Google Cloud

Advanced Logging and Analysis

In this module, we examine some of Google Cloud's advanced logging and analysis capabilities.

Objectives

01 Use Log Explorer features

02 Explain the features and benefits of log-based metrics

03 Define log sinks (inclusion filters and exclusion filters)

04 Explain how BigQuery can be used to analyze logs

05 Use Log Analytics on Google Cloud

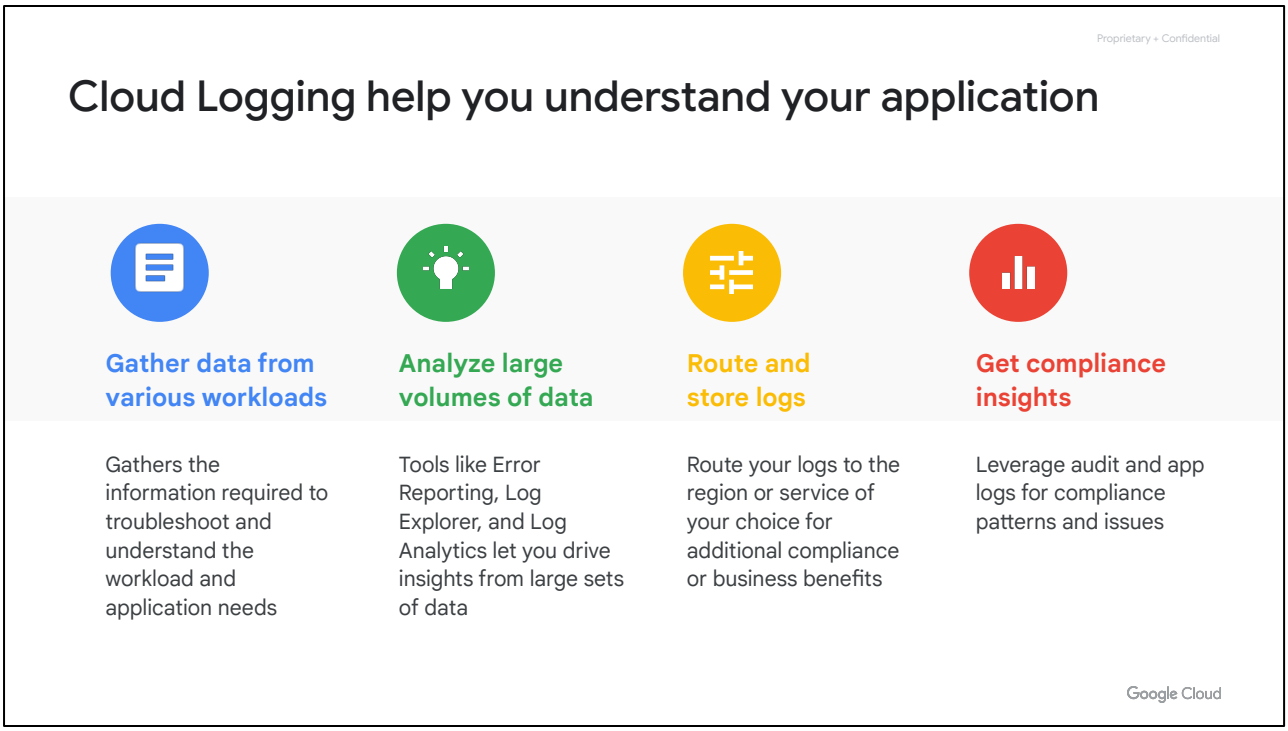
Specifically, in this module you learn to:

- Use Log Explorer features
- Explain the features and benefits of log-based metrics
- Define log sinks (inclusion filters) and exclusion filters
- Explain how Big query can be used to analyze logs
- Export logs to BigQuery for analysis
- Use Log Analytics on Google Cloud

In this section, you'll explore

- Cloud Logging overview and architecture
- Log types and collection
- Storing, routing, and exporting the logs
- Query and view logs
- Using log-based metrics
- Log Analytics

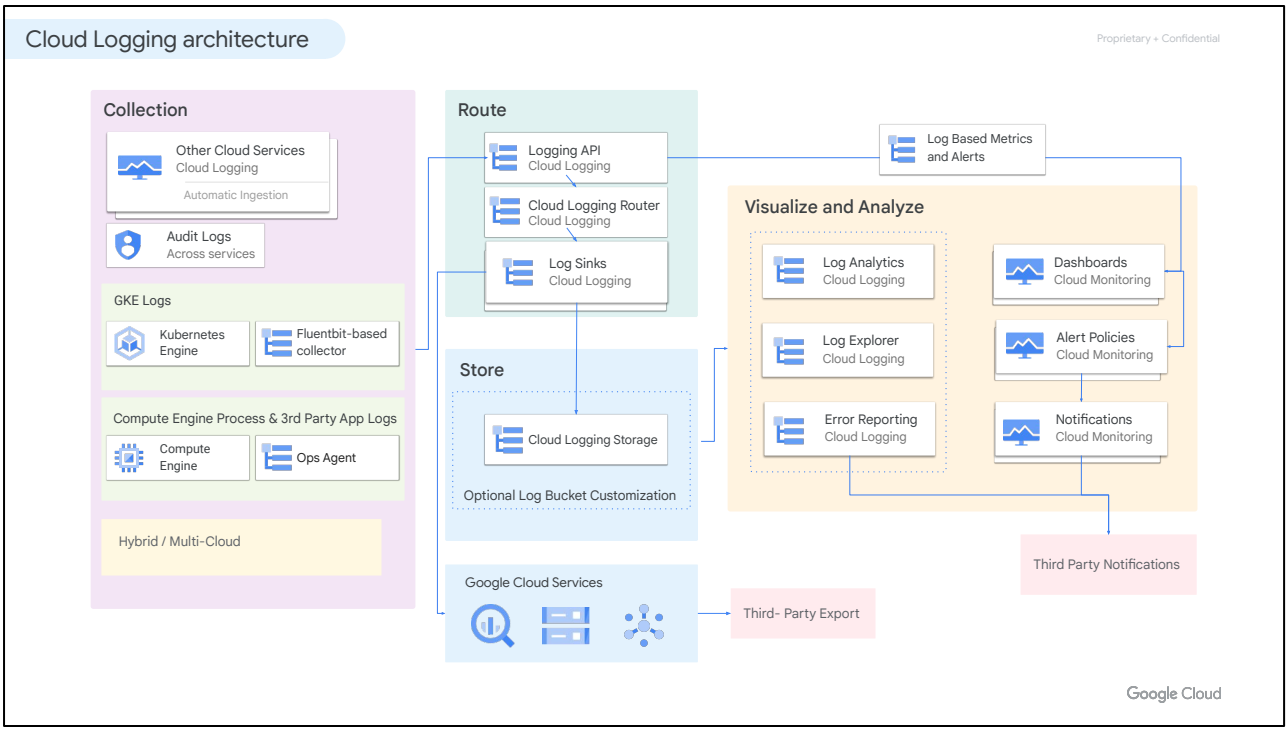
Cloud Logging allows you to store, search, analyze, monitor, and alert on log data and events from Google Cloud. Cloud Logging is a fully managed service that performs at scale and can ingest application and system log data from thousands of VMs. Even better, you can analyze all that log data in real time. Finally we explore a new feature, Log analytics.



Logs is one of the top most visited sections in Google Cloud console and one of most transitional, which indicated that it is an important component of many scenarios. End users need logs for troubleshooting and information gathering but don't want to be overwhelmed with the data. Logs are the pulse of your workloads and application.

Cloud Logging helps to:

- **Gather data from various workloads:** Gathers the information required to troubleshoot and understand the workload and application needs.
- **Analyze large volumes of data:** Tools like Error Reporting, Log Explorer, and Log Analytics let you focus from large sets of data.
- **Route and store logs:** Route your logs to the region or service of your choice for additional compliance or business benefits.
- **Get compliance insights:** Leverage audit and app logs for compliance patterns and issues. We will cover this in the Audit logs module in detail.



Cloud Logging architecture consists of the following components:

- **Log Collections:** These are the places where log data originates. Log sources can be Google Cloud services, such as Compute Engine, App Engine, and Kubernetes Engine, or your own applications.

- **Log Routing:** The Log Router is responsible for routing log data to its destination. The Log Router uses a combination of inclusion filters and exclusion filters to determine which log data is routed to each destination.

- **Log sinks:** Log sinks are destinations where log data is stored. Cloud Logging supports a variety of log sinks, including:
    - Cloud Logging log buckets: These are storage buckets that are specifically designed for storing log data.
    - Pub/Sub topics: These topics can be used to route log data to other services, such as third-party logging solutions.
    - BigQuery: This is a fully-managed, petabyte-scale analytics data warehouse that can be used to store and analyze log data.
    - Cloud Storage buckets: Provides storage of log data in Cloud Storage. Log entries are stored as JSON files.

- **Log Analysis:** Cloud Logging provides several tools to analyze logs.
    - Logs Explorer is optimized for troubleshooting use cases with features like log streaming, a log resource explorer and a histogram for

- visualization.
- Error Reporting help users react to critical application errors through automated error grouping and notifications.
- Logs-based metrics, dashboards and alerting provide other ways to understand and make logs actionable.
- Log Analytics feature expands the toolset to include ad hoc log analysis capabilities.

In this section, you'll explore

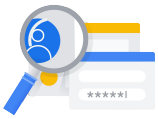
- Cloud Logging overview and architecture
- **Log types and collection**
- Storing, routing, and exporting the logs
- Query and view logs
- Using log-based metrics
- Log Analytics

Let us start with Log Collection and move our way forward to routing and storage and finally visualization.

Available logs

Platform logs

Component logs



Security logs

User-written logs

Multi / Hybrid Cloud logs

The Google Cloud platform logs visible to you in Cloud Logging vary, depending on which Google Cloud resources you're using in your Google Cloud project or organization. Let's explore the key log categories.

Platform logs are logs written by Google Cloud services. These logs can help you debug and troubleshoot issues, and help you better understand the Google Cloud services you're using. For example, VPC Flow Logs record a sample of network flows sent from and received by VM instances.

Component logs are similar to platform logs, but they are generated by Google-provided software components that run on your systems. For example, GKE provides software components that users can run on their own VM or in their own data center. Logs are generated from the user's GKE instances and sent to a user's Cloud project. GKE uses the logs or their metadata to provide user support.

Security logs help you answer "who did what, where, and when."

- Cloud Audit Logs provide information about administrative activities and accesses within your Google Cloud resources.
- Access Transparency provides you with logs of actions taken by Google staff when accessing your Google Cloud content.

