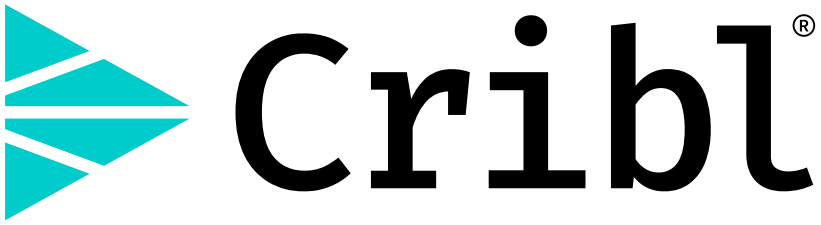


Cribl Lake Documentation Manual

Version: v4.19

Generated: 2026-08-20 23:44:57

1. About Cribl Lake 3
2. Cribl Lake Datasets 6
    2.1. Manage Lake Datasets 8
    2.2. Structure Events for Cribl Lake 17
3. Search Cribl Lake 21
4. Lakehouses in Cribl Lake 25
    4.1. Streamline Logs Storage and Analysis with a Lakehouse 32
5. Cribl Lake Collector 38
6. Cribl Lake Destination 41
7. Ship Data into Cribl Lake Without a Pipeline 45
    7.1. HTTP Direct Access 46
    7.2. Netskope Direct Access 52
    7.3. Zscaler Direct Access 56
    7.4. Splunk Cloud Self Storage (DDSS) Direct Access 60
8. Learn About Storage Locations (Bring Your Own Storage) 65
    8.1. Configure an Amazon S3 Storage Location 67
    8.2. Configure an Azure Storage Location 79
    8.3. About Azure Authentication for Cribl Lake 84
    8.4. BYOS Azure ARM Template Reference 88
9. Integrating Cribl Lake with Cribl Edge 93
10. Configuration Files 95
11. Troubleshooting 96

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 or Direct Access, and stores that data - enabling you to later query the data in Cribl Search, or to 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.

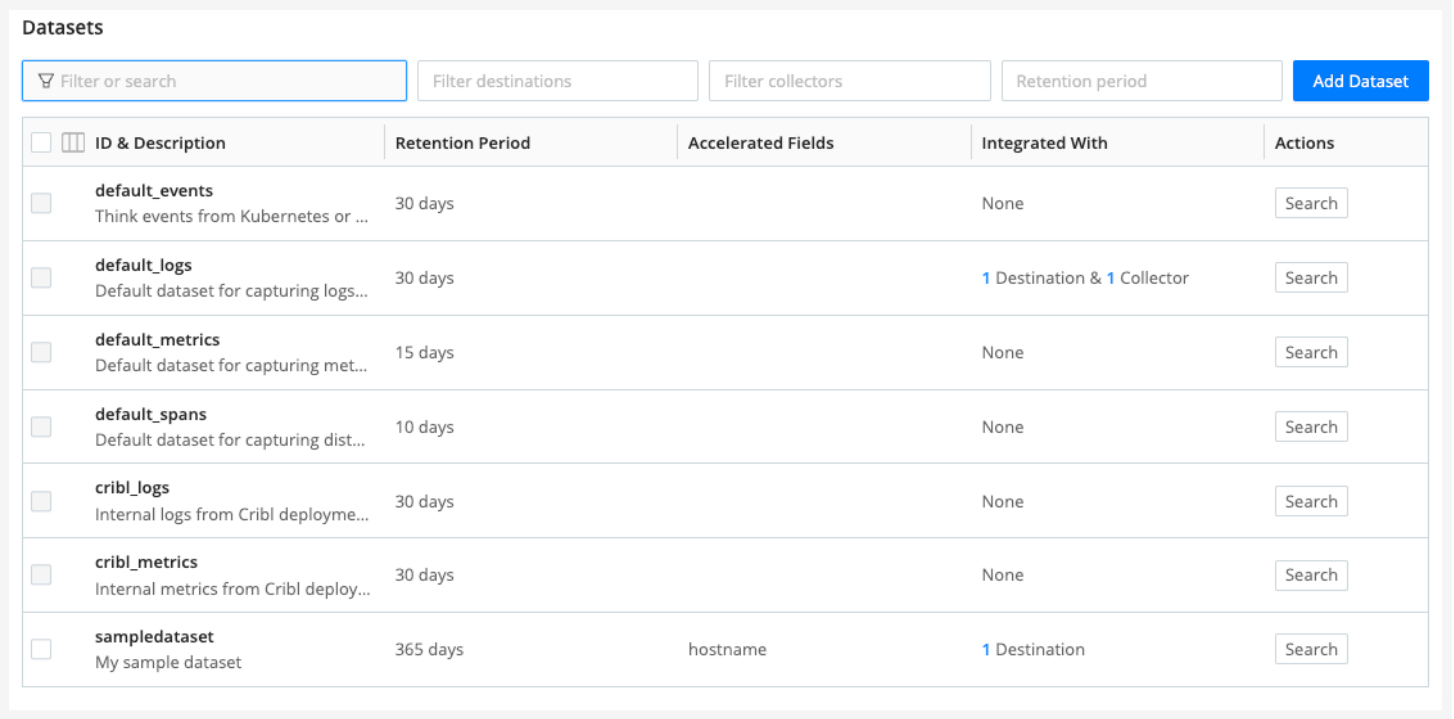
Requirements for Cribl Lake

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.



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 Stream Destination of your choice.

You don't have to think about your partitioning configuration. Cribl Lake takes care of the schema and organizes the data for you.

Depending on your use case, you can store your data in gzip-compressed JSON, Parquet, or DDSS.

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 Dynamic Data Self Storage (DDSS) data to Cribl Lake, use the Lake Splunk Cloud Self Storage option.
- To archive data to Cribl Lake over HTTP, use the Direct Access (HTTP) option.

(This option supports FluentBit, Logstash, Vector, the OpenTelemetry Collector, and a wide variety of other senders. You will find specific configuration settings for Elasticsearch Bulk API and Splunk HEC senders.)

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

2. Cribl 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, create another Cribl.Cloud Workspace.

Lake Dataset Formats

You can set the format of a Lake Dataset to JSON, Parquet, or DDSS.

If you're just exploring, start with the default JSON. But to make the best choice, read these sections beforehand:

- Choose a Lake Dataset Format
- Structure Events for Cribl Lake

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 a retention period of up to 10 years.

If you accelerate a Dataset by linking it to a Lakehouse, the Lakehouse can retain cached data for up to 365 days (but not longer than the Dataset).

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	
`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	Can't be targeted by a Destination.
`cribl_logs`	Logs from the Cloud Leader. Data is stored for 30 days free of charge.	30	Can't 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 page focuses on 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 this parsing is a prerequisite to achieving a Lakehouse performance boost.

You can maintain up to 200 Datasets per Cribl Lake instance (meaning, per Workspace).

Read on to learn how to manage your Lake Datasets:

- Create a New Lake Dataset
- Edit a Lake Dataset
- Delete a Lake Dataset
- Understand Lake Partitions and Lakehouse-Indexed Fields
- Datatypes for Lake Datasets
- Dataset Type for Search (v1 and v2)
- Choose a Lake Dataset Format
- S3 Storage Class for BYOS Datasets

Create a New Lake Dataset

To create a new Lake Dataset:

1. In the Cribl Lake sidebar, select **Datasets**.
2. Select **Add Dataset**.
3. Enter an **ID** for the new Dataset.

   The identifier can contain letters (lowercase or uppercase), numbers, and underscores, up to 512 characters.

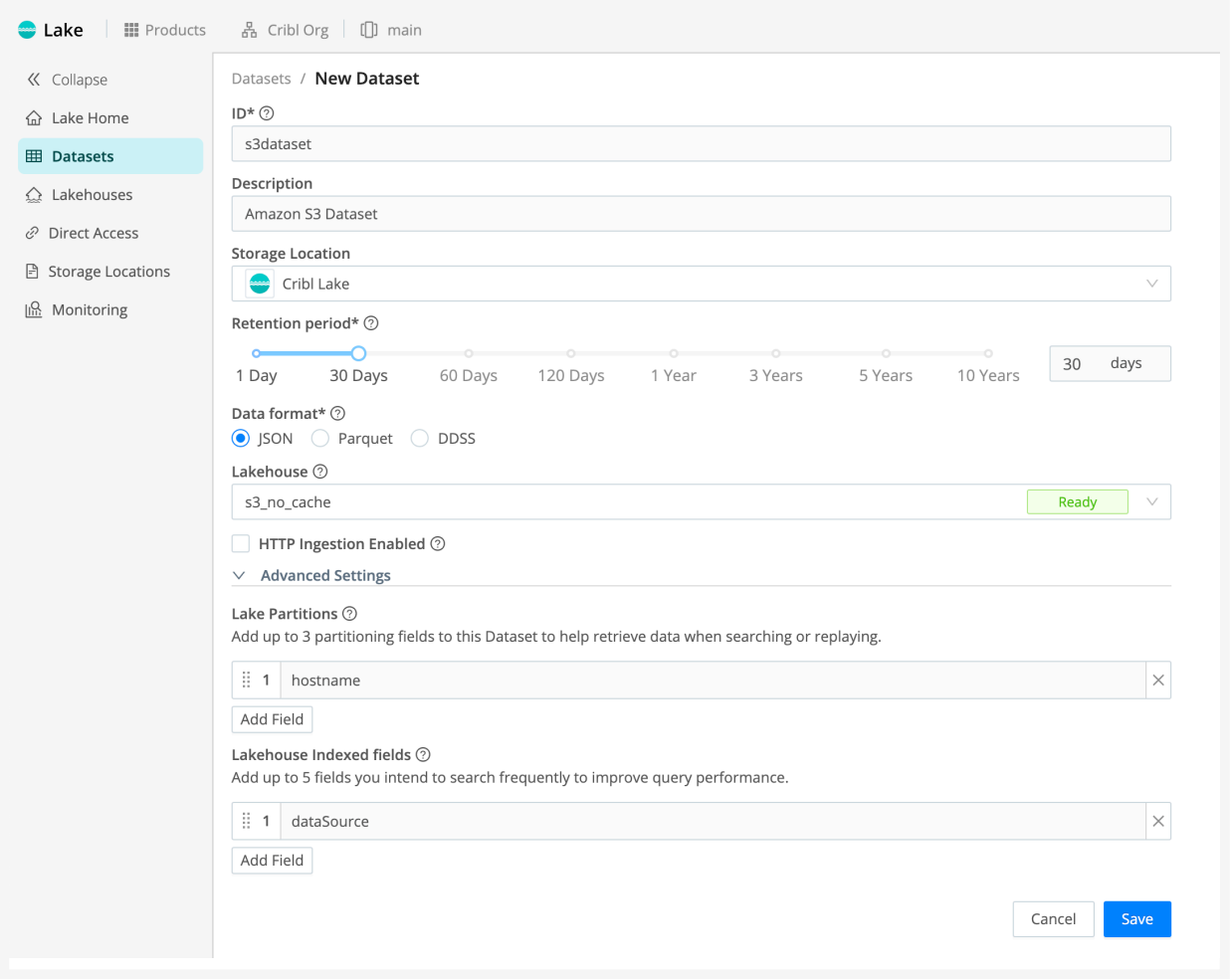
   You can't change the identifier after a Dataset is created, so make sure it is meaningful and relevant to the data you plan to store in it.

   The identifier must be unique and must not match any existing Cribl Search Datasets. Duplicate identifiers can cause problems when searching Cribl Lake data.
4. Optionally, enter a **Description** for the new Dataset.

5. Select a **Storage Location**. The internal `Cribl Lake` storage option is always available on this drop-down, and is the default location if you've configured no external Storage Locations. (Do not select an external Storage Location for a Dataset that you will populate via Direct Access.)
6. If you selected an S3-backed or Azure-backed Storage Location, optionally select a **Storage Class** for objects written to your bucket or container. The default is **Standard** for S3 and **Inferred** for Azure. For details and search compatibility notes, see Storage Class for BYOS Datasets.
7. 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.
8. Select the **Data format**. Keep the default **JSON** to prioritize storage efficiency. Choose **Parquet** to optimize for search performance. Read more: Choose a Lake Dataset Format and Structure Events for Cribl Lake.

   DDSS is available for a Dataset that you will populate via Splunk Cloud Self Storage. The **HTTP Ingestion Enabled** check box will be automatically selected when you configure Direct Access (HTTP) to populate a Dataset.
9. 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 can't link Lakehouses to Datasets populated via Splunk Cloud Direct Access or those using an external Storage Location.
10. If you selected a Lakehouse, define the **Lakehouse retention period**. See Lakehouse Retention for more details.
11. Optionally, in **Advanced Settings**, select up to three **Lake Partitions** for the Dataset.
12. If you are linking the Dataset to a Lakehouse, then in **Advanced Settings**, you have the option to select up to five **Lakehouse Indexed fields**.
13. Confirm with **Save**.



Fill in a few fields and set 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.

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, Storage Location, retention period, partitions, linked Lakehouse, and (for S3- and Azure-backed Storage Locations) Storage Class. You can't modify the identifier of an existing Lake Dataset.
3. Confirm with **Save**.

Change a Lake Dataset 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.

Changing the retention period of the Dataset might affect the retention of its Lakehouse. If you decrease the Dataset retention below the current Lakehouse retention, the Lakehouse retention will be automatically reduced to match.

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

To delete a Lake Dataset:

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**.

Scheduled Deletion

A Lake Dataset is not deleted instantly. Instead, after you select it for deletion, it will be marked with `Deletion in progress`. In this state, you can no longer edit it, or connect it to Collectors or Destinations.

Data that is older than 24 hours will be removed the following day, after midnight UTC. Datasets are only deleted when all of the data in the dataset is 24 hours old or older. Once a Lake Dataset is marked for deletion, you will no longer be charged for any data that it contains.

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

Understand 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.

Configure 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.

The following applies on the Dataset **Processing** tab in Cribl Search (**Datatype Rulesets**). It is separate from **Dataset Type v2 filter rows** (glob plus Datatype ID on the Federated Dataset configuration), which map NDJSON and Parquet objects in Lake storage when you use **Federated Search v2**. See Dataset Type for Search (v1 and v2).

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.

Select the Dataset Type for a Search (v1 and v2)

You choose a **Dataset Type** (v1 or v2) only in Cribl Search, on a federated Dataset that points at your Lake Dataset through the Lake Dataset Provider (`cribl_lake` or equivalent). You'll set the Dataset storage format, retention, partitions, and Lakehouse on the New Dataset window. You'll set the type of your Lake Dataset (v1 or v2) in Cribl Search.

- **Dataset Type v2** supports Lake Datasets stored as JSON (newline-delimited JSON events) or Parquet.
- **DDSS** Lake Datasets support only **Dataset Type v1**. See Splunk Cloud Self Storage (DDSS) Direct Access.

See Dataset Type and filters for Lake Datasets for configuration steps (including the **Filters** table on **Type v2**, where each row sets a **Filter** glob and **Datatype ID**). See Federated Search v2 for limits and migration guidance.

Switch from v1 to v2

To switch an Lake Dataset from **v1** to **v2**, change **Type** on the existing federated Dataset in Search. See Switch a Federated Dataset from v1 to v2.

Choose a Lake Dataset Format

When you create a Lake Dataset, you can set its format to JSON, Parquet, or DDSS. Each format has specific use cases and performance characteristics.

For **Dataset Type v2** in Search, **JSON** Lake Datasets store newline-delimited JSON objects and **Parquet** Lake Datasets store Parquet files. On the federated Dataset, configure the **Filters** table so each row pairs a **Filter** glob with a **Datatype ID** (for example `generic_ndjson` or `cribl_lake_parquet`). See Dataset Type and filters for Lake Datasets. **DDSS** remains **Dataset Type v1** only.

To make the most of JSON or Parquet, preprocess your events in Cribl Stream. Learn more at Structure Events for Cribl Lake.

When to Use JSON

Set your Lake Dataset format to the default JSON when:

- You're just getting started and want to explore Cribl Lake.
- Your data is semi-structured.
- You need a high level of compression, and ordinary search performance.
- You plan to store your original events in the `_raw` field (for compliance, or replay), and drop all other fields.
- You're more likely to retrieve full events, rather than specific fields.

When to Use Parquet

Set your Lake Dataset format to Parquet when:

- Your data is structured as events with multiple fields.
- You want to leverage columnar storage for better analytics performance.
- You expect your queries to filter and aggregate on specific fields.

We don't recommend Parquet if you plan to store unstructured events, like CSV values, or only the `_raw` field in an event. In this scenario, you will see no performance improvement over JSON.

When to Use DDSS

Set your Lake Dataset format to DDSS when you're archiving data from Splunk Cloud using Self Storage. Learn more at Splunk Cloud Self Storage (DDSS) Direct Access.

Storage Class for BYOS Datasets

When you create or edit a BYOS Dataset, you can select the storage class or access tier that Cribl Lake uses when writing objects to your bucket or container. Choosing the right class upfront lets you optimize storage costs without relying on lifecycle policies.

The **Storage Class** dropdown appears in the **Datasets** modal only for Lake Datasets attached to S3-backed or Azure-backed Storage Locations. Managed Lake Datasets do not expose this setting. The options available in the dropdown depend on the Storage Location's provider.

Changing the storage class on an existing Lake Dataset applies only to newly written objects. Cribl Lake does not modify or migrate existing objects in the bucket or container.

Available Storage Classes for S3 BYOS Datasets

Cribl Lake can write data to the following storage classes for S3 BYOS:

Storage Class	AWS Identifier	Notes
Standard	`STANDARD`	Default. General-purpose storage for frequently accessed data.
Intelligent-Tiering	`INTELLIGENT_TIERING`	Automatically moves objects between access tiers based on usage.
Standard-IA	`STANDARD_IA`	Lower cost for infrequently accessed data. Retrieval fees apply.
One Zone-IA	`ONEZONE_IA`	Like Standard-IA, but data is stored in a single Availability Zone.

