

# Optimizing the Costs for Google Cloud Observability



In our final module, we discuss how to optimize the costs for Google Cloud Observability.

# Objectives

01

Analyze resource utilization cost for monitoring related components within Google Cloud.

02

Implement best practices for controlling the cost of monitoring within Google Cloud.



Specifically, you learn to analyze resource utilization costs for operations-related components within Google Cloud and implement best practices for controlling the cost of operations within Google Cloud.

## In this section, you explore



- ✓ Costs and pricing
- ✓ Bill estimation
- ✓ Cost control best practices

Let's start with costs and pricing.

# Cloud Logging pricing

Cloud Logging	
Based on	<ul style="list-style-type: none"><li>• Volume of chargeable logs ingested</li></ul>
Free allotment/ month	<ul style="list-style-type: none"><li>• First 50 GiB per project</li><li>• Logs retained for the default retention period</li></ul>
Examples of usage that incurs cost	<ul style="list-style-type: none"><li>• Cloud Load Balancing logs</li><li>• Custom logs</li><li>• Error reporting</li><li>• The write operation in the Cloud Logging API</li><li>• Log retention</li></ul>

Because Google Cloud Observability services are managed services, their cost is usage-based and not infrastructure-based.

Although Cloud Profiler is offered at no cost, Cloud Logging, Cloud Monitoring, and Cloud Trace has associated costs. Error Reporting supports errors sent by using Cloud Logging and incurs costs associated with this service.

- Logging pricing is based on the volume of chargeable logs ingested.
- Logging incurs cost when using:
  - Cloud Load Balancing logs
  - Custom logs
  - Error reporting costs (if your errors are ingested by Cloud Logging)
  - The write operation in the Cloud Logging API
  - Logs stored beyond 30 days will incur a retention charge for non-required buckets. Refer to the documentation for updated pricing details.

# Cloud Monitoring pricing

Cloud Monitoring	
Based on	<ul style="list-style-type: none"><li>• Volume of chargeable metrics ingested</li><li>• Number of chargeable API calls</li><li>• Execution of Cloud Monitoring uptime checks</li><li>• Metrics ingested by using Google Cloud Managed Service for Prometheus</li></ul>
Free allotment/ month	<ul style="list-style-type: none"><li>• All <a href="#">non-chargeable Google Cloud metrics</a></li><li>• First 150 MiB per billing account</li><li>• First 1 million Read API calls per billing account</li><li>• 1 million uptime check executions per project</li></ul>
Examples of usage that incurs cost	<ul style="list-style-type: none"><li>• Cloud Monitoring custom metrics</li><li>• External metrics written to Cloud Monitoring through API or client libraries</li><li>• The read operation in the Monitoring API (except from the Google Cloud console)</li></ul>

- Cloud Monitoring prices are based on the:
  - Volume of chargeable metrics ingested.
  - Number of chargeable API calls.
  - Execution of Cloud Monitoring uptime checks.
  - Metrics ingested by using Google Cloud Managed Service for Prometheus.
- Example product usage that generates cost through metric volume and API calls includes using:
  - Cloud Monitoring custom metrics.
  - AWS Metrics.
  - The read operation in the Monitoring API, except from Google Cloud console.

# Cloud Trace pricing

Cloud Trace	
Based on	Number of spans ingested and eventually scanned
Free allotment/ month	First 2.5 million spans
Examples of usage that incurs cost	<ul style="list-style-type: none"><li>• Spans</li><li>• Cloud Load Balancing</li><li>• Custom apps</li></ul>

- Trace prices are based on the number of spans ingested and eventually scanned.
- The free allotment was 2.5 million spans.
- Example product usage that generates cost through spans ingested includes adding instrumentation for your:
  - Spans for App Engine apps outside of the default spans
  - Cloud Load Balancing
  - Custom apps