User-written logs are logs written by custom applications and services. Typically, these logs are written to Cloud Logging by using one of the following methods:

- Ops Agent

- Cloud Logging API
- Cloud Logging client libraries

Multi-cloud logs and Hybrid-cloud logs: These refer to logs from other cloud providers like Microsoft Azure and logs from on-premises infrastructure.

Some logs from Google Cloud resources are collected automatically

	Google Kubernetes Engine	Compute Engine	Serverless compute services
Cloud Logging	Logs written to **stdout and stderr** are collected automatically	Install the **Ops Agent** on your VMs	Logs written to **stdout** and **stderr** are collected automatically

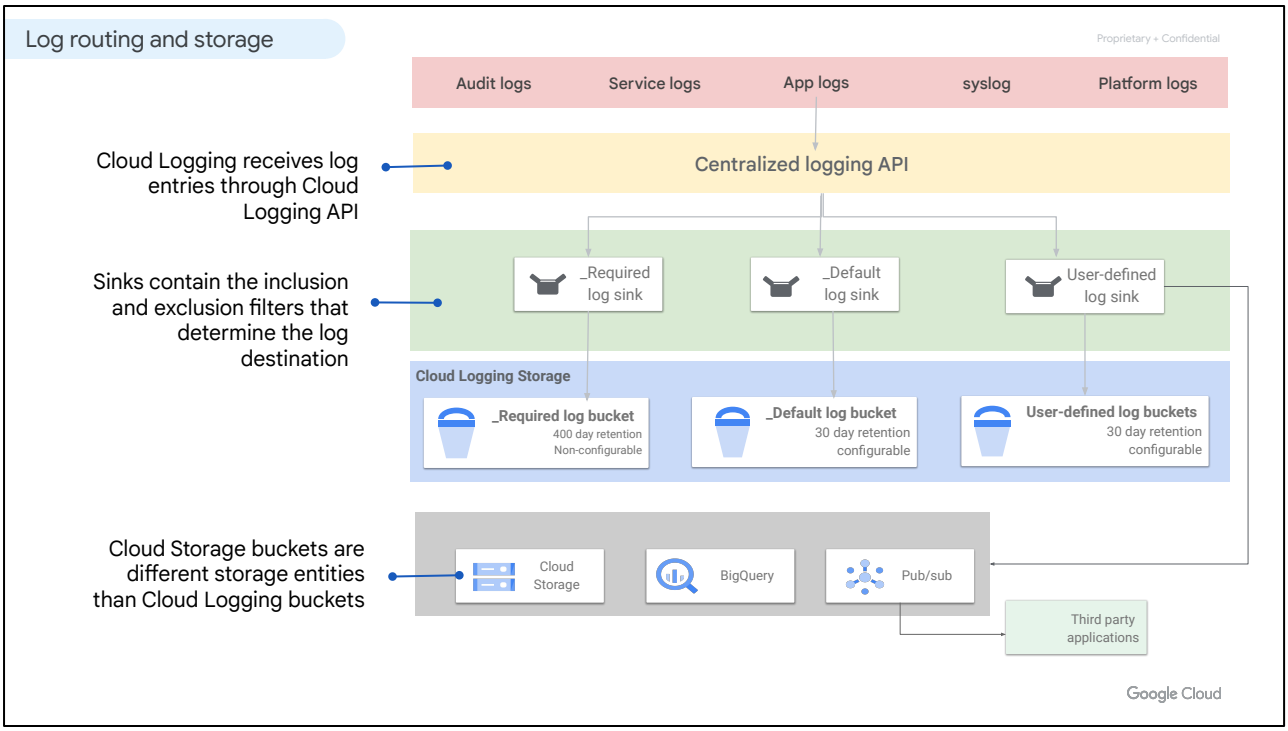
You can programmatically send application logs to Cloud Logging by using client libraries or by using one of the Logging agents. When you can't use them, or when you only want to experiment, you can write logs by using the gcloud logging write command or by sending HTTP commands to the Cloud Logging API endpoint entries.write.

If you're using one of the agents, then your applications can use any established logging framework to emit logs. For example, in container environments like Google Kubernetes Engine or Container-Optimized OS, the agents automatically collect logs from stdout and stderr. On virtual machines (VMs), the agents collect logs from known file locations or logging services like the Windows Event Log, journald, or syslogd. Serverless compute services like Cloud Run and Cloud Run functions, include simple runtime logging by default. Logs written to stdout or stderr will appear automatically in the Google Cloud console.

In this section, you'll explore

- Cloud Logging overview and architecture
- Log types and collection
- **Storing, routing, and exporting the logs**
- Query and view logs
- Using log-based metrics
- Log Analytics

Now that we understand the log collection, let's look at how logs can be routed and exported for long-term storage and analysis.



Let's now explore the log routing and storage section in detail. What we call Cloud Logging is actually a collection of components exposed through a centralized logging API.

log router: Entries are passed through the API and fed to Log Router. Log Router is optimized for processing streaming data, reliably buffering it, and sending it to any combination of log storage and sink (export) locations.

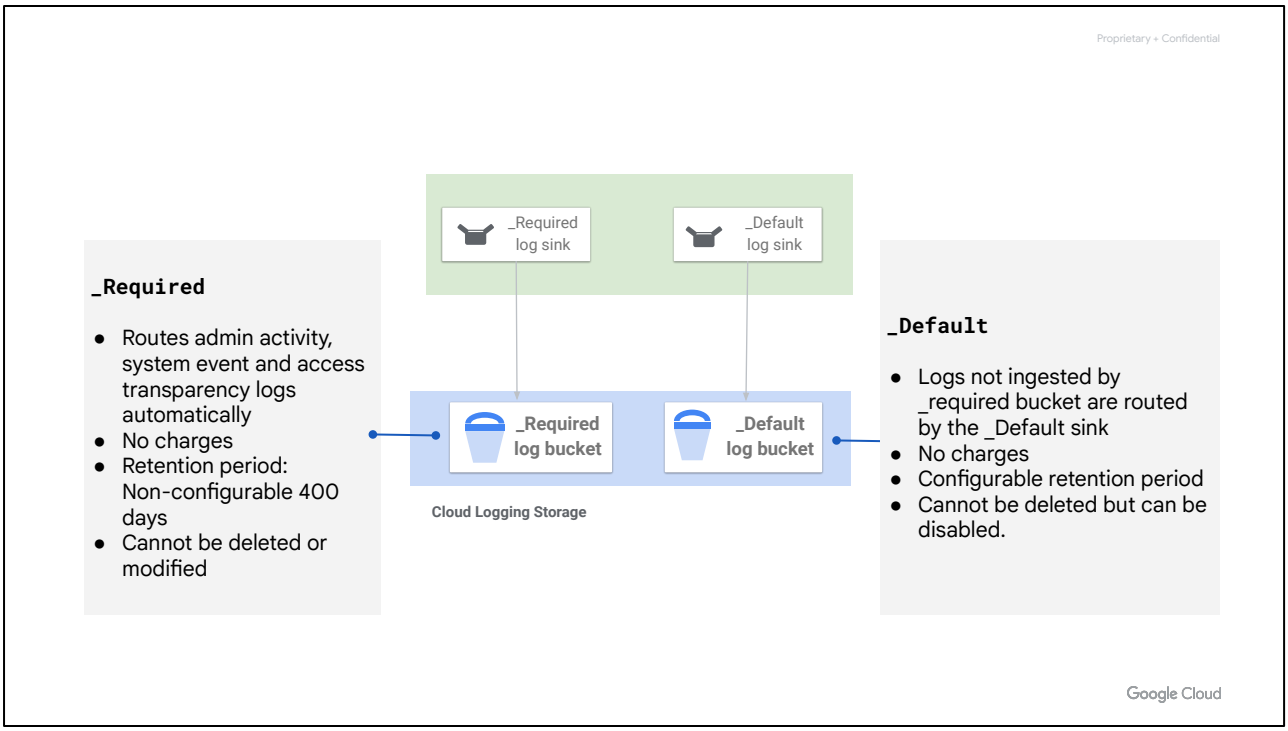
By default, log entries are fed into one of the default logs storage buckets. Exclusion filters might be created to partially or totally prevent this behavior.

Log sinks run in parallel with the default log flow and might be used to direct entries to external locations.

Log storage: Locations might include additional Cloud Logging buckets, Cloud Storage, BigQuery, Pub/Sub, or external projects.

Inclusion and exclusion filters can control exactly which logging entries end up at a particular destination, and which are ignored completely.

Cloud Storage buckets are different storage entities than Cloud Logging buckets.

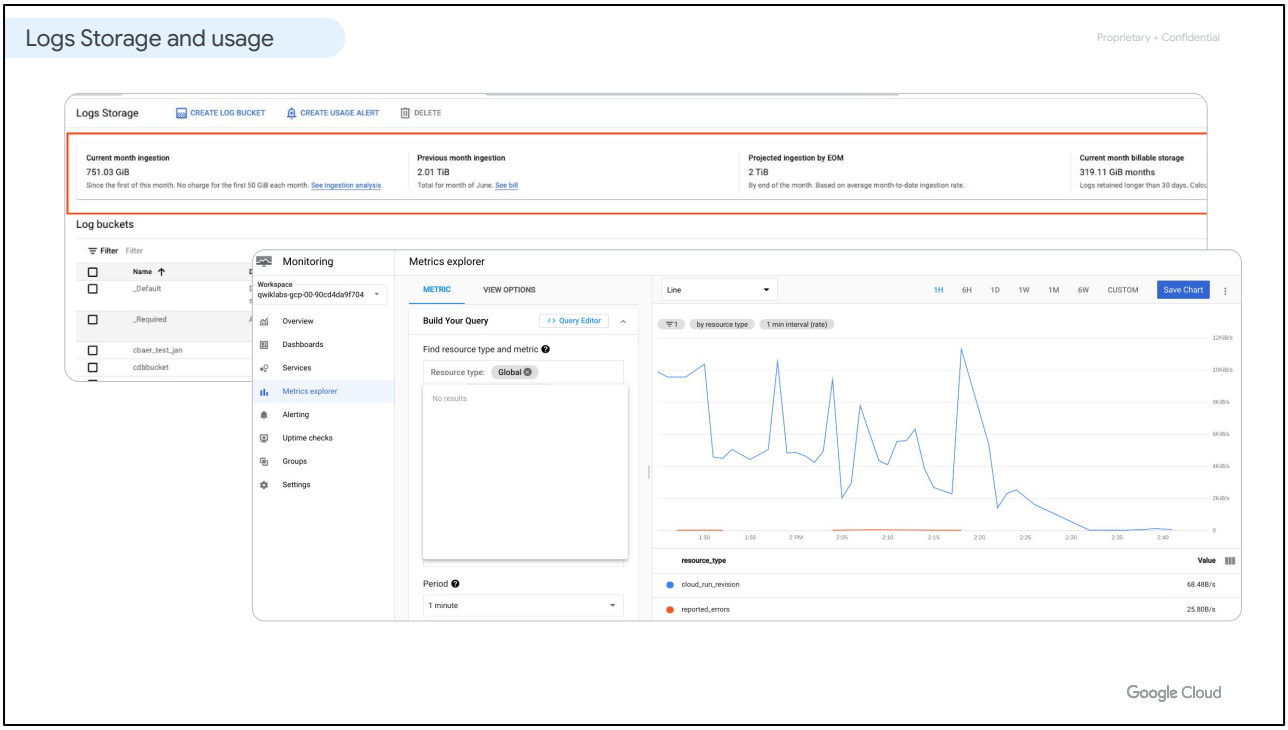


For each Google Cloud project, Logging automatically creates two logs buckets: *_Required* and *_Default*, and corresponding log sinks with the same names. All logs generated in the project are stored in one of these two locations:

- *_Required*: This bucket holds Admin Activity audit logs, System Event audit logs, and Access Transparency logs, and retains them for 400 days. You aren't charged for the logs stored in *_Required*, and the retention period of the logs stored here cannot be modified. You cannot delete or modify this bucket.
- *_Default:* This bucket holds all other ingested logs in a Google Cloud project, except for the logs held in the *_Required* bucket. Standard Cloud Logging pricing applies to these logs. Log entries held in the *_Default* bucket are retained for 30 days, unless you apply custom retention rules. You can't delete this bucket, but you can disable the *_Default* log sink that routes logs to this bucket.