Storage Class	AWS Identifier	Notes
Glacier Instant Retrieval	`GLACIER_IR`	Archival storage with millisecond retrieval.
Glacier Flexible Retrieval	`GLACIER`	Archival storage. Retrieval takes minutes to hours.
Glacier Deep Archive	`DEEP_ARCHIVE`	Lowest cost archival storage. Retrieval takes hours.

The default storage class for S3 is **Standard**.

Objects written to **Glacier Flexible Retrieval** or **Glacier Deep Archive** cannot be searched on demand. Manual restoration in S3 is required before searching.

Available Access Tiers for Azure BYOS Datasets

Cribl Lake can write data to the following access tiers for Azure Blob Storage BYOS. The default access tier for Azure is **Inferred**.

Access Tier	Azure Identifier	Notes
Inferred	`Inferred`	Default. The blob is not explicitly assigned an access tier and inherits the default tier of its storage account.
Hot	`Hot`	Frequently accessed data.
Cool	`Cool`	Infrequently accessed data stored for at least 30 days.
Cold	`Cold`	Rarely accessed data stored for at least 90 days. Lower storage cost than Cool.
Archive	`Archive`	Lowest-cost tier for rarely accessed data. You can't search objects written to the **Archive** access on demand. You must rehydrate them to a warmer tier before searching.

Search Compatibility

For S3 BYOS Datasets, Cribl Search can read objects stored in the following classes: Standard, Intelligent-Tiering, Standard-IA, One Zone-IA, and Glacier Instant Retrieval. Objects stored in Glacier Flexible Retrieval or Glacier Deep Archive are skipped at search time. To search those objects, restore them manually through the AWS S3 Console or API before running a search.

2.2. Structure Events For Cribl Lake

Preprocess your events, optimizing either for long-term retention or frequent access.

Highlights

- Before sending events to Cribl Lake, format them in Cribl Stream.
- For maximum storage efficiency, use JSON, and remove all fields except `_raw`.
- For best search performance, use Parquet, and keep all fields.

Event Structure Matters

Cribl Lake stores data as either gzip-compressed JSON or Parquet files. The format you choose (when creating a Lake Dataset), and how you structure your events in Cribl Stream determines whether you optimize for **storage efficiency** or **search performance**.

Choose Your Format Based on How You'll Use the Data

If you're storing data primarily for compliance or long-term retention, JSON is typically the better choice. JSON with gzip compression produces smaller files when your goal is to minimize storage costs and you don't need to query the data frequently.

If you plan to search and analyze the data regularly, building dashboards, running investigations, or performing ad-hoc queries in Cribl Search, Parquet is often the better option. Parquet has a columnar format that enables faster queries, especially when your events contain structured fields that Cribl Search can filter efficiently.

Consideration	JSON	Parquet
Use case	Compliance, archival, infrequent access	Frequent search, dashboards, analysis
Event structure	Requires `_raw` only	Requires multiple parsed fields
Compression	Excellent with simplified events	Excellent with repeated field values
Search performance	Slower (scans full events)	Faster (columnar filtering, predicate pushdown)

Optimize Events for Storage Efficiency (JSON)

JSON works best when you prioritize compression and storage efficiency over query performance. For compliance or archival use cases, you can simplify events down to just the original log event stored in the `_raw` field. Use a Pipeline in Cribl Stream to remove all top-level fields except `_raw`. This approach preserves the original event for compliance purposes.

Example: Firewall Log for Compliance Storage

A firewall log arrives in Cribl Stream with multiple extracted fields:

```
{
  "_raw": "Jun 12 14:23:01 fw01 %ASA-6-302013: Built outbound TCP connection 847263 f
  "_time": 1718194981,
  "action": "Built",
  "protocol": "TCP",
  "src_ip": "192.168.1.100",
  "dest_ip": "203.0.113.50",
  "dest_port": "443"
}
```

Copy

If you aim for maximum storage efficiency (for example, for archival purposes), you can configure your Stream Pipeline to remove all non-essential fields, and keep only `_raw`.

You can also keep other top-level fields, which can improve search performance later, but will result in larger file sizes in Cribl Lake.

```
{
  "_raw": "Jun 12 14:23:01 fw01 %ASA-6-302013: Built outbound TCP connection 847263 f
  "_time": 1718194981
}
```

Copy

The original log remains intact in `_raw` for compliance. If you ever need to replay this data for analysis, you can re-parse it through Cribl Stream at that time using a Cribl Lake Collector.

If `_raw` contains valid JSON, Cribl Search can extract fields from that JSON. This does not change the guidance above: storing JSON objects in `_raw` is not an efficient way to get structured fields. For best search performance, send structured top-level fields and use Parquet. See Choose a Lake Dataset Format and Federated Search v2.

Optimize Events for Search Performance (Parquet)

Parquet shines when your events are well-structured with multiple fields, particularly fields with repeated values. Unlike JSON, Parquet stores data in columns and compresses repeated values efficiently, which means fields like Kubernetes labels that appear across thousands of events are stored only once per file.

For Parquet to deliver these benefits, your events must arrive at Lake as parsed, structured data with multiple top-level fields. Sending events with only `_raw` to a Parquet Lake Dataset produces larger files than JSON and degrades search performance.

Cribl Search takes advantage of the Parquet structure through predicate pushdown, using min/max values within the files to skip irrelevant data during queries. This can significantly reduce query times when filtering on fields like `namespace`, `pod_name`, or `log_level`.

For tips on searching Parquet data, see Parquet in the Cribl Search docs.

Example: Kubernetes Event for Active Analysis

A Kubernetes log arrives with container metadata:

```
{
  "_raw": "2024-06-12T14:23:01.847Z INFO [main] Application started successfully",
  "_time": 1718194981,
  "kube_namespace": "production",
  "kube_pod": "api-server-7d4b8c6f9-x2vnm",
  "kube_container": "api-server",
  "kube_node": "node-pool-1-abc123",
  "log_level": "INFO",
  "message": "Application started successfully"
}
```

Copy

Keep all these fields intact when sending to a Parquet Dataset. The repeated values in `kube_namespace`, `kube_node`, and `log_level` compress efficiently, and Cribl Search can filter directly on any of these fields without having to re-parse the full `_raw` field.

Understand How Cribl Stream Processes Events

To learn more about how Cribl Stream processes and structures events, see:

- Event Model – Understand the internal structure of events in Stream

- Event Processing Order – See how events flow through Pipelines
- Pipelines – Configure Functions to shape events before they reach Lake

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.

For **Dataset Type v1 or v2** on Lake-backed federated Datasets (including filter rows on **Type v2**), see Connect Cribl Search to Cribl Lake.

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 caching it in a Lakehouse.

Once a Lake Dataset is linked to a Lakehouse, searches against that Dataset will run significantly faster, as long as your query's time range falls within the configured Lakehouse retention period.

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 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

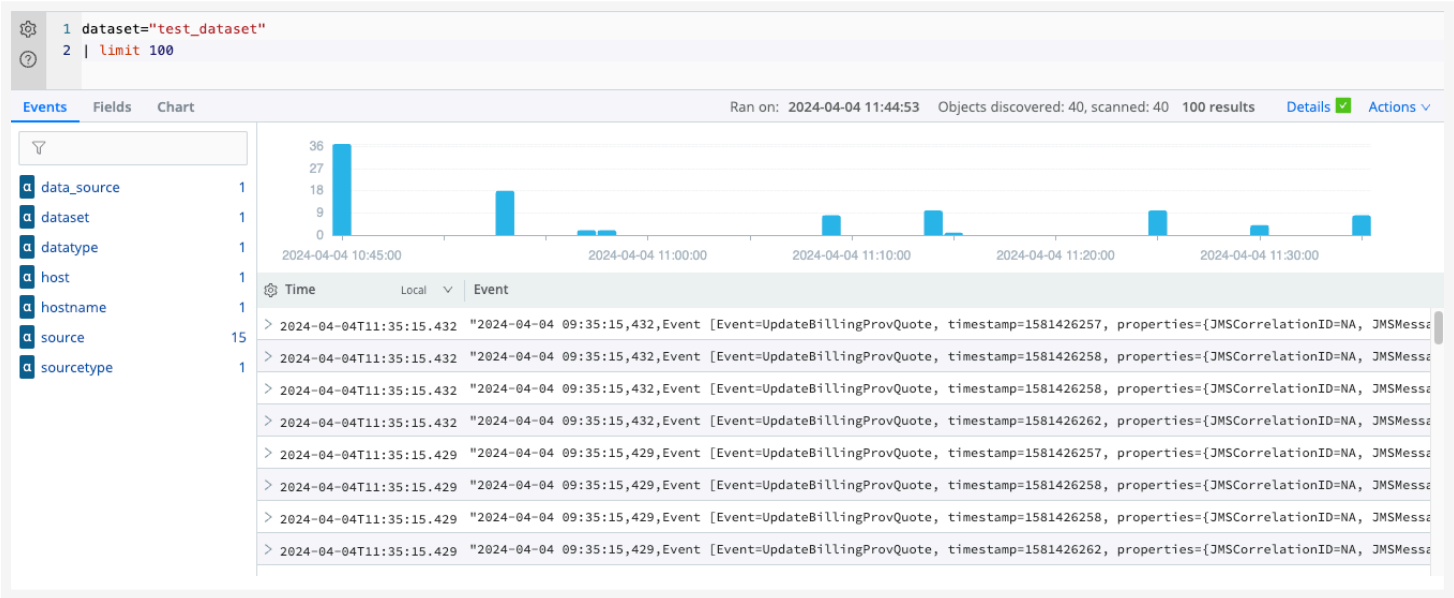
Use these examples as starting points for your own searches.