## Data table

### Currently free

Cloud Profiler

Cloud Audit, access transparency, and BigQuery Data Access logs

Uptime checks and logs analytics when queries running in Cloud Logging

Log exclusions

Exporting logs

Creating and using dashboards

Google Cloud and Anthos metrics and log streams



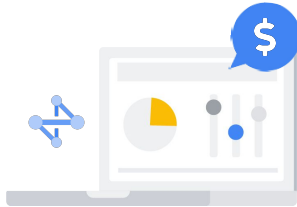
For the latest pricing information, please always check the Google [Documentation](#)

Many functions of Google Cloud Observability are free, including:

- Using Cloud Profiler.
- Collecting and using the Cloud Audit Logs, Access Transparency logs, BigQuery Data Access logs and anything excluded from logs.
- Creating and using dashboards.
- Visualizing Google Cloud and Anthos metrics and log streams.
- App Engine standard trace spans.
- Uptime checks.
- Logs analytics when queries are running in Cloud Logging.

After the free tier finishes, you will be charged for the various operation-related fees. For the latest pricing information, always check the [Google Documentation](#).

# Network Telemetry pricing



VPC Flow logs, Firewall Rules Logging, and Cloud NAT logging cost storage fee.

Logs stored in Cloud Logging are free to generate, but billed for storage per GB.

Network Intelligence Center incurs costs for metrics overlaid on the network topology, Network Analyzer and performance dashboard.

Network Intelligence Center also incurs a cost for running connectivity tests and Firewall Insights.

The networking logs, including VPC Flow logs, Firewall Rules Logging, and Cloud NAT, will cost you the standard log storage fees. However, if you store them in Cloud Logging, they won't cost you anything extra to generate.

If you export the network telemetry logs to an external service, cost is incurred to generate logs. The cost is in addition to any destination or networking fees.

Network Intelligence Center incurs costs for metrics overlaid on the network topology, Network Analyzer and performance dashboard.

Network Intelligence Center also incurs a cost for running connectivity tests and Firewall insights. Refer to the [documentation](#) for more information on pricing models for Firewall insights.



## In this section, you explore

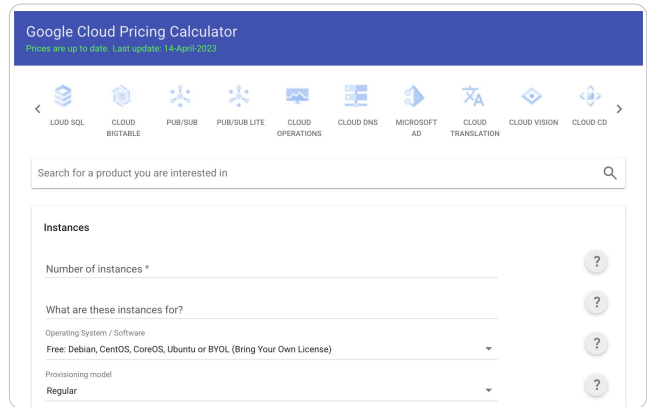


- ✓ Costs and pricing
- ✓ **Bill estimation**
- ✓ Cost control best practices

We covered some of the pure costs. Now let's talk about bill estimation.

## Bill estimation

- The Google Cloud Pricing Calculator is used to get an idea of the costs for hypothetical workloads.
- Start with current usages.
- You can also use the Cloud Estimation API to get billing information.

The screenshot shows the Google Cloud Pricing Calculator interface. At the top, there's a blue header with the title "Google Cloud Pricing Calculator" and a note "Prices are up to date. Last update: 14 April 2023". Below the header is a navigation bar with icons and labels for various services: CLOUD SQL, CLOUD BIGTABLE, PUB/SUB, PUB/SUB LITE, CLOUD OPERATIONS, CLOUD DNS, MICROSOFT AD, CLOUD TRANSLATION, CLOUD VISION, and CLOUD CD. A search bar is positioned below the navigation bar with the placeholder text "Search for a product you are interested in". The main content area is titled "Instances" and contains several input fields: "Number of instances \*" with a question mark icon, "What are these instances for?" with a question mark icon, "Operating System / Software" with a dropdown menu showing "Free: Debian, CentOS, CoreOS, Ubuntu or BYOL (Bring Your Own License)" and a question mark icon, and "Provisioning model" with a dropdown menu showing "Regular" and a question mark icon.

