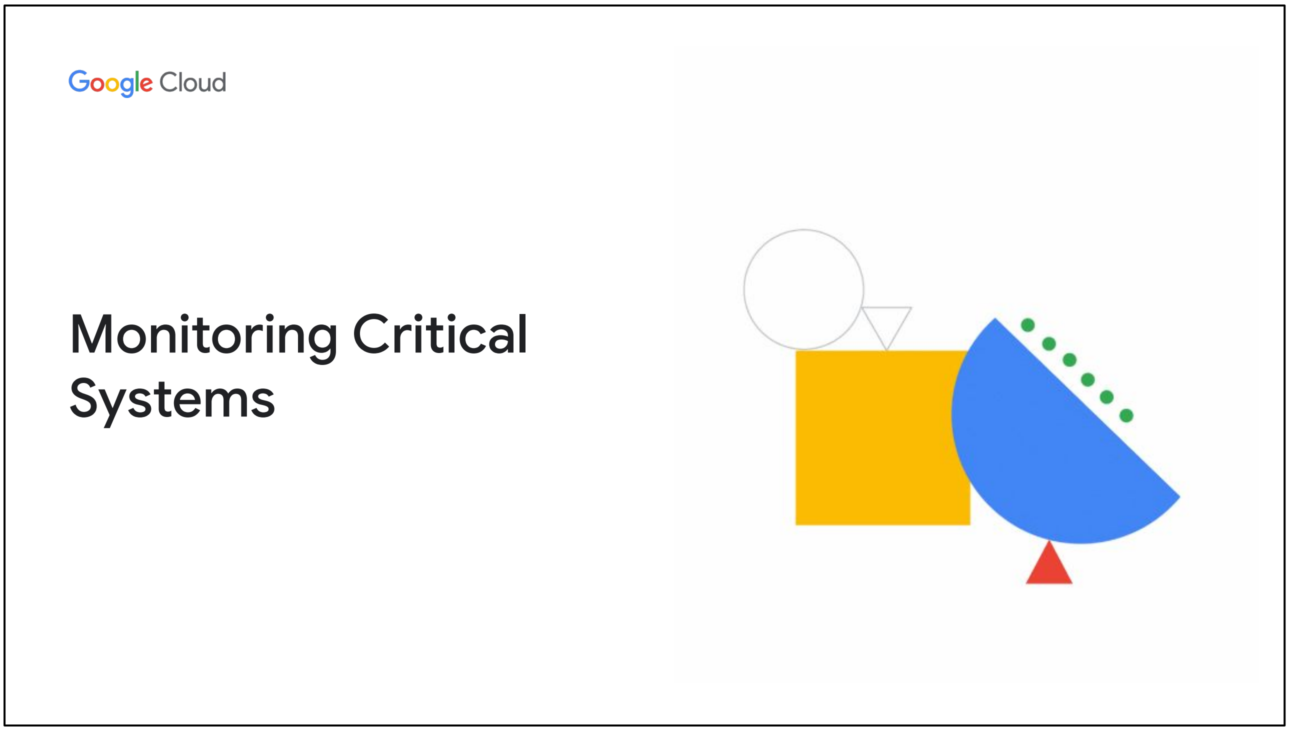
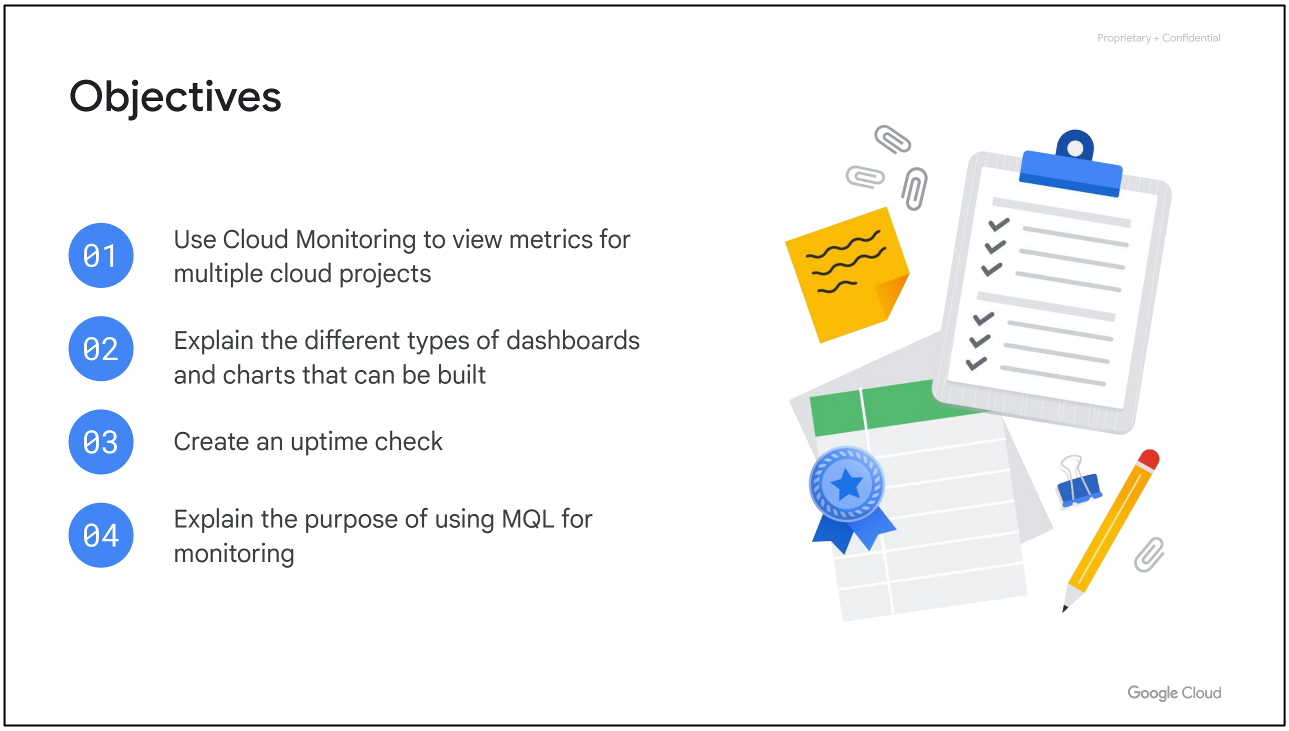


Google Cloud

Monitoring Critical Systems

Let's spend a little time talking about how Google Cloud helps you monitor critical systems.



Objectives

01 Use Cloud Monitoring to view metrics for multiple cloud projects

02 Explain the different types of dashboards and charts that can be built

03 Create an uptime check

04 Explain the purpose of using MQL for monitoring

Monitoring is all about keeping track of exactly what's happening with the resources that we launched inside of Google Cloud.

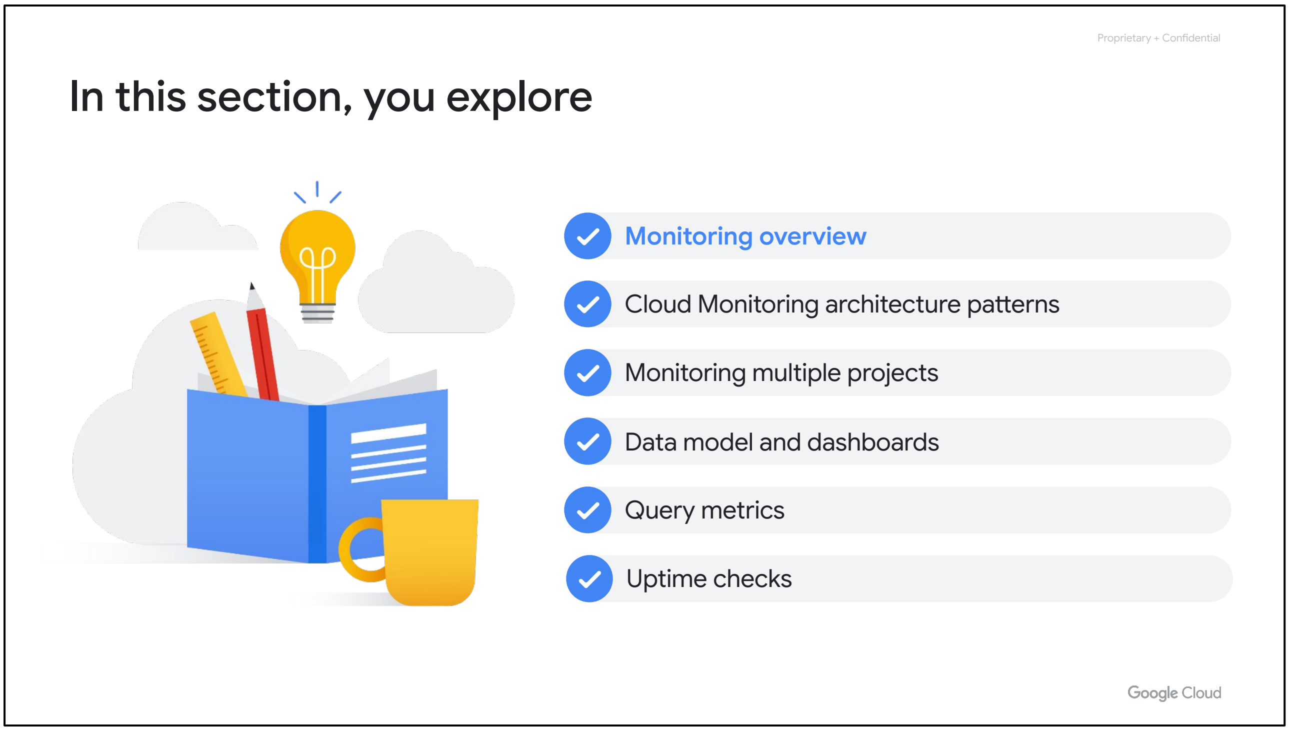
In this module, let's take a look at options and best practices as they relate to monitoring project architectures. It's important to make some early architectural decisions before starting monitoring.

We'll differentiate the core IAM roles needed to decide who can do what as it relates to monitoring. Just like architecture, this is another crucial early step.

We will examine some of the default dashboards created by Google, and see how to use them appropriately.

We will create charts and use them to build custom dashboards to show resource consumption and application load.

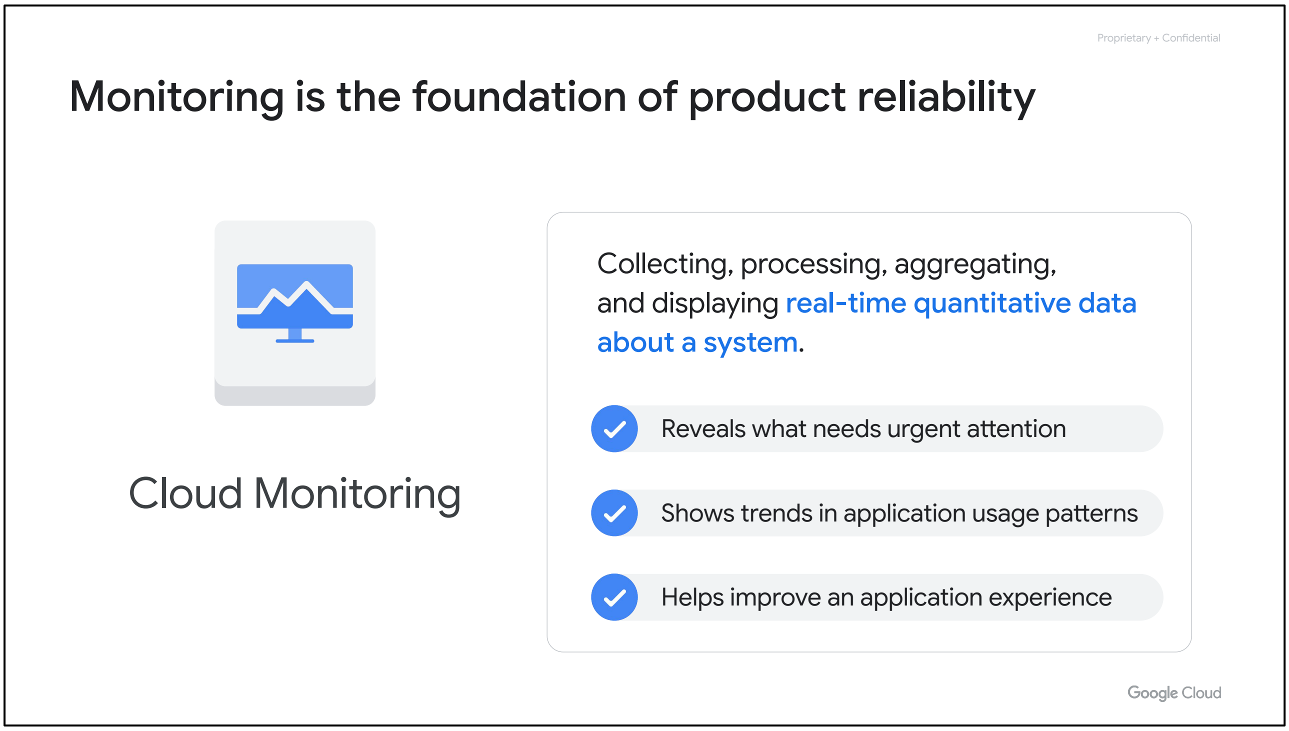
And, we will define uptime checks to track liveliness and latency. We will also cover demonstrate the purpose using MQL for monitoring.



In this section, you explore

- ✓ Monitoring overview
- ✓ Cloud Monitoring architecture patterns
- ✓ Monitoring multiple projects
- ✓ Data model and dashboards
- ✓ Query metrics
- ✓ Uptime checks

We first start with an overview of monitoring. Then, we explore this concept in detail as we advance with the rest of the module.



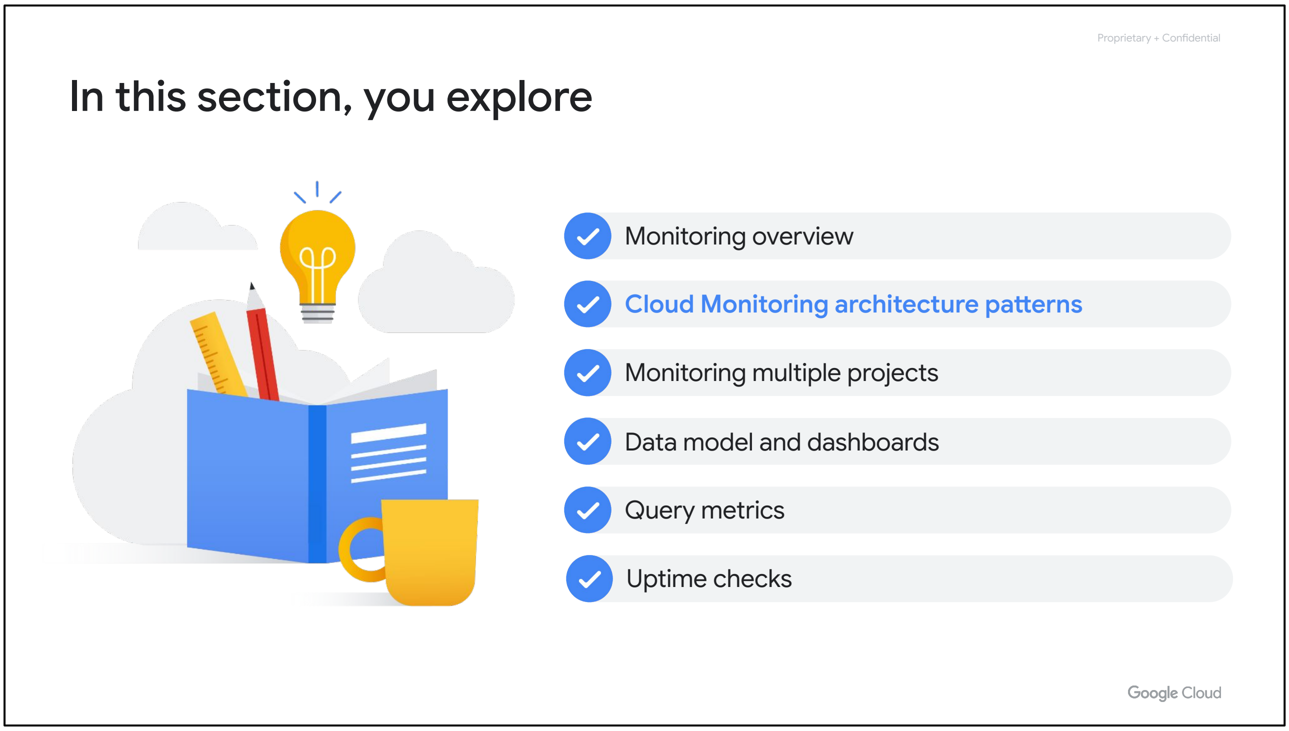
Monitoring is the foundation of product reliability

Cloud Monitoring

Collecting, processing, aggregating, and displaying real-time quantitative data about a system.