Basic Search into Cribl Lake

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

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

Copy



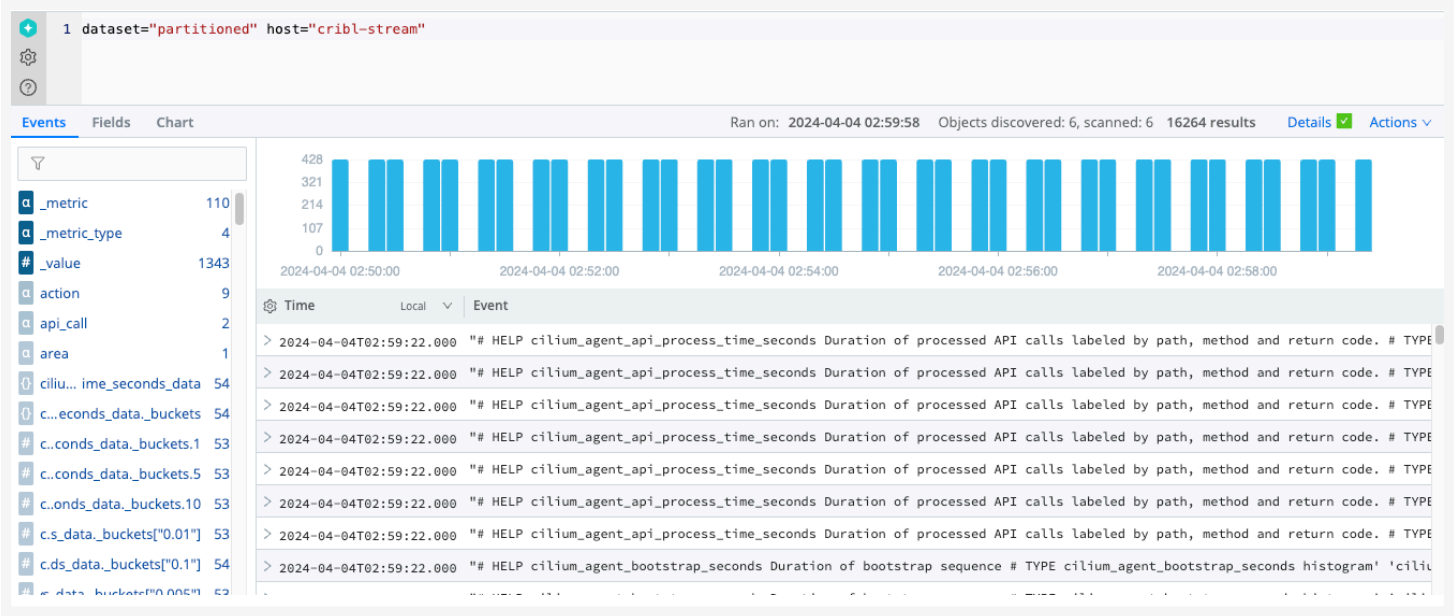
Sample Cribl Lake Search

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 "1713177481843.9AOqxI"
```

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.

Deprecation Notice

For lakehouse workloads, we now recommend using Cribl Search lakehouse engines. **Lakehouses in Cribl Lake will be deprecated in a future release.**

You don't have to take immediate action, though; your existing Lakehouses will continue to work. We'll provide advance notice before you need to make any changes to your existing Lakehouses. Your Cribl Account Team can help with a plan to migrate from Lakehouse to a lakehouse engine.

To speed up searching your Cribl Lake data, you can cache it in a Lakehouse.

For example, Lakehouses can enable efficient storage and fast search of high-volume recent telemetry data. For use case scenarios, see: Streamline Logs Storage and Analysis with a Lakehouse.

You can decide how long to cache data in a Lakehouse by configuring its retention period.

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

Size	Capacity (per day)
XXLarge	9600 GB
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

You can cache data in a Lakehouse for a configurable retention period of 1 to 365 days (but not longer than the retention period of the underlying Lake Dataset).

The default Lakehouse retention is 30 days.

When data ages out of the Lakehouse, it's removed from the cache but remains available in the Lake Dataset (until it reaches the Dataset's own retention limit).

Add a New Lakehouse

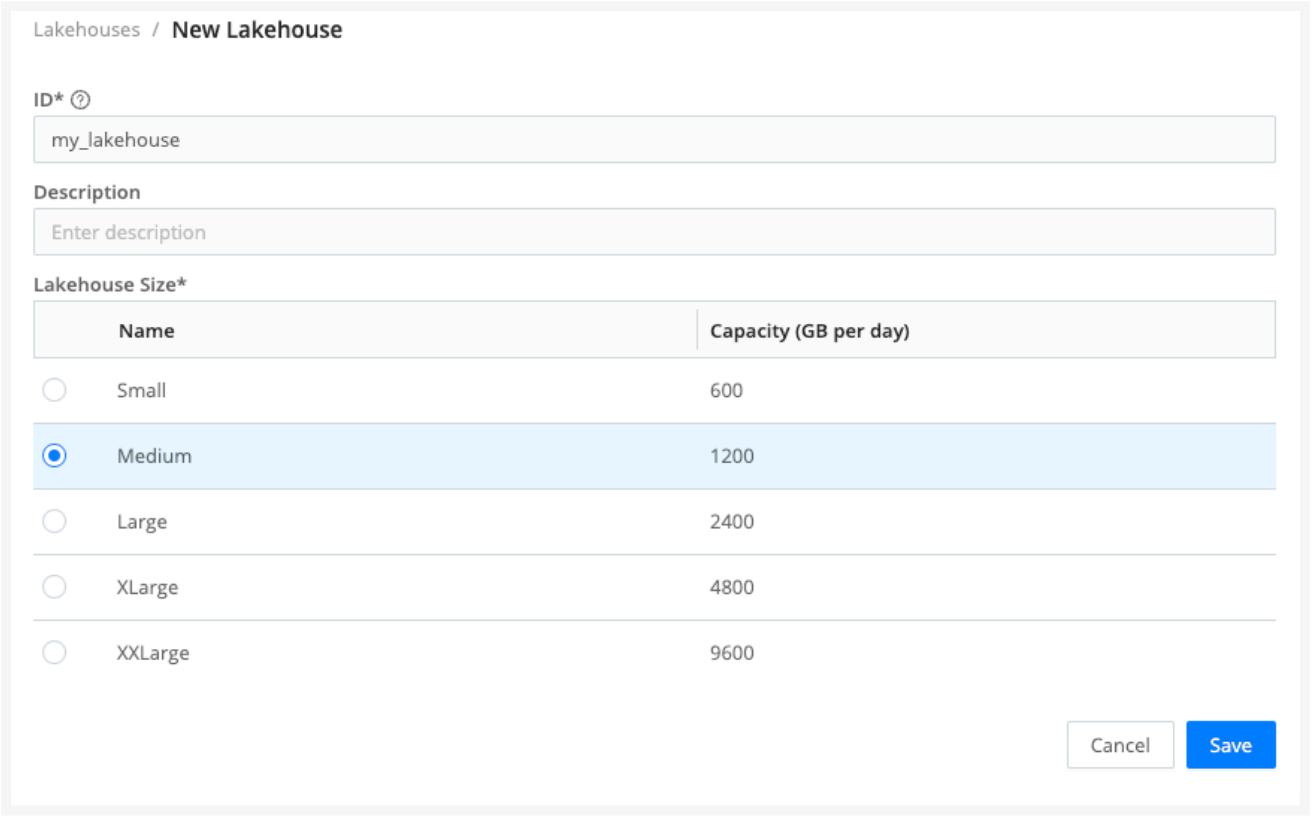
A Lake Dataset can be linked to just one Lakehouse. However, a single Lakehouse can have more than one Lake Dataset linked to it.

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.



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.

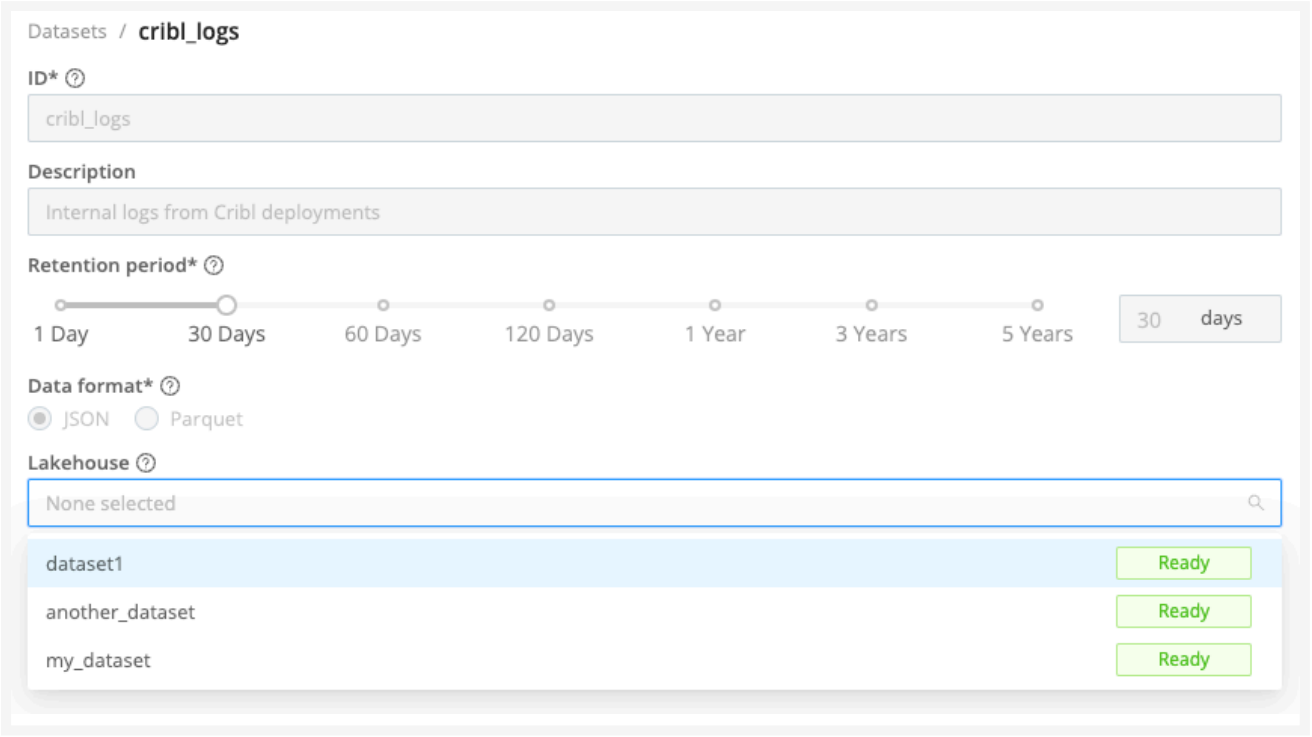
You can link Datasets in two ways, influencing how the Dataset will behave in relation to the Lakehouse:

- 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. Set the **Lakehouse retention period**. See Lakehouse Retention for more details.



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**.
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.
5. Set the **Lakehouse retention period**. See Lakehouse Retention for more details.

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 can't resize it or link it to Lake Datasets.

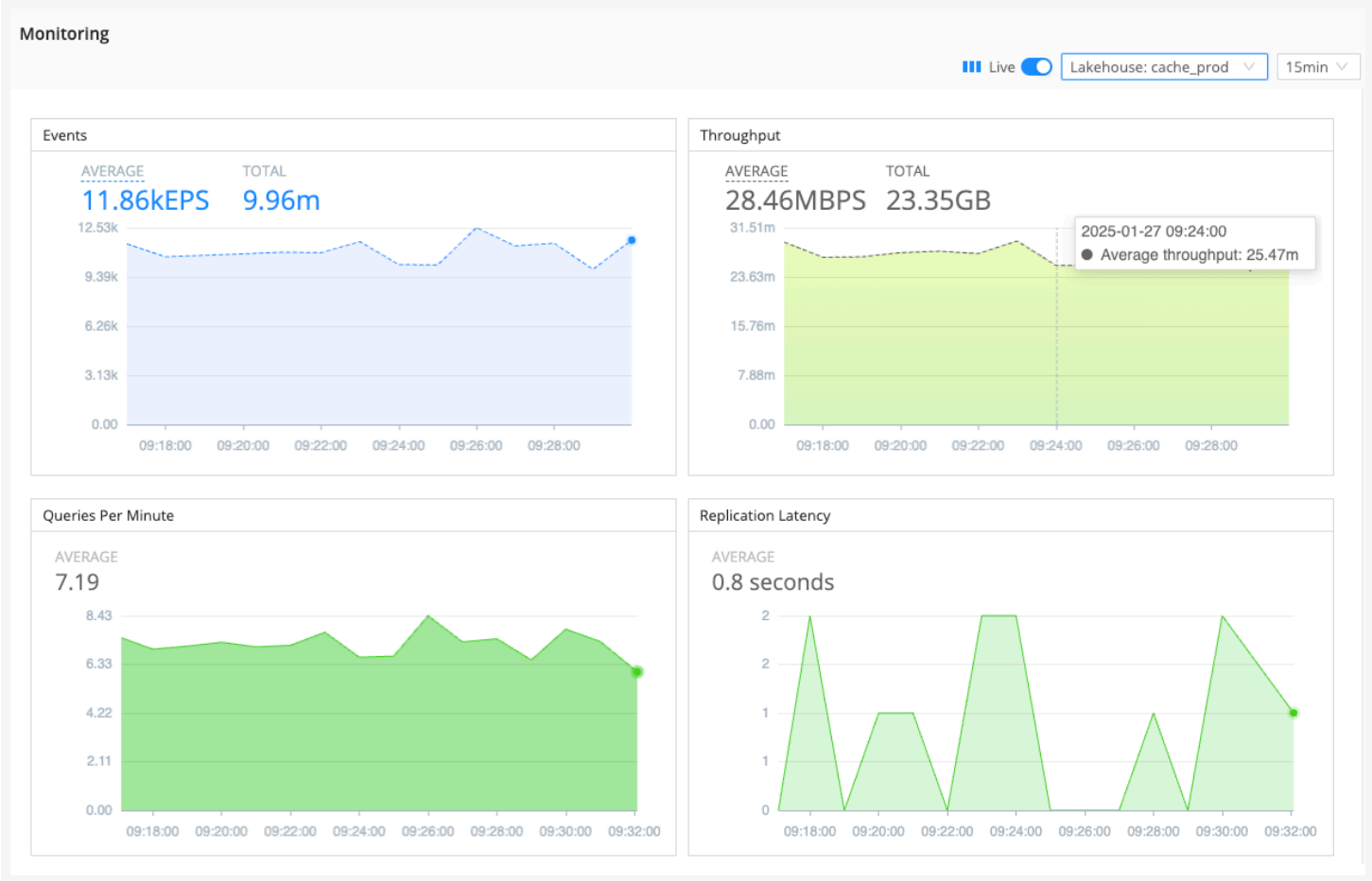
Status	Description
Delayed	Provisioning the Lakehouse has encountered an issue, but it will move to Ready status when the issue is resolved. While the Lakehouse is Delayed, you can resize it, but you can't link it to Datasets.
Terminated	Lakehouse has been marked for deletion.

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.

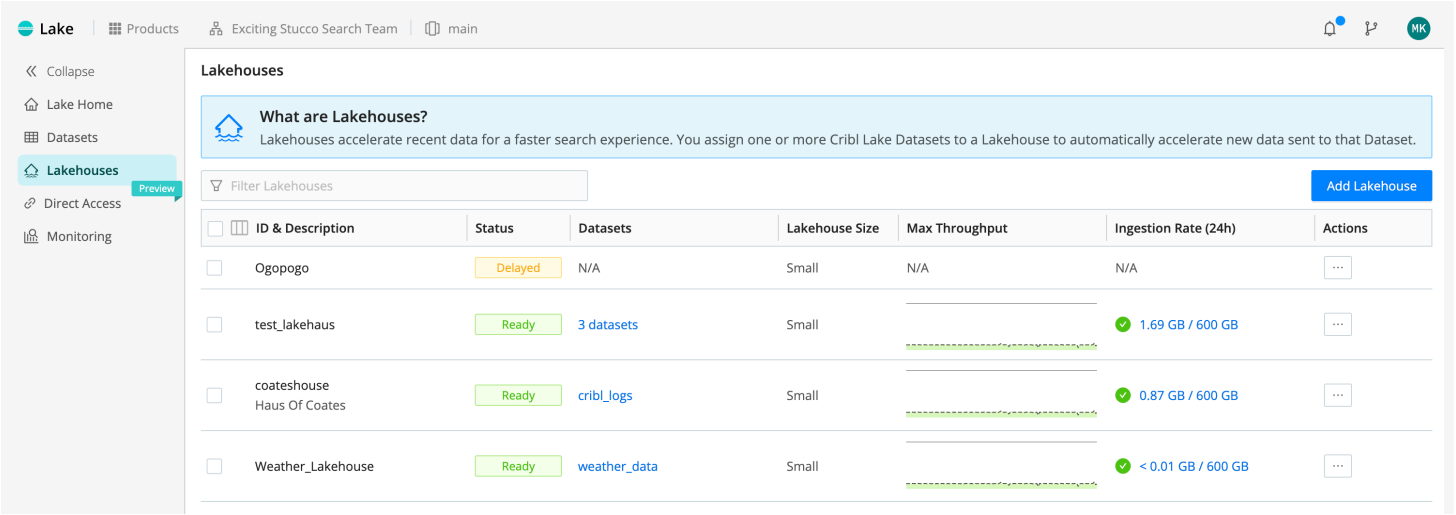
- **Events**: Number of events ingested into the Lakehouse.
- **Throughput**: Number of bytes ingested into the Lakehouse.
- **Queries per Minute**: Number of queries performed on the Lakehouse.
- **Replication Latency**: Latency of ingesting data into the Lakehouse.



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.



4.1. Streamline Logs Storage And Analysis With A Lakehouse

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

Deprecation Notice

For lakehouse workloads, we now recommend using Cribl Search lakehouse engines. **Lakehouses in Cribl Lake will be deprecated in a future release.**

You don't have to take immediate action, though; your existing Lakehouses will continue to work. We'll provide advance notice before you need to make any changes to your existing Lakehouses. Your Cribl Account Team can help with a plan to migrate from Lakehouse to a lakehouse engine.

This is an overview of how to ingest log files into a Lakehouse, and then use Cribl Search to periodically analyze only specific data 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 page demonstrates sequential setup in your upstream logs sender, Cribl Stream, Cribl Lake, and Cribl Search. The broad steps are:

1. Collect and stage logs in the applications and frameworks that generate them..
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.

Collect, Stage, and Shape Logs

Once you've configured log collection on the infrastructure where you run your applications, ingesting logs into Cribl Stream provides a straightforward way to parse events for efficient storage and retrieval in Cribl Lake.

Cribl Stream ships with dedicated Source integrations for common data senders and protocols. Our Integrations page outlines steps for customizing our Sources to ingest data from senders without dedicated counterparts.

Shape with Parsers

With data flowing through Cribl Stream, you can apply the built-in Parser Function to parse events into fields optimized for Cribl Lake storage. This Function enables you to choose from a library of predefined Parsers for common data sources, and the library's UI facilitates customizing, cloning, and adding Parsers to shape your specific data.

Here, we provide specific steps for collecting, staging, and parsing three common Amazon log formats: VPC Flow Logs, CloudTrail logs, and CloudFront logs. Beyond these specific examples, you can use a similar workflow to efficiently store and analyze many other types of logs, by substituting a different Stream Parser or configuring your own.

VPC Flow Logs | CloudTrail Logs | CloudFront Logs

Collect and Stage VPC Flow Logs

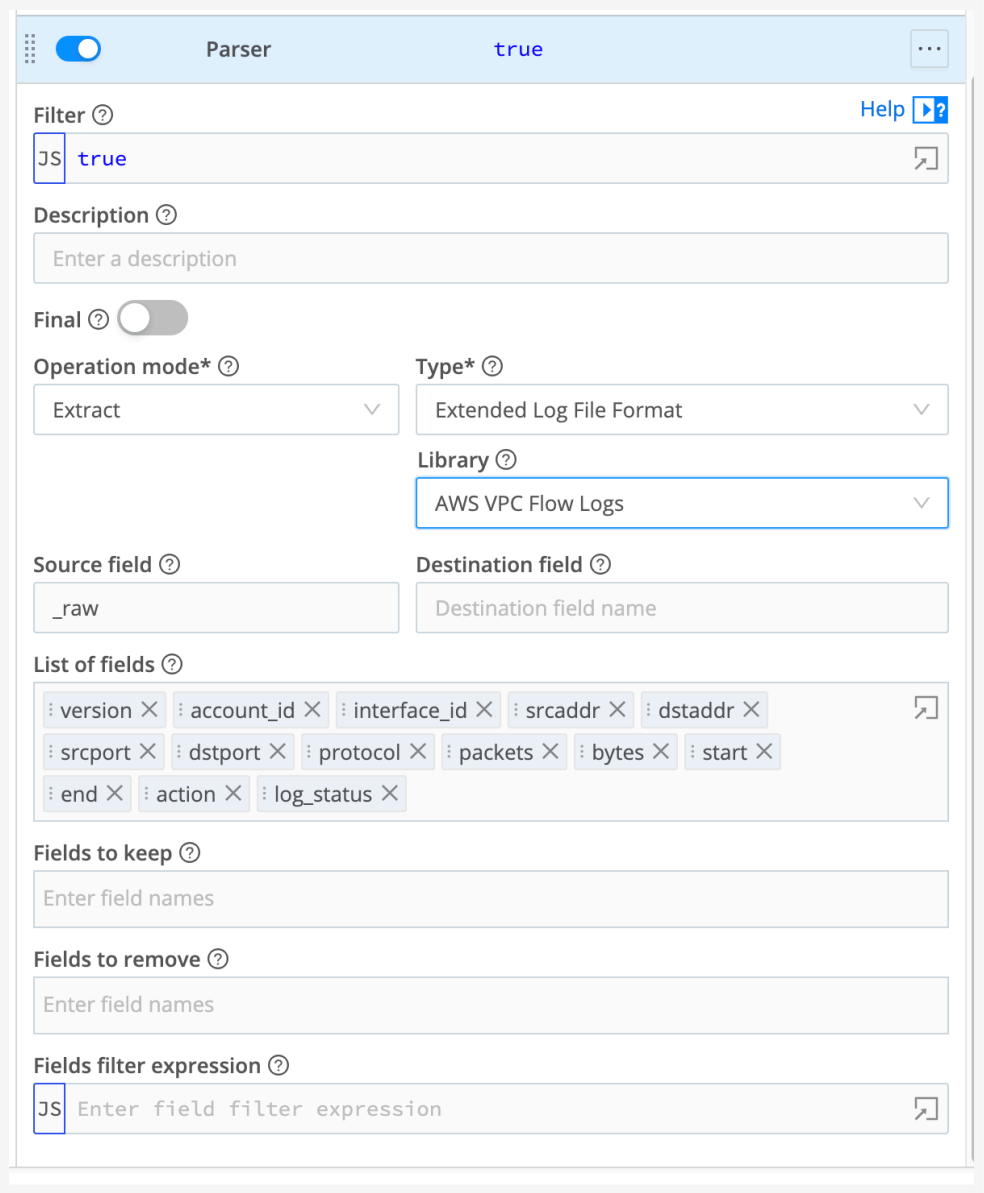
Your first steps take place in the Amazon VPC console or EC2 console. If you're not already collecting VPC Flow Logs, the Amazon Logging IP Traffic Using VPC Flow Logs documentation provides specific instructions.

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 S3 permissions for Cribl Stream to read from the bucket.

Shape (Parse) 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.



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.

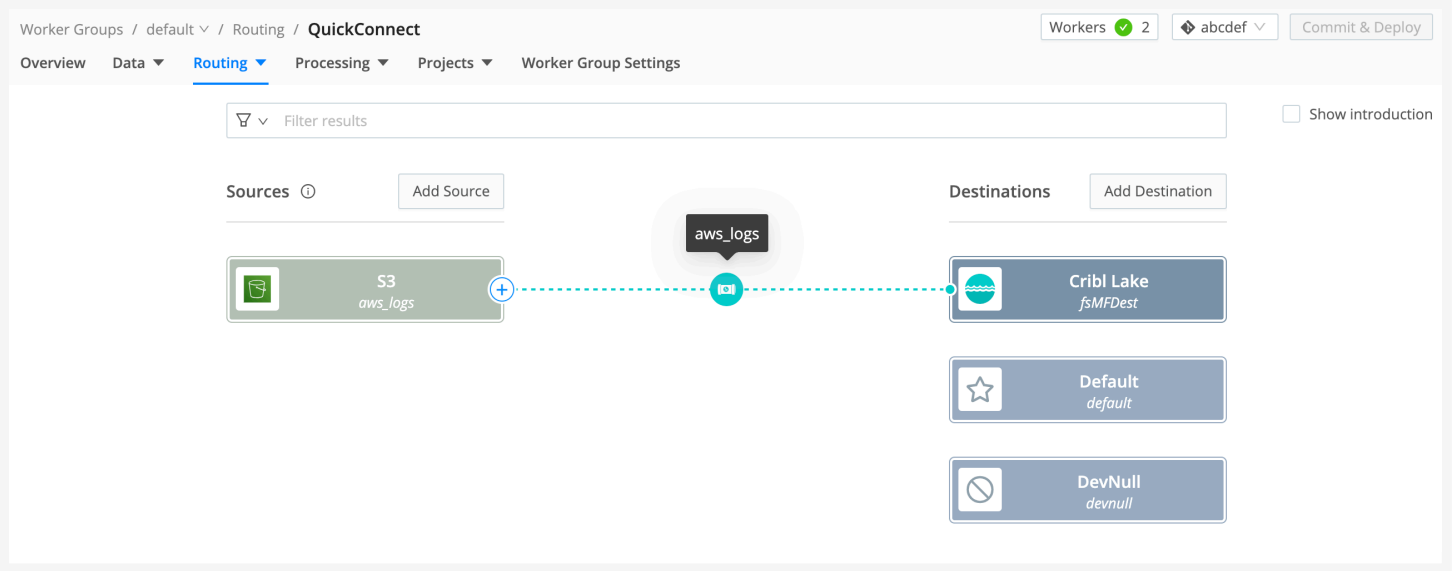
Create a Cribl Lake Dataset

In Cribl Lake, create a Lake Dataset to store your shaped data.

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 page 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 your infrastructure all the way 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 Lake Dataset to the Lakehouse.

Analyze Relevant Data

In Cribl Search, define queries to extract relevant data from your logs. 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.

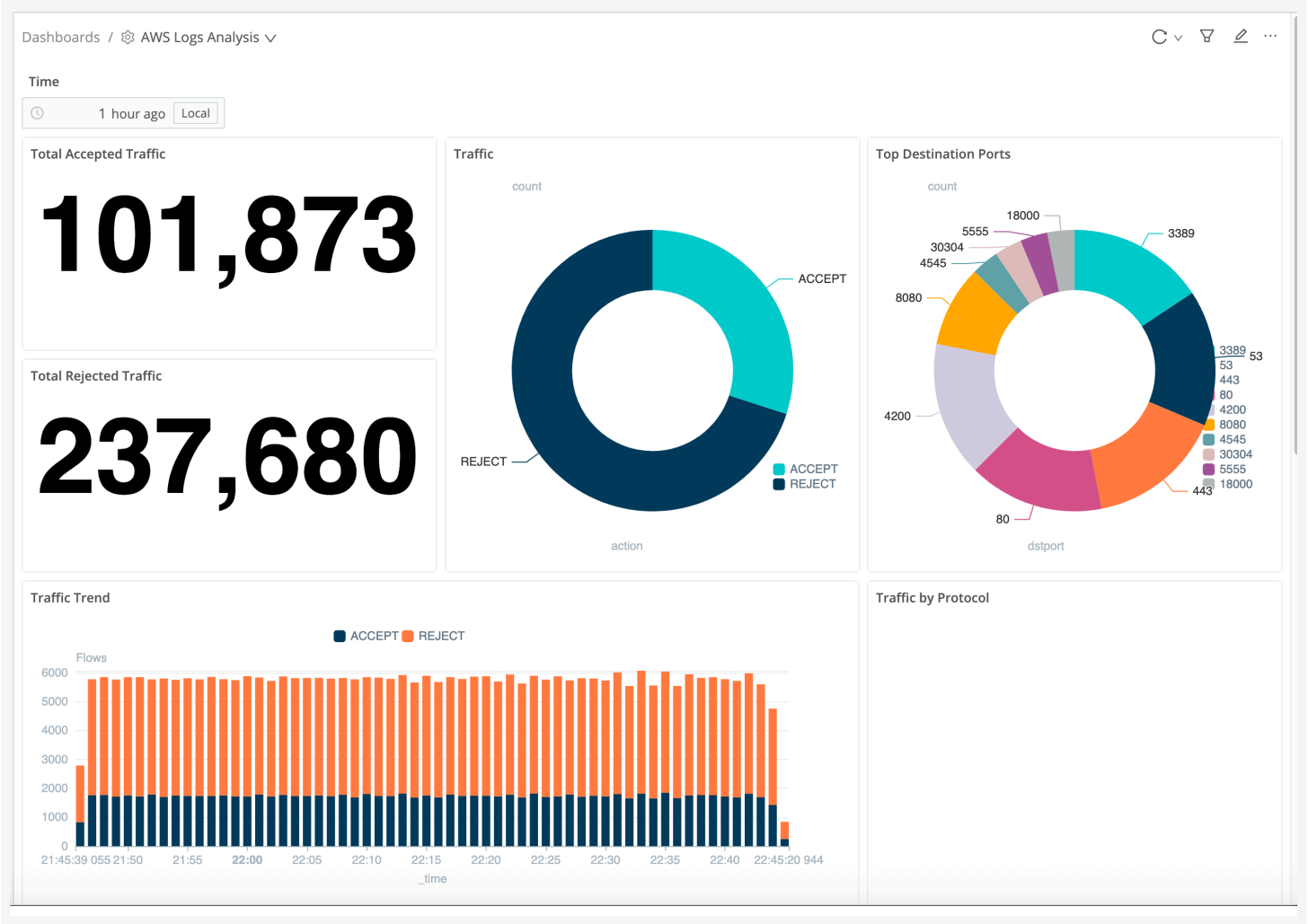
You can also Write Queries Using Cribl Copilot, enabling Cribl AI to suggest KQL queries from your natural-language prompts. If you already have searches in another system of analysis, try asking Copilot to translate them to KQL.

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.

As a starting point for displaying analytics on many log types, 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.



Sample Cribl Search Dashboard

5. 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`.
    - **Storage Location**: Where the Dataset stores its data. The drop-down lists **Cribl Lake** (default) and each external Storage Location your user can access. This workflow is the same when you create a Lake Dataset. After you pick a Storage Location, **Lake Dataset** lists only Datasets that use that location.
    - **Lake Dataset**: Lake Dataset to collect data from (filtered by the selected **Storage Location**).
    - **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.

Filter Lake Data

When you run or schedule a Cribl Lake Collector job, you can set a **Filter** expression. Cribl Stream evaluates that expression against both the Lake object path and the collected event fields. For background, see Filter.

`source` Filters Both Files and Events

Filtering on the `source` field restricts which Lake files Cribl Stream pulls, not only which events it keeps. Lake files are typically stored with extensions such as `.json.gz`, so a filter like `source.endsWith('.log')` can return no results even when events contain a `source` field ending in `.log`.

Avoid filtering on `source` unless you need to. If you must, match both the file extension and the event values you need, for example:

```
source.endsWith('log') || source.endsWith('gz')
```

You can also collect without a `source` filter and filter on `source` later in Routes or Pipelines.

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.

6. 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 delivers 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 receive data from 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 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:

   - On the **Worker Groups** submenu, select **Routing**, then **QuickConnect**.
   - Select **Add Destination** at right.
   - Hover over **Cribl Lake** from the list of tiles.
   - Select **Add New** to open a **New Destination** modal.

   Or, to configure via the Routing UI:

- On the **Worker Groups** submenu, select **Data**, then **Destinations**.
- In the **Destinations** tiles, select **Cribl Lake**.
- 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.

How Cribl Stream Handles the `_time` Field

The Cribl Lake Destination requires `_time` to be a number and not `null`. If an incoming `_time` value is `null` or a string, this Destination will convert the value to `Date.now() / 1000`. This creates a time-based partitioning scheme that is readily searchable.

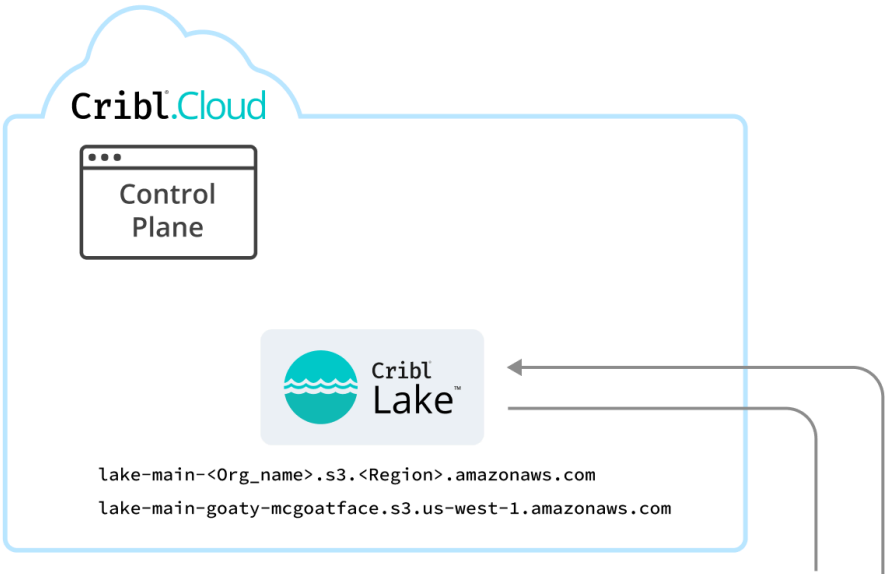
Troubleshooting Cribl Lake Hybrid Access

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 HTTP/S - 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.


Cribl.Cloud
Control Plane
Cribl Lake
lake-main-<Org_name>.s3.<Region>.amazonaws.com
lake-main-goaty-mcgoatface.s3.us-west-1.amazonaws.com

7. Ship Data Into Cribl Lake Without A Pipeline

Ship your data straight into Cribl Lake without using a Pipeline, bypassing Cribl Stream.

You can use Direct Access options to archive certain data directly to Cribl Lake. Here, you bypass 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.

We provide the following Direct Access options for different data sources:

- Direct Access (HTTP): Use this option to ingest data over HTTP from a wide variety of senders. Examples include FluentBit, Logstash, Vector, and the OpenTelemetry Collector. You will find specific configuration settings for Elasticsearch Bulk API and Splunk HEC senders.
- Netskope Direct Access: Use this option to ingest data using Netskope for Cloud Access Security Broker (CASB), data loss prevention (DLP), and network-security telemetry will be able to land that data directly into Cribl Lake for search and analysis without manually managing AWS infrastructure.
- Zscaler Direct Access: Use this option to ingest Zscaler Nanolog Streaming Service (NSS) data directly into Cribl Lake for search and analysis, streamed over HTTP in Splunk HEC format.
- Splunk Cloud Self Storage: Use this option to ingest data from Splunk Cloud DDSS (Dynamic Data Self Storage) indexes.

You can configure one Direct Access Source of each type per Workspace.

7.1. HTTP Direct Access

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

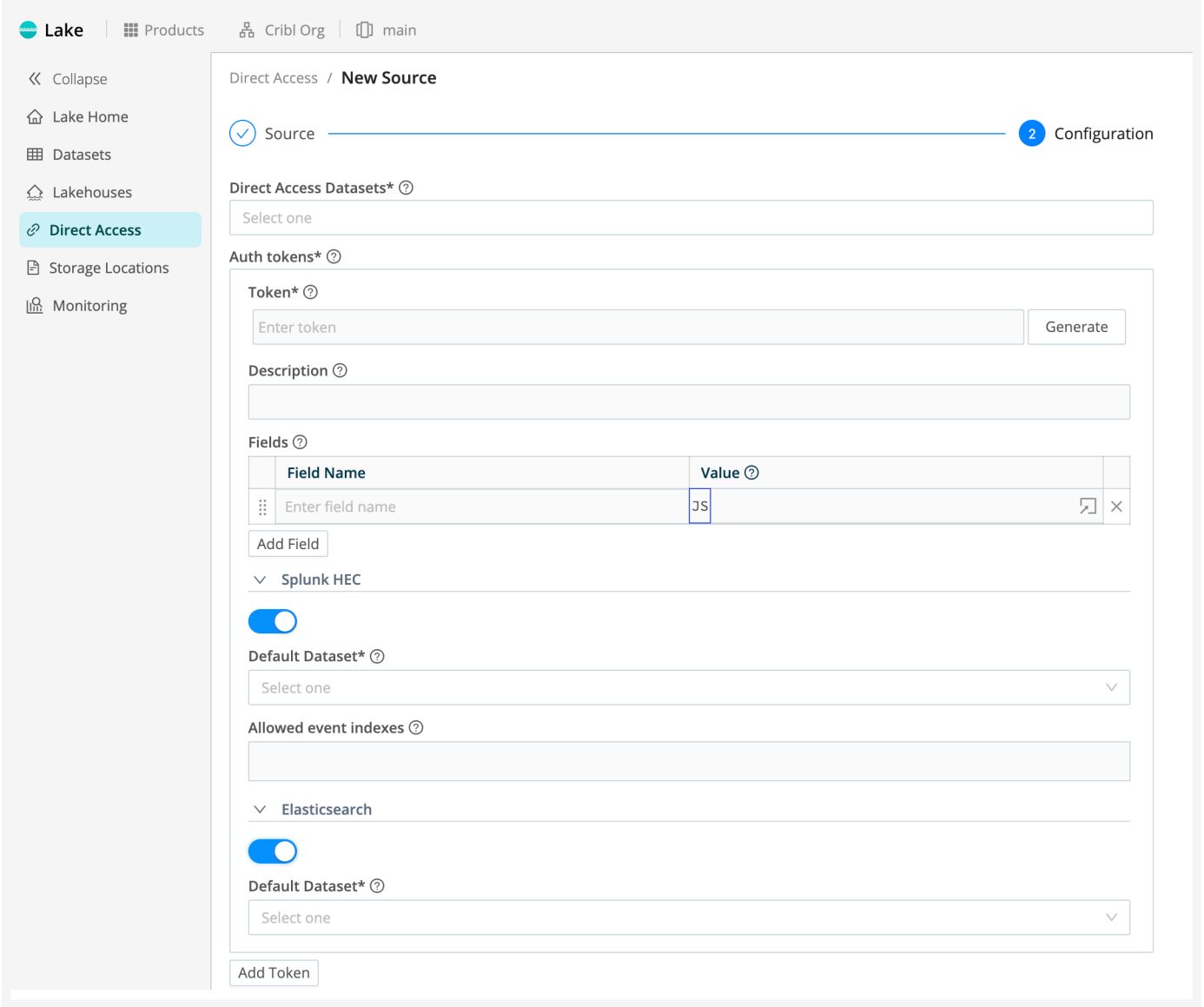
You can use the Direct Access (HTTP) option to archive data directly to Cribl Lake over HTTP from a wide variety of senders. Examples include FluentBit, Logstash, Vector, and the OpenTelemetry Collector. You will find specific configuration settings for Elasticsearch Bulk API and Splunk HEC senders.

This option bypasses routing or processing through Cribl Stream. Once your data is integrated with Cribl Lake, Cribl Stream and Cribl Search can access it as a Lake Dataset.

Create a Direct Access (HTTP) Source

You can configure one Direct Access (HTTP) Source per Workspace. (See other Limitations later in this topic.)

1. From the Cribl Lake sidebar, select **Direct Access**.
2. From the resulting **Direct Access** > **New Source** page, select **HTTP** and **Next**.
3. On the resulting **Configuration** page, you can select up to 10 **Direct Access Datasets**. (After you save this selection here, each targeted Dataset configuration will indicate **HTTP Ingestion Enabled**. For details, see Manage Lake Datasets.)



Direct Access (HTTP): all configuration options

4. In the **Auth tokens** section, you must configure at least one token, but you can add multiple tokens to authorize ingest from multiple senders. For configuration details, see Configure Auth Tokens.

   Each auth token supports an optional **Fields** section, where you can select **Add Field** to tag events with key-value pairs that identify or match this token and sender. You will also see specific options for Splunk HEC and Elasticsearch Bulk API senders.

5. Once you've configured auth tokens, select **Save**.

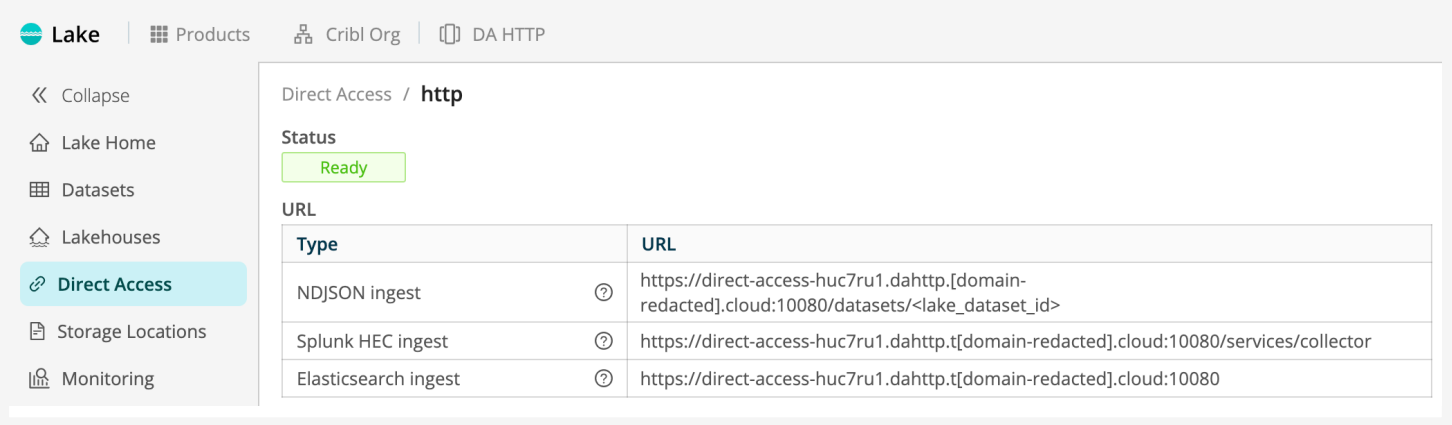
Provisioning and Ready States

For the first few minutes after you save Direct Access (HTTP) Source, you will see a badge showing that the Source is in a `Provisioning` state. In this state, it is setting up infrastructure and deploying configurations. You can't use the Source to ingest data until it shows a `Ready` badge.

Once the Source moves to `Ready`, it is available to ingest data and store it in Lake Datasets.

Examine HTTP Ingest Endpoints

After the Source indicates `Ready`, select it to open its config. Here, the **URL** table shows you the Cribl Lake ingest endpoints that will accept data in NDJSON, Splunk HEC, and Elasticsearch Bulk API formats. As outlined in the following section, you complete each URL by replacing the `<lake_dataset_id>` placeholder with the **ID** of each target Dataset.



Direct Access (HTTP) Source ready to ingest data, with ingest endpoints displayed

Send Data to Lake over HTTP

To ingest data, send a POST request to the endpoint provided in the **URL** of the Direct Access (HTTP) Source. Append the Lake Dataset **ID** to which the data should be sent. You must provide the auth token in the request header, in the format: `Authorization: <my_token>`. Here is an example:

```
curl --location "https://direct-access-huc7ru1.dev.trusting-carver-psu2edw.cribl-stag
--header "Content-Type: application/json" \
--header "Authorization: ••••••" \
--data '{"id": 974027, "name": "jnTdxFQXxd", "value": 0.1772688577715229, "timestamp'
```

Copy

Send Data to Lake with Vector

Vector is one agent that is compatible with Direct Access. To send data to a Cribl Lake Dataset with Vector:

1. In Cribl Lake, from your Direct Access (HTTP) Source page: Copy the **Ingest URL**, appending the **ID** of your target Dataset.
2. In Vector, configure an HTTP sink to the resulting endpoint.
3. As shown in the sample YAML below, your request headers must provide the auth token, in the format: `Authorization: <token>`.

