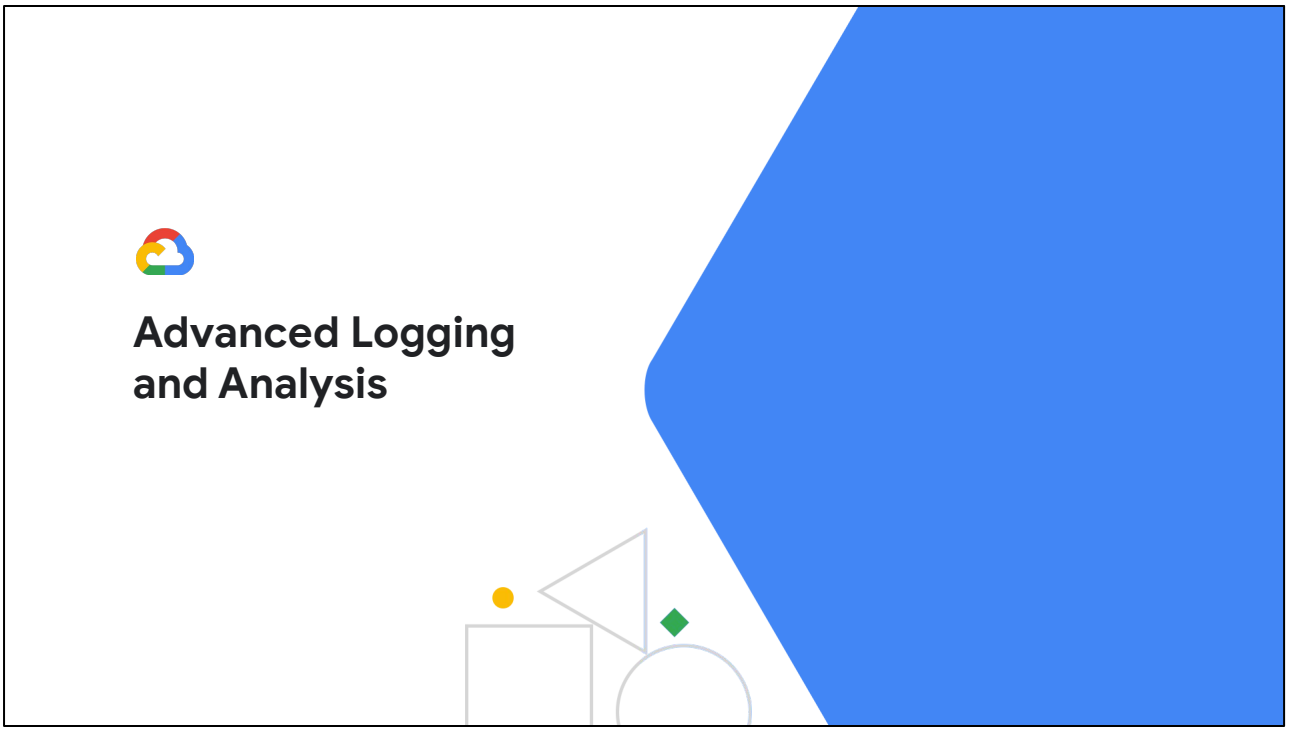


In this module, we examine some of Google Cloud's advanced logging and analysis capabilities.

Agenda

Strategic Logging

- Working with the Log Viewer
- Using Logs-based Metrics

Exporting and Analyzing Logs

Google Cloud

Specifically, in this module you learn to:

- Identify and choose among resource tagging approaches because tags can make locating and tracking resources easier.
- Define log sinks (inclusion filters) to include specific log entries, and exclusion filters to exclude others.
- Create monitoring metrics based on log entries. For example, "this NGINX log entry has appeared x times in the last minute," so we can now tell how many requests our website took.
- And Export logs to BigQuery for long-term storage and SQL-based analysis.

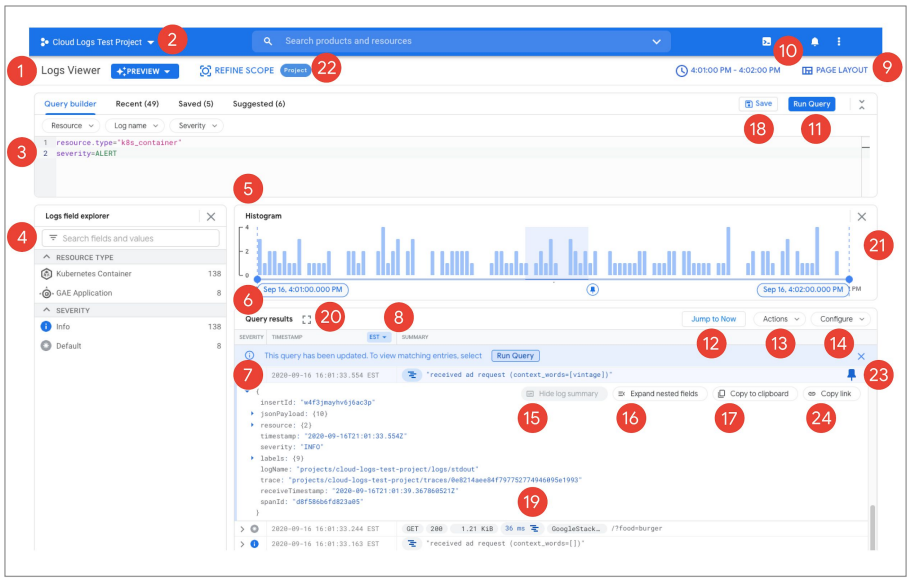
Agenda

- Strategic Logging
  - Working with the Log Viewer
  - Using Logs-based Metrics
- Exporting and Analyzing Logs

Google Cloud

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.

Logs Viewer



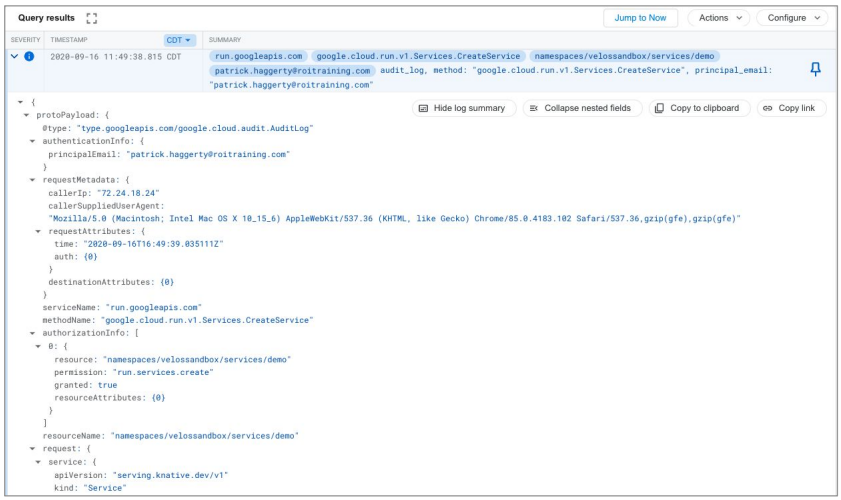
Google Cloud

To start using Google Cloud Logging, start with the Log Viewer. Let's take a tour:

1. **Logs Viewer page**: Lets you build, analyze, and refine queries.
2. **Organization and project selector**: Lets you view logs at an organization or project level.
3. **Query builder**: Lets you build queries using either the drop-down menus or the query builder language. It also features tabs for viewing your Saved and Recent queries.
4. **Logs field explorer**: Lets you see aggregation-based results for the resource.type, resource.labels, logName, and severity fields, and provides a more efficient way to refine a query.
5. **Histogram**: Lets you visualize the frequency of your logs data.
6. **Query results**: Lets you view the retrieved logs from your query.
7. **Log entries**: Lets you view log entries in the structured JSON format.
8. **Time zone**: Lets you change the time zone that logs are displayed in.
9. **Page layout**: Lets you enable and disable the **Histogram** and **Logs field explorer** panels.
10. **Time-range selector**: Lets you restrict results by time range. The default time range is one hour.
11. **Run query**: Lets you run your queries after you have built them in the query-builder pane.
12. **Jump to now**: Lets you perform a forced refresh to include the current time. If the time-range selector uses a custom range and an end time is set, it runs the query with a default time range of one hour. Otherwise, it refreshes with the

1. current start date or duration, and runs the query.
2. **Actions**: Lets you perform certain actions on your logs, such as creating a logs-based metric or a sink destination.
3. **Configure**: Lets you add the value of a log field to the summary line at the beginning or end of the log entry. It also lets you choose to show newest logs either first or last.
4. **Hide log summary**: Lets you hide the log summary line from the query results.
5. **Expand or collapse nested log fields**: Lets you expand or collapse nested fields.
6. **Copy to clipboard**: Lets you copy the log entry in its JSON format.
7. **Save**: Lets you save queries that can be viewed and run from the **Saved** tab.
8. **Trace data**: Lets you view trace details and refine your query based on the trace.
9. **Expand and collapse query results**: Lets you expand the query-results pane to view more log entries.
10. **Adjust time range**: Lets you change the time range used for queries by adjusting the handles. After adjusting the handles, click **Run** to update the time range used in the query.
11. **Refine scope**: Lets you scope your search by logs in your current project only or by one or more storage views.
12. **Pin log entry**: Lets you pin a log entry to the **Query results** and **Histogram** panes. Depending on how your **Query results** pane is configured, Logging pins the log either to the top or to the bottom of the **Query results** pane.
13. **Copy link to a log entry**: Lets you share a link to a log entry.

Entries are returned as LogEntry objects



Google Cloud

The entries returned in the Logs viewer are based on Google's LogEntry datatype. They contain data like the logName, severity, resource.type, and various payload fields.