You cana also use the gcloud CLI to adjust the retention:

gcloud beta logging buckets update _Default --location=global --retention-days=[RETENTION_DAYS]



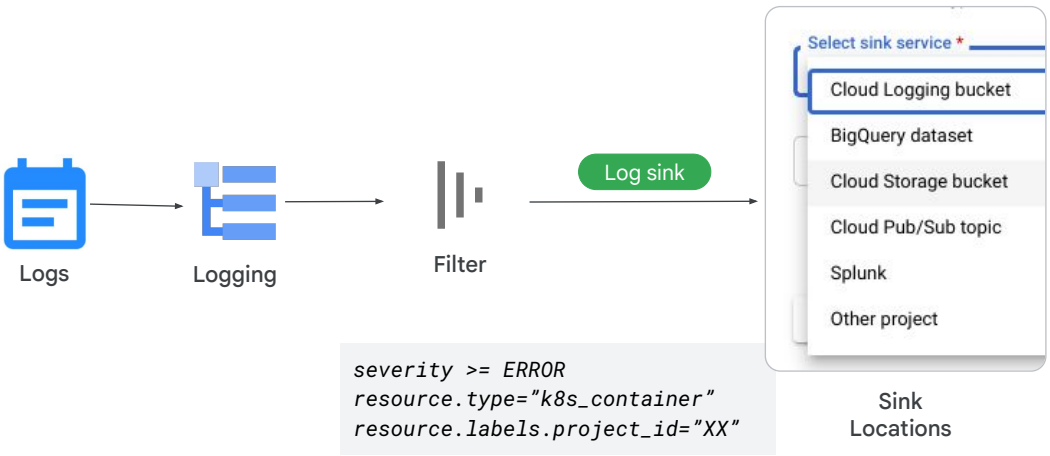
The Logs Storage page displays a summary of statistics for the logs that your project is receiving, including:

- **Current total volume**: The amount of logs your project has received since the first date of the current month.
- **Previous month volume**: The amount of logs your project received in the last calendar month.
- **Projected volume by EOM**: The estimated amount of logs your project will receive by the end of the current month, based on current usage.

You can view the total usage by resource type for the current total volume. The link opens Metrics Explorer, which lets you build charts for any metric collected by your project.

For more information on using Metrics Explorer, see https://cloud.google.com/monitoring/charts/metrics-explorer.

Log router sinks and sink locations



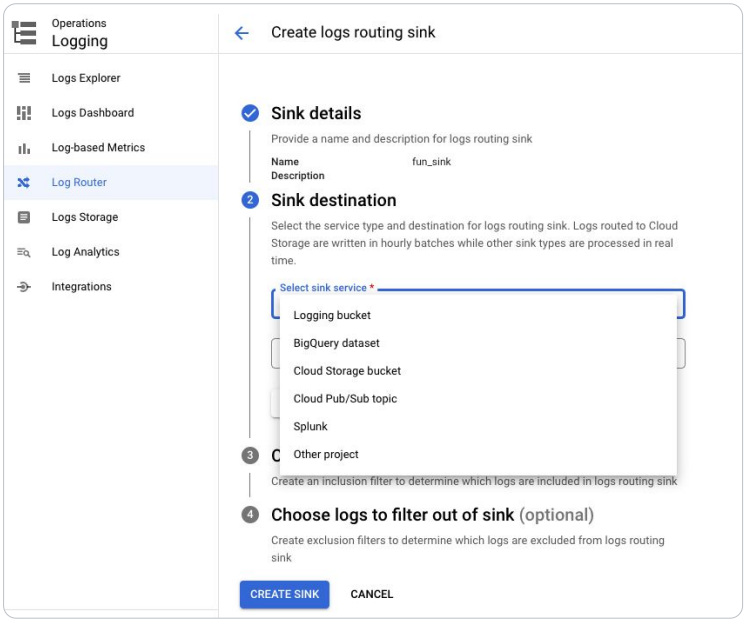
Log Router sinks can be used to forward copies of some or all of your log entries to non-default locations. Use cases include storing logs for extended periods, querying logs with SQL, and access control.

Here, you see we've started creating a sink by generating a log query for a particular subset of entries. We will pass that subset to one of the available sink locations.

There are several sink locations, depending on need:

- **Cloud Logging bucket** works well to help pre-separate log entries into a distinct log storage bucket.
- **BigQuery dataset** allows the SQL query power of BigQuery to be brought to bear on large and complex log entries.
- **Cloud Storage bucket** is a simple external Cloud Storage location, perhaps for long-term storage or processing with other systems.
- **Pub/Sub topic** can export log entries to message handling third-party applications or systems created with code and running somewhere like Dataflow or Cloud Run functions.
- **Splunk** is used to integrate logs into existing Splunk-based system.
- The **Other project** option is useful to help control access to a subset of log entries.

Create a log sink



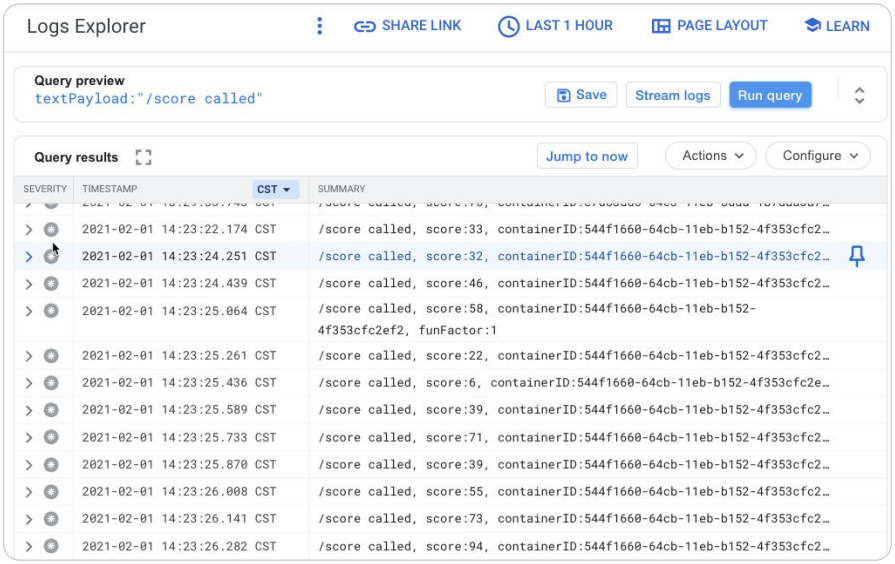
The process for creating log sinks mimics that of creating log exclusions.

It involves writing a **query** that selects the log entries you want to export in Logs Explorer, and choosing a **destination** of Cloud Storage, BigQuery, or Pub/Sub.

The query and destination are held in an object called a **sink**.

Sinks can be created in Google Cloud projects, organizations, folders, and billing accounts.

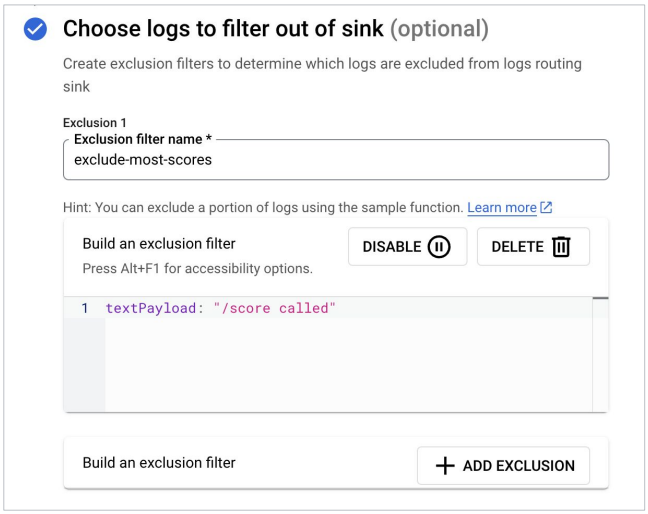
Exclusions: Identify log entries



Use **Logs Explorer** to build a query that selects the logs you want to exclude.

Save the query to use when building the exclusion.

Exclusions: Build the exclusion



Use the **Log Explorer** query to create an exclusion filter that filters the unwanted entries out of the sink. Give the exclusion a name and add the filter for log entries to exclude.

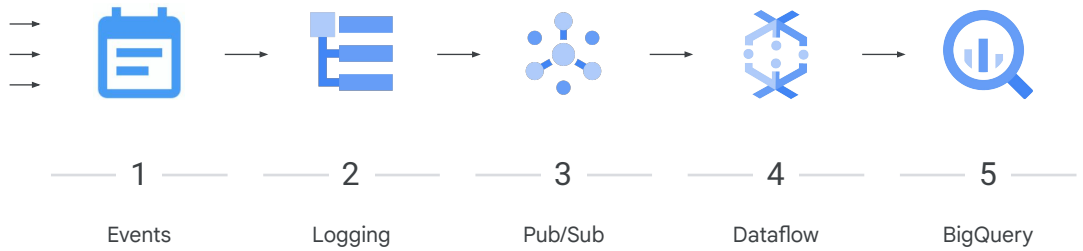
It might be helpful to leave some representative events, depending on the exclusion.

Create the exclusion and it will go into effect immediately. Use the Navigation menu to initiate editing of that entity.

Take care here, because excluded log events will be lost forever.