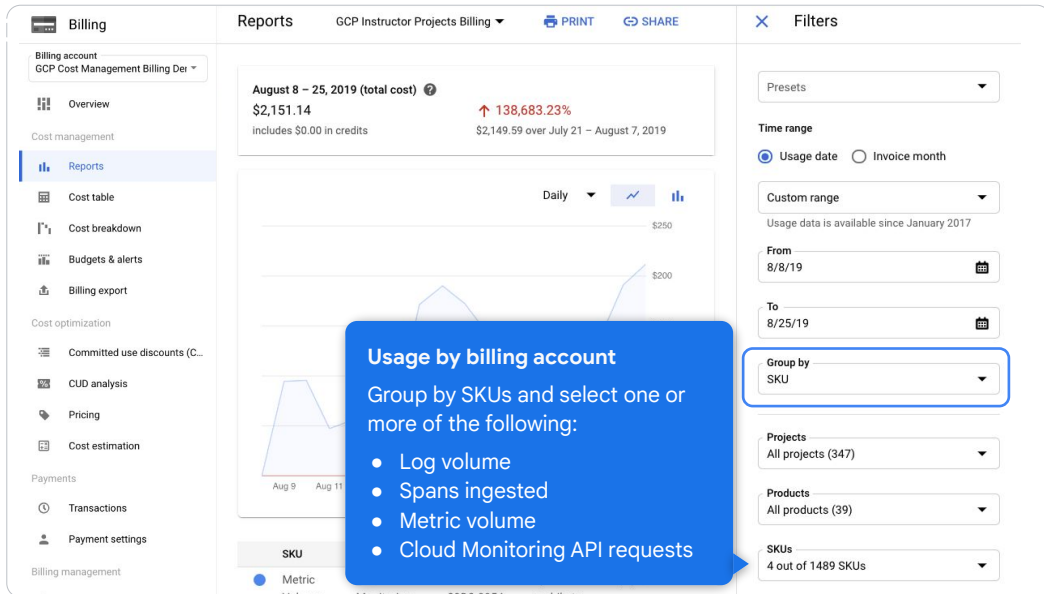
[cloud.google.com/products/calculator](https://cloud.google.com/products/calculator)

When you try to estimate prices in Google Cloud, the page of choice should always be the Google Cloud Pricing Calculator. The Pricing Calculator is accurate, but it's only as accurate as the data that you provide it.

If your operation services are already running on Google Cloud, start by pulling the prices of what you're spending on them. You can look for the requisite data in several places, let us explore these as we move forward in this section.

You can also use the Cost Estimation API. It provides customer-specific estimates that include your discounts. For example, those negotiated as part of a contract and those based on committed usage. These cost estimates can help you make more informed business decisions. For more information, refer to the [documentation](#) linked in the reference material section.