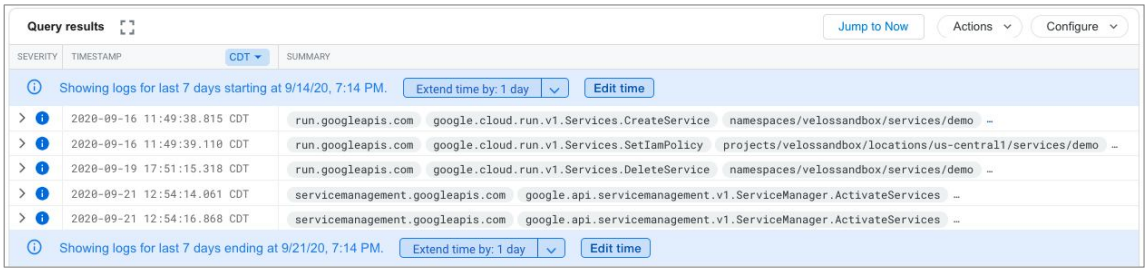
Primary log fields

logName	Resource name of the log to which this log entry belongs (ex: projects/[PROJECT_ID]/logs/[LOG_ID])
insertId	Unique identifier
severity	Entry severity, defaults to LogSeverity.DEFAULT
timestamp/receiveTimestamp	The time the event described by the log entry occurred/was received by Logging
resource.type	The name of a resource type. Example: gce_instance
resource.labels.KEY	The value associated with a resource label key
httpRequest.FIELD	The value of a field in an HttpRequest object (method, url, size, status, etc.)
labels.KEY	The value associated with a label key
operation.FIELD	The value of a field in a LogEntryOperation object
protoPayload.FIELD	Log entry payload represented as a protocol buffer
jsonPayload.FIELD	The value of a field within a JSON object
textPayload	The log entry payload, represented as a Unicode string (UTF-8)

Google Cloud

Here you see common LogEntry properties. Note that *textPayload*, *jsonPayload*, and *textPayload* are mutually exclusive. Also, the information that is typically most interesting and/or most relevant will be found in the provided payload section.

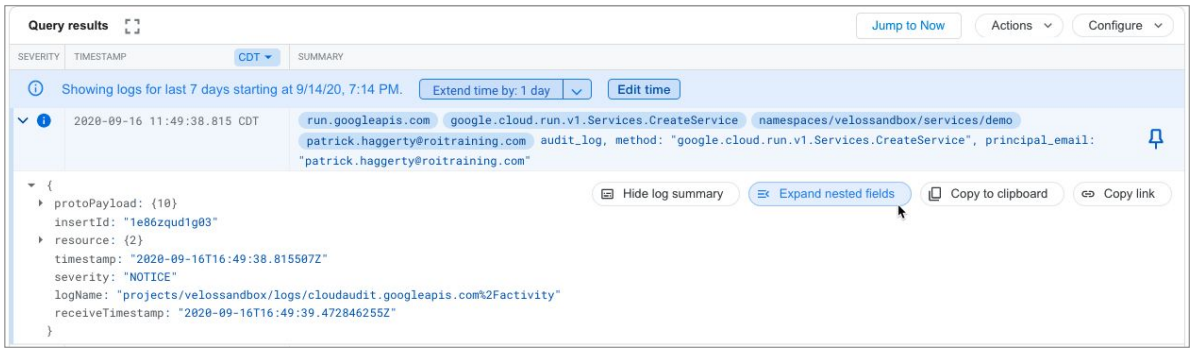
Log entries



Google Cloud

The log-entry table displays an entry line for each log entry. In the line, you see the entry severity, timestamp, and any values for fields that have been promoted to the summary.

Log entry details



Google Cloud

To see the full details for one log entry, click the expander arrow (▸) at the front of the summary line, and then click **Expand nested fields**. The log entry is displayed using JSON format.

Locate (or hide) similar entries

Google Cloud

You can click the value of a specific field in the expanded log entry view and then either show or hide all log entries with the same value. Doing so will modify the log query appropriately.

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 file, or has a known severity, use the query builder drop-down menus

Google Cloud

Ultimately, it's the query that selects the entries displayed by the Logs Viewer. 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.

Using the query builder drop-down menu

Google Cloud

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

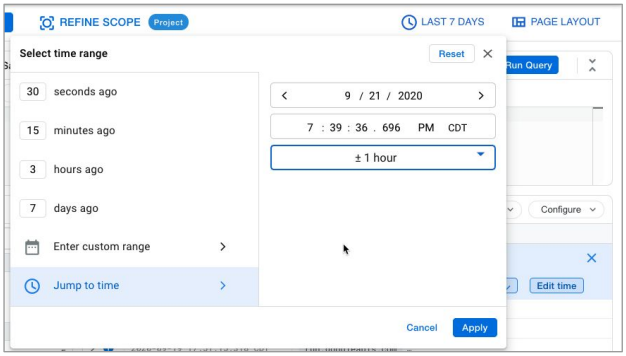
- **Resource**: Lets you specify resource.type. You can select a single resource at a time to add to the **Query builder**. Entries use the logical operator AND.
- **Log name**: Lets you specify logName. You can select multiple log names at once to add to the **Query builder**. When selecting multiple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

Advanced filter comparison operators

=	**Equals**	`resource.type="gce_instance"`
!=	**Does not equal**	`resource.labels.instance_id!="1234567890"`
<=	**Less than equal**	`timestamp <= "2018-08-13T20:00:00Z"`
>=	**More than equal**	`timestamp >= "2018-08-13T20:00:00Z"`
>	**More than**	`timestamp > "2018-08-13T20:00:00Z"`
<	**Less than**	`timestamp < "2018-08-13T20:00:00Z"`
:	**Has**	`textPayload:"GET /check"`

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

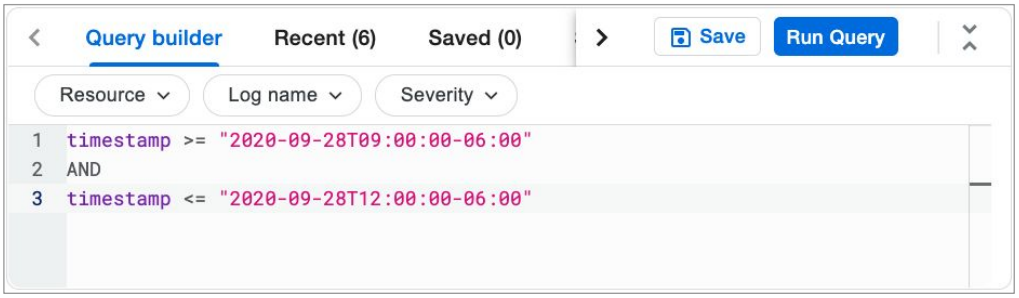
Finding log entries, set the time range



Google Cloud

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 +/- an amount.

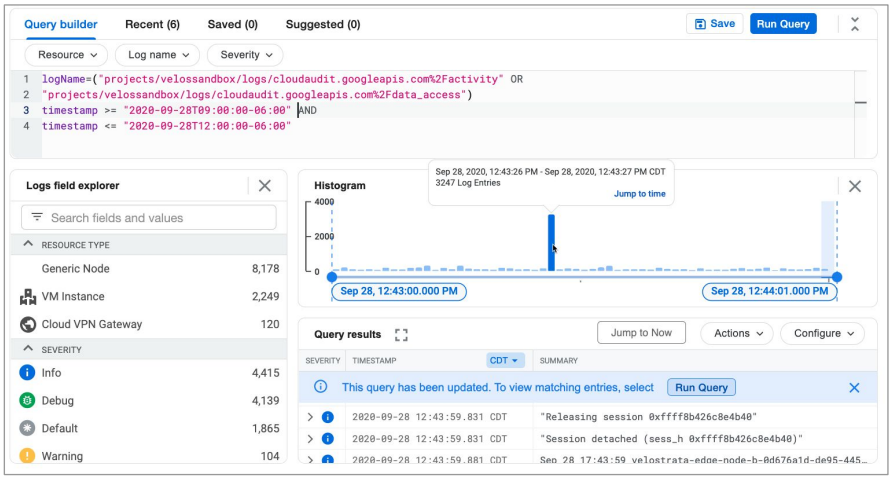
You can also manually restrict the time range



Google Cloud

You can also manually restrict the time range using the timestamp keyword, a comparator, and a time in RFC 3339 format.

Narrow with Logs field explorer and Histogram



Google Cloud

The **Logs field explorer** 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 **Logs field explorer** 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 termination 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. This makes it easier to see trends in your logs data and troubleshoot problems.

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

Advanced filter boolean expressions

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

Google Cloud

Advanced queries support the AND, OR, and NOT boolean expressions for joining queries.

Advanced filter syntax

- *a* **=** *e* means that *a* is a name for the expression *e*.
- *a b* means "*a* followed by *b*."
- *a* | *b* means "*a* or *b*."
- **(** *e* **)** is used for grouping.
- **[** *e* **]** means that *e* is optional.
- **{** *e* **}** means that *e* can be repeated zero or more times.
- "abc" means that **abc** must be written just as it appears.

And, advanced queries can support grouping, optional sections, and repeated values.

The recipe for finding entries

- What do you know about the log entry?
  - Log file, resource, a bit of text?
- Full text searches are slow, but may be effective:
  - `"/score called"`
- If possible, restrict text searches to an entry region:
  - `jsonPayload:"/score called"`
  - `jsonPayload.message="/score called"`

When you're trying to find log entries, start with what you know: the log file name, 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".

Finding entries quickly

- Search on an indexed field
  `httpRequest.status, logName, operation.id, resource.type, timestamp, severity, resource.labels`
- Be specific on which logs you are searching
  `logName="projects/benkelly-test/logs/apache-access"`
- Limit the time range you are 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 log entry's name, resource type, and resource labels.
- As seen in the example, be specific on which logs you are searching by referring to it or them by name.
- Limit the time range you are searching to lessen the log data being queried.

Agenda

Strategic Logging

- Working with the Log Viewer
- Using Logs-based Metrics

Exporting and Analyzing Logs

Google Cloud

Now that we've seen how the Logs Viewer works, let's talk about generating monitoring metrics from logging data.