Log archiving and analysis

Example pipeline



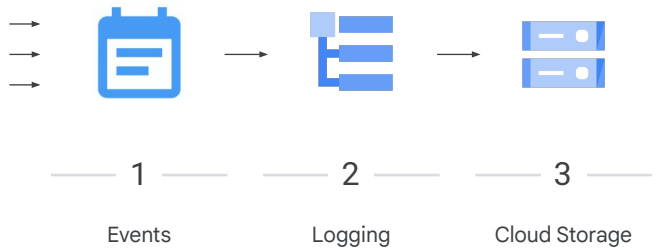
Over the next several slides, we will investigate some possible log export processing options.

Here, for example, we are exporting through Pub/Sub, to Dataflow, to BigQuery. Dataflow is an excellent option if you're looking for real-time log processing at scale.

In this example, the Dataflow job could react to real-time issues, while streaming the logs into BigQuery for longer-term analysis.

Archive logs for long-term storage

Example pipeline

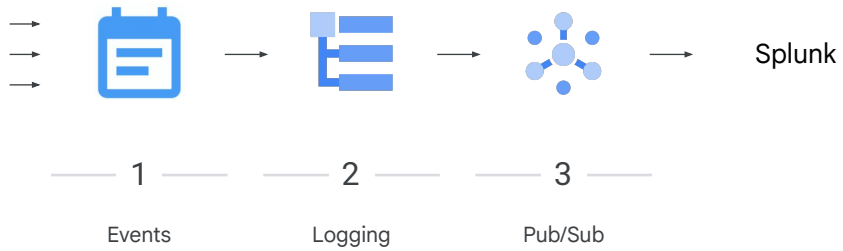


Sink pipelines targeting Cloud Storage tend to work best when your needs align with Cloud Storage strengths. For example, long-term retention, reduced storage costs, and configurable object lifecycles.

Cloud Storage features include automated storage class changes, auto-delete, and guaranteed retention.

Exporting back to Splunk

Example pipeline



Here, we have an example organization that wants to integrate the logging data from Google Cloud, back into an on-premises Splunk instance. You can ingest logs into Splunk you can either stream logs using Pub/Sub to Splunk Dataflow or using the Splunk Add-on for Google Cloud.

Pub/Sub is one of the options available for exporting to Splunk, or to other third-party System Information and Event Management (SIEM) software packages.

Aggregation levels

Project

A **project-level log sink** exports all the logs for a specific project.

A **log filter** can be specified in the sink definition to include/exclude certain log types.

Folder

A **folder-level log sink** aggregates logs on the folder level.

You can also include logs from children resources (subfolders, projects).

Organization

An **organization-level log sink** aggregates logs on the organization level.

You can also include logs from children resources (subfolders, projects).

A common logging need is centralized log aggregation for auditing, retention, or non-repudiation purposes.

Aggregated sinks allow for easy exporting of logging entries without a one-to-one setup. The sink destination can be any of the destinations discussed up to now.

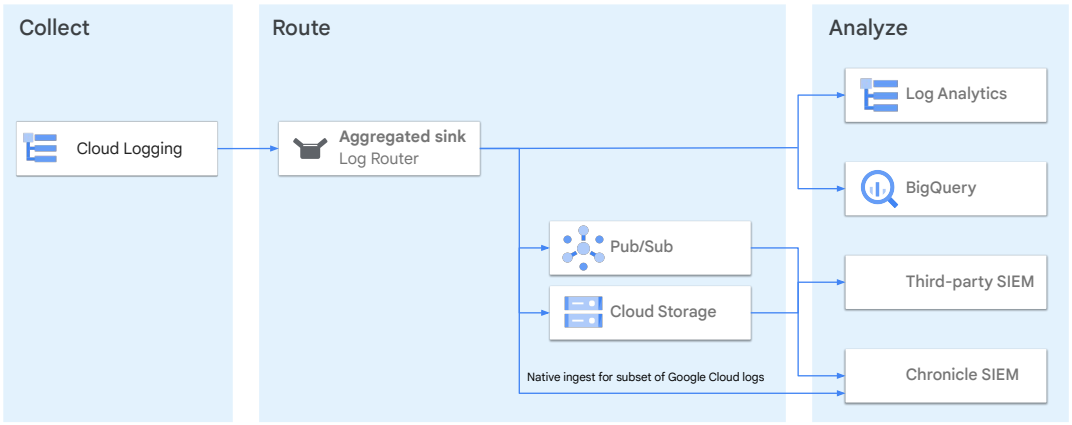
There are three available Google Cloud Logging aggregation levels.

We've discussed a project-level log sink. It exports all the logs for a specific project and a log filter can be specified in the sink definition to include/exclude certain log types.

A folder-level log sink aggregates logs on the folder level and can include logs from children resources (subfolders, projects).

And for a global view, an organization-level log sink can aggregate logs on the organization level and can also include logs from children resources (subfolders, projects).

Security log analytics workflow recommends aggregated sinks



Security practitioners onboard Google Cloud logs for security analytics. By performing security analytics, you help your organization prevent, detect, and respond to threats like malware, phishing, ransomware, and poorly configured assets.

One of the steps in security log analytics workflow is to create aggregate sinks and route those logs to a single destination depending on the choice of security analytics tool, such as Log Analytics, BigQuery, Chronicle, or a third-party security information and event management (SIEM) technology. Logs logs are aggregated from your organization, including any contained folders, projects, and billing accounts.

Field naming

Log entry field	LogEntry type mapping	BigQuery field name
insertId	insertId	insertId
textPayload	textPayload	textPayload
httpRequest.status	httpRequest.status	httpRequest.status
httpRequest. requestMethod.GET	httpRequest. requestMethod.[ABC]	httpRequest. requestMethod.get
resource.labels.moduleid	resource.labels.[ABC]	resource.labels.moduleid
jsonPayload.MESSAGE	jsonPayload.[ABC]	jsonPayload.message
jsonPayload.myField. mySubfield	jsonPayload.[ABC].[XYZ]	jsonPayload.myfield. mysubfield

There are a few naming conventions that apply to log entry fields:

- For log entry fields that are part of the LogEntry type, the corresponding BigQuery field names are precisely the same as the log entry fields.
- For any user-supplied fields, the letter case is normalized to lowercase, but the naming is otherwise preserved.
- For fields in structured payloads, as long as the @type specifier is not present, the letter case is normalized to lowercase, but naming is otherwise preserved. For information on structured payloads where the @type specifier is present, see the Payload fields with @type documentation.

You can see some examples on the current slide.

Last three days from syslog and *apache_access* for a particular *gce_instance*

```
SELECT
  timestamp AS Time, logName as Log, textPayload AS Message
FROM
  (TABLE_DATE_RANGE(my_bq_dataset.syslog_,
    DATE_ADD(CURRENT_TIMESTAMP(), -2, 'DAY'), CURRENT_TIMESTAMP())),
  (TABLE_DATE_RANGE(my_bq_dataset.apache_access_,
    DATE_ADD(CURRENT_TIMESTAMP(), -2, 'DAY'), CURRENT_TIMESTAMP()))
WHERE
  resource.type == 'gce_instance'
  AND resource.labels.instance_id == '1554300700000000000'
ORDER BY time;
```


Here's a sample query over the Compute Engine logs. It retrieves log entries for multiple log types over multiple days.

The query searches the last three days (today -2) of the *syslog* and *apache-access* logs.

The query retrieves results for the single Compute Engine instance ID seen in the *where* clause.

Failed App Engine requests for the last month

```
SELECT
  timestamp AS Time,
  protoPayload.host AS Host,
  protoPayload.status AS Status,
  protoPayload.resource AS Path
FROM
  (TABLE_DATE_RANGE(my_bq_dataset.appengine_googleapis_com_request_log_,
    DATE_ADD(CURRENT_TIMESTAMP(), -1, 'MONTH'), CURRENT_TIMESTAMP()))
WHERE
  protoPayload.status != 200
ORDER BY time
```


In this BigQuery example, we are looking for unsuccessful App Engine requests from the last month.

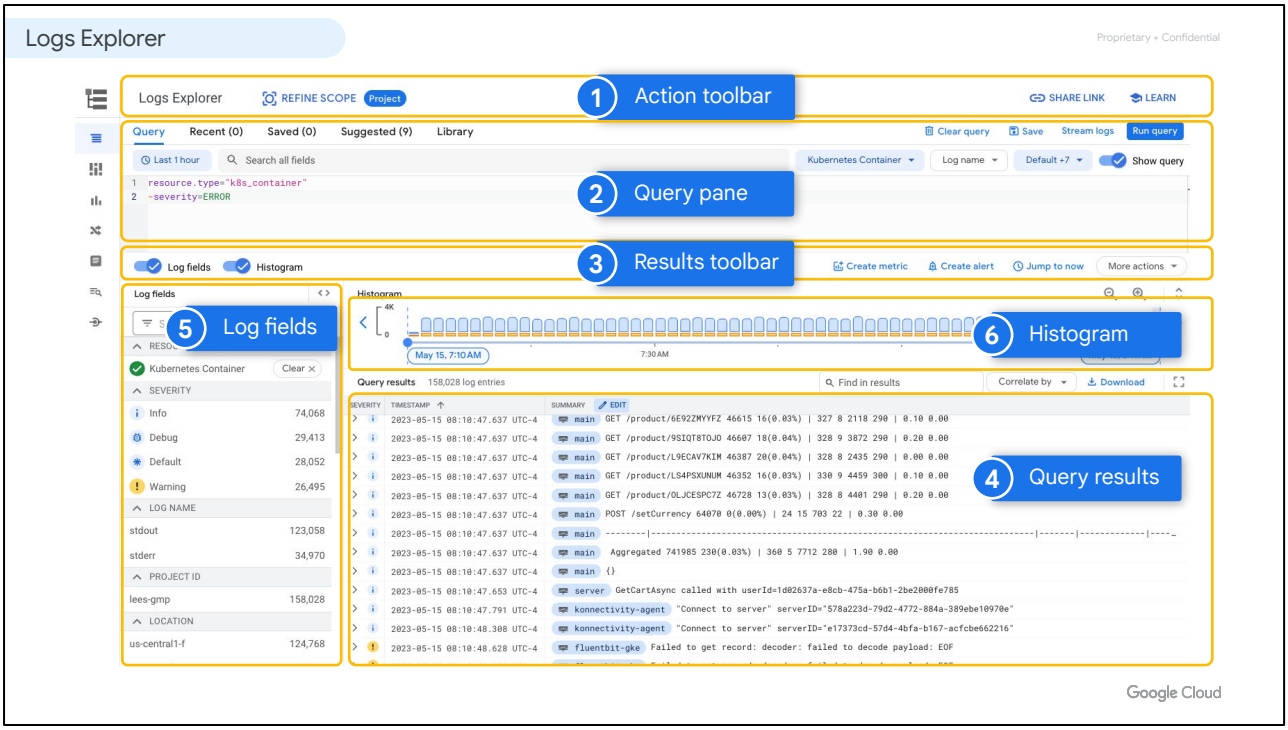
Notice how the *from* clause is constructing the table data range.