- ✓ Reveals what needs urgent attention
- ✓ Shows trends in application usage patterns
- ✓ Helps improve an application experience

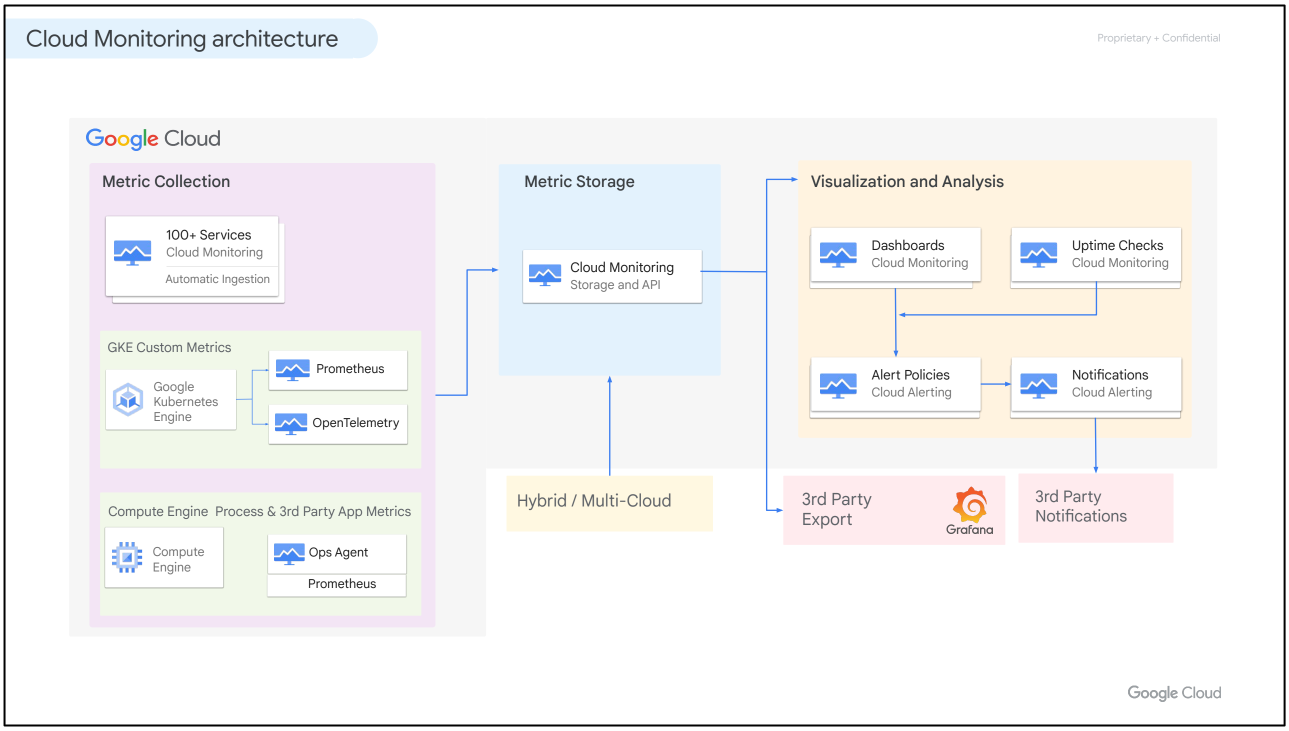
Monitoring is the foundation of product reliability. It reveals what needs urgent attention and shows trends in application usage patterns, which can yield better capacity planning, and generally help improve an application client's experience, and reduce their problems.



In this section, you explore

- ✓ Monitoring overview
- ✓ **Cloud Monitoring architecture patterns**
- ✓ Monitoring multiple projects
- ✓ Data model and dashboards
- ✓ Query metrics
- ✓ Uptime checks

Let's now learn more about observability architecture.

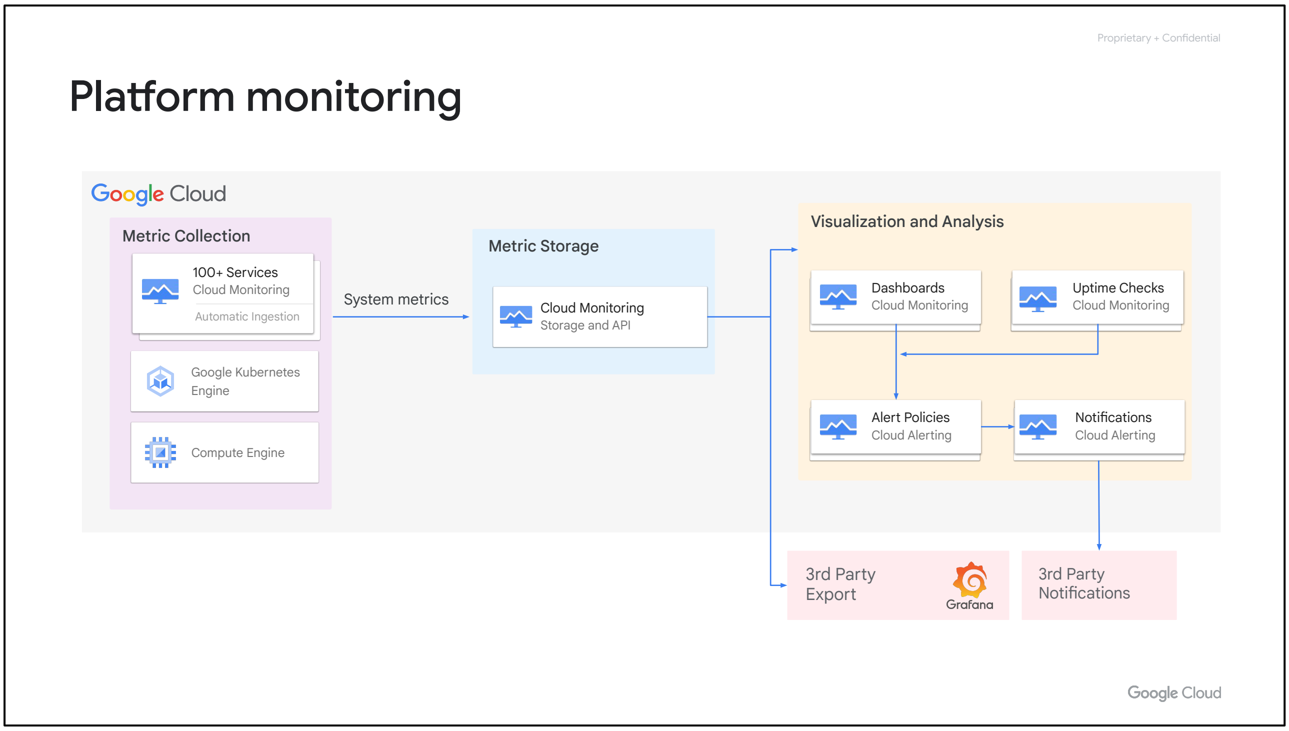


A typical Cloud Monitoring architecture includes the following:

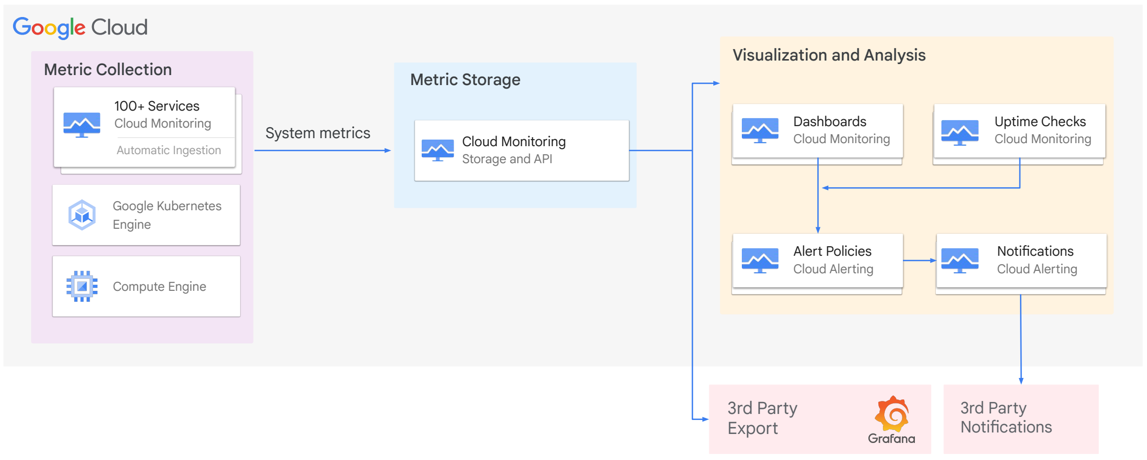
A data collection layer: This layer collects metrics, logs, and traces from cloud-based systems. In Cloud Monitoring, the data collection layer includes Google Cloud services such as Google Kubernetes Engine, Compute Engine, App Engine etc,.

A data storage layer: This layer stores the collected data and routes to the configured visualization and analysis layer. In Cloud Monitoring, this layer includes the Cloud Monitoring API that helps triage the metrics collected to be stored for further analysis.

A data analysis and visualization layer: This layer analyzes the collected data to identify problems and trends and presents the analyzed data in a way that is easy to understand. In Cloud Monitoring, this layer comprise of various features within Cloud Monitoring such as Dashboards to visualize data, Uptime checks to monitor applications, Alerting policies to configure alerts and notifications to notify of events that need immediate attention.



Platform monitoring

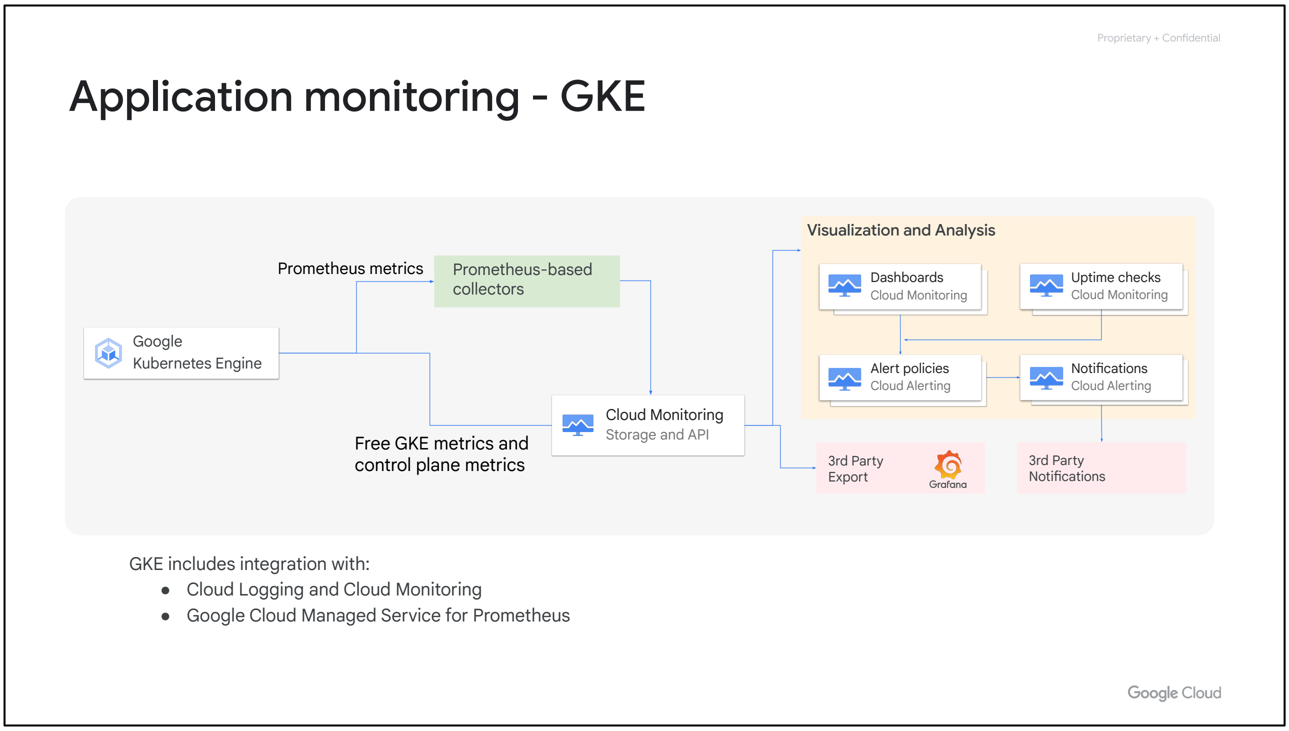


One of the most common uses of Cloud Monitoring is platform monitoring.

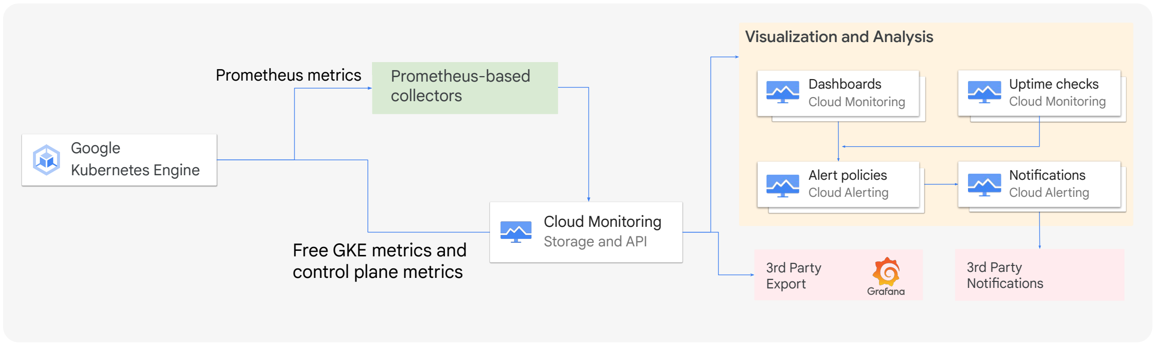
Blackbox monitoring of the platform enables users to get visibility into the performance of their Google Cloud services. With Google Cloud, this is enabled by default and system metrics are automatically collected without any user effort. Google Cloud Monitoring is the recommended solution for Platform monitoring.

System metrics from Google Cloud are available at no cost to customers. These metrics provide information about how the service is operating. Over 1500 metrics across more than 100 Google Cloud services automatically. For example, Compute Engine reports over 25 unique metrics for each virtual machine (VM) instance.

However, if customers, e.g. in traditional enterprise cohorts, are using 3P products for monitoring and want to aggregate their Google Cloud metrics into those partner products, they can use Cloud Monitoring APIs to ingest these metrics.



Application monitoring - GKE


Prometheus metrics
Prometheus-based collectors
Google Kubernetes Engine
Cloud Monitoring Storage and API
Free GKE metrics and control plane metrics
Visualization and Analysis
Dashboards Cloud Monitoring
Uptime checks Cloud Monitoring
Alert policies Cloud Alerting
Notifications Cloud Alerting
3rd Party Export
Grafana
3rd Party Notifications

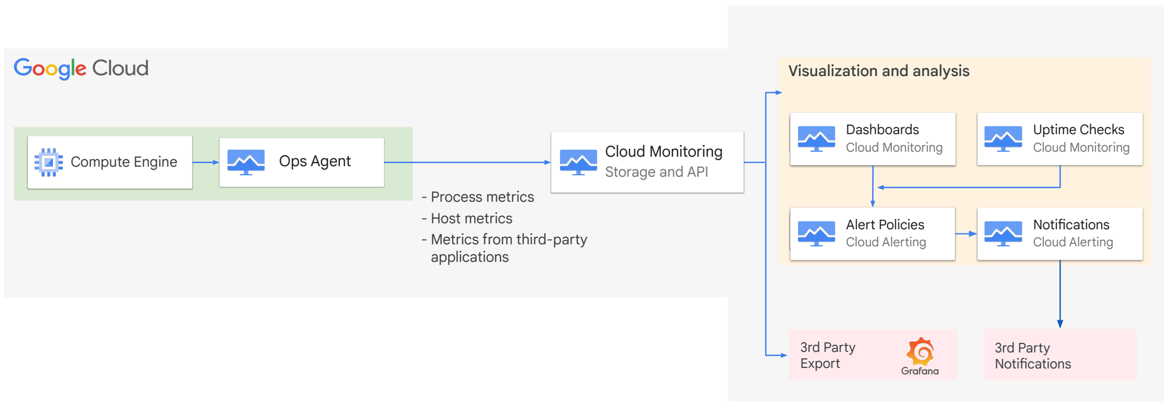
GKE includes integration with:

- Cloud Logging and Cloud Monitoring
- Google Cloud Managed Service for Prometheus

For applications/workloads deployed in GKE, many customers prefer a Prometheus-based solution for monitoring. We fully embrace that monitoring approach and provide customers a new way to leverage Prometheus based monitoring using Google Managed Prometheus (GMP). GMP is a part of Cloud Monitoring and it makes GKE cluster and workload metrics available as Prometheus data.