Key access control roles

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

Google Cloud

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

First, on the logging side:

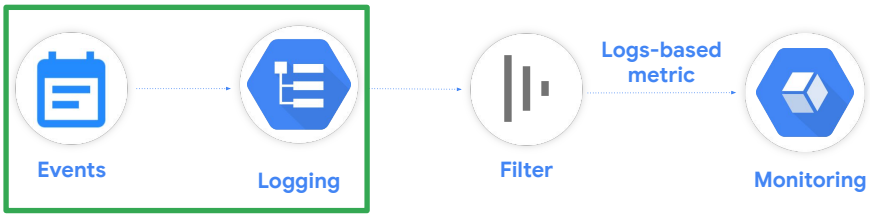
- **Logs Configuration Writers** can list, create, get, update, and delete logs-based metrics.
- **Logs Viewers** can view existing metrics.

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

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

Lecture Notes:
There are IAM permissions for logs at the project level. You cannot allow one person to view some logs and not other logs.
Things are planned to change soon

Logs-based metrics

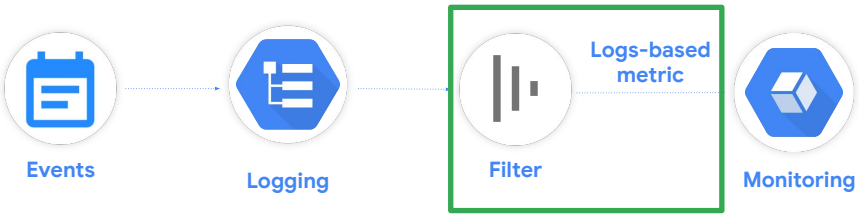


Google Cloud

Logs-based metrics are Cloud Monitoring metrics that are based on the content of log entries.

Resources generate logging events that are streamed into Google Cloud Logging.

Logs-based metrics

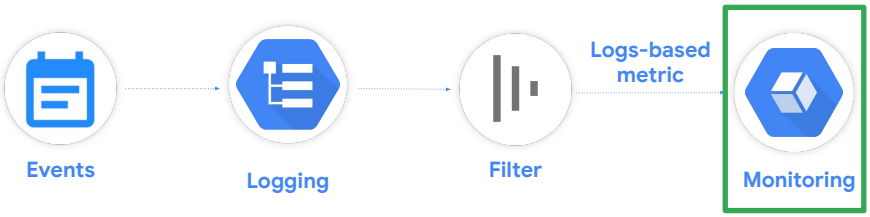


Google Cloud

Logs-based metrics apply a **filter** to locate particular entries.

For example, the metrics might record the number of log entries containing particular messages, or that were generated by a particular resource.

Logs-based metrics



Google Cloud

Once created, you can use logs-based metrics in Cloud Monitoring charts and alerting policies.

Don’t reinvent the wheel!

- Google has a curated list of over 1,000 predefined metrics
  - Check there first!
- After that, can metrics be created from application logs?
  - Logs-based metrics
- Only create custom metrics when it makes sense
  - Remember, they are also charged differently

Google Cloud

Once again, don't reinvent the wheel.

Before creating your own custom logs-based metrics, take a look at Google's curated list of pre-existing metrics. There are over 1,000, and what you're looking for might already be there.

Custom metrics should generally only be used if the metric you need is not already available, or if you need to create application metrics.

Using custom metrics may also increase your monitoring and logging costs.

Basic 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?`);
 });
```

Google Cloud

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

Basic 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?`);
 });
```

Google Cloud

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

Basic 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?`);
 });
```

Google Cloud

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.

Basic 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?`);
 });
```

Google Cloud

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

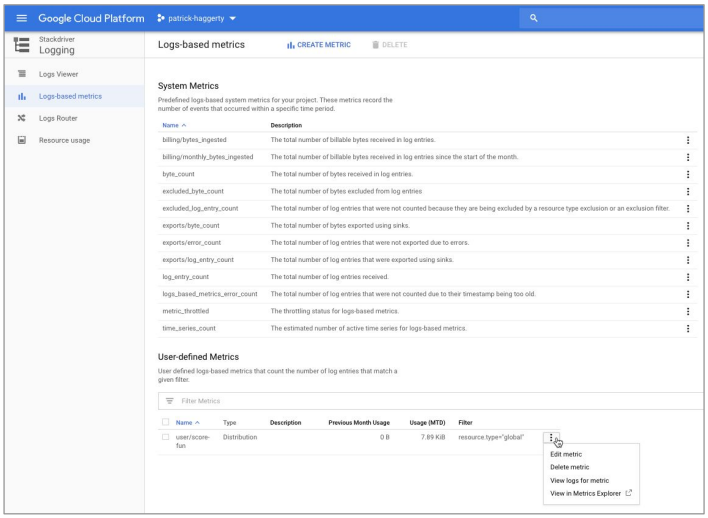
Basic test code

```
//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?`);
 });
```

Google Cloud

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

Logs-based metrics

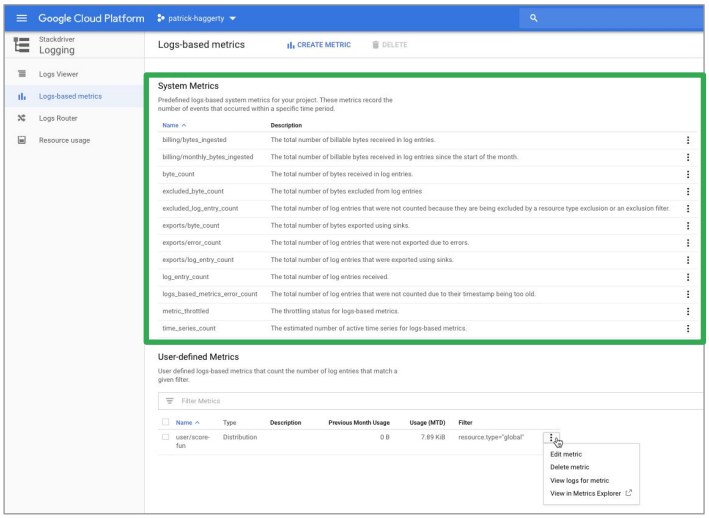


Google Cloud

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

There are two fundamental logs-based metric types:

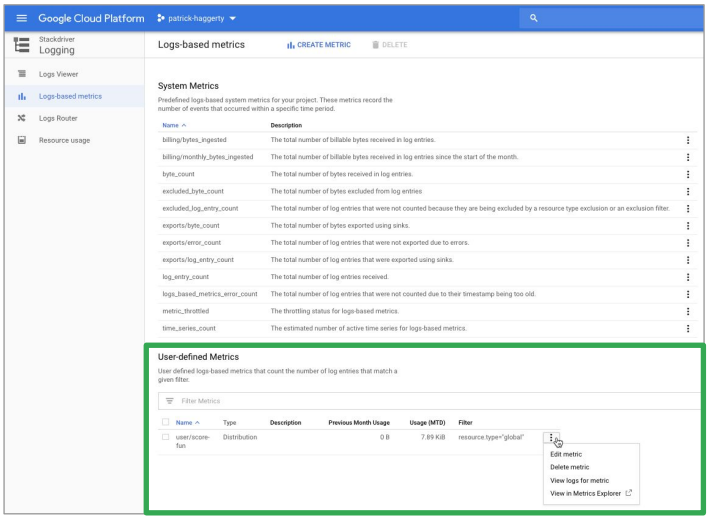
Logs-based metrics



Google Cloud

System logs-based metrics which are predefined by Google and are a standard part of logs-based metrics.

Logs-based metrics

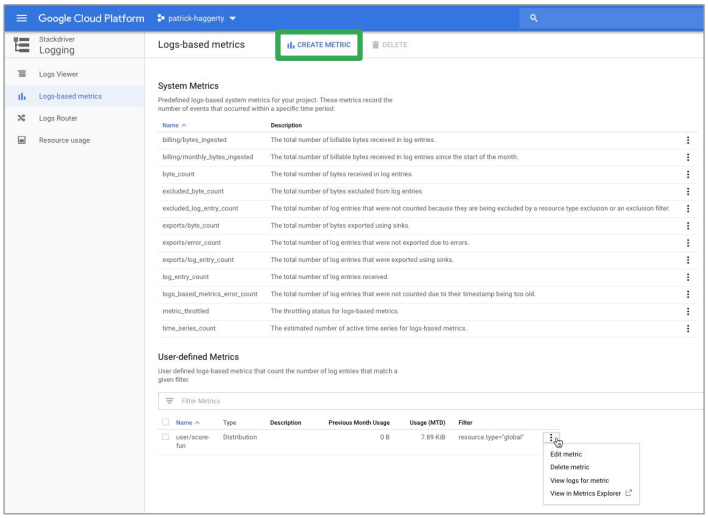


Google Cloud

And then there are **User-defined logs-based metrics,** which are created by a user on a project.