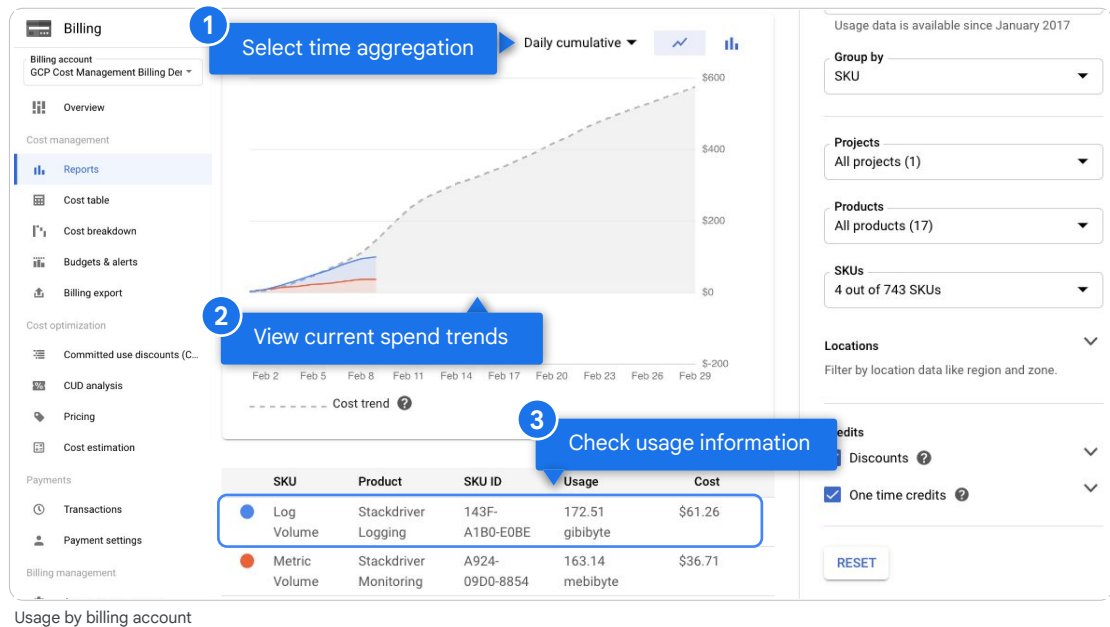
## Reports page



Start by going to your billing account **Reports** page. Set your date range, and then filter by Stock Keeping Unit (SKU). The SKUs you want are:

- **Log volume**
- **Spans ingested**
- **Metric volume**
- **Monitoring API Requests**

## Reports page



Here's another view of the same page.

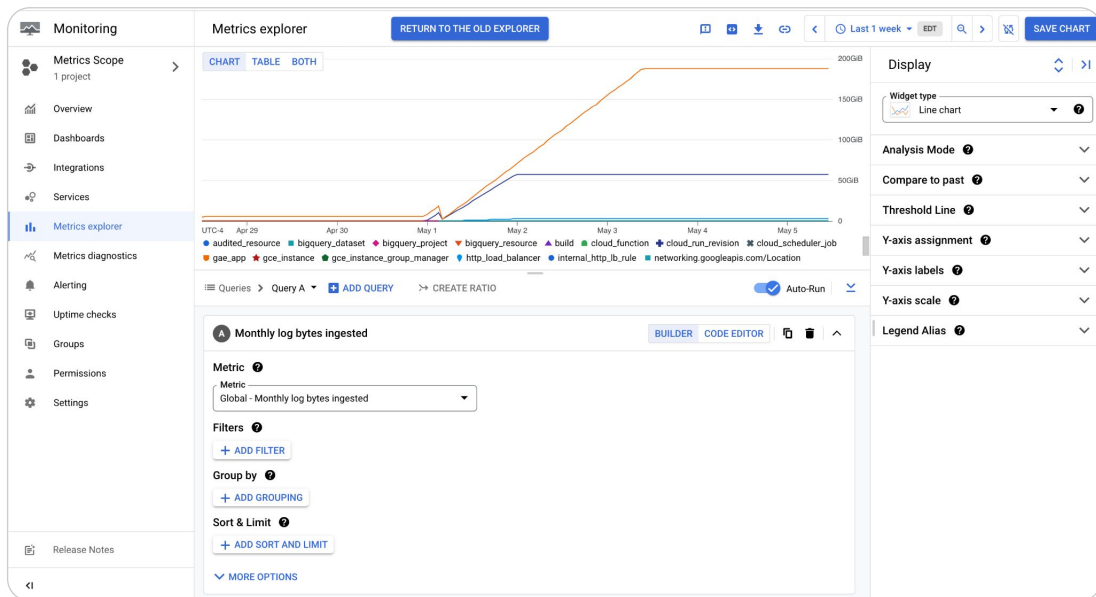
1. In the chart, you can change the view to display the data.
2. When you select **Daily cumulative**, the cost trend line effectively shows where you're headed based on current spend trends. If you recently added logging, adding Daily cumulative might be a good way to estimate what your bill might do.
3. Then, check your usages and costs. Note, if SKU usage is 0 for a metric, then it won't appear in the list.

You can do something similar to see how money was spent in the past month. In this case, the biggest item is log volume.