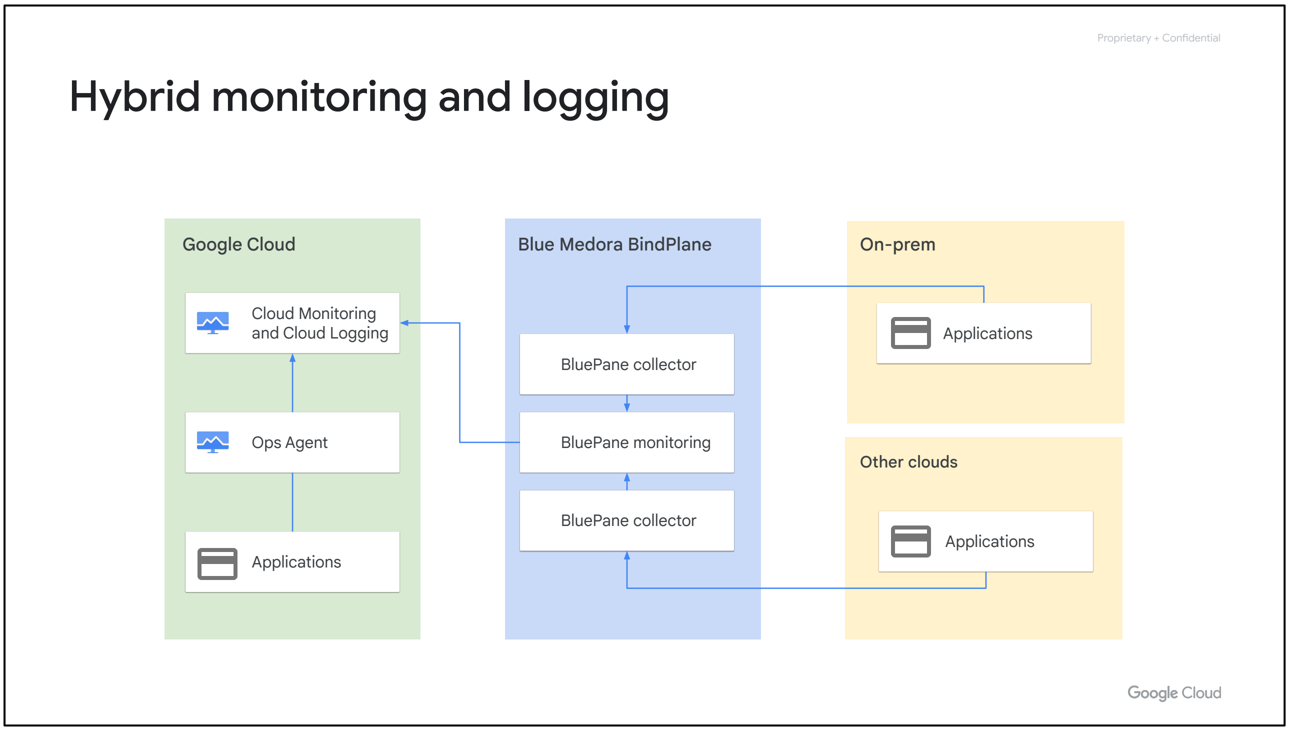
It can ingest monitoring data exposed in Prometheus format, it supports PromQL compatible query language and has natively integrated the Prometheus expression browser, and Prometheus compatible rule evaluation. For application workloads in GKE, we recommend that customers use Google Managed Prometheus.

Application monitoring - Compute Engine



For applications or workloads deployed in Compute Engine, customers should use the Cloud Ops Agent to collect in-process metrics and to collect metrics from third-party applications that run in your VMs. Ops agent today supports more than 30 plugins for different open source and ISV software along with a collection of richer and more fine grained metrics at the OS level for Windows and Linux (many flavors). The Ops agent is based on OpenTelemetry standards so custom applications developed by customers can leverage OTEL client libraries for instrumenting their code and generate the needed telemetry.

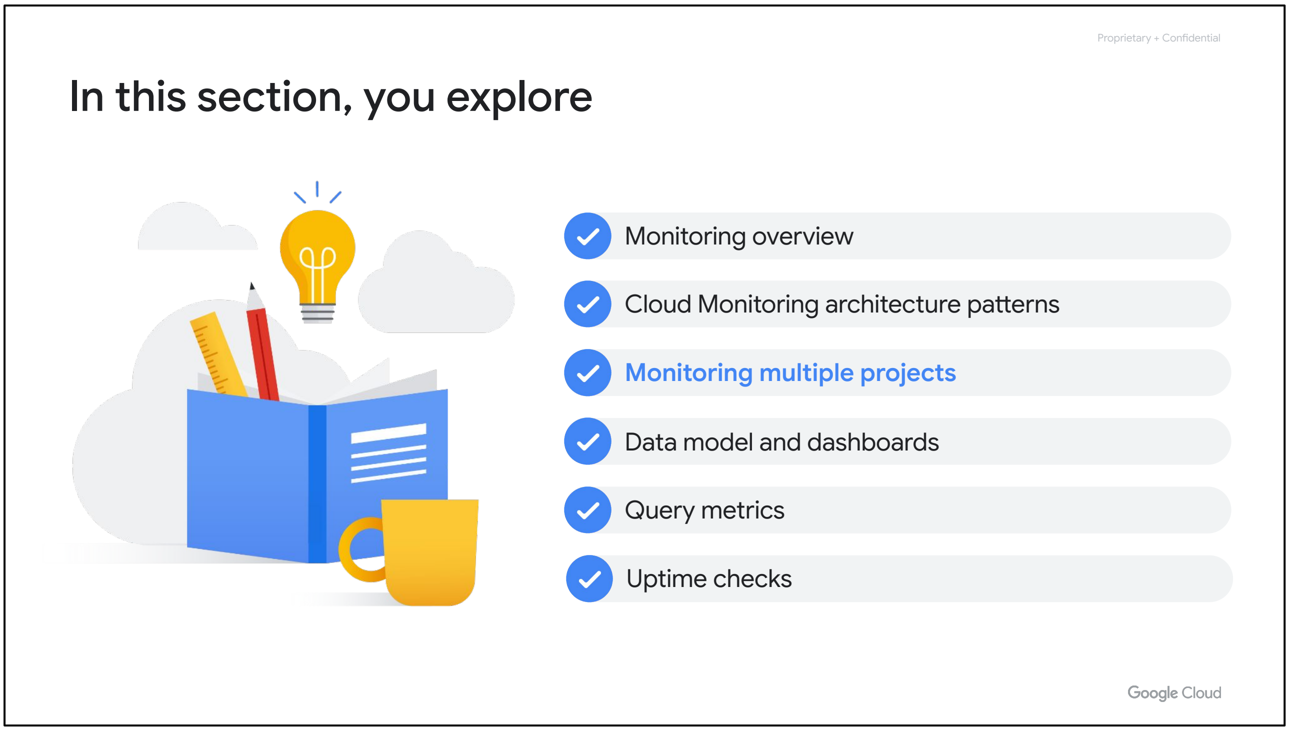
The Ops agent can collect these custom metrics and make them available in Cloud Monitoring as well. While this ecosystem of third-party plugins will continue to expand, if users need support for other software products or services, consider using a partner product like Datadog or NewRelic. If you choose to use partner products, they can collect system metrics from the Google Cloud platform by using the native API-based integrations.


Proprietary + Confidential

Hybrid monitoring and logging

With Google's partner BindPlane by Blue Medora, you can import monitoring and logging data from both on-premises VMs and other cloud providers, such as Amazon Web Services (AWS), Microsoft Azure, Alibaba Cloud, and IBM Cloud into Google Cloud. The following diagram shows how Cloud Monitoring and BindPlane can provide a single pane of glass for a hybrid cloud. This architecture has the following advantages:

- In addition to monitoring resources like VMs, Blue Medora has built-in deep integration for over 150 popular data sources.
- There are no additional licensing costs for using BindPlane. BindPlane metrics are imported into Monitoring as custom metrics, which are chargeable. Likewise, BindPlane logs are charged at the same rate as other Cloud Logging logs.

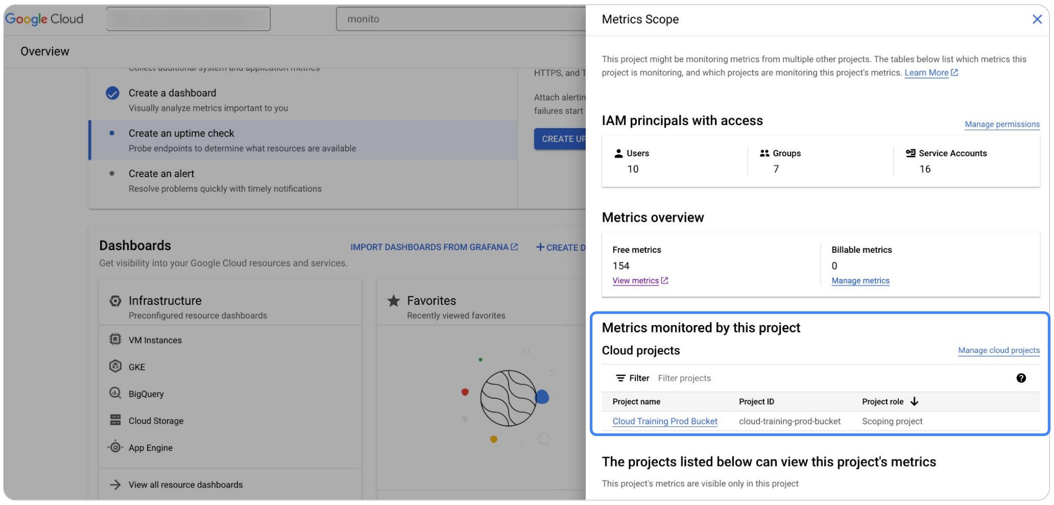


In this section, you explore

- ✓ Monitoring overview
- ✓ Cloud Monitoring architecture patterns
- ✓ **Monitoring multiple projects**
- ✓ Data model and dashboards
- ✓ Query metrics
- ✓ Uptime checks

We have explored a few architecture patterns, let us next explore how you can monitor multiple projects from a single project by using metrics scope.

The default metrics scope contains the current project



Metrics Scope

This project might be monitoring metrics from multiple other projects. The tables below list which metrics this project is monitoring, and which projects are monitoring this project's metrics. Learn More

IAM principals with access — Manage permissions

Users	Groups	Service Accounts
10	7	16

Metrics overview

Free metrics	Billable metrics
154	0
View metrics	Manage metrics

Metrics monitored by this project

Cloud projects — Manage cloud projects

Project name	Project ID	Project role
Cloud Training Prod Bucket	cloud-training-prod-bucket	Scoping project

The projects listed below can view this project's metrics

This project's metrics are visible only in this project

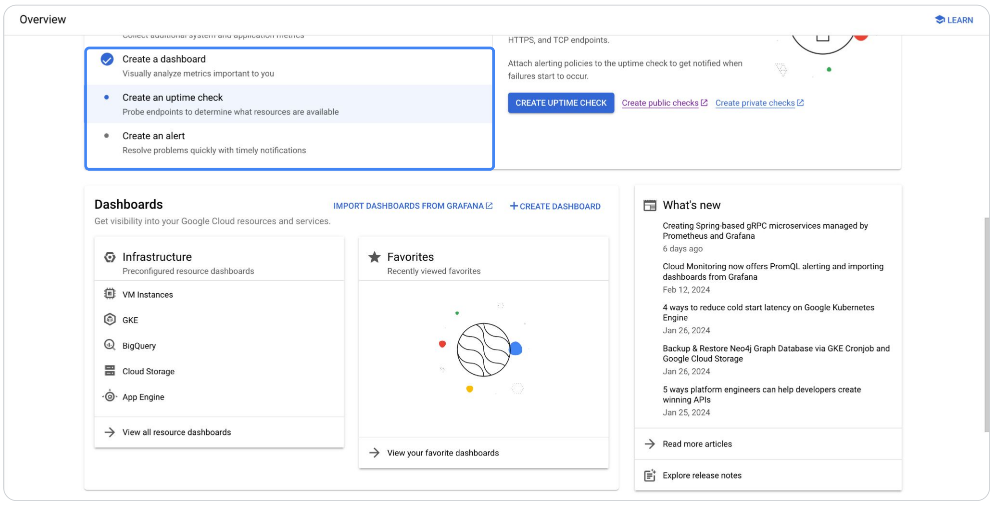
Google Cloud
monito
Overview
Create a dashboard
Visually analyze metrics important to you
Create an uptime check
Probe endpoints to determine what resources are available
Create an alert
Resolve problems quickly with timely notifications
Dashboards
Get visibility into your Google Cloud resources and services.
IMPORT DASHBOARDS FROM GRAFANA
Infrastructure
Preconfigured resource dashboards
VM Instances
GKE
BigQuery
Cloud Storage
App Engine
View all resource dashboards
Favorites
Recently viewed favorites
Filter
Filter projects

Google Cloud

When you go to monitoring settings for a project, you can see that the current metrics scope only has a single project in it, the one it is currently viewing.

When you create a Google Cloud project, that project hosts a metrics scope and becomes the scoping project for that scope.

The default metrics scope contains the current project



Overview

LEARN

Create a dashboard
Visually analyze metrics important to you

Create an uptime check
Probe endpoints to determine what resources are available

Create an alert
Resolve problems quickly with timely notifications

HTTPS, and TCP endpoints.

Attach alerting policies to the uptime check to get notified when failures start to occur.

CREATE UPTIME CHECK | Create public checks | Create private checks

Dashboards
Get visibility into your Google Cloud resources and services.

IMPORT DASHBOARDS FROM GRAFANA | + CREATE DASHBOARD

Infrastructure
Preconfigured resource dashboards
- VM Instances
- GKE
- BigQuery
- Cloud Storage
- App Engine

View all resource dashboards

Favorites
Recently viewed favorites

View your favorite dashboards

What's new

Creating Spring-based gRPC microservices managed by Prometheus and Grafana
6 days ago

Cloud Monitoring now offers PromQL alerting and importing dashboards from Grafana
Feb 12, 2024

4 ways to reduce cold start latency on Google Kubernetes Engine
Jan 26, 2024

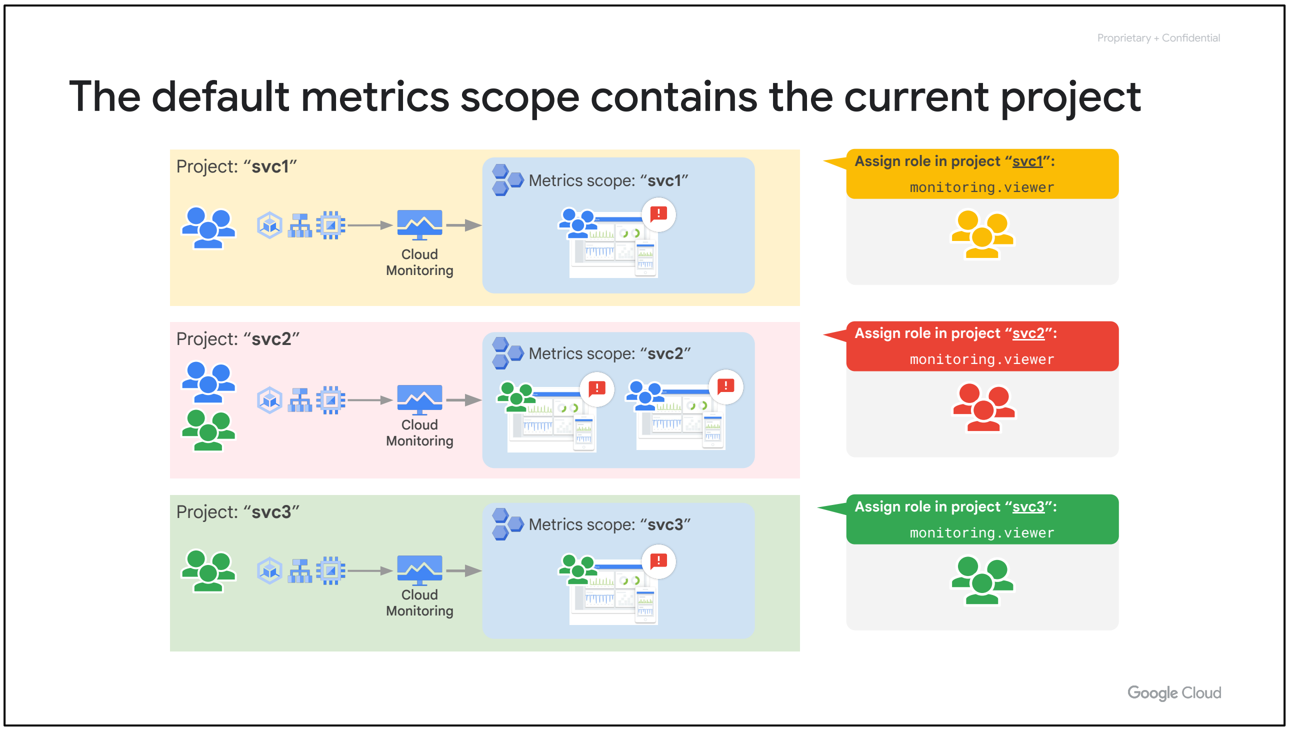
Backup & Restore Neo4j Graph Database via GKE Cronjob and Google Cloud Storage
Jan 26, 2024