The status not equal to 200 is examining the HTTP status for anything that isn't 200. That is to say, anything that isn't a successful response.

In this section, you'll explore

- Cloud Logging overview and architecture
- Log types and collection
- Storing, routing, and exporting the logs
- **Query and view logs**
- Using log-based metrics
- Log Analytics

Once you have collected logs and routed to the right destination, now is the time to query and view logs.



The Logs Explorer interface lets you retrieve logs, parse and analyze log data, and refine your query parameters. The Logs Explorer contains the following panes:

1. **Action toolbar:** Action toolbar to refine logs to projects or storage views, share a link and learn about logs explorer.
2. **Query pane:** Query pane is where you can build queries, view recently viewed and saved queries and a lot more.
3. **Results Toolbar:** This can be used to quickly show or hide logs and histogram pane nad create a log based metric or alert. **Jump to now** option helps query and view the current time results.
4. **Query results:** Is the details of results with a summary and timestamp that helps troubleshoot further.
5. **Log fields**: Log fields pane is used to filter your options based on various factors such as a resource type, log name, project ID, etc,.
6. **Histogram:** Histogram is where the query result is visualized a histogram bars, where each bar is a time range and is color coded based on severity.

Ultimately, it's the query that selects the entries

- Start with what you know about the entry you're trying to find.
- If it belongs to a resource, a particular log name, or has a known severity, use the query builder drop-down menus.

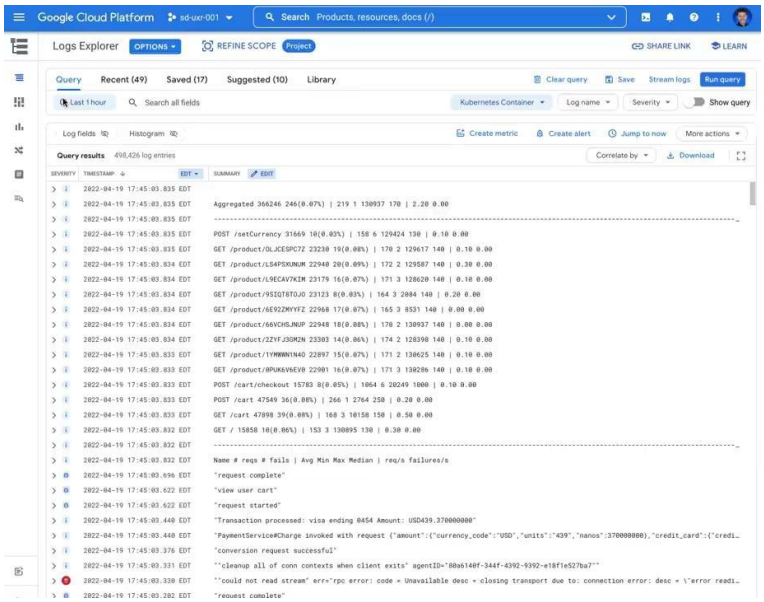
Ultimately, it's the query that selects the entries displayed by Logs Explorer. Queries may be created directly with the Logging Query Language (LQL), using the drop-down menus, the logs field explorer, or by clicking fields in the results themselves.

Start with what you know about the entry you're trying to find. If it belongs to a resource, a particular log file, or has a known severity, use the query builder drop-down menus.

The query builder drop-down menu makes it easy to start narrowing your log choices.

- **Resource**: Lets you specify resource.type. You can select a single resource at a time to add to the **Query builder**. Entries use the logical operator AND.
- **Log name**: Lets you specify logName. You can select multiple log names at once to add to the **Query builder**. When selecting multiple entries, the logical operator OR is used.
- **Severity**: Lets you specify severity. You can select multiple severity levels at once to add to the **Query builder**. When selecting multiple entries, the logical operator OR is used.

Building queries made easier using dropdowns



When you are troubleshooting an issue, finding the root cause often involves finding specific logs generated by infrastructure and application code. The faster you can find logs, the faster you can confirm or refute your hypothesis about the root cause and resolve the issue. Ultimately, it's the query that selects the entries displayed by Logs Explorer. Queries may be created directly with the Logging Query Language (LQL), using the drop-down menus, the logs field explorer, or by clicking fields in the results themselves.

Start with what you know about the entry you're trying to find. If it belongs to a resource, a particular log file, or has a known severity, use the query builder drop-down menus.

The query builder drop-down menu makes it easy to start narrowing your log choices.

To make searching logs and building a query easier, you can use the dropdown selectors.

- **Resource**: Lets you specify resource.type. You can select a single resource at a time to add to the **Query builder**. Entries use the logical operator AND.
- **Log name**: Lets you specify logName. You can select multiple log names at once to add to the **Query builder**. When selecting multiple entries, the logical operator OR is used.
- **Severity**: Lets you specify severity. You can select multiple severity levels at once to add to the **Query builder**. When selecting multiple entries, the logical

- operator OR is used.
- **Simple text search** – Lets you use a simple text search box for global text searches
- **Advanced query** – Lets you toggle to show/hide the Logging query language for the query
- **Date/time picker –** Lets you pick the date/time range in the query builder
- **Dropdown selectors –** Lets you prominently display the resource, logName and severity dropdown selectors
- **Default summary fields** – Lets you disable default summary fields for a more basic log view.

Comparison operators to filter queries

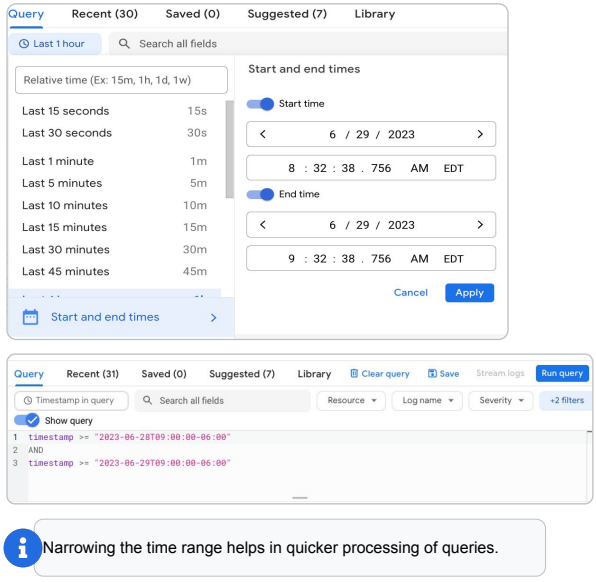
```
[FIELD_NAME] [OP] [VALUE]
```

=	Equals	resource.type="gce_instance"
!=	does not equal	resource.labels.instance_id!="1234567890"
> < >= <=	numeric ordering	timestamp <= "2018-08-13T20:00:00Z"
:	has	textPayload:"GET /check"
:*	presence	jsonPayload.error:*
=~	search for a pattern	jsonPayload.message =~ "regular expression pattern"
!~	search not for a pattern	jsonPayload.message !~ "regular expression pattern"

The next several slides are included for reference. Advanced queries support multiple comparison operators as seen here.

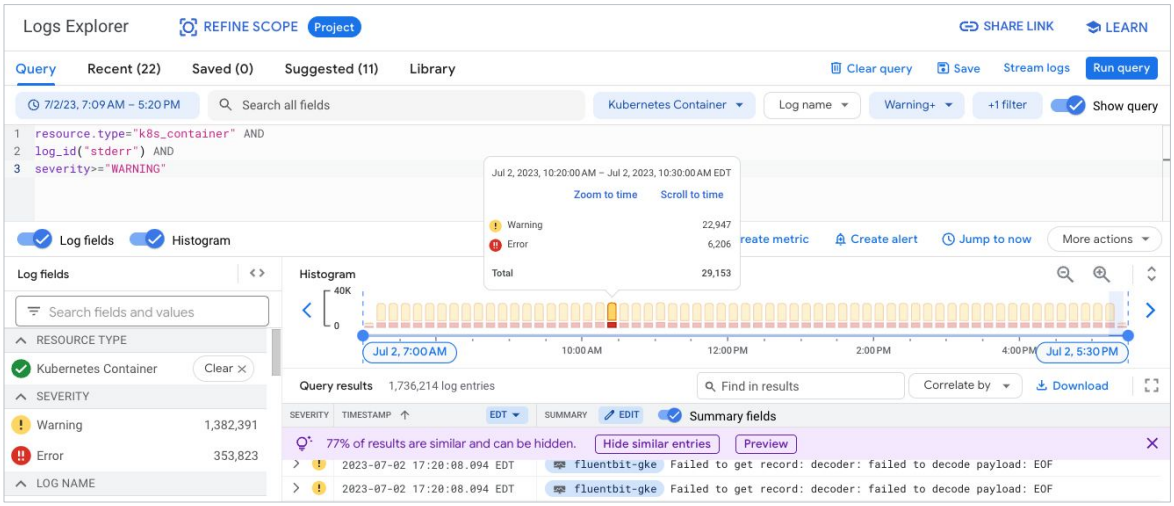
- The equal and not equal operators help filter values that match or not match a value assigned to a field name. These are useful when you search for a specific resource type or id that you want to evaluate.
- The numeric ordering operators are handy when searching for logs filtering a timestamp or duration.
- The colon operation helps check if a value exists. This is useful when you want to match a substring within a log entry field.
- To test if a missing or defaulted field exists without testing for a particular value in the field, use the :* comparison.

Finding log entries, set the time range



If you're looking for a specific set of log entries and have a rough idea when they would have been generated, start by narrowing to a specific time range. You can select one of the pre-created choices, set a custom range, or jump to a particular time.

Narrow with Logs fields and Histogram



Google Cloud

The **Log fields** panel offers a high-level summary of logs data and provides a more efficient way to refine a query. It shows the count of log entries, sorted by decreasing count, for the given log field. The log field counts correspond to the time range used by the **Histogram** panel.

You can add fields from the **Log fields** panel to the **Query builder** to narrow down and refine a query by clicking a field.

When a query is run, the log field counts are incrementally loaded as the log entries are progressively scanned. Once the query is complete, which is indicated by the completion of the blue progress bar, you see the total counts for all log fields.

The histogram panel lets you visualize the distribution of logs over time. Visualization makes it easier to see trends in your logs data and troubleshoot problems. For example, the severity colors make it easy to spot an increasing number of errors even when the volume of requests is relatively constant.

To analyze your log data, point to a bar in the **Histogram** panel and select **Jump to time** to drill into a narrower time range. A new query runs with that time-range restriction.