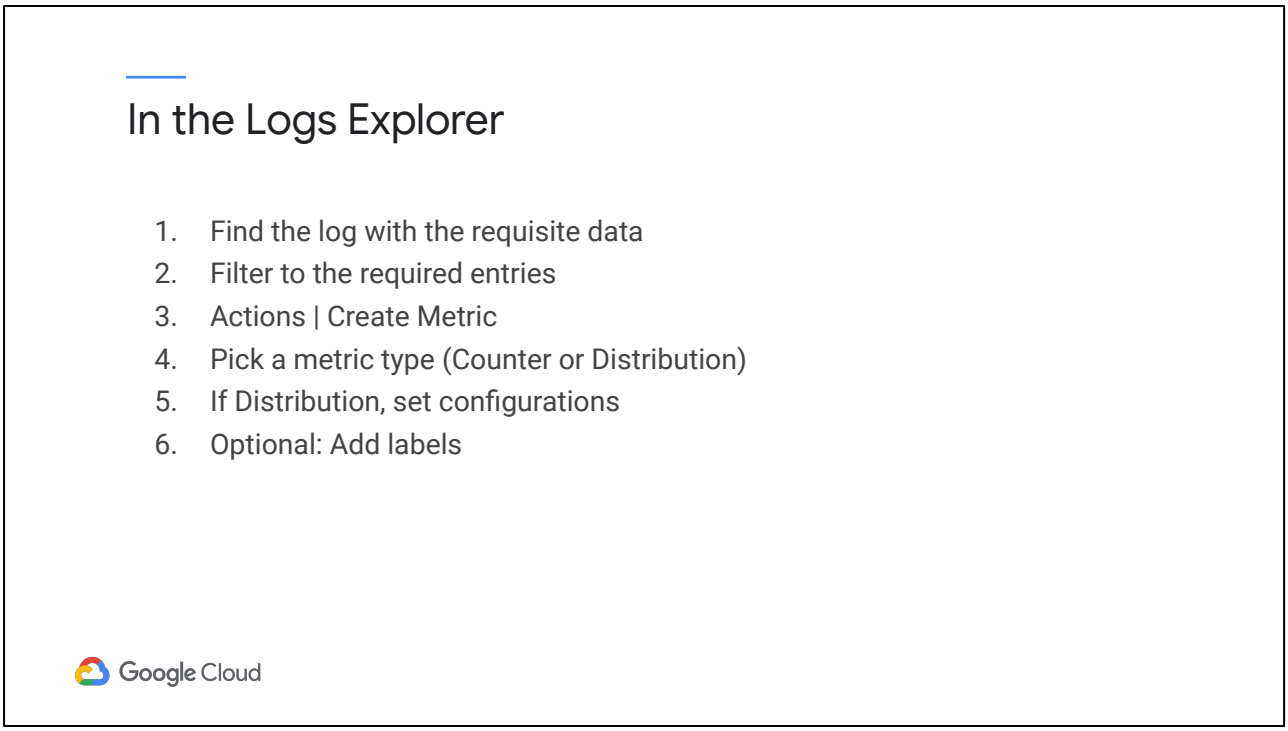
These count the number of log entries that match a given query or keep track of particular values within the matching log entries.

Logs-based metrics



Google Cloud

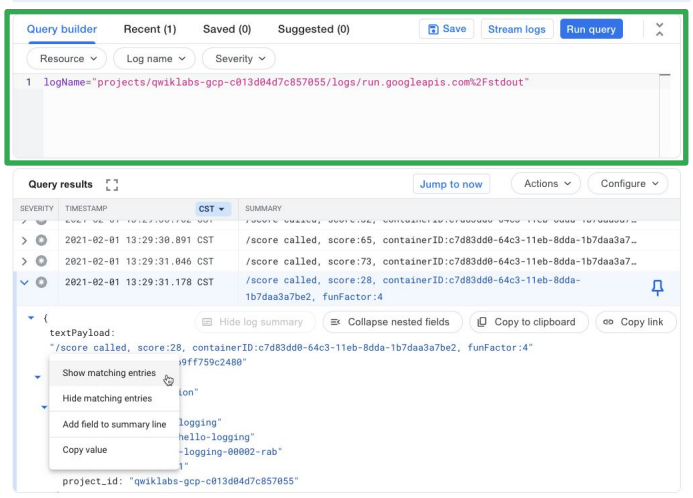
The latter is what we are creating now. You'll note the **Create Metric** button at the top of the interface.



The basic flow for creating logs-based metrics goes something like this:

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

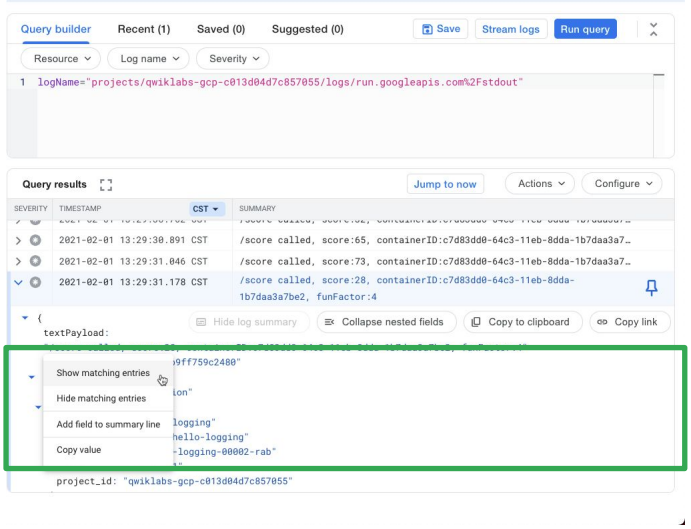
Filtering Entries



Google Cloud

Use the Query builder to access project logs

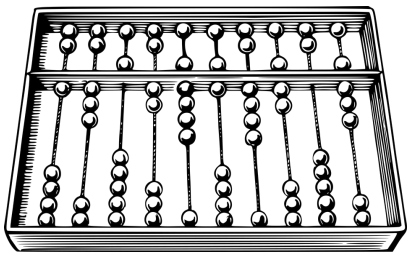
Filtering entries



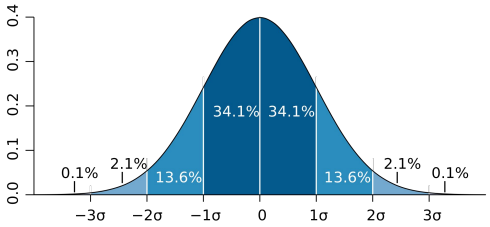
Google Cloud

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

Logs-based metric types



Counter



Distribution

Google Cloud

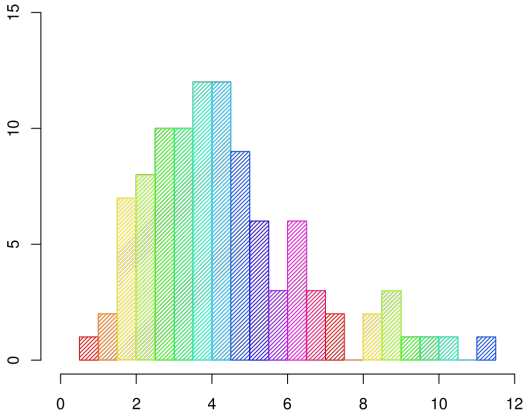
Logs-based metrics can be one of two metric types: **counter** or **distribution**. All predefined system logs-based metrics are the counter type, but user-defined metrics can be either counter or distribution types.

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 records the statistical distribution of the extracted log values in histogram buckets. The extracted values are not recorded individually, but their distribution across the configured buckets are recorded, along with the count, mean, and sum of squared deviations of the values.

Lecture Notes:
Sampling time cannot be set while creating log based metric, nor is it disclosed.

A distribution as a histogram



- Linear: buckets of fixed width
- Exponential:
  - N+2 buckets
  - Upper: scale * (growthFactor ^ i)
  - Lower:scale * (growthFactor ^ (i-1))
- Explicit: Array of bucket boundaries
- (Up to 200 buckets)

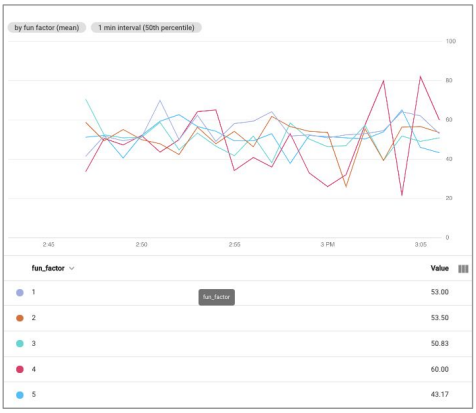
Google Cloud

Distribution metrics include a histogram that counts the number of values that fall in specified ranges (buckets). There are three different ways to specify the boundaries between histogram buckets:

- **Linear**: Every bucket has the same width.
- **Exponential**: Bucket widths increase exponentially for higher values.
- **Explicit**: You list all the boundaries for the buckets in the *bounds* array.

Whichever the methodology, a distribution will support up to 200 buckets.

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



Google Cloud

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

Labels and logs

- Allows for logs-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 10 user-defined labels per metric
- A label cannot be deleted once created
  - And will grow time series significantly

Google Cloud

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

All logs-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 label's value 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).

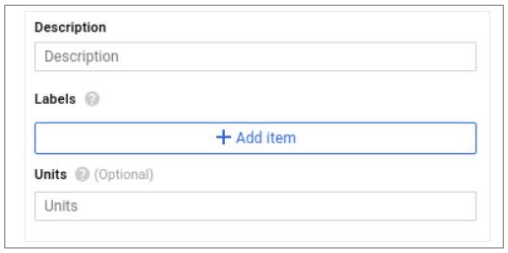
You can extract labels from the LogEntry built-in fields, such as httpRequest.status, or from one of the payload fields, textPayload, jsonPayload, or protoPayload.

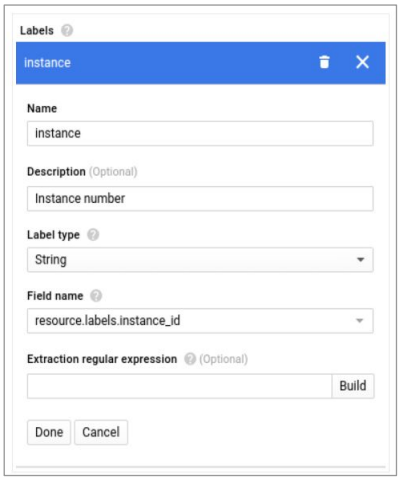
Label with care. A metric can support up to 10 user-defined labels, and once created, a metric cannot be removed. Also, each logs-based metric is limited to about 30,000 active time series.