This insight might raise the question: if money is spent on logging data, where exactly is that data coming from?

To find the answer, check Metrics Explorer.

## View ingestion data in Metrics Explorer



Open **Monitoring** and then **Metrics Explorer**, and set the **Metric** to **Global**.

Then, depending on what you want, select one of the following:

- [Log bytes ingested](#) provides Log bytes ingested in Cloud Logging.
- [Monthly log bytes ingested](#) provides a graph where each point represents the month-to-date sum of log bytes ingested in Cloud Logging. The monthly total is available on the last day of the month, when it also resets.
- [Metric bytes ingested](#) provides chargeable number of bytes of metric data ingested in Cloud Monitoring.
- [Trace spans ingested](#): provides chargeable trace spans ingested in Cloud Trace.
- [Monthly trace spans ingested](#): provides a graph where each point represents the month-to-date sum of trace spans ingested in Cloud Trace. It resets on the last day of the month; the monthly total is found on the last day of the month.

In this example, the orange line represents the highest logging data. Thus, the **gae\_app** is where all the logging data coming from Migrate for Compute Engine. You can find more information on Migrate for Compute Engine in the documentation.

## View monitoring usage by metrics scope

The Metrics Ingested table displays a summary of your metrics ingestion data by resource.

Click **View Bill** to get your project-level usage in detail.

**Overall usage**

Overall metrics ingested	554.01MB	Monthly projection	
154.38MB	Previous month	700.34MB	By end of month
Month to date			↑ 26.41% Versus previous month <a href="#">View Bill</a>

**GCP Projects**

Project name	Project ID	Type	Previous month	Month to date	Monthly projection
▶ VelosSandbox	velossandbox	Host project	554.01MB	154.38MB	700.34MB

[ADD GCP PROJECTS](#)

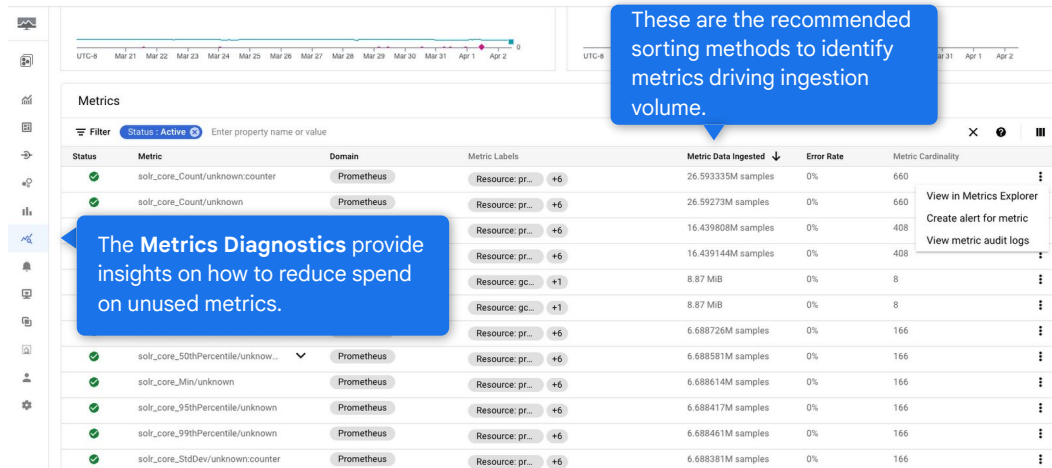
Usage by billing account

We learned that a metrics scope is used in Cloud Monitoring to monitor the resources you care about. The resources could be in a Google Cloud project, an AWS account, or multiple Google Cloud projects and AWS accounts. To view your Monitoring usage by metrics scope, go to **Monitoring** and then **Settings**.

On the **Summary** tab, the **Metrics Ingested** table displays a summary of your metrics ingestion data by resource. This data includes the previous month total usage, the current month to-date usage, and projected usage for the current month.

To get your project-level usage in detail, in the Metrics Ingested table, click **View Bill**. You're taken to the **Cloud Billing Reports** page.