5 ways platform engineers can help developers create winning APIs
Jan 25, 2024

Read more articles

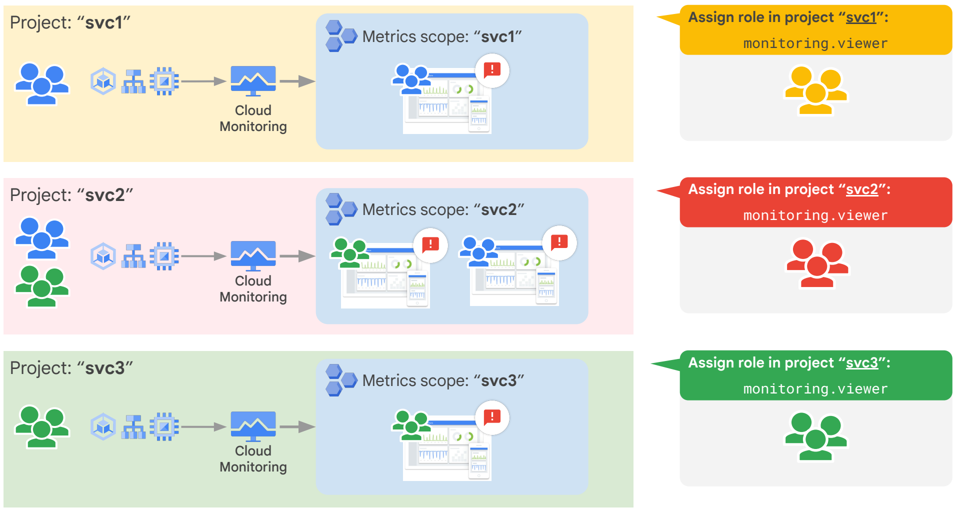
Explore release notes

It stores the alerts, uptime checks, dashboards, and monitoring groups that you configure for the scope.



Proprietary + Confidential

The default metrics scope contains the current project



Google Cloud

For example, if you create a staging project and then access monitoring, you can see the metrics for the resources in the staging project.

This happens for every project you create. Each project creates a metrics scope for itself and hosts monitoring configuration for itself. But what if you want to centralize how that data is stored and how it's accessed?

Since it's possible for one metrics scope to monitor multiple projects, and also a project can be monitored from only a single metrics scope, you will have to decide which relationship will work best for your organizational culture, and this particular project.

Let us explore option one where every project is monitored locally, in that project.

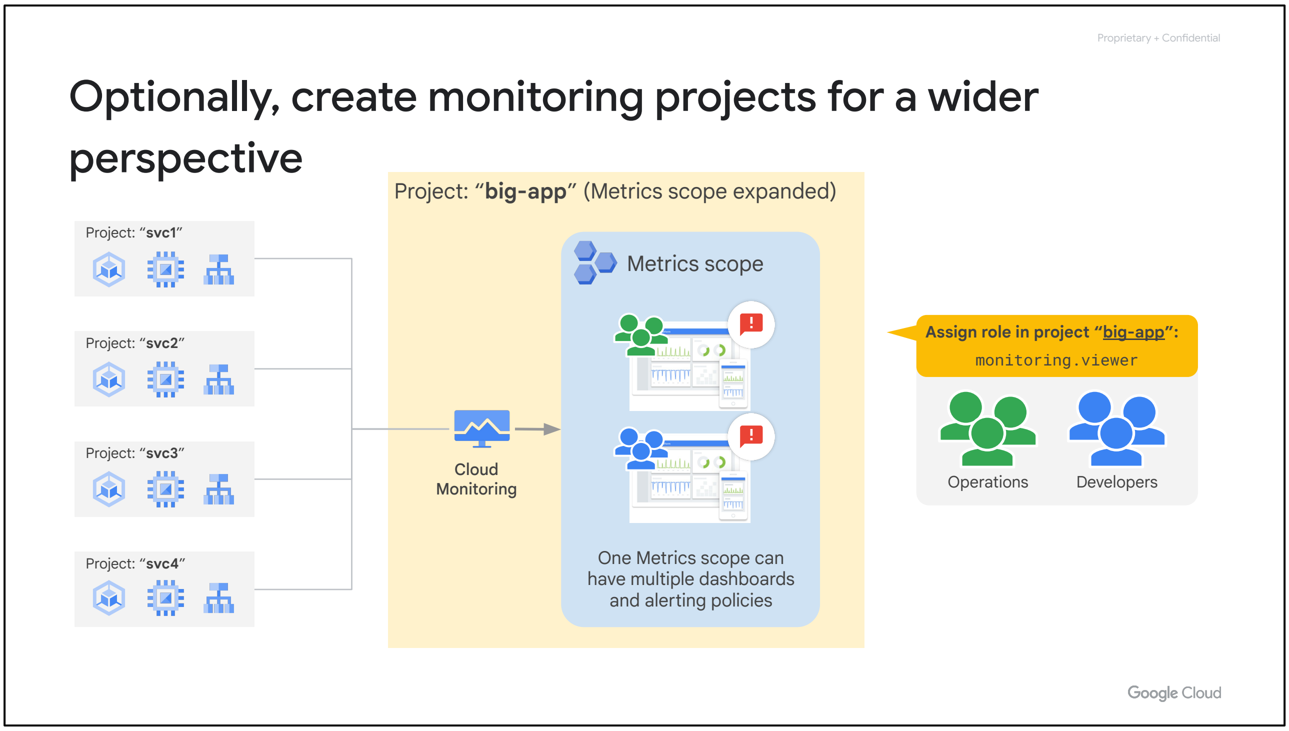
The advantages are clear and obvious separation for each project. If the project contains development-related resources, it's easy to provide access to the dev personnel.

- Project resources and monitoring resources all in the same place.
- Easy to automate, since monitoring becomes a standard part of the initial project setup.

Let us discuss the disadvantage.

- If the application is larger than a single project, you will have limited visibility

- into application performance.



Optionally, create monitoring projects for a wider perspective

Project: "big-app" (Metrics scope expanded)
Project: "svc1"
Project: "svc2"
Project: "svc3"
Project: "svc4"
Metrics scope
Cloud Monitoring
One Metrics scope can have multiple dashboards and alerting policies
Assign role in project "big-app":
monitoring.viewer
Operations
Developers

Strategy B: Single metrics scope for large units of projects, probably an application or application part.

You can add multiple projects to an existing scope. Now, monitoring data for all projects in that scope will be visible. This will let you create dashboards, showing resources from all the projects in the scope, or alerting policies that apply to resources in multiple projects as long as they're in the metrics scope.

Note that the recommended approach for production deployments is to create a dedicated project to host monitoring configuration data and use its metrics scope to set up monitoring for the projects that have actual resources in them. This way, should a project not be necessary anymore and get deleted, the monitoring configuration for all the other projects won't be impacted.

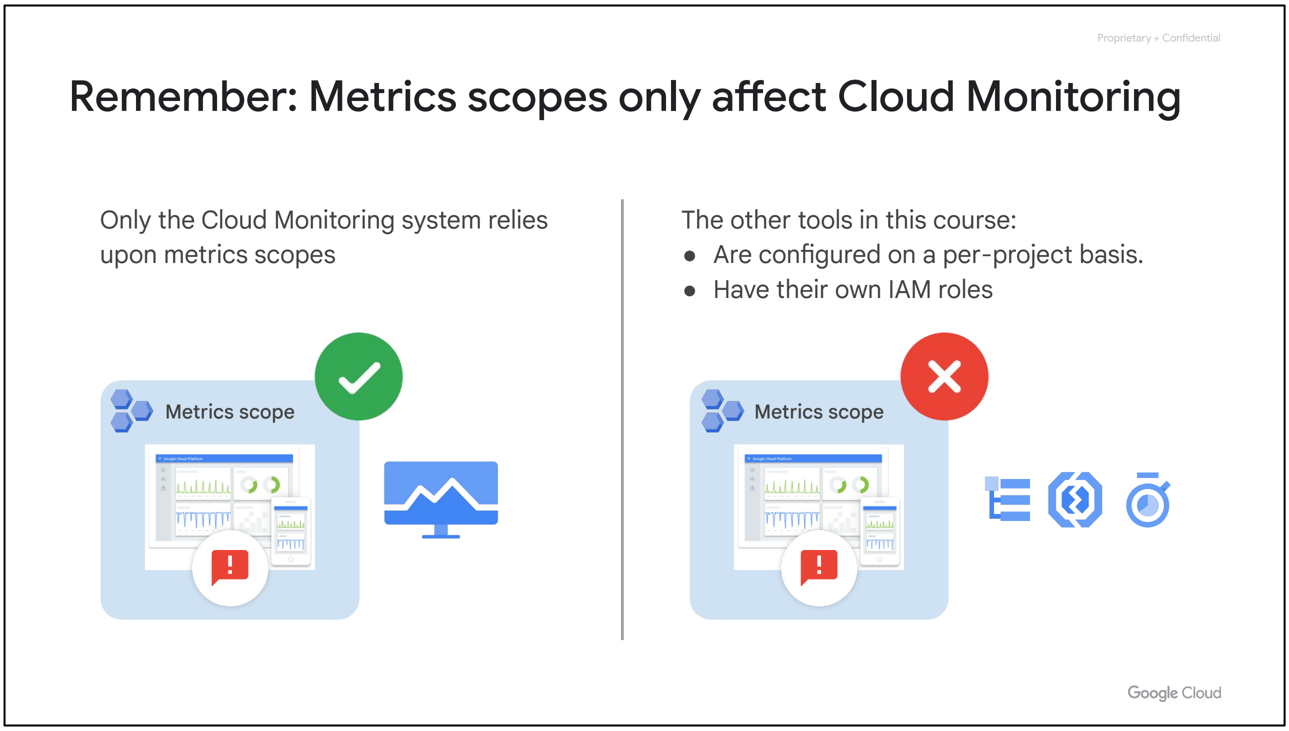
Advantages:

- Single pane of glass that provides visibility into the entire group of related projects.
- Compare non-prod and prod environments easily.

Disadvantages:

- Anyone with IAM permissions to access Cloud Monitoring will be able to see metrics for all environments.
- Monitoring in prod is typically divided among different teams. This approach would not preserve that separation.

Although the metric data and log entries remain in the individual projects, any user who has been granted the role Monitoring Viewer (roles/monitoring.viewer) will have access to the dashboards and have access to all data by default. This means that a role assigned to one person on one project applies equally to all projects monitored by that metrics scope.



Remember: Metrics scopes only affect Cloud Monitoring

Only the Cloud Monitoring system relies upon metrics scopes

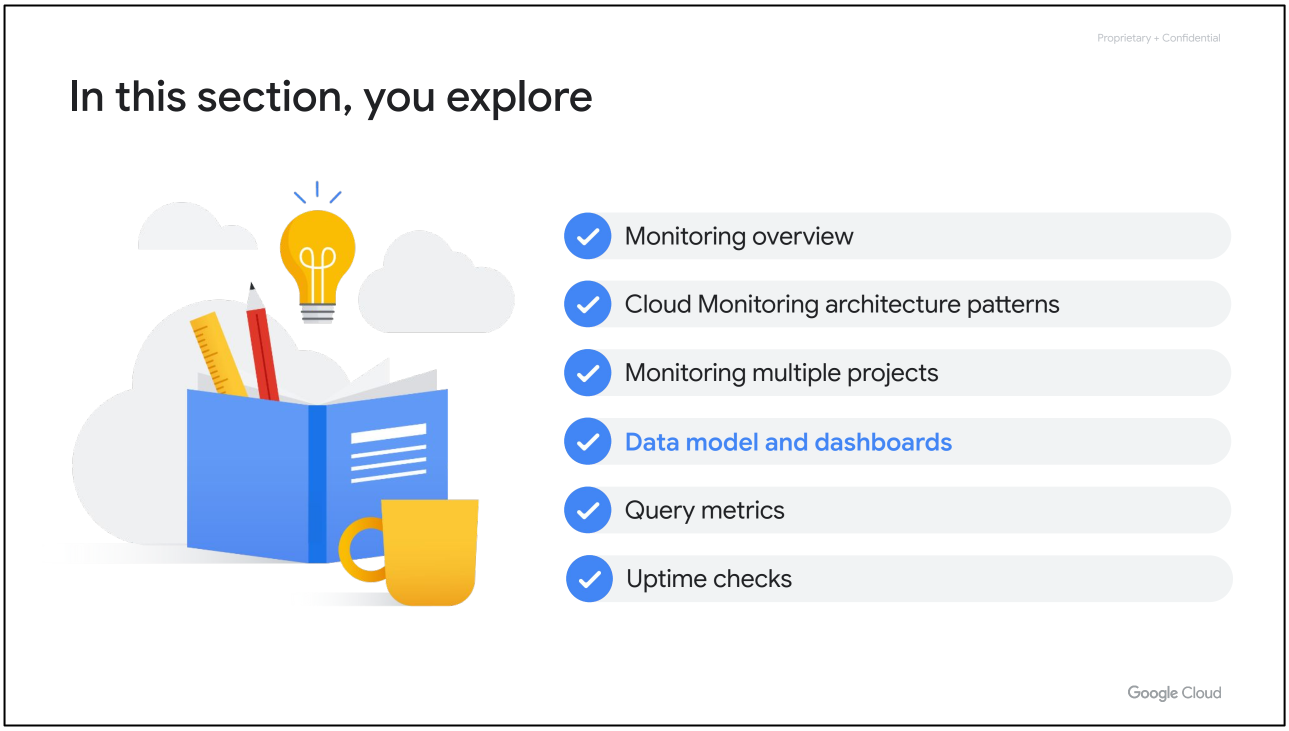
The other tools in this course:

- Are configured on a per-project basis.
- Have their own IAM roles

Metrics scope
Metrics scope

Remember, metrics scope only affects and controls Google Cloud resources related to Cloud Monitoring.

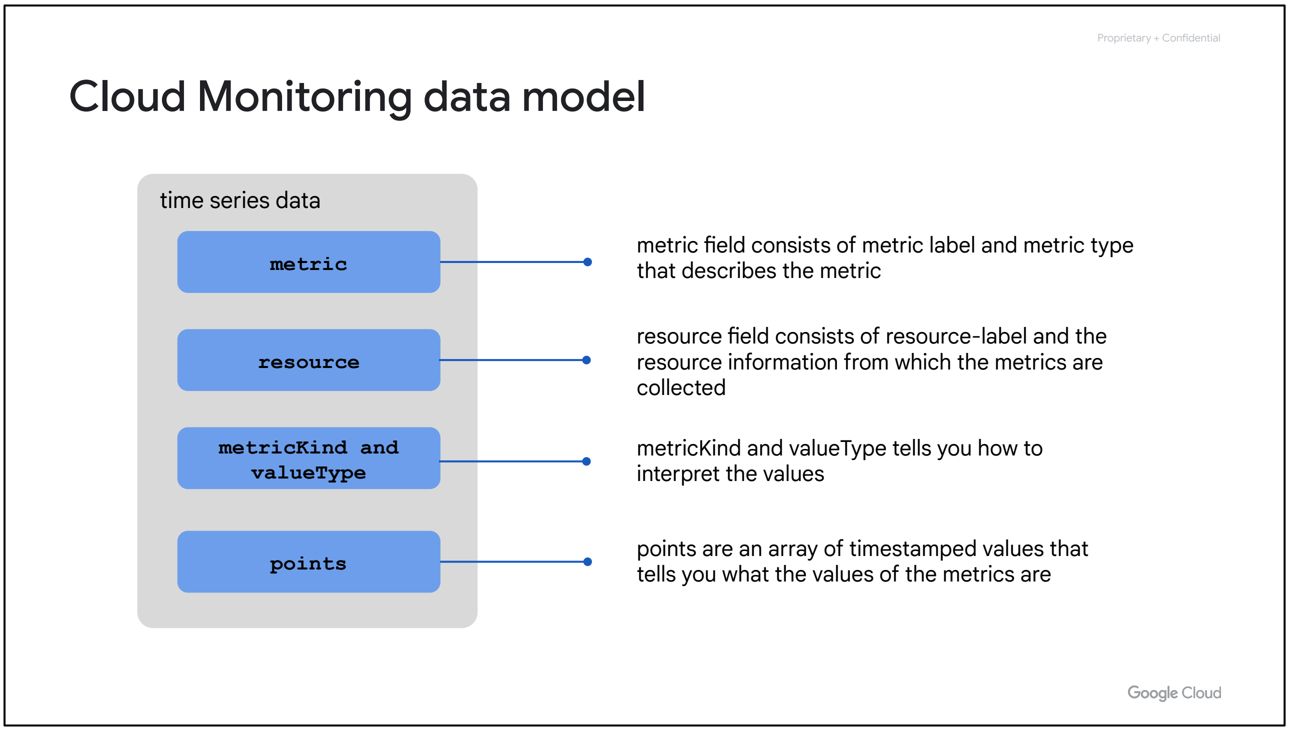
Other tools covered in this course, such as Cloud Logging, Error Reporting, and the Application Performance Management (APM) tools, are strictly project-based and do not rely upon the configuration of the metrics scope or the monitoring IAM roles.