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 logs-based metric





Google Cloud

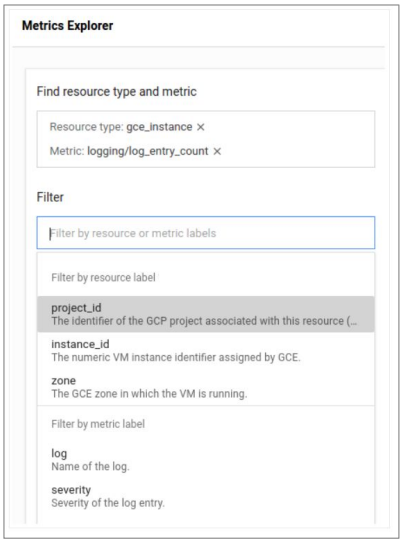
User-defined labels can be created when creating a logs-based metric. The label form requires:

1. **Name**: The identifier which will be used when using the 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 label's value. This field supports autocomplete.
5. **Extraction regular expression**: If your label's value 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.

Lecture Notes:
This is the old UI. This has changed.

Using labels

- User-based metrics can be seen by using filters within Metrics Explorer



Google Cloud

Once created, the label and its time series will be available through the Metrics Explorer and other monitoring services. In this example, you can see the metric is set to **log_entry_count,** and at the bottom, you can filter by the **log** name or **severity** labels.

Agenda

Strategic Logging

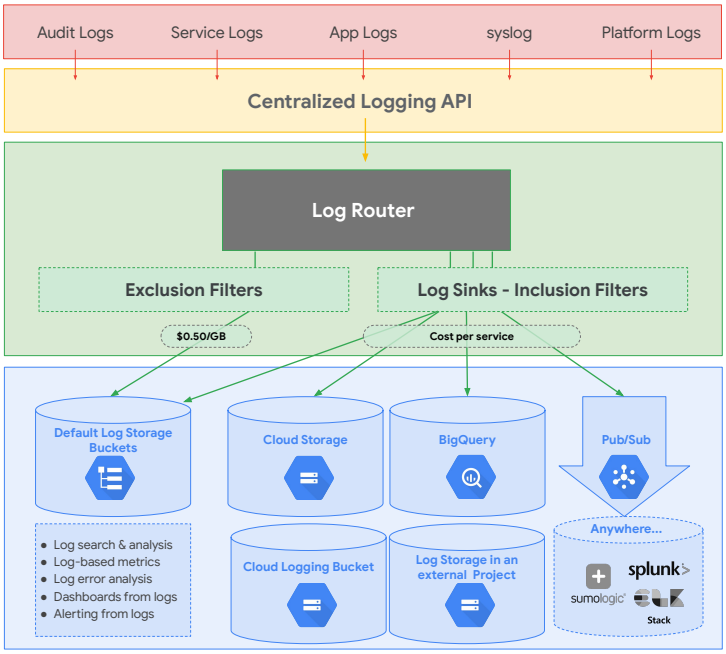
- Working with the Log Viewer
- Using Logs-based Metrics

Exporting and Analyzing Logs

Google Cloud

Now that we understand the core elements involved in strategic logging, let's look at how logs can be exported for long-term storage and analysis.

Logging architecture



Google Cloud

What we call Google Cloud Logging is actually a collection of components exposed through a centralized logging API. Entries are passed through the API and fed to the 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.

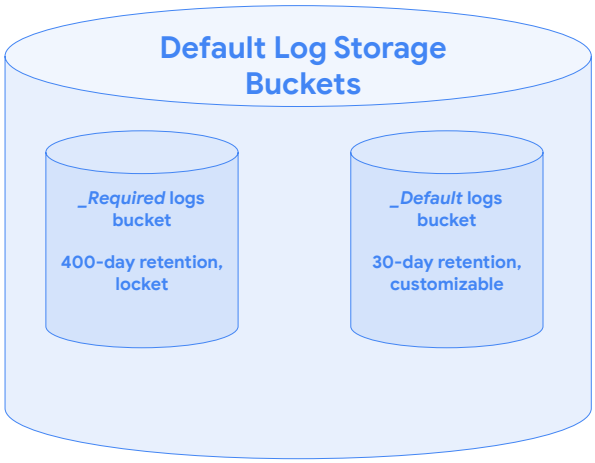
By default, log entries are fed into one of the default log storage buckets. Exclusion filters may be created to partially or totally prevent this behavior.

Log Sinks run in parallel with the default log flow and may be used to direct entries to external locations, including 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.

Lecture Notes:
Why would you exclude entries to logs? Ans: Because of cost considerations. But if logs does not get into the default log storage, you cannot analyze those logs.
Required Logs bucket is always free

Default logs buckets



Google Cloud

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:

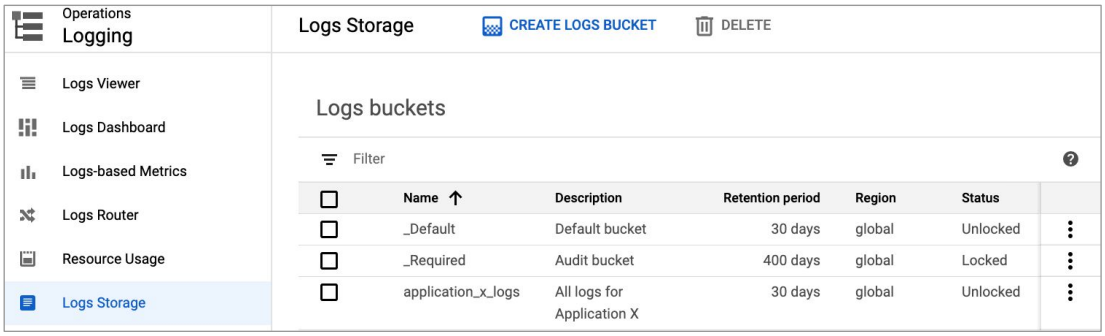
- *_Required*: This bucket holds Admin Activity audit logs, System Event audit logs, and Access Transparency logs, and retains them for 400 days. You aren't charged for the logs stored in *_Required*, and the retention period of the logs stored here cannot be modified. You cannot delete this bucket.
- *_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.

Use gcloud to adjust the retention:

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

Note: Effective March 31, 2021, storage costs will apply to all chargeable logs retained longer than the default retention periods at the rate of $.01 per GiB per month (or fraction thereof). For details, see Logs storage pricing.

Create specialized buckets in current or remote projects:



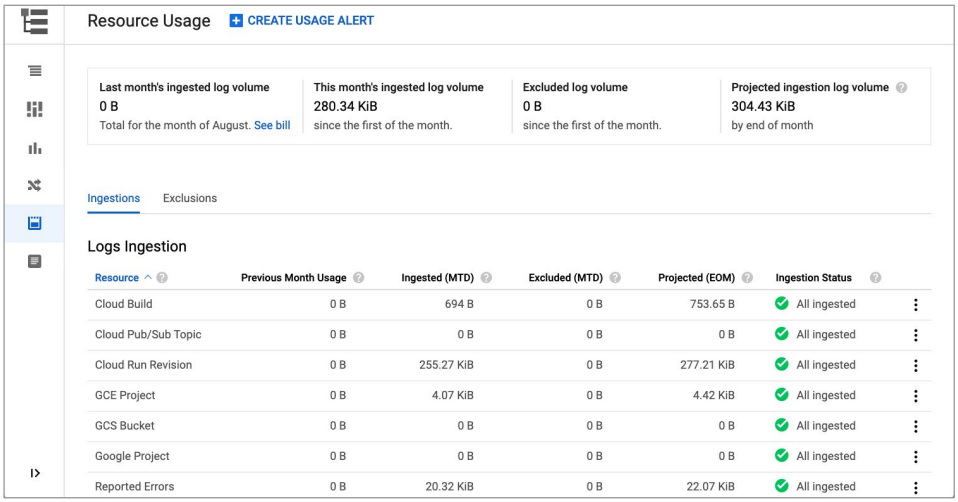
Google Cloud

Logs buckets are containers in your Google Cloud projects that hold your logs data. You can create logs sinks to route all, or just a subset, of your logs to any logs bucket. This flexibility allows you to choose which Google Cloud project your logs are stored in and what other logs are stored with them. Log buckets may also be placed in specific regions for regulatory compliance. Using the gcloud command-line tool and the Google Cloud Console, you can create, update, and delete your custom logs buckets.

Lecture Notes:
This is a recent addition. This also helps in restricting logs to a particular storage region (GDR), and also the retention period.
If you build a special bucket - you would have to go to Logs router and build a sink to send log entries to the special bucket
They are about to introduce "Log Views" which allows you to selectively give access to logs to persons
At this point in time you can restrict view access at the log bucket/storage level? (Need to confirm)
Currently these log storage bucket is not visible today under Cloud storage, but will probably soon be available there.

Org policy can specify the region.

Resource usage

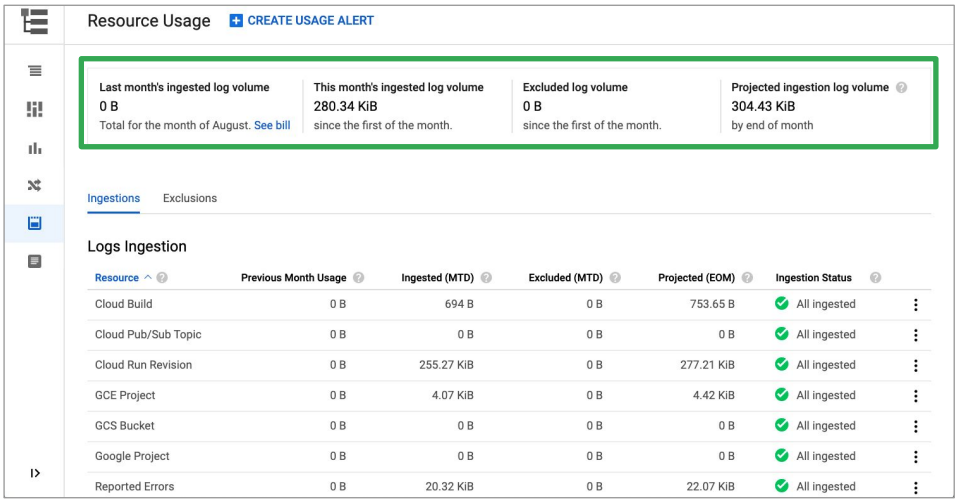


Google Cloud

To track your project's logs volume, go to the **Resource usage** page in the Cloud Logging console.