# View metrics ingestion in Metrics Diagnostics



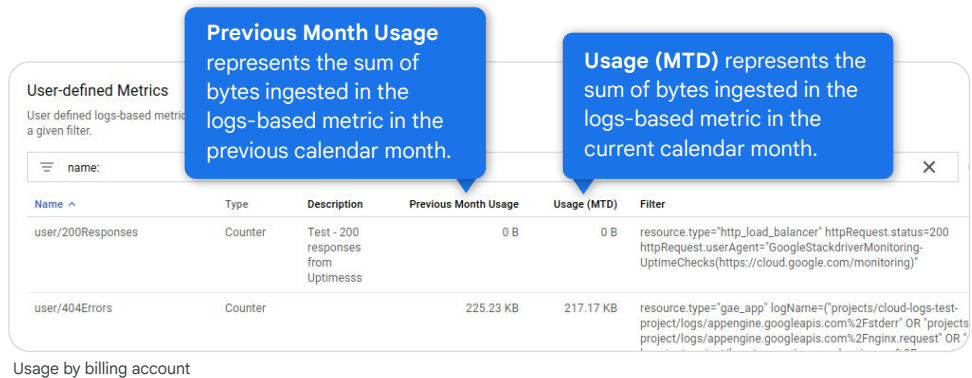
After you know the projects where you're spending on Cloud Monitoring, we want it to be easy to understand which metrics are driving these observability costs. We also want to provide insights on how to reduce spend on unused and noisy metrics.

To get started, go to **Monitoring > Metrics Diagnostics**. This page provides many tools to understand metric ingestion and Monitoring API usage. One tool is a **"Metrics"** table where you can sort and filter metrics by

- volume or samples ingested,
- metric cardinality,
- metric name and domain,
- metric labels,
- project,
- error rate, and more.

We recommend sorting metrics by **"Metric Data Ingested"** in **descending order** to identify exactly which metrics are primarily driving ingestion volume. Whether intended or not, our customers often find that only a few metrics or metric types drive most consumption. These are the ones that are ripe for cost reduction and optimization.

## View Logs-based Metrics usage



**Previous Month Usage** represents the sum of bytes ingested in the logs-based metric in the previous calendar month.

**Usage (MTD)** represents the sum of bytes ingested in the logs-based metric in the current calendar month.

Name ^	Type	Description	Previous Month Usage	Usage (MTD)	Filter
user/200Responses	Counter	Test - 200 responses from Uptimeless	0 B	0 B	resource.type="http_load_balancer" httpRequest.status=200 httpRequest.userAgent="GoogleStackdriverMonitoring-UptimeChecks(https://cloud.google.com/monitoring)"
user/404Errors	Counter		225.23 KB	217.17 KB	resource.type="gae_app" logName=("projects/cloud-logs-test-project/logs/appengine.googleapis.com%2Fstderr" OR "projects/project/logs/appengine.googleapis.com%2Fnginx.request" OR "

Usage by billing account

To see your logs-based metrics usage, go to **Logging > Logs-Based Metrics**.

**Previous Month Usage** represents the sum of bytes ingested in the logs-based metric in the previous calendar month.

**Usage (MTD)** represents the sum of bytes ingested in the logs-based metric in the current calendar month.

Clicking any of the column names lets you sort data in ascending or descending order. For example, if you want to review which metrics ingest the most data, sorting data is helpful.



## In this section, you explore



- ✓ Costs and pricing
- ✓ Bill estimation
- ✓ [Cost control best practices](#)

Finally, let's discuss some billing best practices. We start with what we already discussed in this module, mostly in the last section. You learn to calculate costs, become aware of your operations-related spend and where it goes, and learn what the more expensive items are.

# Controlling logging costs

Best Practices	Description	Example
Exclude logs	Exclude certain logs based on high volume or the lack of practical value.	Common exclusions: <ul style="list-style-type: none"><li>• Load balancers</li><li>• VPC Flow Logs (still incurs costs)</li><li>• Web applications</li></ul>
Export logs	Export logs yet exclude them from being ingested into Cloud Logging.	Export logs to: <ul style="list-style-type: none"><li>• Cloud Storage</li><li>• BigQuery</li><li>• Pub/Sub</li></ul>
Reduce Ops Agent usage	Choose not to add agents in nonessential environments.	Reduce log volumes by not sending the additional logs generated by the Ops Agent to Cloud Logging.