In this section, you explore

- ✓ Monitoring overview
- ✓ Cloud Monitoring architecture patterns
- ✓ Monitoring multiple projects
- ✓ **Data model and dashboards**
- ✓ Query metrics
- ✓ Uptime checks

Now that we can figure out the metrics that Google Cloud has, and can read the documentation to see what each metric actually means, let's put those metrics to work in charts.



Cloud Monitoring data model

metric field consists of metric label and metric type that describes the metric

resource field consists of resource-label and the resource information from which the metrics are collected

metricKind and valueType tells you how to interpret the values

points are an array of timestamped values that tells you what the values of the metrics are

Let us start by understanding the Cloud Monitoring data model.

In general terms, monitoring data is recorded in time series. Each individual time series includes four pieces of information relevant to this discussion:

- The metric field describes the metric itself and records two aspects:
    - The metric-label that represents one combination of label values.
    - The metric type specifies the available labels and describes what is represented by the data points.

- The resource field records:
    - The resource-label represents one combination of label values.
    - The specific monitored resource from which the data was collected.

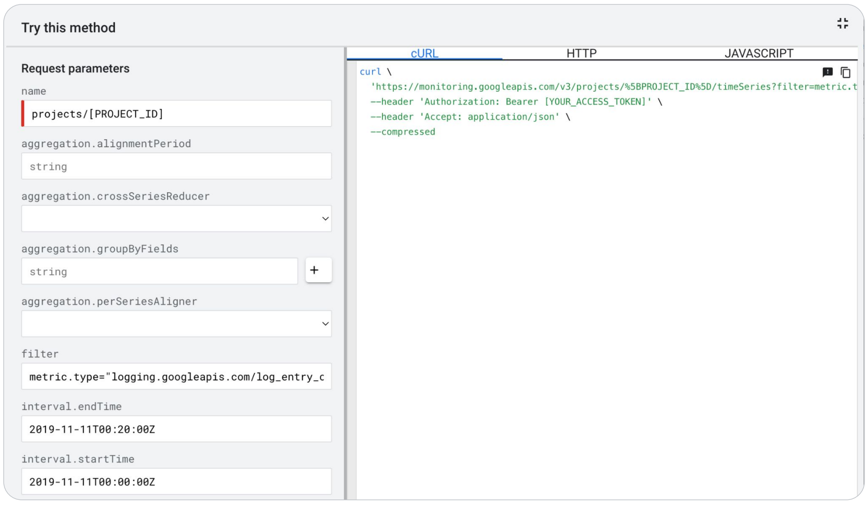
- The metricKind and valueType fields tell you how to interpret the values. The value type is the data type for the measurements. Each time series records the value type (type ValueType) for its data points.
    - For measurements consisting of a single value at a time, the value type tells you how the data is stored:
        - BOOL, a boolean
        - INT64, a 64-bit integer
        - DOUBLE, a double-precision float
        - STRING, a string
    - For distribution measurements, the value isn't a single value but a

    - group of values. The value type for distribution measurements is DISTRIBUTION.

- Each time series includes the metric kind (type MetricKind) for its data points. The kind of metric data tells you how to interpret the values relative to each other. Cloud Monitoring metrics are one of three kinds:
    - A gauge metric, in which the value measures a specific instant in time. For example, metrics measuring CPU utilization are gauge metrics; each point records the CPU utilization at the time of measurement.
    - A delta metric, in which the value measures the change in a time interval. For example, metrics measuring request counts are delta metrics; each value records how many requests were received after the start time, up to and including the end time.
    - A cumulative metric, in which the value constantly increases over time. For example, a metric for “sent bytes” might be cumulative; each value records the total number of bytes sent by a service at that time.

- The points field is an array of timestamped values. The metric type tells you what the values represent. The sample time series has an array with a single data point; in most time series, the array has many more values.

Cloud Monitoring metric type


Try this method

Request parameters

name: projects/[PROJECT_ID]

aggregation.alignmentPeriod: string

aggregation.crossSeriesReducer:

aggregation.groupByFields: string

aggregation.perSeriesAligner:

filter: metric.type="logging.googleapis.com/log_entry_c

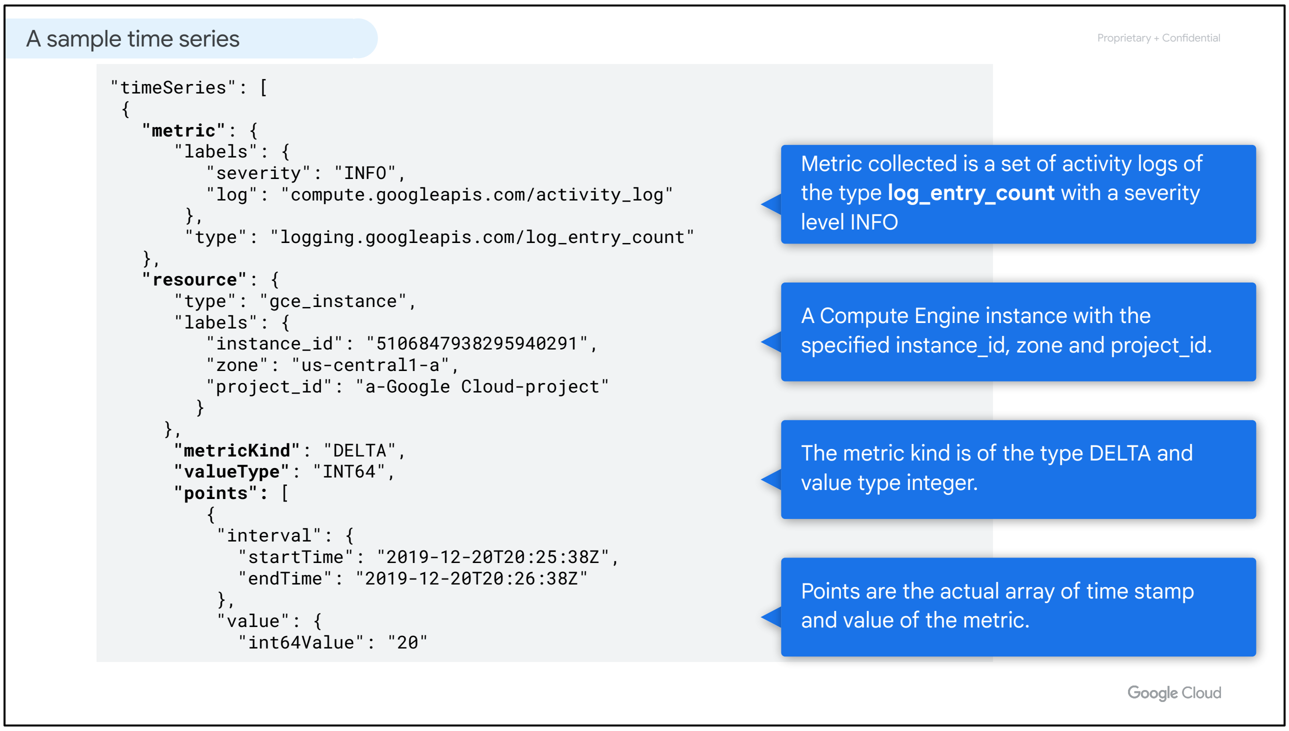
interval.endTime: 2019-11-11T00:20:00Z

interval.startTime: 2019-11-11T00:00:00Z

```
curl \
  'https://monitoring.googleapis.com/v3/projects/%5BPROJECT_ID%5D/timeSeries?filter=metric.t
  --header 'Authorization: Bearer [YOUR_ACCESS_TOKEN]' \
  --header 'Accept: application/json' \
  --compressed
```

Google Cloud

Navigate to the Time Series List API Explorer (usually found in the reference documentation for Google Cloud Monitoring). Locate the API Explorer widget and edit the filter to include the metric type "logging.googleapis.com/log_entry_count" and resource type "gce_instance". Specify the desired start and end time for your data. Upon successful execution, you'll receive the time series data shown on the next slide.



A sample time series

```
"timeSeries": [
 {
   "metric": {
      "labels": {
         "severity": "INFO",
         "log": "compute.googleapis.com/activity_log"
       },
       "type": "logging.googleapis.com/log_entry_count"
   },
   "resource": {
      "type": "gce_instance",
      "labels": {
         "instance_id": "5106847938295940291",
         "zone": "us-central1-a",
         "project_id": "a-Google Cloud-project"
       }
     },
      "metricKind": "DELTA",
      "valueType": "INT64",
      "points": [
         {
          "interval": {
            "startTime": "2019-12-20T20:25:38Z",
            "endTime": "2019-12-20T20:26:38Z"
          },
          "value": {
            "int64Value": "20"
```

Metric collected is a set of activity logs of the type **log_entry_count** with a severity level INFO

A Compute Engine instance with the specified instance_id, zone and project_id.

The metric kind is of the type DELTA and value type integer.

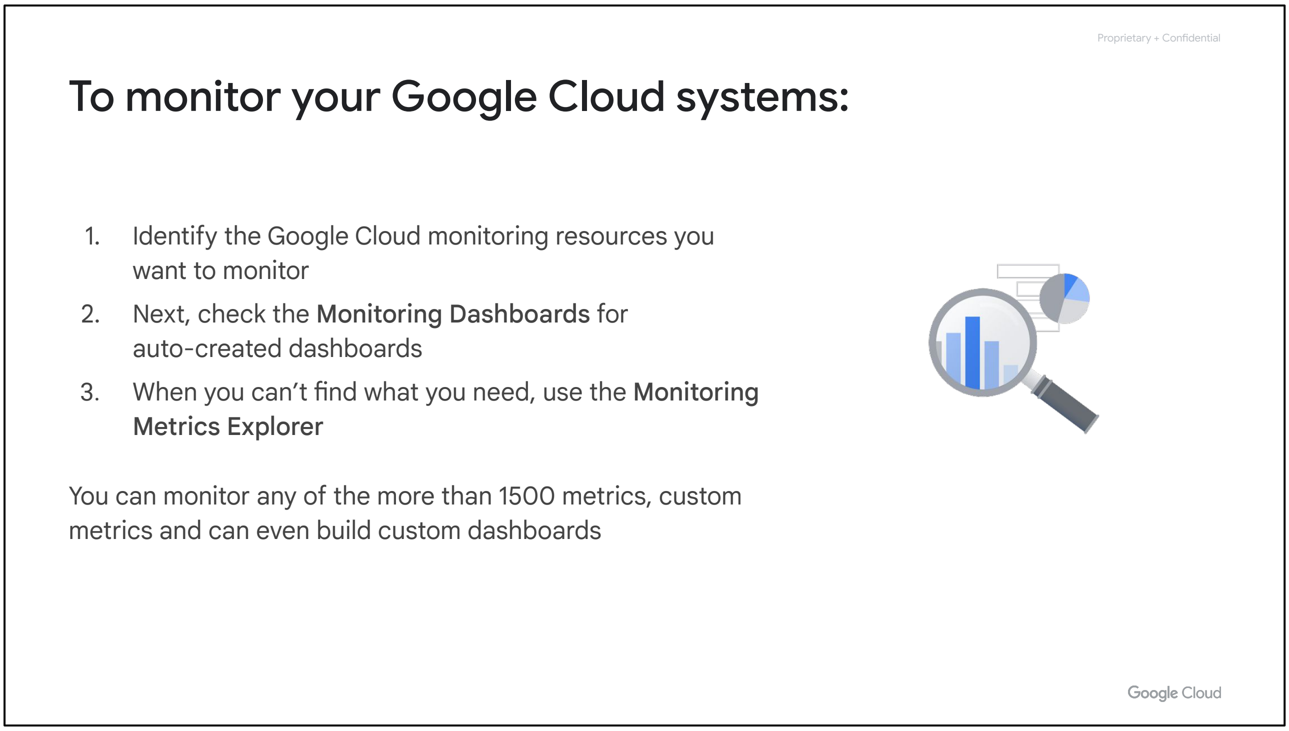
Points are the actual array of time stamp and value of the metric.

Google Cloud

Let us next take a look at an example.

As mentioned earlier monitoring data is recorded in time series. The example shown here is a complete instance of a single time series. Most time series include a lot more data points; this one covers a one-minute interval. All time series have the same structure, with the following fields:

- The metric field indicates that the metric collected is a set of activity logs of the type log_entry_count with a severity level INFO.
- The resource field which indicates that the resource is a Compute Engine instance with the details on the specific instance and project ID.
- The metric kind is of the type DELTA and value type integer. Points are actual array of time stamp and value of the metric.



Proprietary + Confidential

To monitor your Google Cloud systems:

1. Identify the Google Cloud monitoring resources you want to monitor
2. Next, check the **Monitoring Dashboards** for auto-created dashboards
3. When you can't find what you need, use the **Monitoring Metrics Explorer**

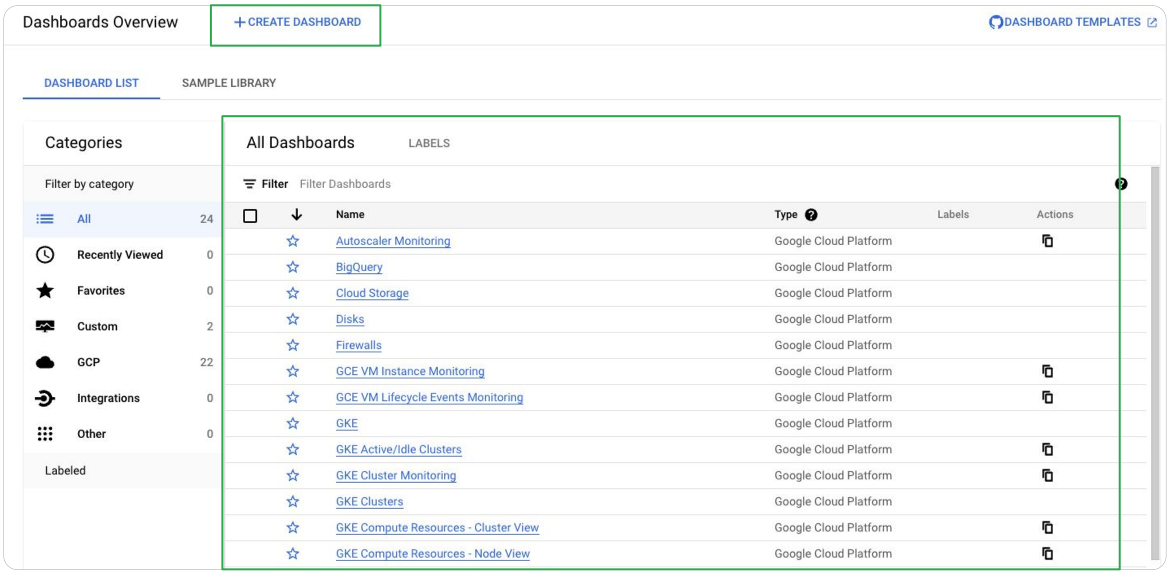
You can monitor any of the more than 1500 metrics, custom metrics and can even build custom dashboards

Google Cloud

- First, identify the Google Cloud Monitoring resources you want to monitor.
- Next, check the Monitoring Dashboards for auto-created dashboards.
- When you can't find what you need, use the Monitoring > Metrics Explorer.