Lecture Notes:
This page no longer exists. So ignore

Resource usage

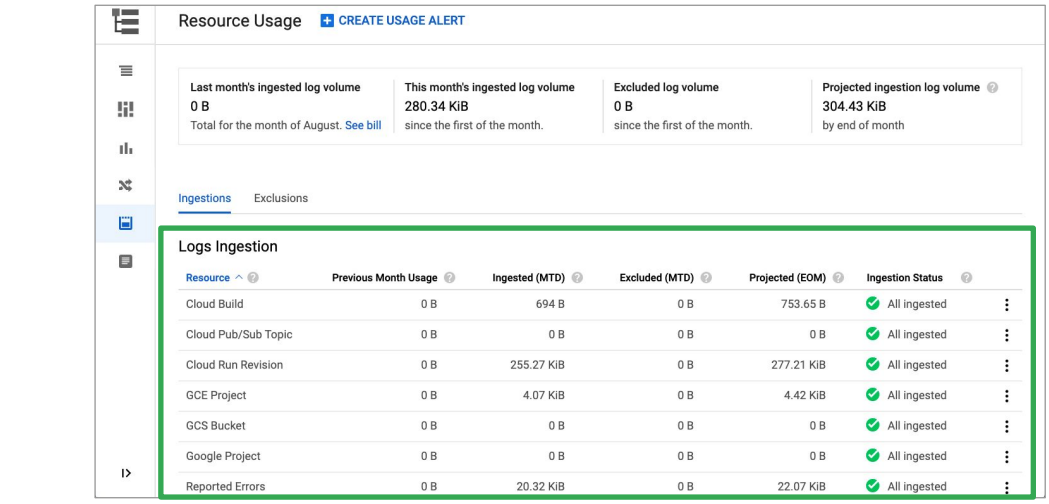


Google Cloud

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

- **Last month's ingested log volume**: The amount of logs your project received in the last calendar month.
- **This month's ingested log volume**: The amount of logs your project has received since the first date of the current month.
- **Excluded log volume**: The amount of logs that you have excluded from your project since the first date of the current month. This number is not included in **This month's ingested log volume**. Excluding logs is covered on the next slide.
- **Projected ingestion log volume**: The estimated amount of logs your project will receive by the end of the current month, based on current usage.

Resource usage



Google Cloud

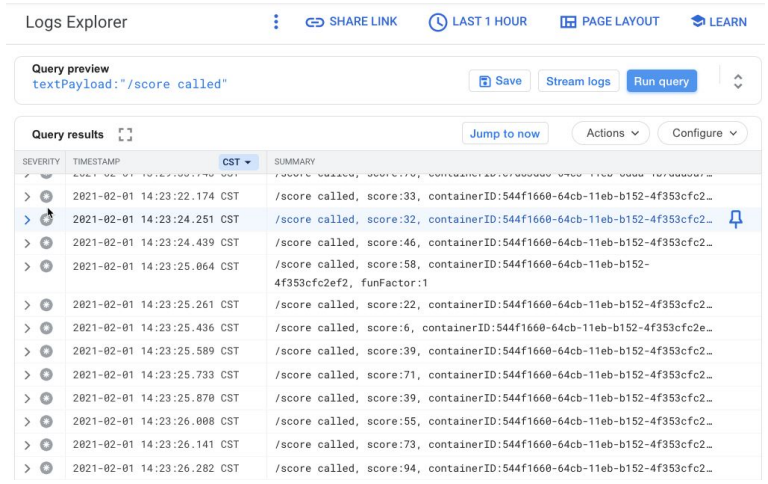
Below that, you have two tabs showing the **Ingestions** and **Exclusions** by type.

One note: The log volumes don't include Admin Activity audit logs or all System Event audit logs.

Those logs are free and cannot be excluded or disabled.

To create logging exclusions, start by clicking the **Exclusions** tab.

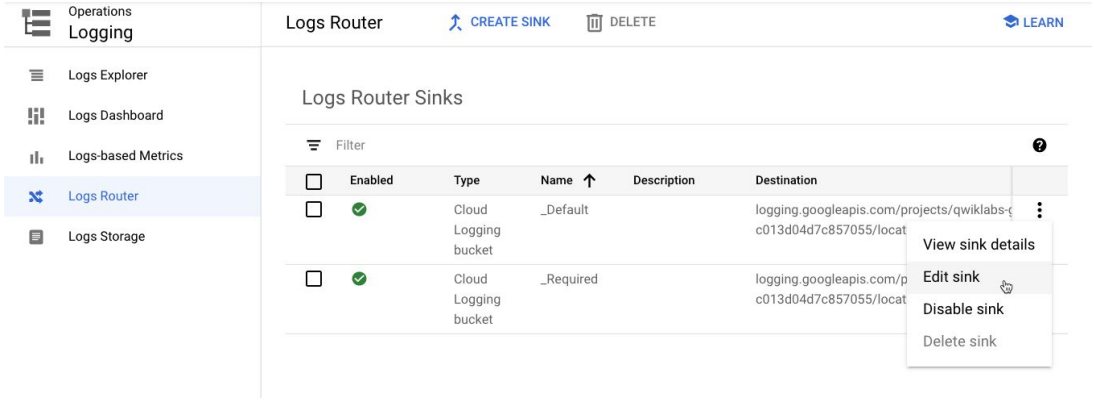
Exclusions - Identify log entries



Google Cloud

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

Exclusions - Edit the target log sink

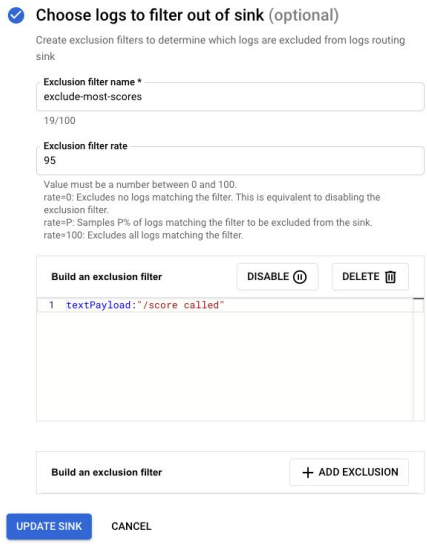


Google Cloud

Use the "hamburger menu" to the right of the target log sink to initiate editing of that entity

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

Exclusions - Build the exclusion



Google Cloud

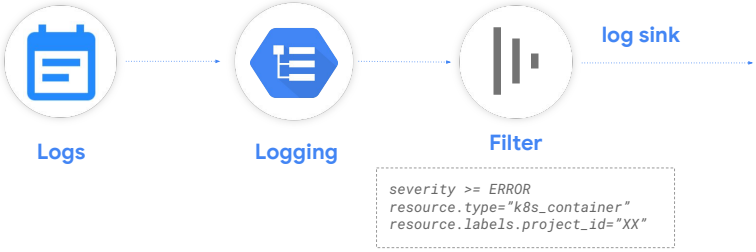
Use the **Log Explorer** query to create an exclusion filter that filters the unwanted entries out of the sink. Give the exclusion a name and description and decide the percentage of log entries to exclude.

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

Create the exclusion and it will go into effect immediately.

Lecture Notes:
Exclusion filter rate - is used when you want to only get a sample population from the overall logs (think very large logs like load balancer logs)

Log router sinks

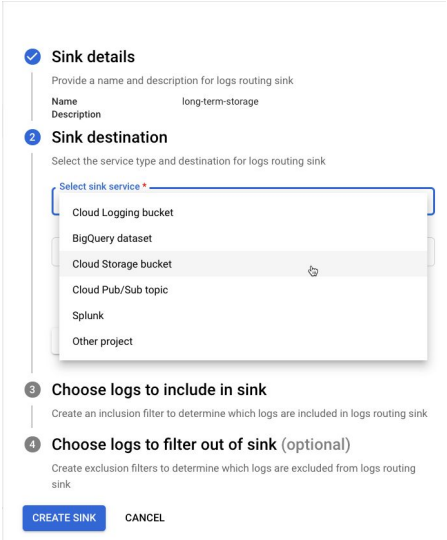


Google Cloud

Logs 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 creating a logs 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

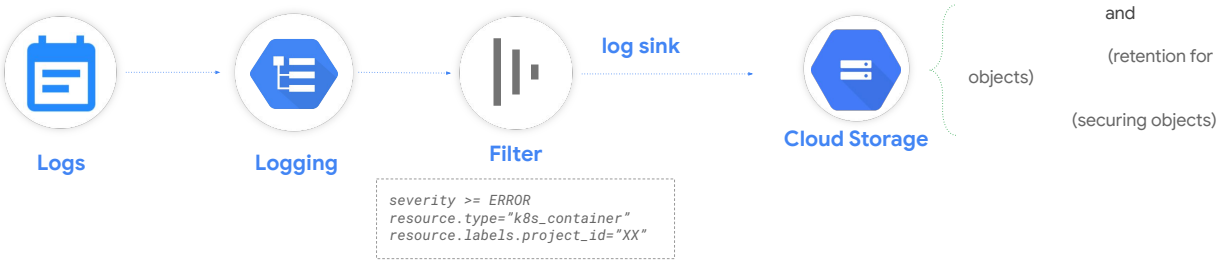


Google Cloud

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.
- **Cloud 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 Functions.
- **Splunk** for integration of logs into existing Splunk-based system.
- **Other project** is useful to help control access to a subset of log entries.