If you exclude log entries, you don't pay for them. However, they will be gone once excluded, so be careful with the log entries you keep.

Here are some of the common exclusions:

- Google has several different load balancers, and they all support various forms of monitoring and logging.
  - a. Frequently, for **Cloud Load Balancing** that support Cloud Logging, all you need is a small percentage. Exclude 90% or more.
  - b. **VPC Flow Logs** is another good example. Again, you can often exclude more than 95% of the entries and still get enough to monitor the VPC. Also, remember that your exclusion can filter on entry contents. Perhaps you only want to retain logs from sources with a CIDR outside your network. If you send your VPC Flow Logs to Cloud Logging, the charges for the generation of VPC Flow Logs instances are waived. Only logging charges apply. However, if you send them and then exclude your VPC Flow Logs from Cloud Logging, VPC Flow Logs charges apply. To lower the bill, they have to be deactivated completely. For more details, refer to the [documentation](#) provided at the end of this module.
  - c. Another common exclusion is **web applications and services** that are logging HTTP 200 OK from requests. Frequently, OK messages don't provide much insight, and they can generate numerous entries.

Log exports are free, but not the target resource. You can export logs yet exclude them from being ingested into Cloud Logging. You can retain the logs in Cloud Storage and BigQuery or use Pub/Sub to process the logs while excluding the logs from Cloud Logging, which can reduce costs. The fee associated with log exports are:

- Storage fees in Cloud Storage.
- Storage, streaming, and query fees in BigQuery.
- Pub/Sub message and networking egress fees.

In fact,

- 2 TiBs of data access log data stored in Logging would cost about \$1,000.
- The same 2 TiBs stored in a regional, standard class bucket would cost about \$40. With archival, it would be much cheaper.

You can reduce log volumes by not sending the additional logs generated by the Ops Agent to Cloud Logging. For example, you might reduce log volumes by choosing not to add the Ops Agent to VMs in your development or other nonessential environments to Cloud Logging. Your virtual machines continue to report the standard logs to Cloud Logging, but don't report logs from third-party apps nor the syslog.

# Controlling monitoring costs

Best Practices	Description
Optimize metrics and label usage.	<ul style="list-style-type: none"><li>• Limit custom metric labels.</li><li>• Select labels thoughtfully.</li></ul>
Reduce Ops Agents usage.	<ul style="list-style-type: none"><li>• Consider reducing the volume of metrics by not sending detailed metrics.</li><li>• Reduce the number of VMs using Ops Agents.</li></ul>
Reduce custom metrics usage.	Reduce the number of custom monitoring metrics that your apps send.

How you use labels on Monitoring custom metrics can affect the volume of time series that are generated.

- Given a custom metric with two labels (cost\_center and env values), you can calculate the maximum number of time series by multiplying the cardinality of both labels. If there are 11 cost\_center values and 5 env values, that means that up to 55 time series can be generated. Adding additional metric labels can add significant metric volume and, therefore, increase the cost.
- Where possible, limit the number of custom metric labels.
- Select labels thoughtfully to avoid label values with high cardinality. For example, using user\_id as a label results in at least one time series for each user. If you have significant traffic, the number could be very large.

Metrics sent from the Ops Agent are chargeable metrics. Although installing them is undoubtedly a best practice, there will be exceptions. Weigh the advantages and disadvantages, and remember that both the agents are customizable through configuration files.

- If you don't need the detailed system metrics or metrics from the third-party apps for certain VMs, reduce the volume by not sending these metrics.
- You can also reduce the metric volumes by reducing the number of VMs using the Ops Agent.
- For example, you can reduce metric volumes by choosing not to add Google Cloud projects in your development or other nonessential environments to

- Cloud Monitoring. Also, you can choose not to include the monitoring agent in VMs in development or other nonessential environments.