```
cribl_lake_http:
    type: http
    inputs: ["<vector_sink_or_transform"]
    request:
      headers:
        authorization: "<token>"
    encoding:
      codec: "json"
      json:
        pretty: true
    uri: https://direct-access-huc7ru1.<tenant_id>.cribl.cloud:10080/datasets/<datase
```

Copy

Configure Auth Tokens

Here, you define the authentication token required to access each corresponding sender's inbound data. Ensure that the token is valid and has the necessary permissions for data retrieval. Select **Add Token** to define more tokens. For each token, you can configure the following settings, with specific options available for Splunk HEC and Elasticsearch senders.

Token: Enter your token here. You also have the option to **Generate** a new token.

Description: Optionally, enter a summary of this token's purpose. Recommended, because the description becomes a friendly display name for each token.

Fields: Fields to add to events referencing this token. Each field is a **Name/Value** pair.

Fields that you specify here will normally override fields of the same name in events. However, you can specify that event fields' values should prevail.

In particular, where inbound events have no `index` field, this Source adds one with the literal value `default`. You can override this value by using **Add Field** to specify an `index` field, and then setting its **Value** to an expression of the following form: `index == 'default' ? 'myIndex' : index`

Splunk HEC Token Options

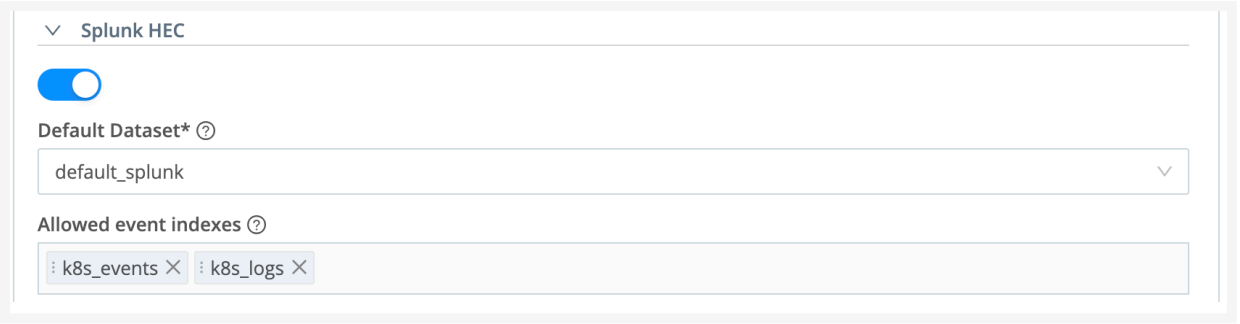
The Direct Access (HTTP) Source will inspect each Splunk HEC event, looking for a field named `index` that is sent by a Splunk Heavy Forwarder. Ideally,the `index` field's value should match an identically named Cribl Lake Dataset.

If Direct Access finds no `index`, or if the `index` value does not match a Cribl Lake Dataset, it will store the data in the Dataset that you select in the **Default Dataset** drop-down.

Allowed Event Indexes

In this optional field, you can specify which Splunk HEC event indexes are permissible for events ingested using this token. (Cribl Lake will reject requests whose indexes don't match strings or patterns here.) Leave empty to accept all events. You can enter multiple index values, and you can use wildcards (`*`) for pattern matching.

Events lacking any `index` field will still be ingested into the `hec_syslog_da` fallback Dataset.

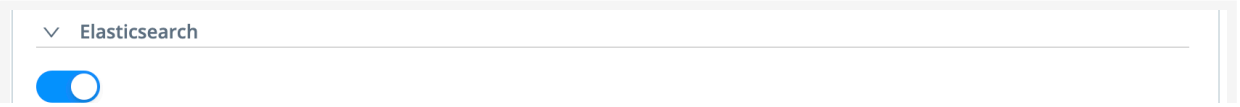


Splunk HEC configuration options

Elasticsearch Token Options

For data sent via the ElasticSearch Bulk API, the Direct Access (HTTP) Source will inspect each event, looking for a field named `index` that is sent by a compatible agent. Ideally,the `index` field's value should match an identically named Cribl Lake Dataset.

If Direct Access finds no `index`, or if the `index` value does not match a Cribl Lake Dataset, it will store the data in the Dataset that you select in the **Default Dataset** drop-down.



7.2. Netskope Direct Access

Ship your Netskope data straight into Cribl Lake without a Pipeline, bypassing Cribl Stream.

Use the Netskope Direct Access option to land Netskope data directly into Cribl Lake for search and analysis, without manually managing AWS infrastructure. The Netskope Direct Access source supports the following Netskope telemetry:

- Cloud Access Security Broker (CASB)
- Data loss prevention (DLP)
- Network security

How Netskope Direct Access Works

Netskope Direct Access in Cribl is comprised of four main steps. Here's an overview of the steps you'll complete in this guide:

1. In Netskope, create a Netskope Log Streaming stream with an **Amazon S3 destination**.
2. In Cribl Lake, create a Netskope Direct Access Source and generate an External ID for Netskope. Cribl provisions a managed S3 bucket and IAM role.
3. Create a Lake Dataset on the auto-created Storage Location to hold the data.
4. Add Cribl connection details in Netskope to enable data transfer.

Learn About the Implementation

The Netskope Direct Access implementation has the following details:

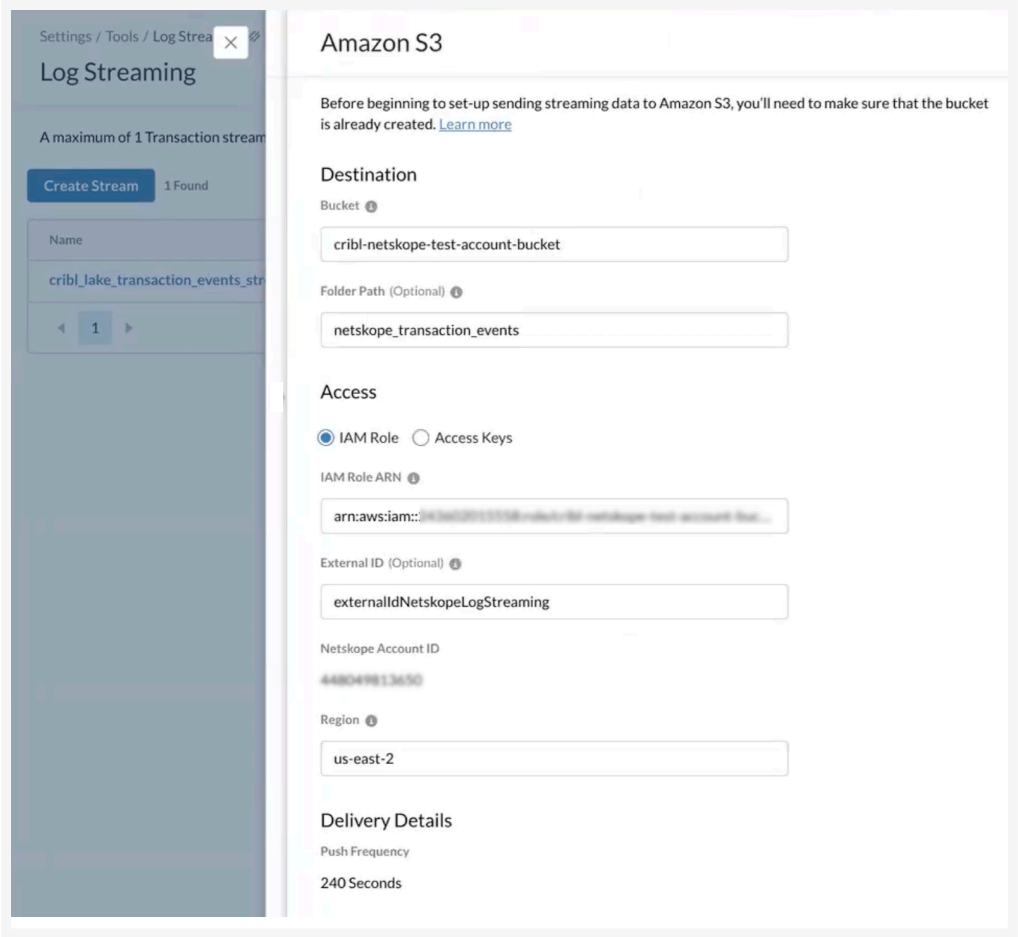
- You can configure one Netskope Source per Cribl Lake Workspace.
- You can have many Datasets per Source.
- Data flows directly from Netskope into your S3 bucket; it doesn't pass through Cribl infrastructure.
- Netskope Direct Access only supports v1 Datasets.

Create a Netskope Log Streaming Stream

You'll need to set up a few things in Netskope before you create the Direct Access Source in Cribl.

1. In Netskope, create a Log Streaming stream. Select **Amazon S3** as the **Destination**.
2. Don't add a **Folder Path** yet – this will be the Dataset name.
3. Copy the Netskope Account ID and the Region to use in Cribl.

This is the view in Netskope you'll need:



Select the Log Streaming stream in Netskope and open the Destination

See the Netskope Stream Logs to Amazon S3 documentation for details on where to find these fields.

Create a Netskope Direct Access Source

Create the Source in Cribl Lake so that Cribl can provision the managed S3 bucket and IAM role that Netskope writes to.

1. In Cribl Lake, select **Direct Access** in the left navbar.
2. On the **Source** step, select **Netskope**, then select **Next**.
3. On the **Configuration** step, enter the required fields. You'll need access to your Netskope stream destination to collect some of these field values.

Field	Description	Notes
Netskope Account ID	The Netskope Account ID associated with the Log Streaming stream. To find this ID, consult the Netskope docs: Stream Logs to Amazon S3.	Required.

Field	Description	Notes
Description	A descriptive name for this direct access connection.	Optional.
Region	The Region where the Netskope AWS bucket resides.	Required. Must match the Region you configured in the Netskope stream.
External ID	The Netskope ID you'll use to form the connection to Cribl Lake. Generate an ID in Cribl and paste it into the **External ID** field on the Netskope Amazon S3 Destination form.	Required.

4. Select **Save**. After you select **Save**, the Netskope Source enters a `Provisioning` state while Cribl sets up infrastructure. Provisioning usually completes in a few minutes.
5. When the provisioning completes, you can open the Netskope Direct Access Source to see the **IAM role**.

Create a Netskope Dataset

Create a Dataset on the auto-created Storage Location to hold and organize the incoming Netskope data.

1. In Cribl Lake, navigate to **Datasets**.
2. Select **New Dataset**.
3. Enter an ID. You will copy this value into the **Folder Path (Optional)** field in Netskope.
4. Select the Netskope Storage Location.
5. Add a retention period and select **Save**.

For more detailed information on Datasets, see Cribl Lake Datasets.

Add Cribl Connection Details in Netskope

These steps are required to get data flowing correctly.

Add the IAM Role in Netskope

1. Once the Source is `Ready` in Cribl, open it and copy the **IAM role** field value.
2. Navigate to Netskope.
3. Paste the IAM role in the Netskope **IAM role ARN** field.
4. Select **Save** in Netskope.

Add the Folder Path in Netskope

1. In Cribl Lake, open the Netskope Dataset.
2. Copy the Dataset ID.
3. Navigate to Netskope and paste the Dataset ID into the **Folder Path (Optional)** field. This grants Netskope the ability to write data to Cribl Lake.

Provisioning Lifecycle

A Netskope Direct Access Source will have one of these statuses after you select **Save**:

Status	Description
`Provisioning`	Cribl is provisioning the S3 bucket. Credentials are computed and visible to the Cribl Admin.
`Ready`	The S3 bucket is reachable, S3 inventory is configured. You can attach Datasets to this Source.
`Failed`	The provisioning timed out or hit an unrecoverable error. You must delete the Source and try again.

Query Netskope Data with Cribl Search

Once your Netskope data lands in Cribl Lake, your Netskope Dataset is available in Cribl Search as a `cribl_lake` Dataset. For details on querying your data, see Search Cribl Lake.

7.3. Zscaler Direct Access

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

Use the Zscaler Direct Access option to land Zscaler Nanolog Streaming Service (NSS) data directly into Cribl Lake for search and analysis, without manually managing infrastructure. Zscaler NSS streams your security event logs to a Cribl Lake ingest endpoint over HTTP, using the Splunk HTTP Event Collector (HEC) format. Events are parsed at ingest, so you can query top-level fields in Cribl Search without search-time extraction.

How Zscaler Direct Access Works

Zscaler Direct Access in Cribl consists of a few main steps. Here's an overview of the steps you'll complete in this guide:

1. Create a Lake Dataset with a `sourcetype` partition to hold the Zscaler data.
2. In Cribl Lake, create a Zscaler Direct Access Source and generate an auth token.
3. In Zscaler NSS, create a Cloud NSS Feed that points to the Cribl Lake ingest endpoint.

Learn About the Implementation

The Zscaler Direct Access implementation has the following details:

- You can configure one Zscaler Source per Cribl Lake Workspace.
- You can have one Dataset per Source.
- Data flows from Zscaler NSS into Cribl Lake over HTTP; it doesn't pass through Cribl Stream.
- Supported Zscaler NSS sourcetypes:
  - `zscalernss-web`
  - `zscalernss-fw`
  - `zscalernss-dns`
  - `zscalernss-tunnel`

How Events Are Parsed

When Zscaler NSS events arrive at the Direct Access ingest endpoint, Cribl:

- Sets `_time` from the event's own timestamp (`time` or `datetime`, depending on sourcetype), using the `local` timezone of the Cribl.Cloud Organization. For example, if the Organization is in `us-east-*`, events are stamped with Eastern Time.
- Promotes fields from the event payload to the top level (for example, `user`, `login`, `action`).

- Drops empty values such as `N/A`, `NA`, and `None`.
- Preserves `sourcetype` so you can distinguish web, firewall, DNS, and tunnel events in Search.

Create a Zscaler Dataset

First, define a Dataset with a `sourcetype` partition. Zscaler NSS identifies each feed type with a `sourcetype` value, and Cribl uses that field to organize and query the data.

1. In Cribl Lake, navigate to **Datasets**.
2. Select **New Dataset**.
3. Enter an ID.
4. Add a default retention period.
5. Select a data format. Selecting **Parquet** may result in smaller files, reducing total overall storage.
6. In **Advanced**, create a Lake Partition. Set the value to `sourcetype`.
7. Select **Save**.

Supported `sourcetype` values are `zscalernss-web`, `zscalernss-fw`, `zscalernss-dns`, and `zscalernss-tunnel`.

For more detailed information on Datasets, see Cribl Lake Datasets.

Create a Zscaler Direct Access Source

Create the Source in Cribl Lake so that Cribl can provision the ingest endpoint that Zscaler NSS writes to.

1. In Cribl Lake, select **Direct Access** in the left navbar.
2. On the **Source** step, select **Zscaler**, then select **Next**.
3. On the **Configuration** step, enter the required fields:
    1. Add Datasets in the **Direct Access Datasets** drop-down.
    2. In the **Auth tokens** section, select **Generate** under **Token**.
    3. Copy the token value; you'll paste it into the Zscaler Cloud NSS Feed configuration.
    4. Optionally, add a description and fields to events that reference the token.
    5. Select a **Default Dataset** this Source will reference.
4. Select **Save**. The Zscaler Source enters a `Provisioning` state while Cribl sets up infrastructure. Provisioning usually completes in a few minutes.
5. When the provisioning completes, open the Zscaler Direct Access Source to see the ingest **URL**.

Create a Cloud NSS Feed in Zscaler NSS

Set up a data stream in Zscaler NSS Cloud to send data to your Cribl Lake ingest endpoint. Before you start, ensure you have admin permissions in your Zscaler environment.

1. Log in to Zscaler.
2. In the left menu, select **Administration**, then **Nanolog Streaming Service** under **Cloud Configuration**.
3. Select the **Cloud NSS Feeds** tab, then select **Add Cloud NSS Feed**.
4. Enter the following Cloud NSS Feed configuration:
    - Add a **Feed Name**.
    - Set **SIEM TYPE** to **Splunk**.
    - Enter the **API URL** in the form `https://default.<workspace>.<cribl_cloud_name>.cribl.cloud:<port>/services/collector`, using the ingest URL from your Cribl Lake Zscaler Source.
    - Add two HTTP headers:
        - `Authorization` with the value of the token from the Cribl Lake Zscaler Direct Access Source.
        - `Content-Encoding` with the value `gzip`.
5. Scroll down, then select a **Log Type**.
6. Disable **JSON Array Notation**.
7. Select **Save**. The feed appears in the Cloud NSS Feed list.

Test the Connection

1. In Zscaler NSS, open the Cloud NSS Feed you created.
2. Select the **Test Connectivity** icon in the far-right column. You'll see a success message at the top of the console.
3. In Cribl Lake, confirm that the test event lands in your Zscaler Dataset.

Provisioning Lifecycle

A Zscaler Direct Access Source will have one of these statuses after you select **Save**:

Status	Description
`Provisioning`	Cribl is provisioning the ingest endpoint and deploying configurations.
`Ready`	The ingest endpoint is reachable. You can attach Datasets to this Source.

Status	Description
`Failed`	The provisioning timed out or hit an unrecoverable error. You must delete the Source and try again.

Query Zscaler Data with Cribl Search

Once your Zscaler data lands in Cribl Lake, your Zscaler Dataset is available in Cribl Search as a `cribl_lake` Dataset. Because events are parsed at ingest, you can filter on promoted fields and on `sourcetype` directly. For example:

```
dataset="your_zscaler_dataset" sourcetype="zscalernss-web"
| limit 100
```

Copy

For details on querying your data, see Search Cribl Lake.

Goat-to Resources

Cribl Stream and Zscaler Nanolog Streaming Service	Blog
Cribl Stream Zscaler NSS Integration	Doc
Zscaler: Understanding Nanolog Streaming Service	Doc
Zscaler: About Cloud NSS Feeds	Doc

7.4. Splunk Cloud Self Storage (DDSS) Direct Access

Bring your Splunk data straight into Cribl Lake, bypassing Cribl Stream processing.

You can use the Splunk Cloud Self Storage option to archive one or more Splunk Cloud indexes directly to Cribl Lake. This option bypasses routing or processing it 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 Splunk Cloud Direct Access

To set up this integration, you need a Splunk index (new or existing).

You can configure one Splunk Cloud Direct Access Source per Workspace.

See other Limitations later in this topic.

Configure Splunk Cloud Direct Access

To configure this integration, you'll work back and forth between Splunk Cloud and Cribl Lake to share linking identifiers. The major steps are:

1. In Splunk Cloud, get your Splunk Cloud ID.
2. In Cribl Lake, create a Cribl Lake Source.
3. In Splunk Cloud, connect your Cribl Lake bucket.
4. In Cribl Lake, provision your Lake.
5. In Splunk Cloud, test and submit the connection.
6. In Cribl Lake, create a Cribl Lake Dataset.

Get Your Splunk Cloud ID

You'll need to find and copy your **Splunk Cloud ID**, which is the first part of your Splunk Cloud URL and the 12-character S3 bucket string.

For example, if you administer Splunk Cloud Platform for Hungry Goats Computing, and your Splunk Cloud URL is `hungrygoatscomputing.splunkcloud.com`, then your Splunk Cloud ID is `hungrygoatscomputing-{12-character s3 string}`.

You can retrieve the AWS bucket prefix using one of the following methods:

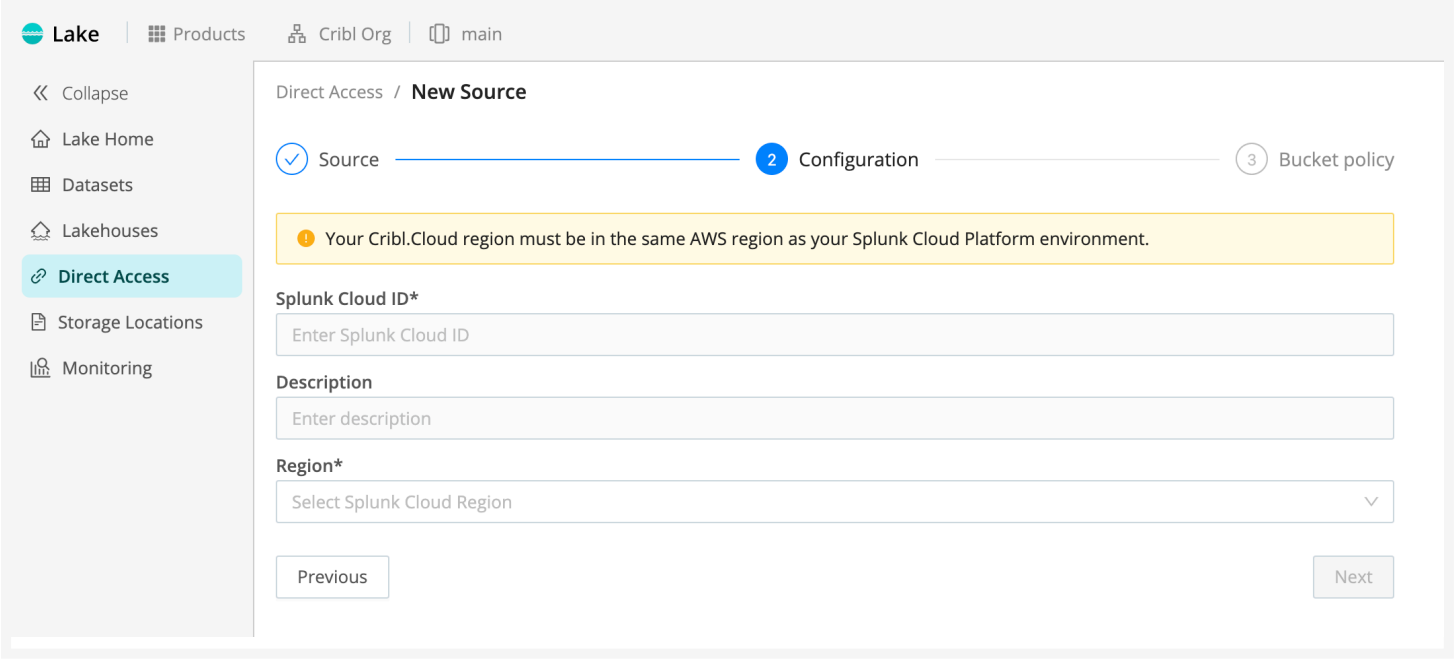
API Splunk UI

Use the Splunk Cloud Platform **Admin Config Service (ACS)** API to manage DDSS self storage locations programmatically. For endpoint and payload details, see the Splunk docs on how to manage DDSS self storage locations in their Admin Config Service manual.

Create Cribl Lake Source

In Cribl Lake, create the integration with your Splunk Cloud instance:

1. From the sidebar, select **Direct Access**.
2. Select the **Splunk Cloud Self Storage** Source, and select **Next**.
3. In the **Splunk Cloud ID** field, paste your 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.



Configuring Cribl Lake matching storage location

Connect Splunk Cloud to Your Cribl Lake Storage Location

Next, return to your Splunk Cloud Platform, to point this environment to the Storage Location 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.

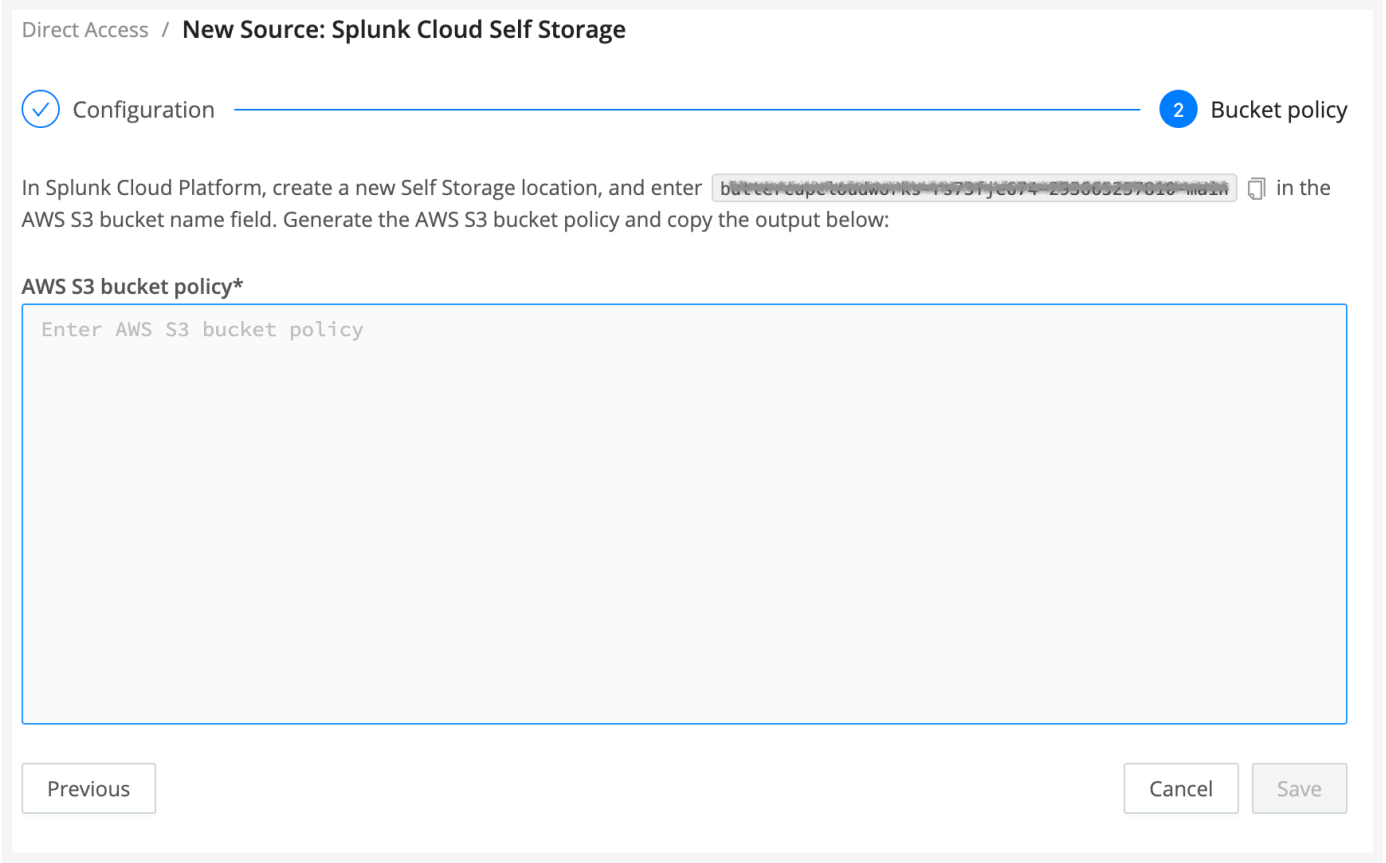
3. Copy the resulting bucket policy to your clipboard for the next section.

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.

   Cribl provisions the underlying object storage for you during this step. Cribl-managed Lake storage is encrypted at rest by default.



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 verify that Splunk Cloud Platform has permissions to write to Cribl Lake.

1. If you see a success message, select **Submit** to finish linking your Self Storage location to Cribl Lake.
2. 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. Each Splunk index maps to a single Lake Dataset, and no additional Search dataset setup is required. 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 your Splunk index name from Set Up Splunk Self Storage Location. (Otherwise, you will be unable to search or replay your data.)
- Set the **Storage Location** to the default `Cribl Lake`, or make no selection on this drop-down.
- Set the **Data format** to `DDSS`.

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

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.

Limitations of Splunk Cloud Direct Access

A Lake Dataset created and populated via Splunk Cloud 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 populated via Direct Access must use the default `Cribl Lake` Storage Location, not an external bucket.

A Lake Dataset created via Splunk Cloud Direct Access can't be assigned to a Lakehouse.

For Cribl-managed storage, retention is enforced at the Dataset level only.

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.

8. Learn About Storage Locations (Bring Your Own Storage)

Create Lake Datasets on storage that you directly own, achieving compliance while taking advantage of streamlined management in Cribl Lake.

You can host Cribl Lake Datasets on **Amazon S3** or **Azure**. Storage Locations are available on an Enterprise billing plan. For details, see Cribl Pricing.

Prerequisite: Configure a Storage Location by Provider

First, you need to configure at least one Storage Location. Choose a provider and follow the steps to configure the Storage Location. Then, return to this topic to connect Lake Datasets.

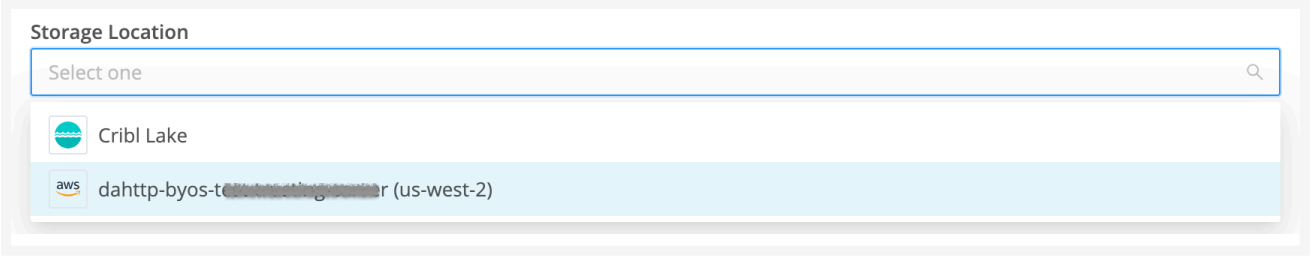
- Configure an Amazon S3 Storage Location
- Configure an Azure Storage Location

Connect Lake Datasets to Storage Locations

You can link multiple **Connected Datasets** per Storage Location.

Follow the procedure in Manage Lake Datasets to create a new Dataset, or to edit an existing Dataset. On the Dataset configuration page, select a configured **Storage Location** from the drop-down. Then save or resave the Dataset.

If you don't make a selection on this drop-down, the Dataset will use the internal `Cribl Lake` default location.



Connecting a Dataset to a Storage Location

The Dataset will now use the selected (or default) Storage Location to store its data. On the **Storage Location** page, this Dataset will appear in the right column of **Connected Datasets**.

As with other Datasets, you can write to this Dataset from Cribl Stream with the Cribl Lake Destination. And you can access the Dataset contents from Cribl Search and Stream (subject to some Limitations).

The **Retention period** that you set in the Cribl Lake Dataset configuration is reflected in your cloud provider's object lifecycle or retention UI where applicable. After you've connected a Dataset to an external bucket, avoid changing retention outside Cribl Lake in ways that conflict with Lake Dataset lifecycle expectations.

Detach a Storage Location Connection

If you choose to detach a Storage Location, all associated Datasets will also be removed from Cribl Lake. However, your data will not be deleted from the underlying bucket or storage container.

For steps to detach, see your storage location provider:

- Detach an AWS Storage Location
- Detach an Azure Storage Location

API Support for Storage Locations

For automation and scripting, Members with appropriate Permissions can use Cribl `storageLocations` REST API endpoints. See Amazon S3 storage locations for an example request body.

Limitations on Storage Locations

- Each Storage Location must map to one bucket or equivalent container in the target cloud.
- You can't populate an external storage container/bucket using Cribl Lake Direct Access.
- You can't assign storage containers/buckets for these Connected Datasets to Lakehouses.
- Data sent to an external storage container/bucket through the Cribl Lake Destination will be accessible to Cribl Search, and to Cribl Stream for replay through its Cribl Lake Collector. However, any data that you send directly to the storage container/bucket outside of Cribl products will not be visible to Cribl Search or Stream.
- Provider-specific limits (supported regions, authentication, and setup paths) are documented on Amazon S3 storage locations and Azure storage locations.

8.1. Configure An Amazon S3 Storage Location

This topic explains how to configure Amazon S3 as a Cribl Lake Storage Location.

Use an Amazon S3 bucket as a Storage Location for Cribl Lake Datasets so you keep direct ownership of your data for regional or compliance purposes, while using Cribl Lake provisioning, retention, access control, and analytics.

For connecting Datasets, detaching Storage Locations, API overview, and general BYOS limits, see Storage Locations.

Using any S3 bucket with a Cribl Lake Storage Location will incur AWS charges in addition to Cribl Lake credits.

Configure an Amazon S3 Storage Location

When you follow the recommended path in this section, Cribl Lake will simplify creating a **new** bucket on Amazon S3. After you configure initial settings in the Lake UI, Cribl will provide a ready-made CloudFormation template that you can download, then launch in the AWS CloudFormation console to create your infrastructure-as-code.

If you prefer to configure the bucket yourself on S3, see Setting Up a Dedicated Bucket Outside Cribl Lake.

Each Storage Location maps to one bucket. So you'll repeat the configuration steps for each external bucket you want to create.

Select a Storage Provider

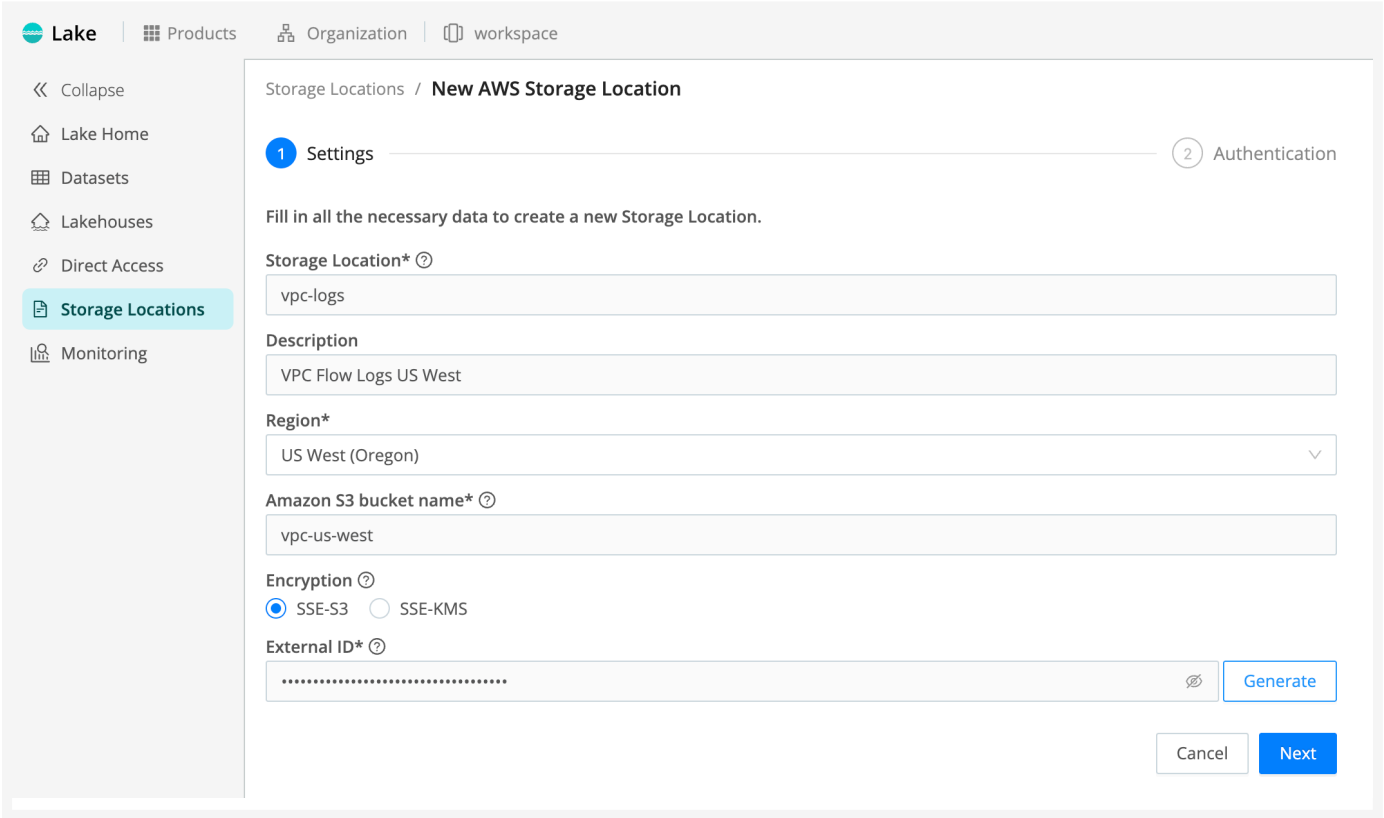
1. Navigate to **Lake** > **Storage Locations** > **New Storage Location**.
2. Select **AWS**, then **Next.**

Configure Settings

Setting	Description	Example
Storage Location	A name for your storage location. This is arbitrary, and should be unique in your Cribl Lake instance.	`aws-storage`

Setting	Description	Example
Description	Optional describer that helps other users understand the purpose of this Storage Location.	Data stored in AWS S3.
Region	Select a supported AWS Region in which to create the bucket.	US West (Oregon)
Bucket Name	Enter a name for the S3 bucket. The **Bucket Name** is arbitrary, but must be unique on AWS. We recommend giving your buckets a distinctive identifier. (For detailed naming recommendations, see General Purpose Bucket Naming Rules from AWS.)	`us-west-demo`
Encryption	Optionally customize the **Encryption** method on your bucket. It defaults to `SSE-S3` for Amazon S3–managed keys; you can instead select `SSE-KMS` for AWS Key Management Service (KMS) for enhanced lifecycle control (key rotation and revocation), auditability, and IAM policy customization through AWS interfaces. For details, see Using Server-Side Encryption with AWS KMS Keys (SSE-KMS) from AWS.	`SSE-S3`
External ID	Generate or enter an **External ID**. The **External ID** is arbitrary, with no uniqueness requirement. However, to provide an extra layer of security, Cribl recommends using a random UUID here. You can use the **Generate** button for this purpose. For details about external IDs, see How to Use External ID When Granting Access to Your AWS Resources from AWS.	`a1b2c3d4-e5f6-7890-abcd-ef1234567890`

When you finish the **Settings** form, select **Next** to open the **Authentication** page.

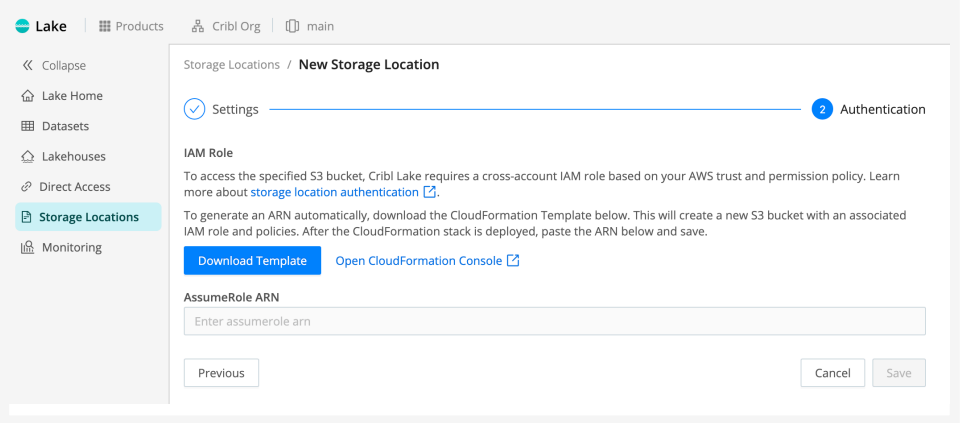


Configure Authentication and Infrastructure

From **New Storage Location > AWS > Authentication** in Cribl Lake, you create your Amazon S3 infrastructure, and you then authenticate Cribl Lake on it.

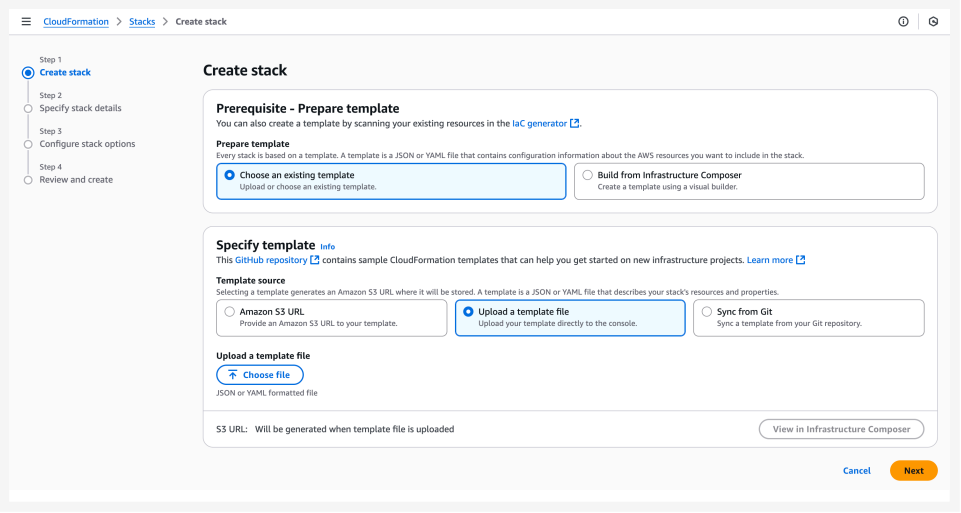
For details about required AWS permissions, see Authentication.

1. Select **Download Template** and save the resulting CloudFormation template locally.
2. Select **Open CloudFormation Console**.



Screenshot of Authentication page, the second step in configuring a new Storage Location in Cribl Lake.

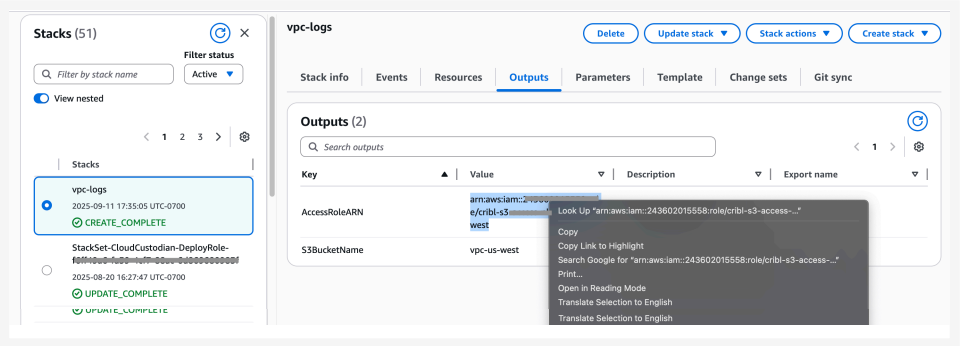
3. On the resulting AWS CloudFormation **Stacks** > **Create stack** page, select **Upload a template file**, then **Choose file**.



Screenshot of AWS CloudFormation “Create stack” page, showing selected radio button to upload template from Cribl Lake.

4. Upload the template you saved, then select **Next**.
5. In the resulting CloudFormation **Configure stack options** sequence, accept defaults, or adjust them to match your AWS practices. Then select **Next**.
6. On the **Specify stack details** page, assign your stack an arbitrary name. Then complete the **Configure stack options** sequence, accepting defaults and acknowledgments as appropriate.
7. On the **Review and create** page, select **Submit**.

   On the CloudFormation **Stacks** page, you will now see your new bucket progress from `CREATE_IN_PROGRESS` status to `CREATE_COMPLETE`.
8. Select the **Outputs** tab, and copy your new bucket's **AccessRoleARN** value to the clipboard.



Screenshot of newly created AWS stack's Outputs tab, showing how to copy the bucket's AccessRole ARN value.

9. Back on the Cribl Lake **Authentication** page, paste this ARN into the **AssumeRole ARN** field.
10. Select **Save** to link your Cribl Lake Storage Location with your S3 bucket.

    Your new Storage Location will appear on the left side of the Lake **Storage Locations** page.

> The Retention period that you set in the Cribl Lake Dataset configuration will automatically be reflected in the Amazon S3 Console's **Management** tab. After you've connected a Dataset to an external S3 bucket, do not change the retention period directly in the S3 Console.

When you attach a Dataset to this Storage Location, you can also select an S3 storage class (Standard, Intelligent-Tiering, Glacier, and others) to control how Cribl Lake stores objects in your bucket. For details, see Storage Class for BYOS Datasets.

Supported AWS Regions

Each Amazon S3 Storage Location can map to a bucket in one of the following AWS Regions:

- US East (Virginia)
- US East (Ohio)
- US West (Oregon)

- Canada (Central)
- Europe (Frankfurt)
- Europe (London)
- Europe (Zurich)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)

FedRAMP-Supported Regions

In Cribl.Cloud Gov (FedRAMP Moderate), Cribl Lake storage locations that use Amazon S3 support the following AWS regions:

- `us-east-1` – US East (N. Virginia)
- `us-east-2` – US East (Ohio)
- `us-west-2` – US West (Oregon)

Cribl.Cloud Government currently does not support AWS GovCloud regions (such as `us-gov-west-1`, `us-gov-east-1`) for BYOS storage locations.

Detach an AWS Storage Location

To detach this storage location:

1. Remove any Datasets with connected Collectors and Destinations.
2. Open the Storage Location and select **Detach**.
3. Select the checkbox and type **DETACH** in the text field.
4. Go to your AWS S3 account and revoke Cribl Lake permissions for the bucket.

API Support for Amazon S3 Storage Locations

For automation and scripting, Members with appropriate Permissions can use Cribl `storageLocations` REST API endpoints. However, in this case, Cribl does not generate a CloudFormation template. So you will need to explicitly create corresponding S3 buckets and IAM roles with appropriate trust policy and permissions. (See Authentication for required permissions.)

Create a Storage Location via API

Here is an example request to create a Storage Location:

```
POST /products/lake/lakes/{lakeId}/storage-locations
```

Copy

Your request body should include the Storage Location details. Here's an example:

```
{
  "id": "byosLocation",
  "provider": "aws-s3",
  "credentials": {
    "method": "auto",
    "roleToAssume": "arn:aws:iam::1111111111:role/test"
  },
  "config": {
    "bucketName": "byosbucket",
    "region": "us-west-2"
  }
}
```

Copy

Attach a Dataset via API

When adding a Dataset via API, you reference the target Storage Location instead of a bucket name. Here is an example YAML configuration:

```
datasets:
  - id: my_dataset
    storageLocation: my-byos-location
    retention: 30d
    format: json
    storageClass: STANDARD
```

Copy

Set Up a Dedicated Bucket Outside Cribl Lake

If you follow the procedure in Configure an Amazon S3 Storage Location, the provided CloudFormation template will create your new Amazon S3 bucket's structure for you.

For advanced users who prefer to manage AWS resources directly, you can create a **new**, **dedicated** bucket and IAM role on S3, then link it as a Cribl Lake Storage Location. This is a more complex path, which requires you to explicitly configure access control on AWS.

Using the AWS Management Console or AWS API, you would:

1. Create a simple bucket configuration in an AWS Region that Cribl Lake supports.
2. Create a corresponding IAM role with an appropriate trust policy and permissions. (See Authentication for required permissions.)

Do not connect an **existing** S3 bucket that contains other data. Cribl Lake requires full control over lifecycle rules and S3 Inventory configuration, so each Cribl Lake Storage Location must be backed by a bucket dedicated exclusively to Cribl Lake.

Authentication

Cribl Lake accesses Amazon S3 buckets using STS Assume Role. If you follow the procedure in Configure an Amazon S3 Storage Location, the provided CloudFormation template will create the required resources and permissions for you.

IAM Role Permissions

Read the remainder of this section only if you choose to configure access control on an S3 bucket yourself (see Setting Up a Dedicated Bucket Outside Cribl Lake). These instructions assume that you are permitted to view and manage access details on both AWS and your Cribl Organization.

Your IAM role must provide the permissions listed in the following example. Replace the `<bucketName>` placeholder with your own bucket's name:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "BucketAdminPermissions",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/Actor": "admin"
        }
      },
      "Action": [
        "s3:PutLifecycleConfiguration",
        "s3:GetLifecycleConfiguration",
        "s3:PutInventoryConfiguration",
        "s3:GetInventoryConfiguration",
        "s3:PutBucketNotification",
        "s3:GetBucketNotification",
        "s3:GetEncryptionConfiguration",
        "s3:GetBucketTagging"
      ],
      "Resource": "arn:aws:s3:::<bucketName>",
      "Effect": "Allow"
    },
    {
      "Sid": "BucketObjectLevelReadWritePermissions",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:GetObjectTagging",
        "s3:PutObjectTagging",
        "s3:AbortMultipartUpload"
      ],
      "Resource": "arn:aws:s3:::<bucketName>/*",
      "Effect": "Allow"
    },
    {
      "Sid": "BucketLevelReadPermissions",
      "Action": ["s3:ListBucket", "s3:GetBucketLocation", "s3:ListBucketMultipartUplo
      "Resource": "arn:aws:s3:::<bucketName>",
      "Effect": "Allow"
    }
  ]
`
```

```
}
```

Copy

Also, for Cribl resources to be able to assume your IAM role, the IAM role must provide `sts:AssumeRole` and `sts:TagSession` permissions to these Cribl roles:

- `role/saas/cribl-lake-admin-<workspaceId>` (manages your bucket properties, like lifecycle and inventory).
- `role/<workspaceId>-*` (Workers from all Cribl Stream Worker Groups need read/write access to your bucket).
- `role/search-exec-<workspaceId>` (Cribl Search needs read/write access to be able to perform searches against your bucket).

IAM Role Trust Policy

If you choose to configure access control on an S3 bucket yourself, you must provide a trust policy comparable to the following example. (See also the Restrictive Trust Policy option.) Replace the `<CRIBL_AWS_ACCOUNT>`, `<workspaceId>`, and `<externalId>` placeholders as follows:

1. In your Cribl.Cloud Organization, select **Products** from the top bar, then select **Workspace** from the sidebar.
2. To insert the `<CRIBL_AWS_ACCOUNT>`, select **Trust** from the sidebar. Then, from the displayed **Worker ARN** field, copy the numeric value that follows the string `arn:aws:iam::`.
3. To insert the `<workspaceId>`, select the Workspaces button on the top bar. In the resulting modal, copy the **ID** of the appropriate Workspace.
4. To insert the `<externalId>`, switch to your Storage Location config, and copy the **External ID** value you defined there. (For details, see Configure Initial Settings.)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CriblLakeAdminAssumeRole",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<CRIBL_AWS_ACCOUNT>:role/saas/cribl-lake-admin-<workspac
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "<externalId>",
          "aws:RequestTag/Actor": "admin"
        }
      }
    },
    {
      "Sid": "CriblLakeAdminTagSession",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<CRIBL_AWS_ACCOUNT>:role/saas/cribl-lake-admin-<workspac
      },
      "Action": "sts:TagSession",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/Actor": "admin"
        }
      }
    },
    {
      "Sid": "CriblWorkerAssumeRole",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<CRIBL_TRUST_ACCOUNT>:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "<externalId>"
        },
        "StringLike": {
```