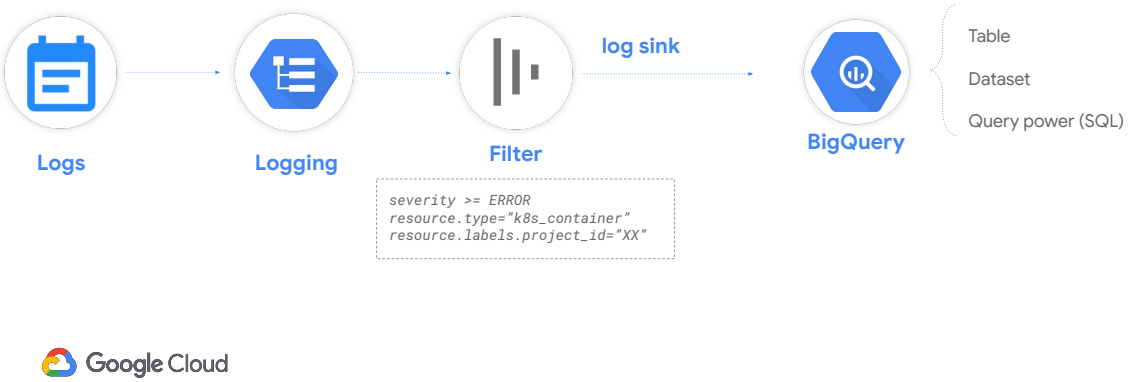
Cloud Storage works well for general storage:



Google Cloud

Cloud storage works well for general log storage. It allows the control of bucket location, storage class, object lifecycle rules, and bucket locks. The files can also be easily processed with a number of tools and products.

BigQuery for easy warehousing and analysis

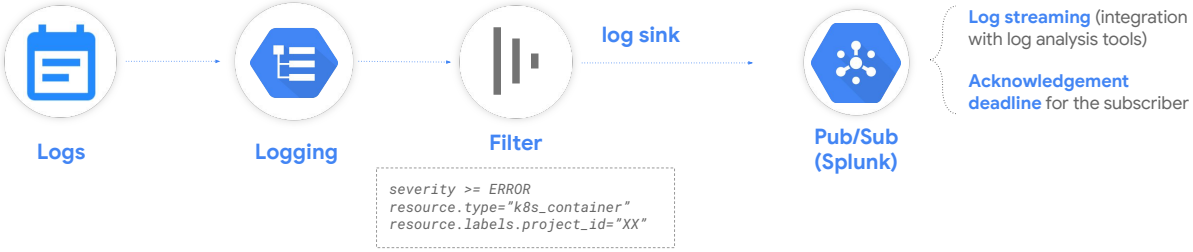


Google Cloud

BigQuery is a common log sink because it allows both long-term, cost-effective storage and the ability to implement powerful analytics with SQL queries. BigQuery also supports analysis with machine learning, and works well as a back end for visualization.

Lecture Notes:
Log Query is good for general purpose queries. But if you need high end, sophisticated queries, BigQuery is the best fit

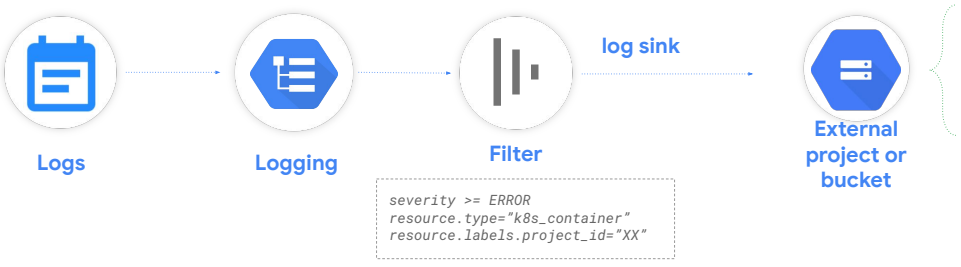
Pub/Sub to connect with external systems and applications



Google Cloud

Pub/Sub allows logging events to be streamed asynchronously to code, event processing pipelines created in Cloud Functions, Cloud Run, or Dataflow, or to third-party log analysis tools.

Log exports



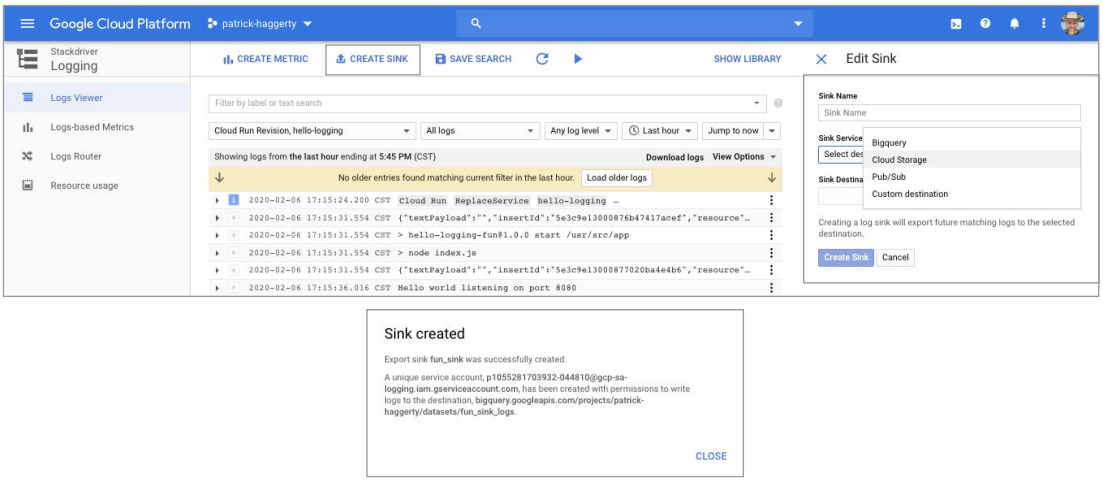
Google Cloud

Using a specialized bucket or external project is a great way to limit access and to control log entry location.

Lecture Notes:
Use-case - this is for securing logs - allow users to view only the logs, not anything else. (Today access to logs is at the project level)

With the new "log views" capability, this may no longer be needed

Create a log sink



Google Cloud

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 the Logs Viewer, 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.

Lecture Notes:
This is old. We now have a new way.

Log archiving and analysis

Example pipeline

Google Cloud

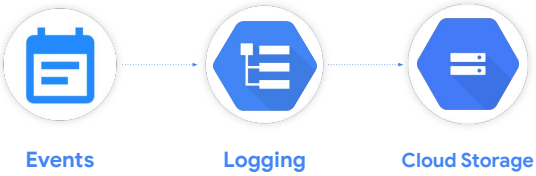
Over the next several slides, we will investigate some possible log export processing options.

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



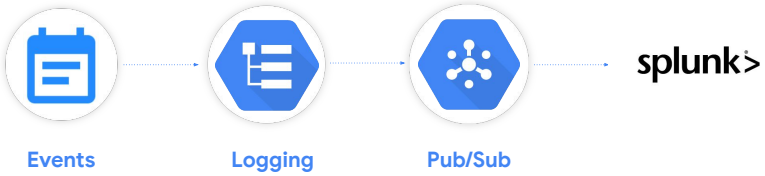
Google Cloud

Sink pipelines targeting Cloud Storage tend to work best when your needs are in line with the Cloud Storage strengths, like 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



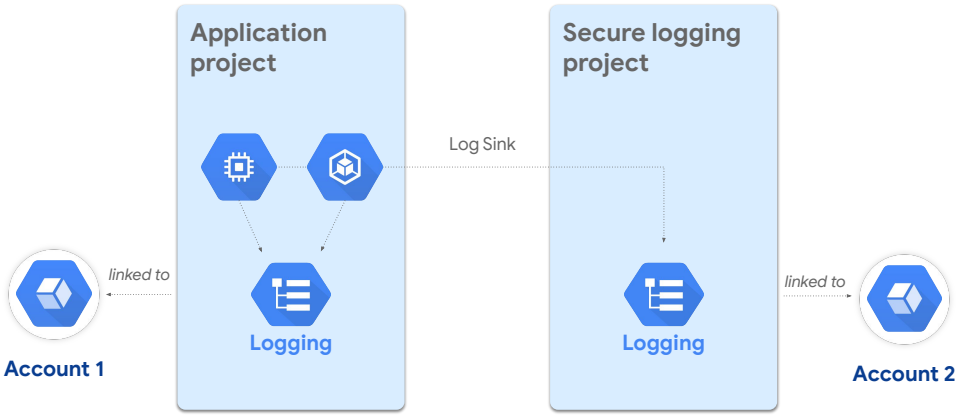
Google Cloud

Here, we have an example organization that wants to integrate the logging data from Google Cloud, back into an on-premises Splunk instance.

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.

Lecture Notes:
Watch your network fees.

Security logging



Google Cloud

Setting up a Sink to an external project can allow for easier security segregation of logging entries.

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

Aggregated sinks allows 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.

A project-level log sink we've discussed; 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's 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).

Lecture Notes:
By default, logging is at the project level. If you need at the folder or org level use aggregated sinks.
In this case you would have to send logs from each of the projects to a separate logging project.
Reference: https://cloud.google.com/logging/docs/export/aggregated_sinks

Aggregated sinks

- Export log entries for multiple projects, folders, up to the organization or billing account level

```
gcloud logging sinks create [SINK_NAME] \
storage.googleapis.com/[BUCKET_NAME] --include-children \
--folder=[FOLDER_ID] --log-filter="logName:activity"
```

- `--folder` could also be `--organization` and `--billing-account`
- Need *Logs Configuration Writer* IAM role for parent

Google Cloud

An aggregated sink can export log entries from all the projects, folders, and billing accounts of a Google Cloud organization. For instance, you might aggregate and export audit log entries from all an organization's projects to a central location.