Supported Boolean operators

AND	`textPayload:("foo" AND "bar")`
NOT	`textPayload:("foo" AND NOT "bar")`
OR	`textPayload:("foo" OR "bar")`

Ensure to use the all caps for the operator name.

The NOT operator has the highest precedence, followed by OR and AND in that order.

The Boolean operators AND and OR are short-circuit operators.

Advanced queries support the AND, OR, and NOT boolean expressions for joining queries. Ensure to use the all caps for the operator name.

A couple of things to keep in mind include:

- Ensure to use the all caps for the operator name.
- The NOT operator has the highest precedence, followed by OR and AND in that order.
- The Boolean operators AND and OR are short-circuit operators.

The recipe for finding entries

- What do you know about the log entry?
  - Log filename, resource, a bit of text?
- Full text searches are slow, but may be effective:
  - **"/score called"**
- Use indexed SEARCH function for complete text matches, because they perform a case-insensitive match
  - SEARCH(textPayload, "hello world")
- If possible, restrict text searches to an log field
  - **jsonPayload:"/score called"**
  - **jsonPayload.message="/score called"**

When you're trying to find log entries, start with what you know: the log filename, resource name, even a bit of the contents of the logged message might work.

Full text searches are slow, but they may be effective. For example, you might search for "/score called".

If possible, restrict text searches to an entry region, like jsonPayload:"/score called", or even better, jsonPayload.message="/score called".

You can use the built-in SEARCH function to find strings in your log data:

- SEARCH([query])
- SEARCH([field], [query])

Both forms of the SEARCH function contain a query argument, which must be formatted as a string literal. In the first form, the entire log entry is searched. In the second form, you specify the field in the log entry to search. The Logging query language supports different ways that you can search your log data. When searching for a string, it is more efficient to use the SEARCH function than to perform a global search or a substring search. However, you can't use use the SEARCH function to match non-text fields.

Finding entries quickly

- Search on an indexed field

```
httpRequest.status, logName, operation.id, resource.type, timestamp, severity
```

- Apply constraints on **`resource.type`** and **`resource.labels`** field

```
resource.type = "gke_cluster"
resource.labels.namespace = "my-cool-namespace"
```

- Be specific on which logs you're searching

```
logName="projects/benkelly-test/logs/apache-access"
```

- Limit the time range that you're searching

```
timestamp >= "2018-08-08T10:00:00Z" AND timestamp <= "2018-08-08T10:10:00Z"
```


Some tips on finding log entries quickly:

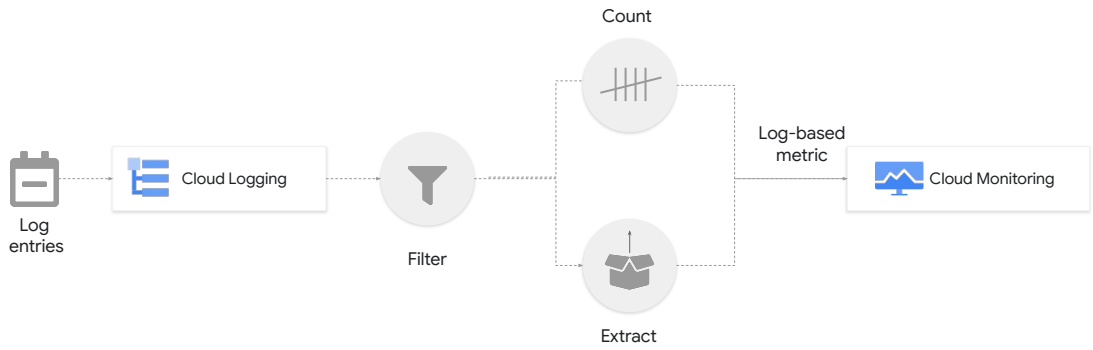
- Search for specific values of indexed fields, like the name of the log entry, resource type, and resource labels.
- Constraints on resource.type and resource.labels, resource.type = "gke_cluster" and resource.labels.namespace = "my-cool-namespace"
- These fields are preferentially indexed in our storage and can make a huge difference for query performance.
- As seen in the example, be specific on which logs you're searching by referring to it or them by name.
- Limit the time range that you're searching to reduce the log data that is being queried.

In this section, you'll explore

- Cloud Logging overview and architecture
- Log types and collection
- Storing, routing, and exporting the logs
- Query and view logs
- **Using log-based metrics**
- Log Analytics

Let's next focus about generating monitoring metrics from logging data.

Logs-based metrics



Logs-based metrics derive metric data from the content of log entries.

Logs-based metrics derive metric data from the content of log entries. For example, metrics can track the number of entries that contain specific messages or extract latency information that is reported in the logs.

These metrics transform into time series data and use it in Cloud Monitoring Charts and Alerting Policies.

There are two types of log-based metrics:

- System-defined log-based metrics, provided by Cloud Logging for use by all Google Cloud projects.
  System-defined log-based metrics are calculated only from logs that have been ingested by Logging. If a log has been explicitly excluded from ingestion by Cloud Logging, it isn't included in these metrics.
- User-defined log-based metrics, created by you to track things in your Google Cloud project that are of particular interest to you. For example, you might create a log-based metric to count the number of log entries that match a given filter.

Logs-based metrics are suitable in different cases

Count the occurrences	Observe trends in your data	Visualize extracted data
Count the occurrences of a message, like a warning or error, in your logs and receive a notification when the number of occurrences crosses a threshold.	Observe trends in your data, like latency values in your logs, and receive a notification if the values change in an unacceptable way.	Create charts to display the numeric data extracted from your logs.

Log-based metrics are suitable when you want to do any of the following:

- Count the occurrences of a message, like a warning or error, in your logs and receive a notification when the number of occurrences crosses a threshold.
- Observe trends in your data, like latency values in your logs, and receive a notification if the values change in an unacceptable way.
- Create charts to display the numeric data extracted from your logs.

Key access control roles

- Logs Configuration Writer
  - List, create, get, update, and delete log-based metrics.
- Logs Viewer
  - View existing logs.
- Monitoring Viewer
  - Read the time series in log-based metrics.
- Logging Admin, Editor, and Owner
  - These are broad-level roles that can create log-based metrics.

A refresher of the key IAM roles that relate to logging and monitoring.

First, on the logging side:

- **Logs Configuration Writers** can list, create, get, update, and delete log-based metrics.
- **Logs Viewers** can view existing metrics.

On the monitoring side, **Monitoring Viewers** can read the time series in log-based metrics.

And finally, **Logging Admins**, **Editors**, and **Owners** are all broad-level roles that can create log-based metrics.

Log-based metric types

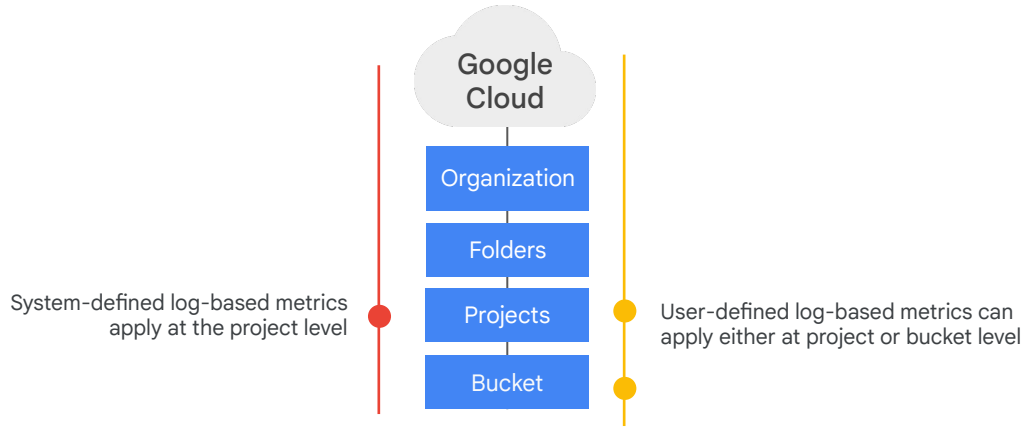
There are three types of log-based metrics: **counter** or **distribution**. All predefined system log-based metrics are the counter type, but user-defined metrics can be either counter, distribution or boolean types.

Counter metrics count the number of log entries matching an advanced logs query. So, if we simply wanted to know how many of our "/score called" entries were generated, we could create a counter.

Distribution metrics record the statistical distribution of the extracted log values in histogram buckets. The extracted values are not recorded individually. Their distribution across the configured buckets is recorded, along with the count, mean, and sum of squared deviations of the values.

Boolean metrics record where a log entry matches a specified filter.

Scope of log-based metrics



System-defined log-based metrics apply at the Google Cloud project level. These metrics are calculated by the Log Router and apply to logs only in the Google Cloud project in which they're received.

User-defined log-based metrics can apply at either the Google Cloud project level or at the level of a specific log bucket:

- Project-level metrics are calculated like system-defined log-based metrics; these user-defined log-based metrics apply to logs only in the Google Cloud project in which they're received.
- Bucket-scoped metrics apply to logs in the log bucket in which they're received, regardless of the Google Cloud project in which the log entries originated.

With bucket-scoped log-based metrics, you can create log-based metrics that can evaluate logs in the following cases:

- Logs that are routed from one project to a bucket in another project.
- Logs that are routed into a bucket through an aggregated sink.

A sample test code

```
//Basic NodeJS app built with the express server
app.get('/score', (req, res) => {
 //Random score, the contaierID is a UUID unique to each
 //runtime container (testing was done in Cloud Run).
 //funFactor is a random number 1-100
 let score = Math.floor(Math.random() * 100) + 1;
  console.log(`/score called, score:${score},
     containerID:${containerID}, funFactor:${funFactor}`);
   //Basic message back to browser
 res.send(`Your score is a ${score}. Happy?`);
 });
```

Track the request on the on the '/score' path

If a request matches, generate 1-100 score

Define log entry fields

Send a msg containing the score

Before we create a log-based metric, let's generate some logging entries. Here we see a basic NodeJS app built with the simple and lightweight Express web server. The app is run as a managed container on the Cloud Run service.

The code watches for a request to come into the server on the '/score' path.

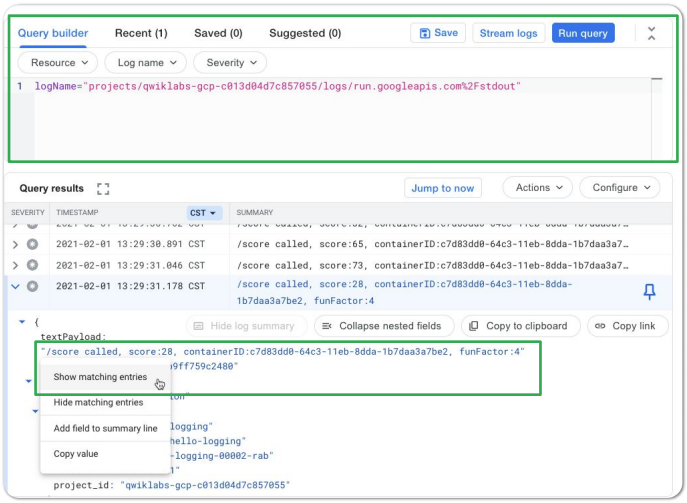
When a /score request arrives, the code generates a random 1-100 **score**, and it then creates a log entry.