```
          "aws:PrincipalArn": "arn:aws:iam::<CRIBL_AWS_ACCOUNT>:role/<workspaceId>-*"
        }
      }
    },
    {
      "Sid": "CriblSearchAssumeRole",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<CRIBL_AWS_ACCOUNT>:role/search-exec-<workspaceId>"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "<externalId>"
        }
      }
    }
  ]
}
```

Copy

Restrictive Trust Policy

The preceding model trust policy enables all Cribl Stream Worker Groups in the current Workspace to write to the configured bucket. If you instead wish to limit write permissions to only certain Worker Groups, modify the model policy's `CriblWorkerAssumeRole` element as shown in the following listing.

8.2. Configure An Azure Storage Location

This topic explains how to configure Microsoft Azure as a Cribl Lake Storage Location.

You can set an Azure Blob Storage container as the Storage Location for a Cribl Lake Dataset. This enables you to keep your data stored in Azure, but be able to search it in Cribl as a Dataset.

To connect Datasets, detach Storage Locations, and general BYOS limits, see Learn About Storage Locations.

Why Use an Azure Storage Location

By adding an Azure Storage Location to your Cribl Lake, you can:

- Keep log and security data in your own Azure storage account for compliance, but use Cribl for search and analytics over that data.
- Use Cribl Lake and Cribl Search to turn low-cost Azure Blob storage into a searchable archive.

Before You Begin

Complete these steps before configuring an Azure Storage Location in Cribl:

1. Log in to your Azure tenant and navigate to **Object storage** > **Blob Storage**.
2. Create a Storage Account and Blob container.
3. Go to **Access Control (IAM)**, then:
    1. If you want customer-managed encryption keys, create an **Azure Key Vault** and key.
    2. Assign RBAC roles at the right scopes:
        - `Storage Blob Data Contributor` at the Container.
        - `Storage Blob Data Owner` at the Storage Account.
        - `Key Vault Crypto User` at the Key Vault.

Configure an Azure Storage Location

By following the recommended setup in this section, Cribl Lake will simplify creating a **new** Azure Blob Storage location for BYOS Lake Datasets.

Select a Storage Provider

1. Navigate to **Lake** > **Storage Locations** > **New Storage Location**.
2. Select **Azure**, then **Next.**

Configure Settings

Setting	Description	Example
Storage Location	A unique name for this storage location in Cribl Lake.	`azure-lake-byos`
Description	An optional description for this storage location.	`Azure Storage in Cribl Lake.`
Resource Group Name	The name of the resource group that will contain the storage account. Resource groups are logical organization structures for containers.	`azure-blobs`
Storage Account Name	The storage account.	`azure`
Region	The Azure region where your storage account resides. You can select any publicly available Azure region. Choose the region that matches your data residency and compliance requirements, because a storage account's region is fixed when the account is created and can't be changed afterward.	`West US 2`
Tenant ID	The ID of your Microsoft Entra directory, which is a string of hyphen-separated numbers and letters.	`aa1b23c4-5d67-890e-ab12-345c67de8906` See Microsoft's documentation on How to Find your tenant ID.
Subscription ID	The ID associated with your subscription agreement with Microsoft to use their cloud services.	`aaaa0a0a-bb1b-cc2c-dd3d-eeeeee4e4e4e` See Microsoft's documentation on How to find your Azure subscription.

Authenticate the Storage Account Access

After you configure the initial settings, you will authorize Cribl.Cloud in your Microsoft account, then generate an ARM template that you can download. Finally, launch the template in the Azure Data Storage console to create your infrastructure-as-code.

1. Select **Authorize Cribl.Cloud** to launch a new window and sign into your Microsoft account. This creates an enterprise application for your Cribl.Cloud organization in your Microsoft Azure account. For details about what admin consent grants and does not grant, see About Azure Authentication for Cribl Lake.
2. Once consent is confirmed, select **Generate ARM Template**, which downloads a JSON file containing the required IAM policies and roles. This will automatically provide Cribl Lake with access when you upload the JSON file to the Azure Storage account.

About the ARM Template

When you create an Azure Storage Location in Cribl Lake, the system generates a customized Azure Resource Manager (ARM) template for your environment. Deploying this template in Azure is a required step in the Cribl Lake Azure BYOS onboarding flow.

To learn about the ARM template, the resources it creates, storage account configuration, role assignments, inventory policy, and detailed steps on deploying it, see BYOS Azure ARM Template Reference.

Deploy the Custom Template in Microsoft Azure

To deploy the template in Azure:

1. In Microsoft Azure, navigate to **Deploy a Custom Template**.
2. Select **Build your own template in the editor**.
3. Select **Load file** and load the `.json` ARM template file you downloaded from Cribl.
4. Select the target **Subscription** and **Resource Group**.
5. Confirm the template loaded in Azure properly, then select **Save**.

To deploy the template in the command line:

Azure CLI

```
az deployment group create \
  --resource-group <your-resource-group> \
  --template-file CriblLakeByosAzureArmTemplate.json
```

Copy

PowerShell

```
New-AzResourceGroupDeployment `
  -ResourceGroupName <your-resource-group> `
  -TemplateFile CriblLakeByosAzureArmTemplate.json
```

Copy

Validate and Save the Azure Storage Location

Return to Cribl and move to Step 3: Confirm Template Deployment.

1. Check the box to tell Cribl that you've deployed the ARM template in Microsoft Azure.

2. Select **Save**.

Cribl will validate the configuration and connection to your Azure Storage account. At this step, Cribl will display any missing permissions if they are present.

When the connection is successful, you'll see the Tenant ID in the list of Storage Locations.

When you attach a Dataset to this Storage Location, you can also select an Azure access tier (Inferred, Hot, Cool, Cold, or Archive) to control how Cribl Lake stores objects in your container. For details, see Storage Class for BYOS Datasets.

Supported Azure Regions

You can select any publicly available Azure region when you create an Azure Storage Location. There is no fixed subset of supported regions.

Select the region where your storage account already resides. Because the region of a storage is set when you create the account, and can't be changed afterward, plan your region choice around your data residency, latency, and compliance requirements. To keep data in its region of origin, create a storage account, along with its Cribl Lake Storage Location, in each region where you need to land data.

When you save the Storage Location, Cribl validates that your storage account exists in the region you selected. If the storage account isn't in that region, Cribl reports an error so you can correct the selection before finishing setup.

Existing Storage Locations in previously supported regions are unaffected.

FedRAMP Support

FedRAMP deployments of Cribl.Cloud can't use Cribl Lake Azure storage locations.

Detach an Azure Storage Location

To detach this storage location:

1. Remove any Datasets with connected Collectors and Destinations.
2. Open the Storage Location and select **Detach**.
3. Select the checkbox and type **DETACH** in the text field.
4. Select **Detach**. Next, revoke Cribl access in Azure by following the steps in the Azure documentation: Delete an enterprise application.
5. In Cribl, confirm you revoked access in Azure.

Attach a Dataset via API

When you add a Dataset via API, you'll reference the target Storage Location instead of a container name. You can set the Azure access tier with the `storageClass` field. Valid values are `Inferred`, `Hot`, `Cool`, `Cold`, and `Archive`. If you omit the field, the Dataset defaults to `Inferred`. Here is an example YAML configuration:

```
datasets:
  - id: my_dataset
    storageLocation: my-azure-byos-location
    retention: 30d
    format: json
    storageClass: Cool
```

Copy

Next Steps

Now that you've successfully created an Azure Storage Location, you can attach a Dataset to it. See Connect Datasets to Storage Locations for details.

8.3. About Azure Authentication For Cribl Lake

Learn how Cribl Lake authenticates to your Azure Blob Storage using a Cribl-managed Enterprise Application.

IT and security teams can use this topic when evaluating the Azure admin consent step in Configure an Azure Storage Location.

Summary

Aspect	Detail
Authentication method	AWS IAM Outbound Identity Federation + Azure Workload Identity Federation
Credential type	Short-lived federated tokens (no stored secrets)
Admin consent	One-time. Creates a Service Principal in your tenant
Entra ID access	Token exchange only. No management API or Graph access
Storage access	Scoped to explicitly assigned storage accounts via ARM template RBAC
Revocation	Remove RBAC roles or delete the Enterprise Application

What Is the Cribl Lake Enterprise Application?

To read and write data in your Azure Blob Storage containers, Cribl Lake uses AWS IAM Outbound Identity Federation with Azure Entra ID Workload Identity Federation. This is a standard cross-cloud authentication pattern for organizations that run infrastructure on both AWS and Azure.

Cribl maintains a multi-tenant Azure App Registration in the Cribl Entra ID tenant. When your administrator approves the admin consent URL, Azure creates a Service Principal (a machine identity) in your tenant. That Service Principal:

- Acts as the local representation of the Cribl application in your directory.
- Does not sign in interactively.
- Does not appear as a user in your directory.
- Holds only the RBAC assignments you configure with the Cribl ARM template.

Microsoft recommends this pattern for SaaS applications and cross-cloud integrations that authenticate to Azure with federated credentials.

What Admin Consent Does

Admin consent creates a Service Principal in your tenant. It does not grant storage access. Storage access comes only from the RBAC roles you assign when you deploy the ARM template.

When an admin approves the consent URL:

- Azure creates a Service Principal in your tenant linked to the Cribl App Registration.
- The Service Principal is registered as a Workload Identity, not a user identity.
- No permissions are granted at this step. You configure RBAC separately with the ARM template.

Consent is a one-time action. You do not need administrator intervention again unless you remove and re-add the application.

Required Azure Roles

Creating the Service Principal in your tenant requires one of these roles:

- Global Administrator
- Privileged Role Administrator
- Cloud Application Administrator (if configured in your tenant's consent policy)

What Admin Consent Does Not Grant

Approving the Cribl Lake Enterprise Application does not grant:

Concern	Status
Access to your Entra ID instance (users, groups, policies)	Not granted
Microsoft Graph API access (mail, Teams, SharePoint)	Not requested
Delegated user permissions (acting on behalf of any user)	Not used
Modification of Azure AD configuration or security policies	Not possible
Access to Azure Key Vault, Azure SQL, or non-storage services	Not granted
Standing administrative access to your Azure environment	Not granted
Access to any storage account not explicitly assigned via ARM template	Not possible

How Authentication Stays Scoped

Workload Identity Federation is a credential exchange mechanism, not an API access grant. Entra ID acts as a token broker, not as a resource that Cribl accesses.

Approving the Cribl Enterprise Application does not open new endpoints, network paths, or public API surfaces in your Azure environment. The token exchange endpoint (`https://login.microsoftonline.com/{tenant-id}/oauth2/v2.0/token`) is a pre-existing, Microsoft-hosted public endpoint available for every Entra ID tenant. It is part of the Microsoft identity platform and OAuth 2.0 authorization code flow.

Adding the Cribl Enterprise Application:

- Does **not** create a new public endpoint.
- Does **not** enable inbound network connectivity to your tenant.
- Does **not** expose your Entra ID instance beyond Microsoft's existing infrastructure.
- Does **not** grant Cribl access to Microsoft Graph or any Entra ID management surface.

In practice:

- The Cribl Service Principal authenticates with a federated credential (an AWS-signed JWT), not a client secret or certificate stored in your tenant.
- The only Entra ID interaction is the token exchange endpoint (`/oauth2/v2.0/token`), which validates the JWT and returns a scoped Bearer token. That endpoint is the standard OAuth 2.0 token endpoint, not a management API.
- The Bearer token grants access only to Azure Blob Storage on storage accounts where you assigned RBAC roles. It cannot call Microsoft Graph, Entra ID management APIs, or other Azure services.
- No client secret or certificate is stored in your Entra ID instance. The federated identity credential references an external AWS OIDC issuer with a case-sensitive subject and audience match.
- Before issuing a token, the federated credential validates three claims: **issuer** (AWS OIDC provider URL), **subject** (specific Cribl IAM role ARN), and **audience** (`api://AzureADTokenExchange`). All three must match.

Microsoft has documentation on federated identity credentials which describes this pattern as eliminating the maintenance burden of manually managing credentials and the risk of leaking secrets.

How Authentication Works

Cribl Lake runs on AWS infrastructure. When it needs to access your Azure Blob Storage:

1. **Cribl obtains a short-lived identity token** from AWS Security Token Service (STS). This signed JWT is valid for about 15 minutes.
2. **Cribl exchanges the JWT with Azure Entra ID** for a Bearer access token. Azure validates the token against the federated identity credentials on the Cribl App Registration. The Bearer token is valid for about 1 hour.

3. **Cribl generates a scoped User Delegation SAS token** with the Bearer token. The SAS token is limited to the configured storage account and containers.
4. **Workers use the SAS token** to read and write blobs directly to your storage. Cribl does not store long-term Azure credentials.

Permissions Granted via the ARM Template

After admin consent, deploying the BYOS Azure ARM Template assigns RBAC roles to the Service Principal:

Role	Scope	Purpose
Storage Blob Data Contributor	Storage Account	Read/write blob data in configured containers
Storage Account Contributor	Storage Account	Manage lifecycle policies and inventory configuration

These roles are scoped to the storage account created by the ARM template. They do not provide access to other resources in your Azure subscription.

Revoke Access

To remove Cribl access:

1. Remove the RBAC role assignments from the storage account. Data access ends immediately.
2. Delete the Enterprise Application from your Entra ID tenant to remove the Service Principal.

Both actions are under your control and take effect immediately. For related steps in Cribl Lake, see Detach an Azure Storage Location.

Goat-to Resources

Configure an Azure Storage Location — Doc

BYOS Azure ARM Template Reference — Doc

Microsoft: Federated Identity Credentials Overview — Doc

Microsoft: Application Objects and Service Principals — Doc

Microsoft: Identity Platform OAuth 2.0 Token Endpoint — Doc

8.4. BYOS Azure ARM Template Reference

This topic is a reference guide for the Azure ARM template generated during Azure BYOS setup.

Overview

When you create an Azure Storage Location in Cribl Lake, the system generates a customized Azure Resource Manager (ARM) template for your environment. This template provisions the Azure infrastructure required for Cribl Lake BYOS, including the storage account, inventory reporting, lifecycle management, and role assignments for the Cribl service principal.

Deploying this template is a required step in the Azure BYOS onboarding flow. It ensures your Azure environment is configured correctly before Cribl Lake begins writing or reading data.

Prerequisites

Before deploying the ARM template, ensure you have:

- An active Azure subscription.
- A resource group in the Azure region where you want to store data.
- Permissions to create storage accounts and assign roles within the target resource group (typically Owner or User Access Administrator + Contributor).
- The ARM template JSON file downloaded from the Cribl Lake UI. See Authenticate the Storage Account Access.

The resource group location determines where the storage account is created. Choose a region that aligns >with your data residency requirements.

Resources Created

The template creates the following resources in your Azure subscription:

Resource	Type	Purpose
Storage account	`Microsoft.Storage/storageAccounts`	Primary storage for Cribl Lake Datasets

Resource	Type	Purpose
Blob service	`Microsoft.Storage/storageAccounts/blobServices`	Configures blob storage settings
Inventory container	`Microsoft.Storage/storageAccounts/blobServices/containers`	Stores daily inventory reports used for billing
Inventory policy	`Microsoft.Storage/storageAccounts/inventoryPolicies`	Generates daily Parquet reports of all blobs
Lifecycle policy	`Microsoft.Storage/storageAccounts/managementPolicies`	Auto-deletes inventory reports after 7 days
Role assignment (data)	`Microsoft.Authorization/roleAssignments`	Grants Cribl read/write access to blob data
Role assignment (management)	`Microsoft.Authorization/roleAssignments`	Grants Cribl management access to the storage account

Storage Account Configuration

The storage account is created with the following security settings:

- **SKU**: Standard_LRS (locally redundant storage)
- **Kind**: StorageV2
- **Minimum TLS version**: 1.2
- **Public blob access**: Disabled
- **HTTPS only**: Enabled
- **Access tier**: Hot
- **Encryption**: Enabled (Microsoft-managed keys)

The template does not create a data container for your Datasets. Cribl Lake creates Dataset containers automatically when you configure Datasets against this storage location.

Role Assignments

The template assigns two Azure RBAC roles to the Cribl service principal:

Role	Scope	Purpose
Storage Blob Data Contributor	Storage account	Read, write, and delete blob data (Dataset operations)
Storage Account Contributor	Storage account	Manage storage account settings, lifecycle policies, and inventory configuration

The service principal ID is unique to your Cribl Workspace and is embedded in the template when you download it. This principal is authorized in your tenant when you visit the consent URL during onboarding.

Do not modify the `servicePrincipalObjectId` value. This is the Cribl-managed identity that your Workspace uses to access your storage.

Inventory Policy

The inventory policy generates a daily Parquet report containing metadata for all block blobs in the storage account:

- **Schedule**: Daily
- **Format**: Parquet
- **Fields captured**: `Name`, `Content-Length`, `Creation-Time`
- **Destination**: `inventoryreports` container
- **Exclusions**: Blobs within the `inventoryreports/` prefix (avoids self-referencing)

Cribl uses these inventory reports to meter storage usage for billing. The reports are stored in your storage account — the billing pipeline reads them from there.

The first inventory report may take 24–48 hours to appear after deployment. Azure may skip a daily run if the previous report is still being generated.

Inventory Cleanup

A lifecycle management policy automatically deletes inventory reports older than 7 days.

Outputs

The template returns the following values after successful deployment:

Output	Description	Used For
storageAccountName	Name of the created storage account	Cribl Lake storage location configuration
subscriptionId	Azure subscription ID	Azure SDK authentication
resourceGroupName	Resource group name	Azure SDK authentication
tenantId	Azure AD tenant ID	Reference and validation

These values are used by Cribl Lake to connect to your storage account. If you deploy via the Azure Portal, you can find them in the deployment Outputs tab.

Safe to Customize

- **Resource group and region**: Deploy to any resource group in a supported Azure region.

Do Not Modify

- **Service principal object ID**: Must match your Cribl Workspace identity.
- **Role assignments**: Required for Cribl Lake to operate. Removing or narrowing these will cause connectivity failures.
- **Inventory policy**: Required for billing - disabling this will prevent Cribl from metering usage.
- **Storage account name**: Generated by Cribl to match your storage location configuration.

Security Considerations

The template follows Azure security best practices:

- **No public blob access**: All blob access requires authentication.
- **TLS 1.2 enforced**: Connections using older TLS versions are rejected.
- **HTTPS only**: Unencrypted HTTP traffic is not accepted.
- **Encryption at rest**: All blob data is encrypted with Microsoft-managed keys by default.

- **Scoped permissions**: Role assignments are scoped to the storage account, not the subscription or resource group.

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.

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. Configuration Files

Configuration files store the configuration changes locally on your system. They reflect the changes you make to configuration settings in the UI.

You can preview how settings are translated into configuration changes in the Git Changes modal whenever you commit your changes via the Version Control option.

Configuration file/folder	Description
`lake-config.yml`	Configuration for Cribl Lake.
`lakes.yml`	Configuration for individual Cribl Lake datasets.

11. Troubleshooting

View known issues, filtered for Cribl Lake. Check whether bugs are resolved, discover fix versions, and find temporary workarounds for open issues.