You can monitor any of the more than 1500 metrics, custom metrics and can even build custom dashboards.

Predefined and custom dashboards



Categories	
All	24
Recently Viewed	0
Favorites	0
Custom	2
GCP	22
Integrations	0
Other	0
Labeled	

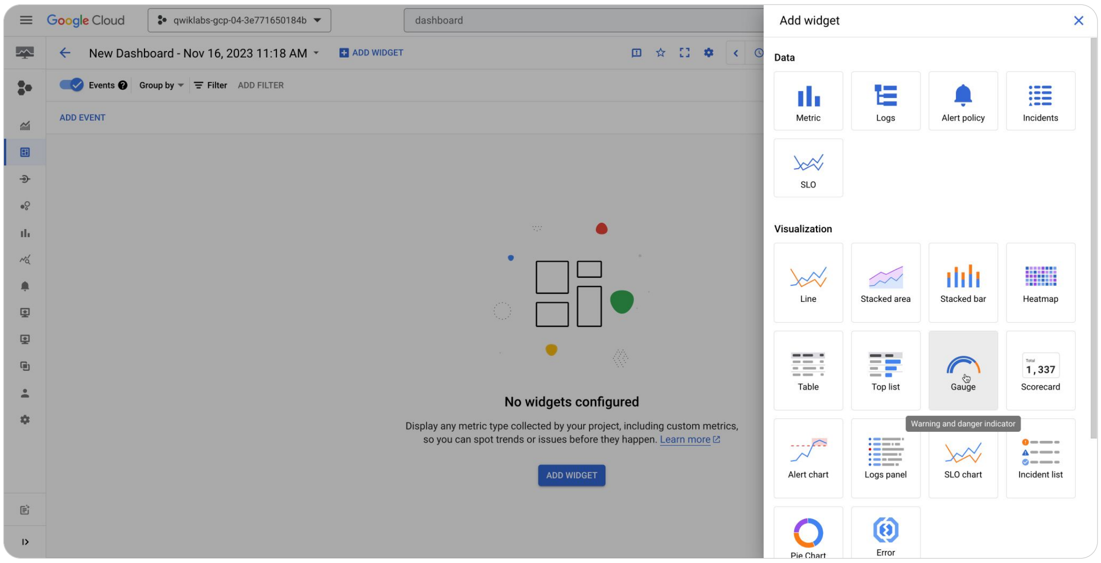
Name	Type	Labels	Actions
Autoscaler Monitoring	Google Cloud Platform		
BigQuery	Google Cloud Platform		
Cloud Storage	Google Cloud Platform		
Disks	Google Cloud Platform		
Firewalls	Google Cloud Platform		
GCE VM Instance Monitoring	Google Cloud Platform		
GCE VM Lifecycle Events Monitoring	Google Cloud Platform		
GKE	Google Cloud Platform		
GKE Active/Idle Clusters	Google Cloud Platform		
GKE Cluster Monitoring	Google Cloud Platform		
GKE Clusters	Google Cloud Platform		
GKE Compute Resources - Cluster View	Google Cloud Platform		
GKE Compute Resources - Node View	Google Cloud Platform		

Google Cloud

Dashboards are a way for you to view and analyze metric data that is important to you. They give you graphical representations on the main signal data in such a way as to help you make key decisions about your Google Cloud-based resources.

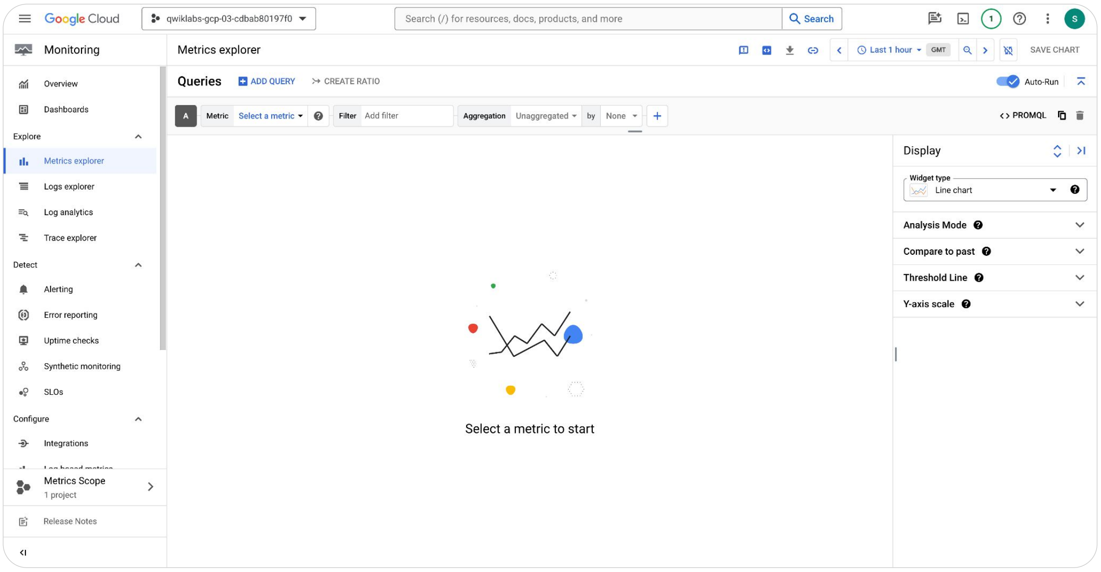
One of the changing aspects of monitoring is Google's commitment to providing more opinionated default information. Google Cloud sees that your project contains Compute Engine VMs, or a GKE Cluster, so Monitoring auto-creates dashboards for you that radiate the information that Google thinks is important for those two resource types. As you add more resources, Google will continue to add more default dashboards. These dashboards form a great monitoring foundation on which you can build.

Dashboard Builder



You can also use the Dashboard Builder to visualize application metrics that you are interested in. You can select the chart type and filter the metrics based on your requirements.

Metrics Explorer



Google Cloud

Frequently, the easiest way to start chart creation is to build an ad-hoc chart with Google's Metrics Explorer. Metrics Explorer lets you build charts for any metric collected by your project. With it, you can:

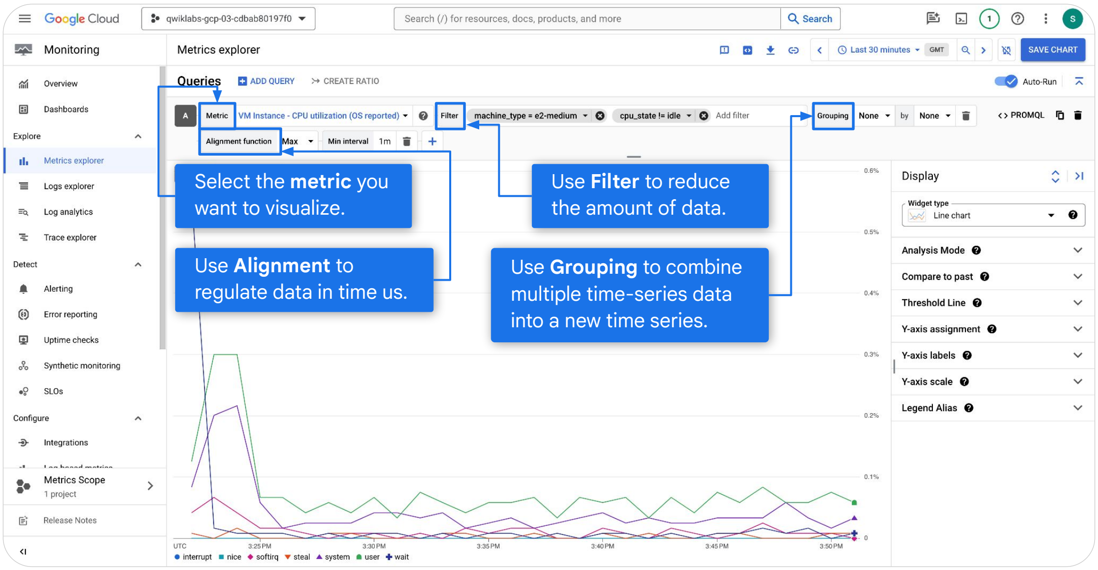
- Save charts you create to a custom dashboard.
- Share charts by their URL.
- View the configuration for charts as JSON.

Most importantly, you can use Metrics Explorer as a tool to explore data that you don't need to display long term on a custom dashboard.

As seen on this slide, the Metrics Explorer interface consists of three primary regions:

- A configuration region, where you pick the metric and its options
- The chart that displays the selected metric
- The display panel, where you can configure the axis, set a threshold lines, and more

Metrics Explorer



Metric: VM Instance - CPU utilization (OS reported)

Filter: machine_type = e2-medium; cpu_state != idle; Add filter

Grouping: None by None

Alignment function: Max; Min interval: 1m

Select the **metric** you want to visualize.

Use **Filter** to reduce the amount of data.

Use **Alignment** to regulate data in time us.

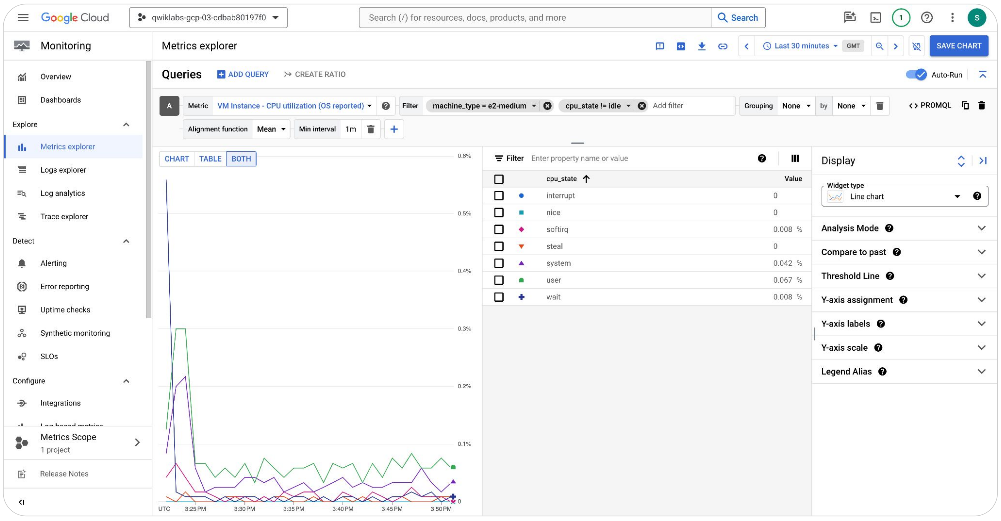
Use **Grouping** to combine multiple time-series data into a new time series.

You define a chart by specifying both what data should display and how the chart should display it.

1. **Metric:** To populate the chart, you must specify at least one pair of values, the *monitored resource type* (or *monitored resource*, or just *resource*), and the *metric type* (also called the *metric descriptor*, or just *metric*).
2. **Filter:** You can reduce the amount of data returned for a metric by specifying a filter. Filtering removes data from the chart by excluding time series that don't meet your criteria. The result is fewer lines on the chart and, hopefully, a better signal to noise ratio. When you click in the **Filter** field, a panel that contains lists of criteria by which you can filter appears. In broad strokes, you can filter by resource group, by name, by resource label, and by the metric label.
   a. In this example, we filter by machine type. The zone can then be compared to a direct value, like "=e2-medium," or by using the “=” operator, to any valid regular expression. You can check the documentation If you want to see the fully supported filter syntax.
3. **Grouping:** You can reduce the amount of data returned for a metric by combining different time series. To combine multiple time series, you typically specify a grouping and a function. Grouping is done by label values. The function defines how all time-series data within a group are combined into a new time series.
4. **Alignment:** Alignment creates a new time series in which the raw data has been regularized in time so it can be combined with other aligned time series. Alignment produces time series with regularly spaced data.

Alignment

Break data into regular time buckets



Metric: VM Instance - CPU utilization (OS reported)
Filter: machine_type = e2-medium, cpu_state != idle
Grouping: None by None
Alignment function: Mean
Min interval: 1m
Widget type: Line chart

	cpu_state	Value
●	interrupt	0
■	nice	0
◆	softirq	0.008 %
▼	steal	0
▲	system	0.042 %
●	user	0.067 %
✚	wait	0.008 %

Google Cloud

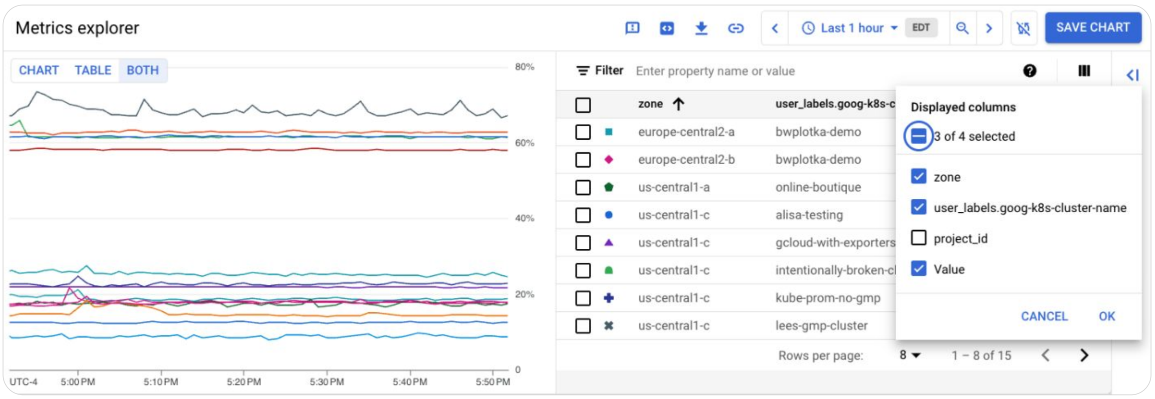
A time series is a set of data points in temporal order. To align a time series is to break the data points into regular buckets of time, the *alignment period*. Multiple time series must be aligned before they can be combined.

Alignment is a prerequisite to aggregation across time series, and monitoring does it automatically, by using default values. You can override these defaults by using the alignment options, which are the Alignment function and the Min alignment period**.**

- The **alignment period** determines the length of time for subdividing the time series. For example, you can break a time series into one-minute chunks or one-hour chunks in the **Min interval** field. The data in each period is summarized so that a single value represents that period. The default alignment period, which is also the minimum, is one minute. Although you can set the alignment interval for your data, time series might be realigned when you change the time interval displayed on a chart or change the zoom level.

- The **alignment function** determines how to summarize the data in each alignment period. The functions include the sum, the mean, and so forth. Valid alignment choices depend on the kind and type of metric data a time series stores.

Legend configuration



		zone ↑	user_labels.goog-k8s-c
☐	■	europe-central2-a	bwplotka-demo
☐	◆	europe-central2-b	bwplotka-demo
☐	⬟	us-central1-a	online-boutique
☐	●	us-central1-c	alisa-testing
☐	▲	us-central1-c	gcloud-with-exporters
☐	●	us-central1-c	intentionally-broken-cl
☐	✚	us-central1-c	kube-prom-no-gmp
☐	✖	us-central1-c	lees-gmp-cluster

Rows per page: 8 ▾ 1 – 8 of 15

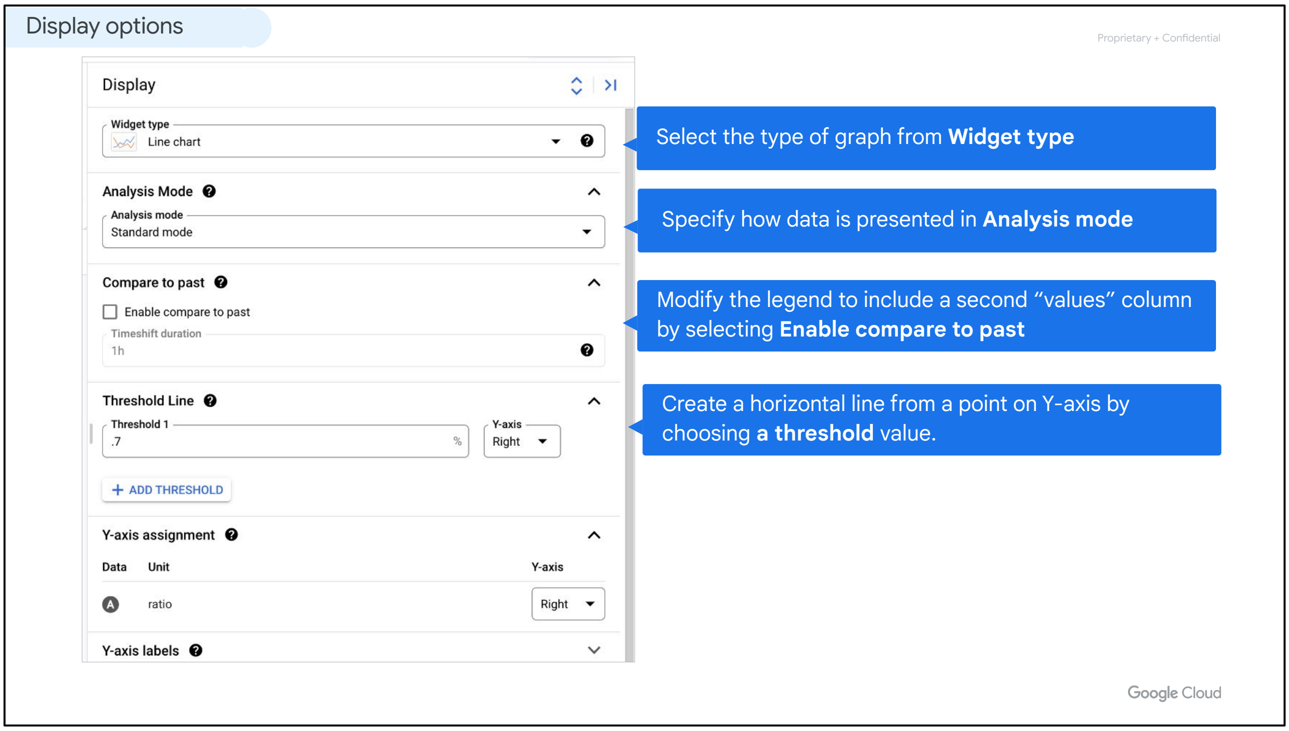
Displayed columns

3 of 4 selected

- [✓] zone
- [✓] user_labels.goog-k8s-cluster-name
- [ ] project_id
- [✓] Value

CANCEL OK

Click any of the legend column headers to sort by that field. The legend columns included in the chart's display are configurable.