Custom metrics can increase spend. Newer metrics created using OpenTelemetry support sampling to help reduce volume. When you instrument more apps to send metrics, more custom monitoring metrics are generated. If you want to reduce metric volumes, you can reduce the number of custom monitoring metrics that your apps send.

# Controlling costs for Google Cloud Managed Service for Prometheus

Best Practices	Description
Reduce the number of time series.	<ul style="list-style-type: none"><li>• Increase the length of the sampling period.</li><li>• Set the scraping interval on a per-job or a per-target basis.</li></ul>
Reduce the number of samples collection.	<ul style="list-style-type: none"><li>• Do not send detailed system metrics from third-party apps.</li><li>• Set the scraping interval on a per-job or a per-target basis.</li></ul>
Configure local aggregation (self-managed collection only).	Aggregate high-cardinality metrics locally by using recording rules, flags, and environment variables.

Open source Prometheus documentation rarely recommends filtering metric volume, which is reasonable when costs are bounded by machine costs. But when paying a managed-service provider on a unit basis, sending unlimited data can cause high bills.

To reduce the number of metrics, you can do the following:

- Modify your scrape configs to scrape fewer targets.
- Filter exported metrics when you using managed collection or self-deployed collection.

Managed Service for Prometheus charges on a per-sample basis. Here are some of the ways in which you can reduce the number of samples ingested:

- a. Increasing the length of the sampling period. For example:
  - i. Changing a 10-second sampling period to 30 seconds can reduce your sample volume by 66%, without much loss of information.
- b. Setting the scraping interval on a per-job or a per-target basis.
  - i. For managed collection, you set the scrape interval in the PodMonitoring resource by using the interval field.

If you're configuring the service by using self-deployed collection, for example with kube-prometheus, prometheus-operator, or by manually deploying the image, then

you can reduce your samples sent to Managed Service for Prometheus by aggregating high-cardinality metrics locally.

- Use recording rules, flags, and environment variables to aggregate data to Monarch.

Refer to the [documentation](#) for more information.

# Controlling Trace costs

Best Practices	Description
<b>Use OpenTelemetry sampling.</b>	Use the sampling feature to reduce the volume of trace ingestion. <ul style="list-style-type: none"><li>• Reduce sampling to one-fourth.</li><li>• Use OpenTelemetry to specify sampling.</li></ul>
<b>Use Cloud Trace API span quotas.</b>	Enforce span quotas to ensure that your project does not cross quota limit.
<b>Optimize third party callers.</b>	Understand how instrumented apps interact to assess the effect of the number of spans ingested.

[Trace](#) charges are based on the number of trace spans ingested and scanned.

Use sampling to reduce the volume of traces ingested. Sampling is a crucial part of a tracing system, because it provides insight into the breakdown of latency caused by app components, such as RPC calls. Sampling is not only a best practice for using Cloud Trace: you might reduce your span volume for cost-reduction reasons too.

- For example, with a popular web application with 5000 queries/second, you might gain enough insight from sampling 5% of your app traffic instead of 20%. A smaller sample reduces the number of spans ingested into Trace to one-fourth.
- The OpenTelemetry lets you specify a sampling rate.

You can enforce span quotas with the API-specific quota page in the Google Cloud console. Setting a quota that is lower than the default product quota means that you guarantee that your project won't go over the specific quota limit. Quotas are a way to ensure that your costs are expected.

Your app might be called by another app. If your app reports spans, the number of spans reported by your app might depend on the incoming traffic that you receive from the third-party app. For example, if you have a frontend microservice that calls a checkout microservice, and both are instrumented with OpenTelemetry, the sampling rate for the traffic is at least as high as the frontend sampling rate. Understanding how instrumented apps interact lets you assess the effect of the number of spans



ingested.

## Recap

01

Analyze resource utilization cost for monitoring related components within Google Cloud.

02

Implement best practices for controlling the cost of monitoring within Google Cloud.



In this module, you learned how to:

- Analyze resource utilization costs for operations-related components within Google Cloud.
- And implement best practices for controlling the cost of operations within Google Cloud.