Earlier code, not shown on this slide, created a unique identifier for the container serving this request in containerID and a random value called funFactor.

The log entry contains the text "/score called", the random score, the container ID, and the fun factor.

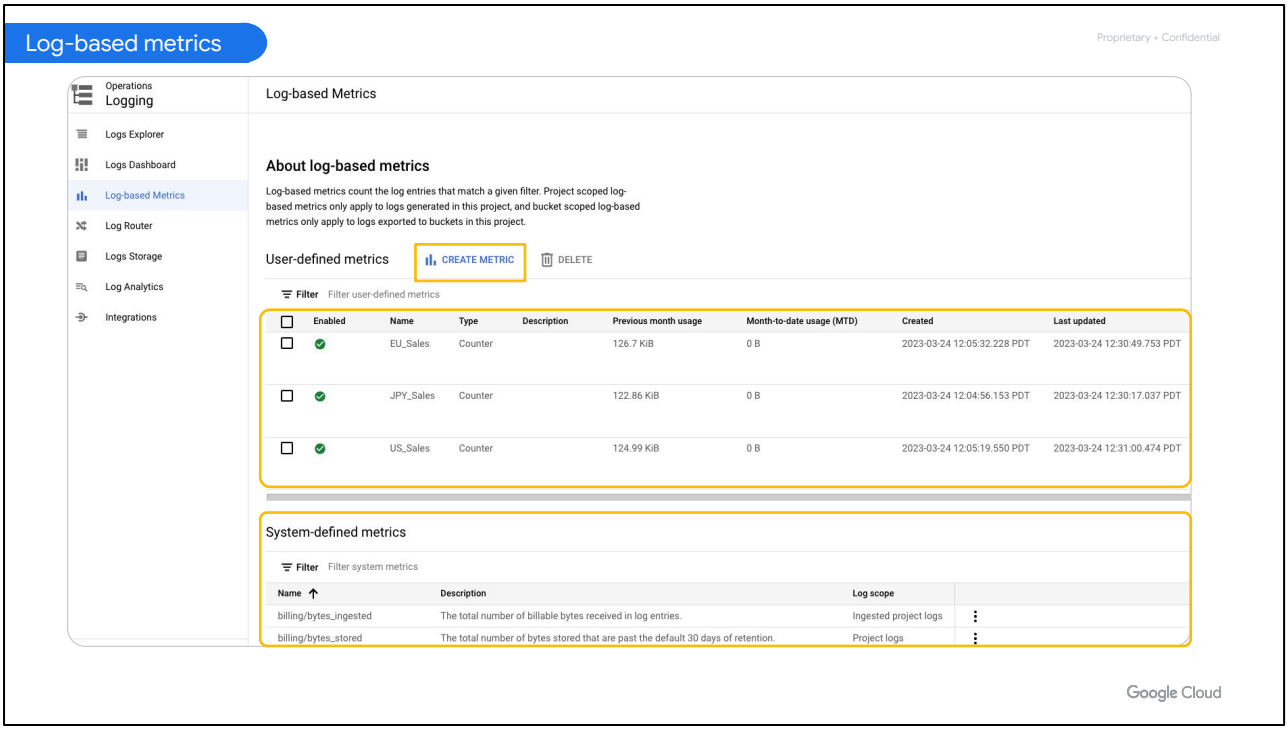
Lastly, a basic message, also containing the score, is sent back to the browser.

Filtering entries



Use the Query builder to access project logs.

In the list of entries, we've located one of the "/score called" entries. Now we can filter to select those entries by clicking "/score called", and selecting **Show matching entries**.



Imagine we've generated some load on our Cloud Run sample application, and we'd like to use the log events to generate a log-based metric.

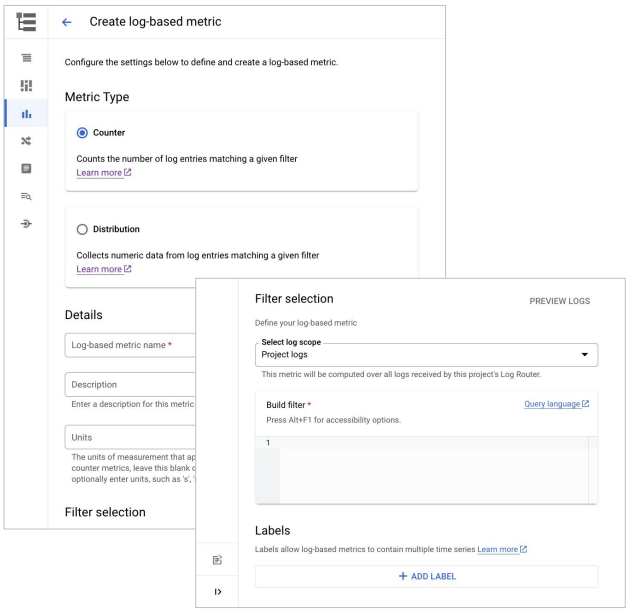
There are two fundamental log-based metric types:

- **System log-based metrics** which are predefined by Google and are a standard part of log-based metrics.
- And then there are **User-defined log-based metrics,** which are created by a user on a project. These metrics count the number of log entries that match a given query or track particular values within the matching log entries.

The latter is what we are creating now. Note the **Create Metric** button at the top of the interface.

In Logs Explorer

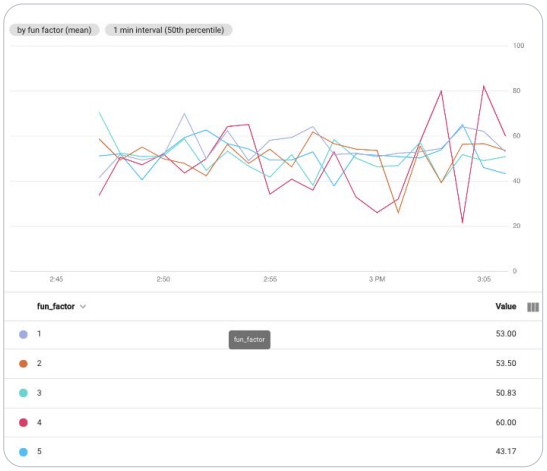
1. Find the log with the requisite data
2. Filter to the required entries
3. **Actions** | **Create Metric**
4. Pick a metric type (Counter or Distribution)
5. If Distribution, set configurations
6. You can also add labels



This is an example of a basic flow for creating log-based metrics:

1. You start by finding the log with the requisite data.
2. Then you filter it to the required entries.
3. Create a metric.
4. Pick your metric type (Counter or Distribution).
5. If Distribution, then set configurations.
6. And finally, add labels as needed.

Labels (for example, group-by, or filter)



Like many cloud resources, labels can be applied to log-based metrics. Their prime use is to help with group-by and filtering tasks in Cloud Monitoring.

Labels and logs

- Labels allows for log-based metrics to contain a time series for each label.
- Two types of labels applied:
  - Default
  - User-defined
- User-defined labels can be either of the following:
  - The entire contents of a named field in the LogEntry object.
  - A part of a named field that matches a regular expression.
- You can create up to ten user-defined labels per metric.
- A label cannot be deleted once created.
  - Grows the time series significantly.

Labels allow log-based metrics to contain multiple time series—one for each label value.

All log-based metrics come with some default labels and you can create additional user-defined labels in both counter-type and distribution-type metrics by specifying extractor expressions. An extractor expression tells Cloud Logging how to extract the value of the label from log entries. You can specify the label's value as either of the following:

- The entire contents of a named field in the LogEntry object.
- A part of a named field that matches a regular expression (regexp).

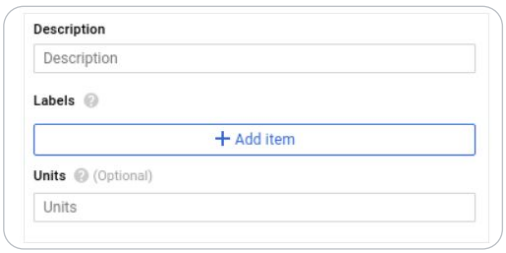
You can extract labels from the LogEntry built-in fields, such as httpRequest.status, or from one of the payload fields, textPayload, jsonPayload, or protoPayload.

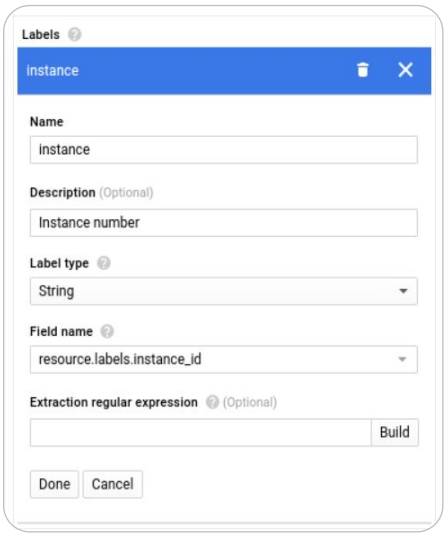
Label with care. A metric can support up to ten user-defined labels, and once created, a metric cannot be removed. Also, each log-based metric is limited to about 30,000 active time series.

Each label can grow the time series count significantly. For example, if your log entries come from 100 resources, such as VM instances, and you define a label with 20 possible values, then you can have up to 2,000 time series for your metric.

Creating user-defined labels

User-defined labels can be created when creating a log-based metric.





User-defined labels can be created when creating a log-based metric. The label form requires:

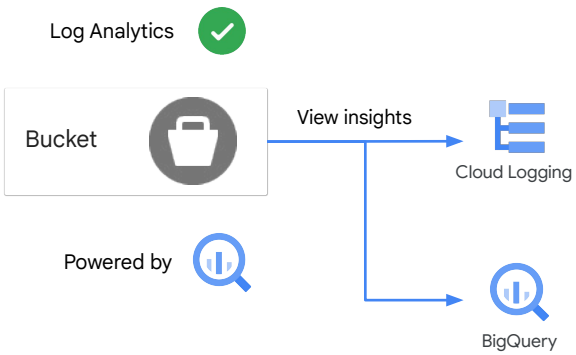
1. **Name**: The identifier which will be used to label in Monitoring.
2. **Description**: Describe the label. Try to be as specific as possible.
3. **Label type**: Choose **String**, **Boolean**, or **Integer**.
4. **Field name**: Enter the name of the log entry field that contains the value of the label. This field supports autocomplete.
5. **Extraction regular expression**: If the value of your label consists of the field's entire contents, then you can leave this field empty. Otherwise, specify a regular expression (regexp) that extracts the label value from the field value.