Display options

Proprietary + Confidential

Display

Widget type: Line chart

Analysis Mode
Analysis mode: Standard mode

Compare to past
☐ Enable compare to past
Timeshift duration: 1h

Threshold Line
Threshold 1: .7 %
Y-axis: Right

+ ADD THRESHOLD

Y-axis assignment

Data	Unit	Y-axis
A	ratio	Right

Y-axis labels

Select the type of graph from **Widget type**

Specify how data is presented in **Analysis mode**

Modify the legend to include a second "values" column by selecting **Enable compare to past**

Create a horizontal line from a point on Y-axis by choosing **a threshold** value.

Google Cloud

A **chart's widget type** and **its analysis mode** setting determine how the chart displays data. For example, when you create a line chart, each time series is shown by a line with a unique color. However, you can configure a line chart to display statistical measures such as the mean and moving average.

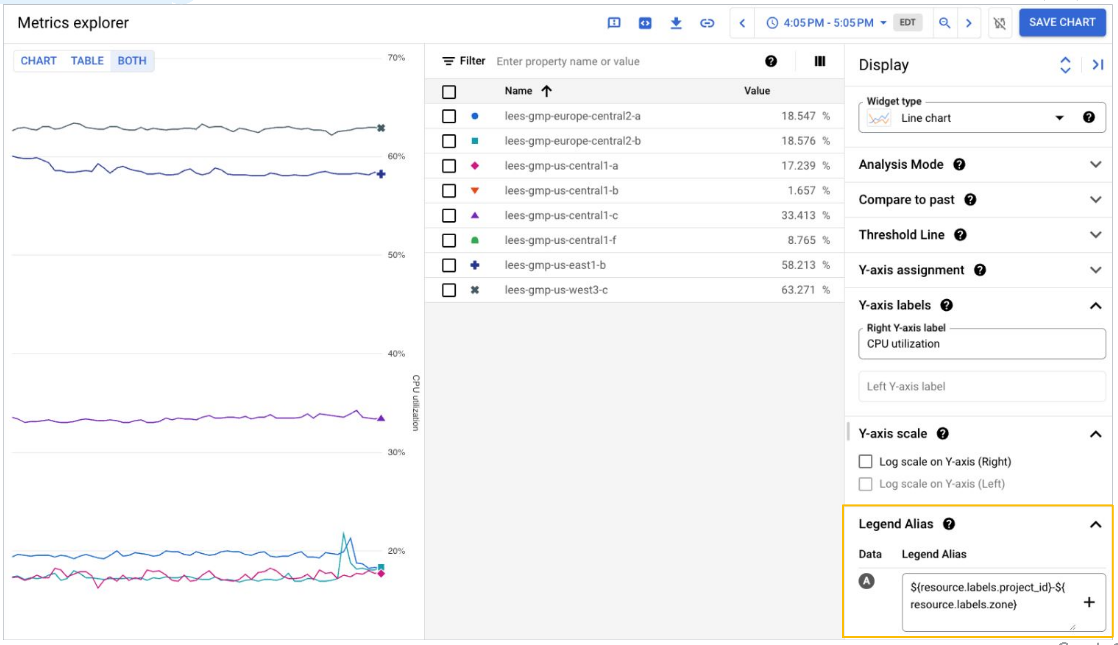
There are three analysis modes:

- **Standard mode** displays each time series with a unique color.
- **Stats mode** displays common statistical measures for the data in a chart.
- **X-Ray mode** displays each time series with a translucent gray color. Each line is faint, and where lines overlap or cross, the points appear brighter. Therefore, this mode is most useful on charts with many lines. Overlapping lines create bands of brightness, which indicate the normal behavior within a metrics group.

Threshold line: The Threshold option creates a horizontal line from a point on the Y-axis. The line provides a visual reference for the chosen threshold value. You can add a threshold that refers to a value on the left Y-axis or the right Y-axis.

Compare to past: When you use **Compare to Past** mode on a chart, the legend is modified to include a second "values" column. The current Value column becomes Today, and the past values column is named appropriately—for example, Last Week.

Legend Alias



	Name	Value
●	lees-gmp-europe-central2-a	18.547 %
■	lees-gmp-europe-central2-b	18.576 %
◆	lees-gmp-us-central1-a	17.239 %
▼	lees-gmp-us-central1-b	1.657 %
▲	lees-gmp-us-central1-c	33.413 %
●	lees-gmp-us-central1-f	8.765 %
✚	lees-gmp-us-east1-b	58.213 %
✖	lees-gmp-us-west3-c	63.271 %

Display

Widget type: Line chart

Analysis Mode

Compare to past

Threshold Line

Y-axis assignment

Y-axis labels
- Right Y-axis label: CPU utilization
- Left Y-axis label

Y-axis scale
- Log scale on Y-axis (Right)
- Log scale on Y-axis (Left)

Legend Alias

Data	Legend Alias
A	${resource.labels.project_id}-${resource.labels.zone}

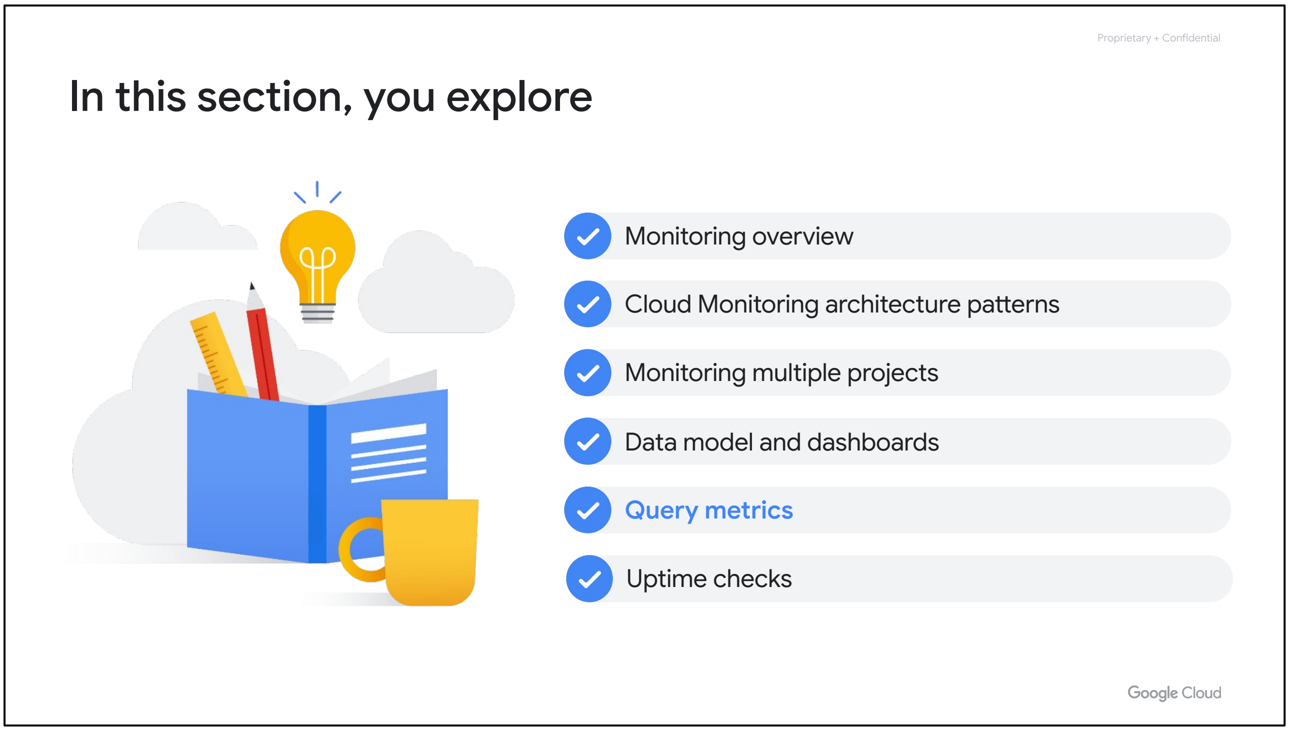
The **Legend Alias** field lets you customize a description for the time series on your chart. These descriptions appear on the tooltip for the chart and on the chart legend in the **Name** column. By default, the descriptions in the legend are created for you from the values of different labels in your time series. Because the system selects the labels, the results might not be helpful to you. To build a template for descriptions, use this field.

You can enter plain text and templates in the **Legend Alias** field. When you add a template, you add an expression that is evaluated when the legend is displayed.

To add a legend template to a chart, do the following:

1. In the **Display** pane, expand expand **Legend Alias**.
2. Click **+** (Plus) and select an entry from the menu.

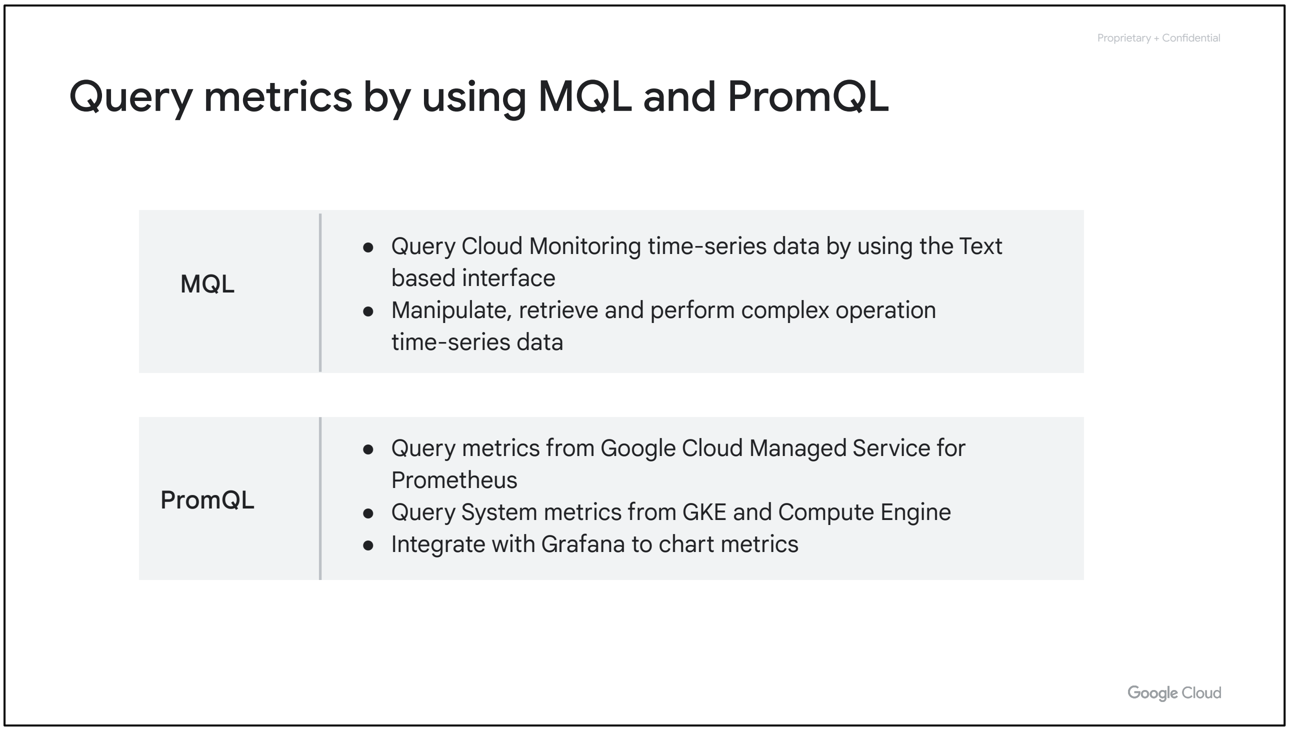
In this example, you see the mix of the variable, ${resource.labels.zone} - ${metric.labels.instance_name}. You can see the resulting output in the chart legend.



In this section, you explore

- ✓ Monitoring overview
- ✓ Cloud Monitoring architecture patterns
- ✓ Monitoring multiple projects
- ✓ Data model and dashboards
- ✓ **Query metrics**
- ✓ Uptime checks

Before we wrap our discussion on dashboards, charts, and the Metrics Explorer, let's examine a more versatile way of interacting with metrics by leveraging the query languages such as Monitoring Query Language (MQL) and PromQL.


Proprietary + Confidential

Query metrics by using MQL and PromQL

MQL	● Query Cloud Monitoring time-series data by using the Text based interface ● Manipulate, retrieve and perform complex operation time-series data
PromQL	● Query metrics from Google Cloud Managed Service for Prometheus ● Query System metrics from GKE and Compute Engine ● Integrate with Grafana to chart metrics

Google Cloud

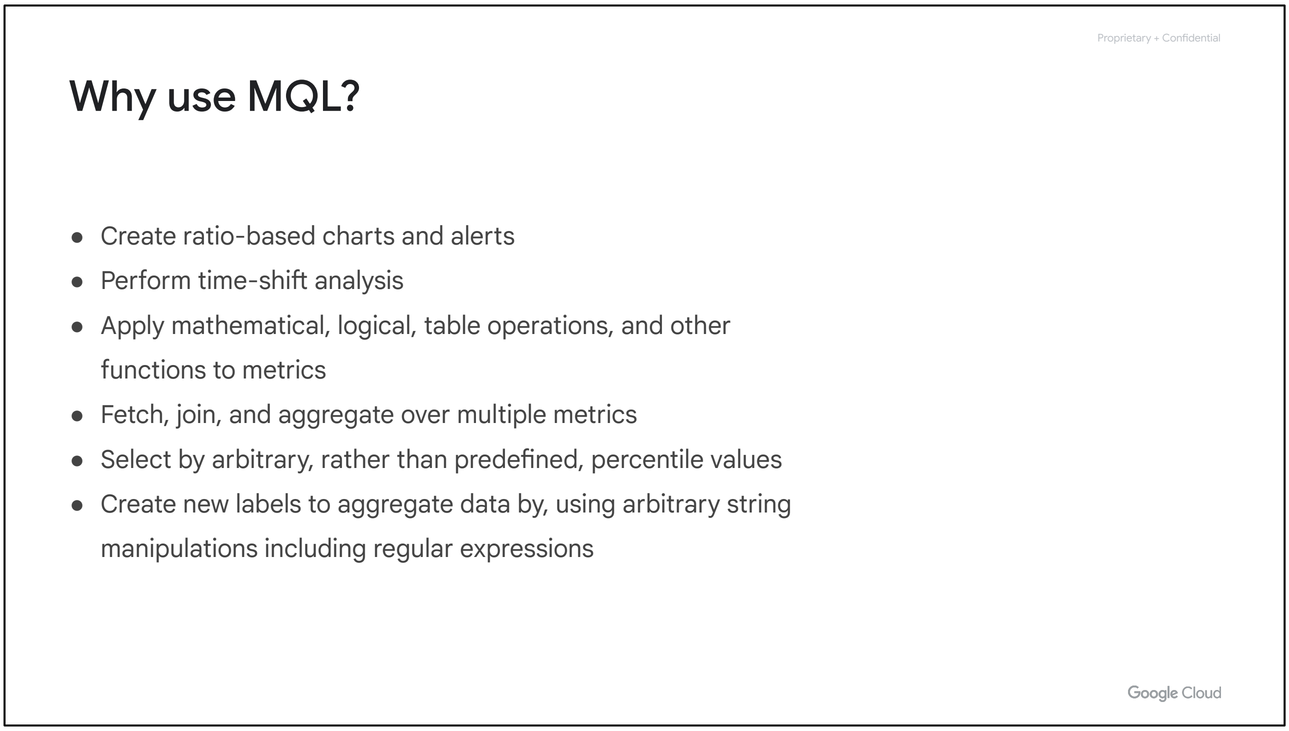
MQL: MQL is an advanced query language. Monitoring Query Language (MQL) provides an expressive, text-based interface to Cloud Monitoring time-series data. By using MQL, you can retrieve, filter, and manipulate time-series data.