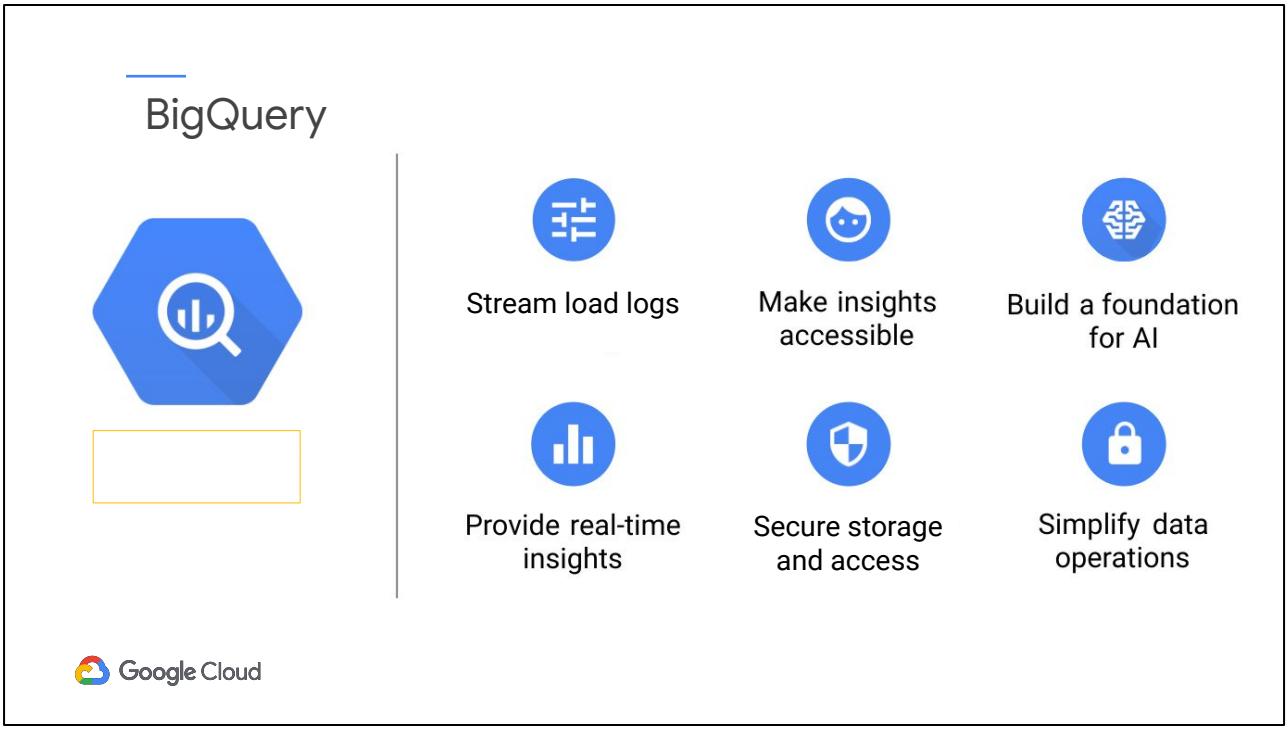
The destination for log sinks has to be created before the sink. Once again, the supported destinations are a Cloud Storage bucket, Pub/Sub topic, or BigQuery table.

To create an aggregated sink in Google Cloud folders, billing accounts, or organizations, you can use either the Cloud Logging API or gcloud command-line tool.

Here, we see an example using gcloud. You would need to supply the sink name, sink destination bucket name, logs query, and the ID of the folder, billing account, or organization. Here, we are filtering for the logName activity.

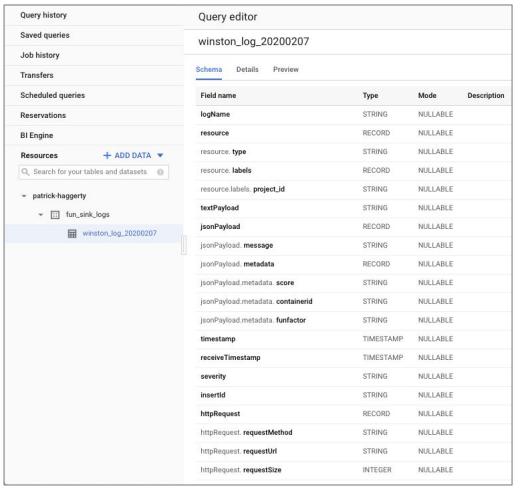
Other valid options besides folder include organization and billing accounts.

You would need the **Logs Configuration Writer** Cloud IAM role for the parent to create the sink.



We've mentioned several times that a common export destination for logs is BigQuery. BigQuery has many features that can be of use when processing exported logging data. It supports stream loading logs to support easy access to real-time insights, while serving as a good foundation for making those insights easily accessible. BigQuery can store data both securely and inexpensively. It can form an excellent foundation for AI system training data and simplify data operations through its easy-to-use, ANSI 2011 compliant, SQL interface. BigQuery query results can also be visualized using tools such as Data Studio and Looker.

Table schema based on LogEntry



Google Cloud

BigQuery table schemas for exported logs are based on the structure of the LogEntry type and the contents of the log payloads.

Cloud Logging also applies some special rules to shorten BigQuery schema field names for audit logs.

You can view the table schema by selecting a table with exported log entries in the BIgQuery web UI as seen on this slide.

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

Google Cloud

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;
```

Google Cloud

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

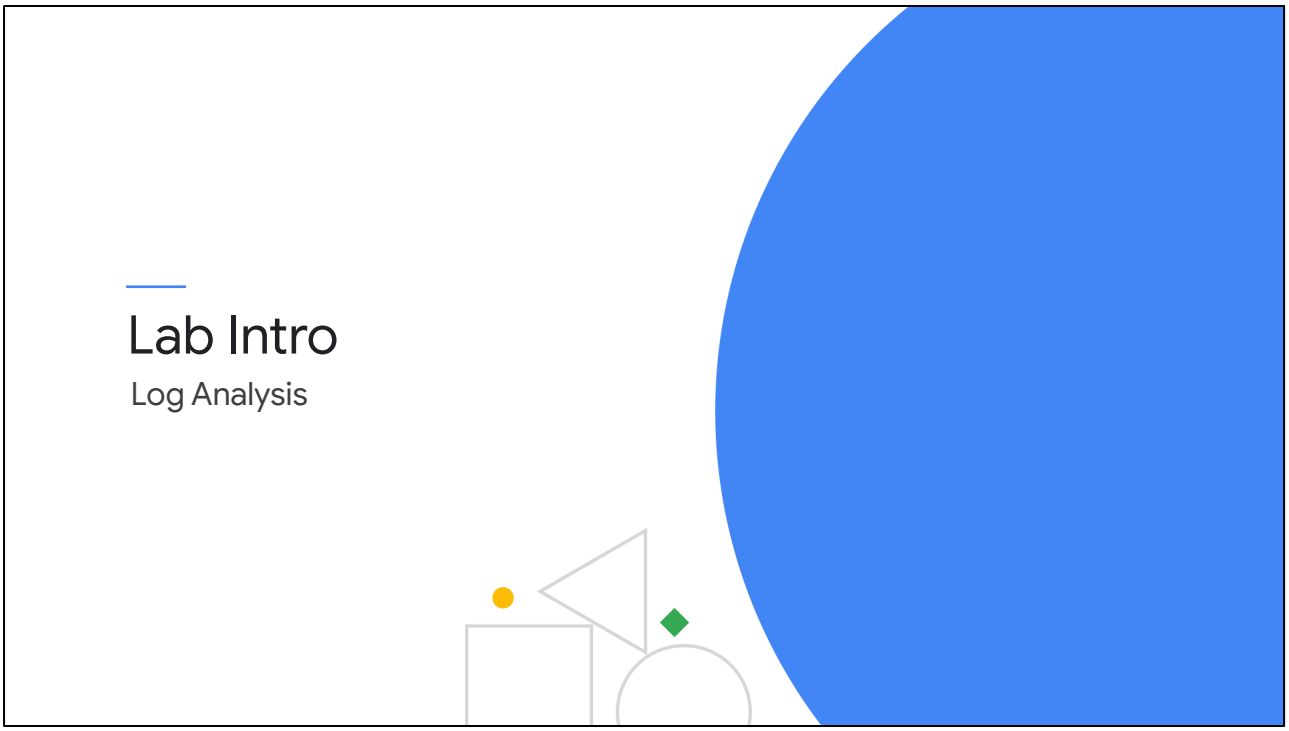
```
SELECT
  timestamp AS Time,
  protoPayload.host AS Host,
  protoPayload.status AS Status,
  protoPayload.resource AS Path
FROM
  (TABLE_DATE_RANGE(my_bq_dataset.appengine_googleapis_com_request_log_,
    DATE_ADD(CURRENT_TIMESTAMP(), -1, 'MONTH'), CURRENT_TIMESTAMP()))
WHERE
  protoPayload.status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

Google Cloud

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 lab, you generate logging entries from an application, filter and analyze logs, and export logs to a BigQuery sink.

Quiz

You want to be able to compare resource utilization for VMs used for production, development, and testing?

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 Functions to build reports based on the VM tags

Quiz

You want to be able to compare resource utilization for VMs used for production, development, and testing?

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 Functions to build reports based on the VM tags

Quiz

Your governance team has mandated you save your log data for 5 years for compliance reasons. Where would the best place to export it be?

A. Cloud Storage in a multi-region bucket
B. Cloud Storage in a single-region bucket using the archival storage class
C. BigQuery
D. You don't need to export it, just set the retention policy in Logging to 5 years

Quiz

Your governance team has mandated you save your log data for 5 years for compliance reasons. Where would the best place to export it be?

A. Cloud Storage in a multi-region bucket
B. Cloud Storage in a single-region bucket using the archival storage class
C. BigQuery
D. You don't need to export it, just set the retention policy in Logging to 5 years

Quiz

You want to use the logs to monitor application usage in real time. Where would the best export sink be?

A. Cloud Storage
B. Pub/Sub
C. BigQuery
D. Spanner

Quiz

You want to use the logs to monitor application usage in real time. Where would the best export sink be?

A. Cloud Storage
B. Pub/Sub
C. BigQuery
D. Spanner

Quiz

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

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

Learned how to...

- Identify and choose among resource tagging approaches
- Define log sinks (inclusion filters) and exclusion filters
- Create metrics based on logs
- Export logs to BigQuery

Google Cloud