In this section, you'll explore

- Cloud Logging overview and architecture
- Log types and collection
- Storing, routing, and exporting the logs
- Query and view logs
- Using log-based metrics
- Log Analytics

Next us next take a look at one of the new feature in Cloud Logging, Log Analytics.

Perform analytics on log data using Log Analytics



Log Analytics gives you the analytical power of BigQuery within the Cloud Logging console and provides you with a new user interface that's optimized for analyzing your logs.

When you create a bucket and activate analytics on it, Cloud Logging makes the logs data available in both the new Log Analytics interface and BigQuery; you don't have to route and manage a separate copy of the data in BigQuery. You can still query and examine the data as usual in Cloud Logging with the Logging query language.

Cloud Logging and Log Analytics use cases

Troubleshooting

Logs Explorer

Get to the root cause with search, filtering, histogram and suggested searches.

Log Analysis

Log Analytics

Analyze application performance, data access and network access patterns.

Reporting

BigQuery link and Looker products

Use the same logs data in Log Analytics directly from BigQuery to report on aggregated application and business data found in logs.

- Logs are written to the Logging API via client libraries, stdout/fluentbit agent or directly via API,. Logs Explorer helps with troubleshooting and getting to the root cause with search, filter, histogram and suggested search.
- Log Router routes logs to the Logging Sink for the Logs Bucket. Log Analytics, analyze application performance, data access and network access patterns.
- Log Analytics pipeline maps logs to BigQuery tables (JSON, STRING, INT64, RECORD, etc..) and writes to BigQuery. Use the same logs data in Log Analytics directly from BigQuery to report on aggregated application and business data found in logs.

How different is analytics-enabled bucket log data from logs routed to BigQuery

- Log data in BigQuery is managed by Cloud Logging.
- BigQuery ingestion and storage costs are included in your Logging costs.
- Data residency and lifecycle are managed by Cloud Logging.

The logs data in your analytics-enabled buckets is different than logs routed to BigQuery via traditional export in the following ways:

- Log data in BigQuery is managed by Cloud Logging.
- BigQuery ingestion and storage costs are included in your Logging costs.
- Data residency and lifecycle are managed by Cloud Logging.

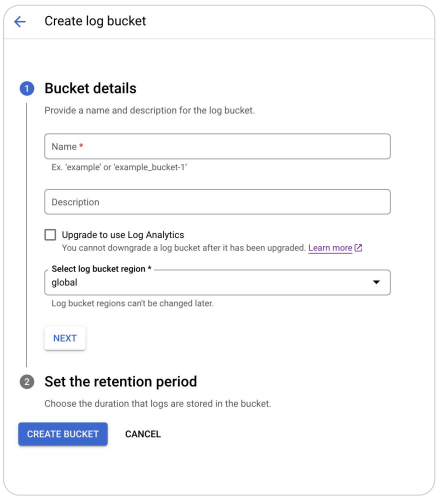
You can query your logs on Log Analytics-enabled buckets directly in Cloud Logging via the new Log Analytics UI. The Log Analytics UI is optimized for viewing unstructured log data. You can also access your logs data in BigQuery using a read-only view if you want to combine your logs data with other data in BigQuery.

You can turn on or off access to your analytics-enabled buckets in BigQuery by turning on or off the option that connects the logs to BigQuery. When the option is enabled, you can query the logs directly from BigQuery, including joining your logs data with other BigQuery datasets.

Creating a log-analytics enabled bucket

1. Create a log bucket with Log Analytics enabled.
2. Create a sink to route logs to the newly created bucket.
3. Check **Upgrade to use Log Analytics.**

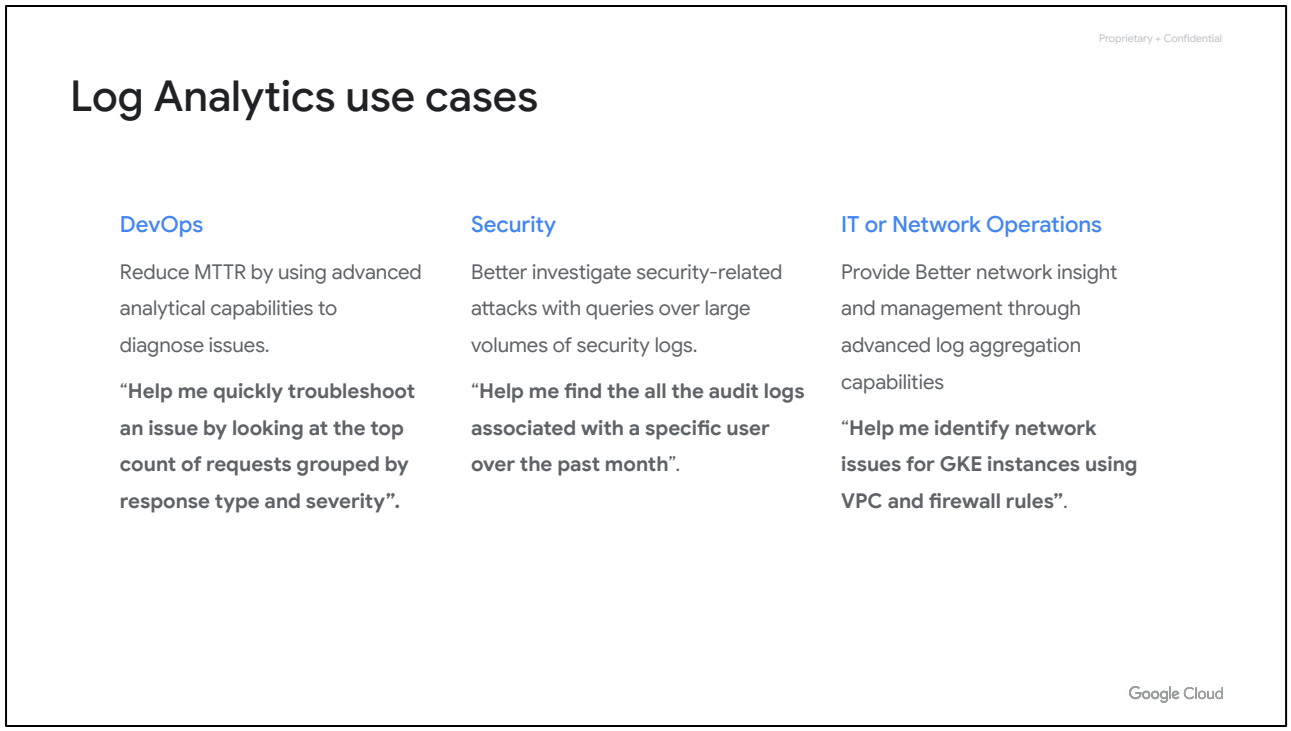
You can't downgrade the log bucket to remove the use of Log Analytics.



To create an analytics-enabled bucket by using the console:

1. Navigate to **Logs Storage**.
2. Click **Create log bucket**.
3. Select **Upgrade to use Log Analytics**.

Note: Upgrading a bucket to use Log Analytics is permanent. You can't downgrade the log bucket to remove the use of Log Analytics.

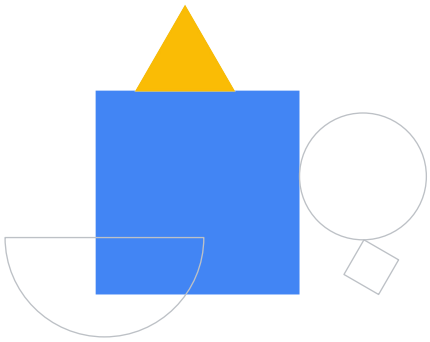


Log Analytics is useful in multiple aspects. Let's look at different fields' perspectives, starting with DevOps.

- **DevOps**
  - For a DevOps specialist it is important to quickly troubleshoot an issue that requires to reduce Mean Time to Repair (MTTR). Log Analytics includes capabilities to count the top requests grouped by response type and severity, which allows engineers to diagnose the issues.
- **Security**
  - A security personal is interest in finding all the audit logs associated with a specific use over the past month. Log Analytics help better investigate the security -related attacks with queries over large volumes of security data. For more information, refer to the documentation.
- **IT/Network Operations**
  - A IT/Network Operations is interested in identifying network issues for GKE instances that are using VPC and firewall rules. Log Analytics in this case provides better network insights and management through advanced log aggregation capabilities. For more information, refer to the documentation.

Lab Intro

Log Analytics on Google Cloud



In this lab you will learn about the features and tools provided by Cloud Logging to gain insight of your applications.

Knowledge Check

Quiz | Question 1

Question

You want to compare resource utilization for VMs used for production, development, and testing. What should you do?

A. Add a label called “state” to your VMs with the values “dev”, “test”, and “prod” and group by that label in your monitoring chart.

B. Put those resources in different projects and use Dataflow to create an aggregation of log values for each.

C. Name the VMs with a prefix like “dev-”, “test-”, and “prod-” and filter on the name property when reporting.

D. Export all machine logs to Cloud Storage and use Cloud Run functions to build reports based on the VM tags.

Quiz | Question 1

Answer

You want to compare resource utilization for VMs used for production, development, and testing. What should you do?

A. Add a label called “state” to your VMs with the values “dev”, “test”, and “prod” and group by that label in your monitoring chart. 

B. Put those resources in different projects and use Dataflow to create an aggregation of log values for each.

C. Name the VMs with a prefix like “dev-”, “test-”, and “prod-” and filter on the name property when reporting.

D. Export all machine logs to Cloud Storage and use Cloud Run functions to build reports based on the VM tags.

Quiz | Question 2

Question

Your manager wants a daily report of resource utilization by application. Where would the best export sink be?

A. Cloud Storage

B. Pub/Sub

C. BigQuery

D. Spanner

Quiz | Question 2

Answer

Your manager wants a daily report of resource utilization by application. Where would the best export sink be?

A. Cloud Storage

B. Pub/Sub

C. BigQuery 

D. Spanner

Recap

01 Use Log Explorer features

02 Explain the features and benefits of log-based metrics

03 Define log sinks (inclusion filters and exclusion filters)

04 Explain how BigQuery can be used to analyze logs

05 Use Log Analytics on Google Cloud

In this module, you learned how to:

- Use Log Explorer features
- Explain the features and benefits of log-based metrics
- Define log sinks (inclusion filters) and exclusion filters
- Explain how Big query can be used to analyze logs
- Export logs to BigQuery for analysis
- Use Log Analytics on Google Cloud

Reference materials

- Log Analytics samples