PromQL provides an alternative to the Metrics Explorer menu-driven and Monitoring Query Language (MQL) interfaces for creating charts and dashboards.

You can use PromQL to query and chart Cloud Monitoring data from the following sources:

- Google Cloud services, like Google Kubernetes Engine or Compute Engine, that write metrics described in the lists of Cloud Monitoring system metrics.
- User-defined metrics, like log-based metrics and Cloud Monitoring user-defined metrics.
- Google Cloud Managed Service for Prometheus, the fully managed multi-cloud solution for Prometheus from Google Cloud. For information about the managed service, including support from PromQL, see Google Cloud Managed Service for Prometheus.

You can also use tools like Grafana to chart metric data ingested into Cloud Monitoring. Available metrics include metrics from Managed Service for Prometheus and Cloud Monitoring metrics documented in the lists of metrics.


Proprietary + Confidential

Why use MQL?

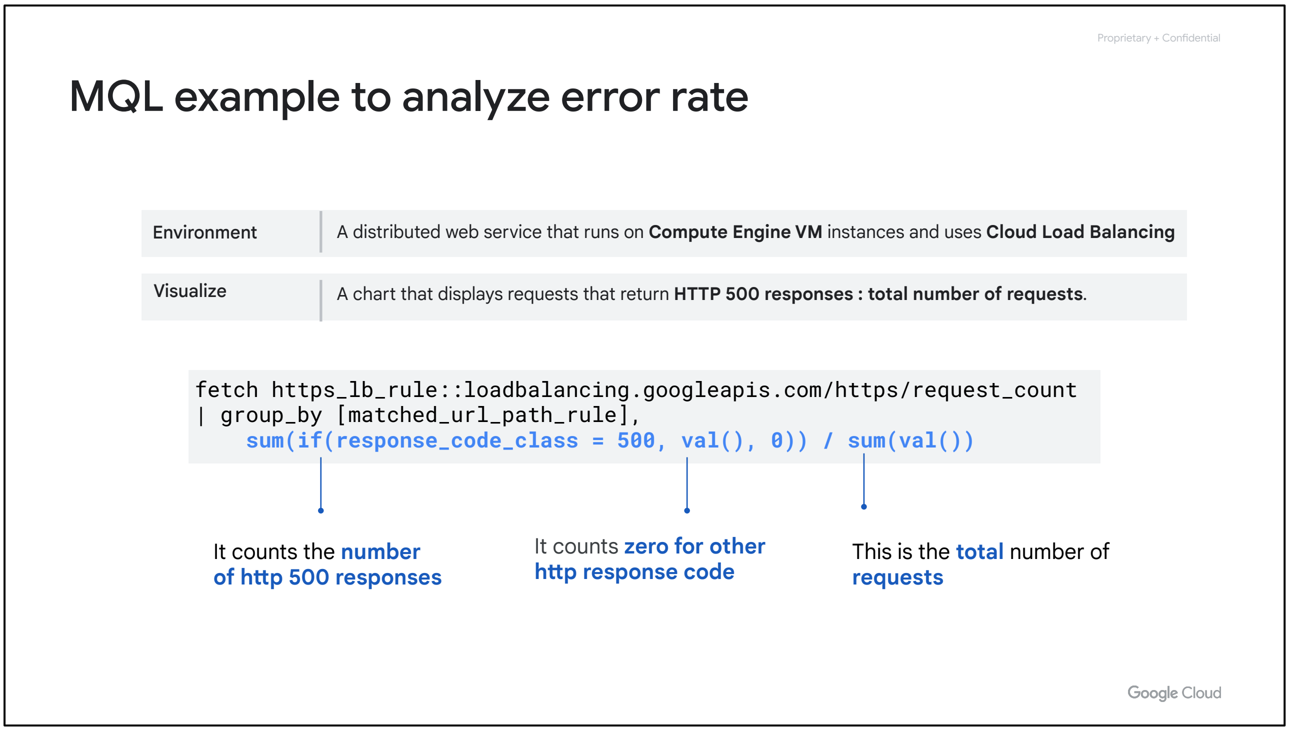
- Create ratio-based charts and alerts
- Perform time-shift analysis
- Apply mathematical, logical, table operations, and other functions to metrics
- Fetch, join, and aggregate over multiple metrics
- Select by arbitrary, rather than predefined, percentile values
- Create new labels to aggregate data by, using arbitrary string manipulations including regular expressions

Google Cloud

These are a few examples of when you can use MQL:

- Create ratio-based charts and alerts
- Perform time-shift analysis (compare metric data week over week, month over month, year over year, etc.)
- Apply mathematical, logical, table operations, and other functions to metrics
- Fetch, join, and aggregate over multiple metrics
- Select by arbitrary, rather than predefined, percentile values
- Create new labels to aggregate data by, using arbitrary string manipulations including regular expressions

Whether you need to perform joins, display arbitrary percentages, or even make advanced calculations, the use cases for MQL are unlimited.


Proprietary + Confidential

MQL example to analyze error rate

Environment	A distributed web service that runs on **Compute Engine VM** instances and uses **Cloud Load Balancing**
Visualize	A chart that displays requests that return **HTTP 500 responses : total number of requests**.

```
fetch https_lb_rule::loadbalancing.googleapis.com/https/request_count
| group_by [matched_url_path_rule],
    sum(if(response_code_class = 500, val(), 0)) / sum(val())
```

It counts the **number of http 500 responses**

It counts **zero for other http response code**

This is the **total** number of **requests**

Google Cloud

MQL is built using operations and functions. Operations are linked together by using the common “pipe” idiom, where the output of one operation becomes the input to the next. Linking operations makes it possible to build complex queries incrementally. In the same way you compose and chain commands and data through pipes on the Linux command line, you can fetch metrics and apply operations by using MQL.

For a more advanced example, suppose you built a distributed web service that runs on Compute Engine VM instances and uses Cloud Load Balancing, and you want to analyze error rate, which is one of the SRE “golden signals.”

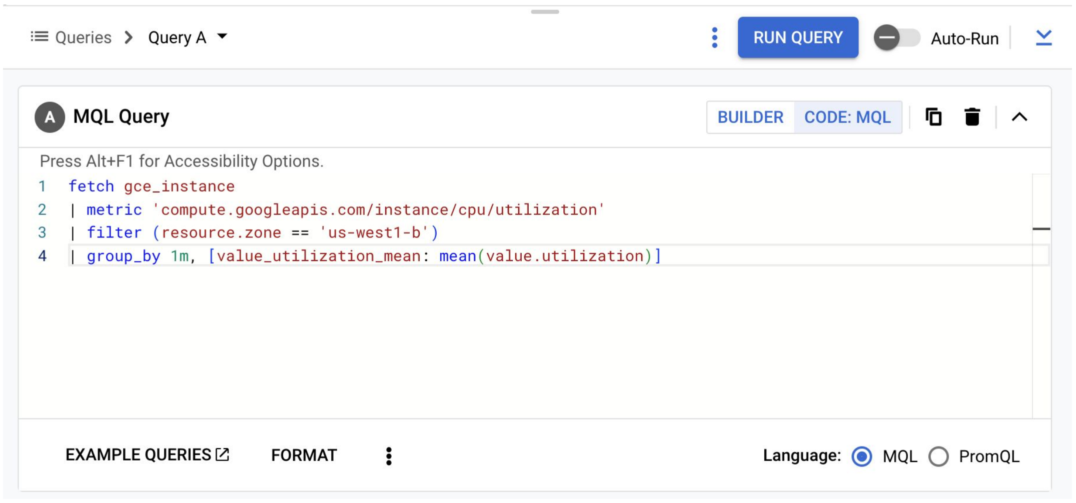
You want to see a chart that displays the ratio of requests that return HTTP 500 responses (internal errors) to the total number of requests; that is, the request-failure ratio. The loadbalancing.googleapis.com/https/request_count metric type has a response_code_class label, which captures the class of response codes.

This query uses an aggregation expression built on the ratio of two sums:

- The first sum uses the if function to count 500-valued HTTP responses and a count of 0 for other HTTP response codes. The sum function computes the count of the requests that returned 500.
- The second sum adds up the counts for all requests, as represented by val().

The two sums are then divided, which results in the ratio of 500 responses to all responses.

Accessing MQL and PromQL interface



Press Alt+F1 for Accessibility Options.

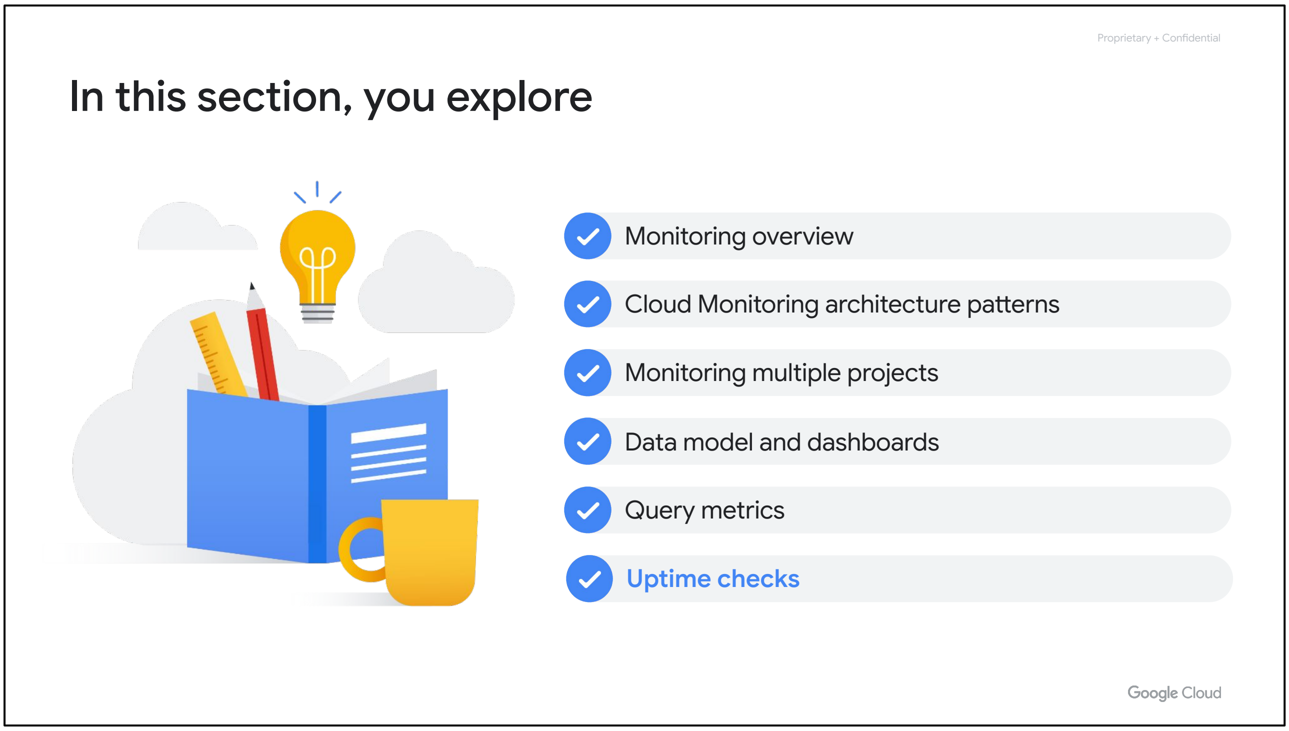
```
fetch gce_instance
| metric 'compute.googleapis.com/instance/cpu/utilization'
| filter (resource.zone == 'us-west1-b')
| group_by 1m, [value_utilization_mean: mean(value.utilization)]
```

Language: [✓ MQL] PromQL

Google Cloud

You can use queries by navigating to Metrics explorer and simply click the CODE button. You can use the radio button to switch between MQL and PromQL.

We will cover PromQL in a later module.

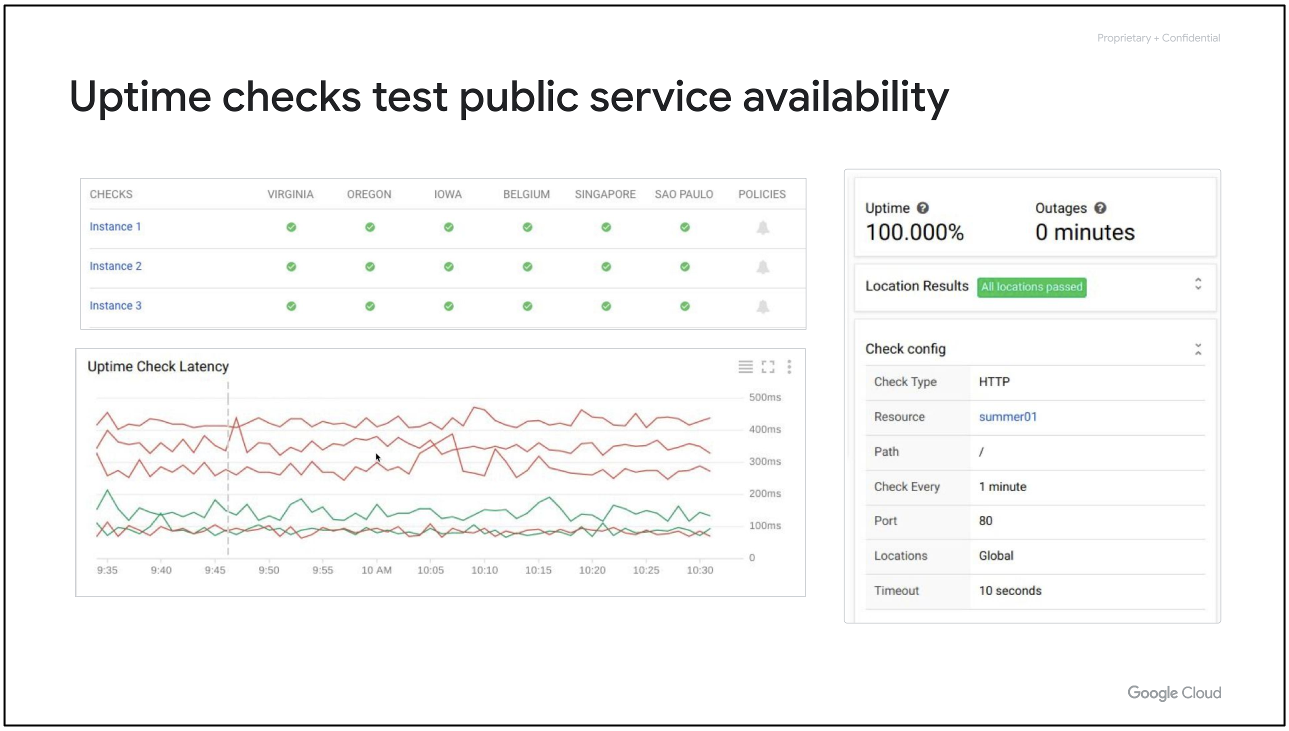


In this section, you explore

- ✓ Monitoring overview
- ✓ Cloud Monitoring architecture patterns
- ✓ Monitoring multiple projects
- ✓ Data model and dashboards
- ✓ Query metrics
- ✓ **Uptime checks**

Google Cloud

Another monitoring component before we wrap this module is uptime checks.



Uptime checks test public service availability

CHECKS	VIRGINIA	OREGON	IOWA	BELGIUM	SINGAPORE	SAO PAULO	POLICIES
Instance 1	✓	✓	✓	✓	✓	✓	🔔
Instance 2	✓	✓	✓	✓	✓	✓	🔔
Instance 3	✓	✓	✓	✓	✓	✓	🔔

Uptime 100.000%

Outages 0 minutes

Location Results All locations passed

Check config

Check Type	HTTP
Resource	summer01
Path	/
Check Every	1 minute
Port	80
Locations	Global
Timeout	10 seconds

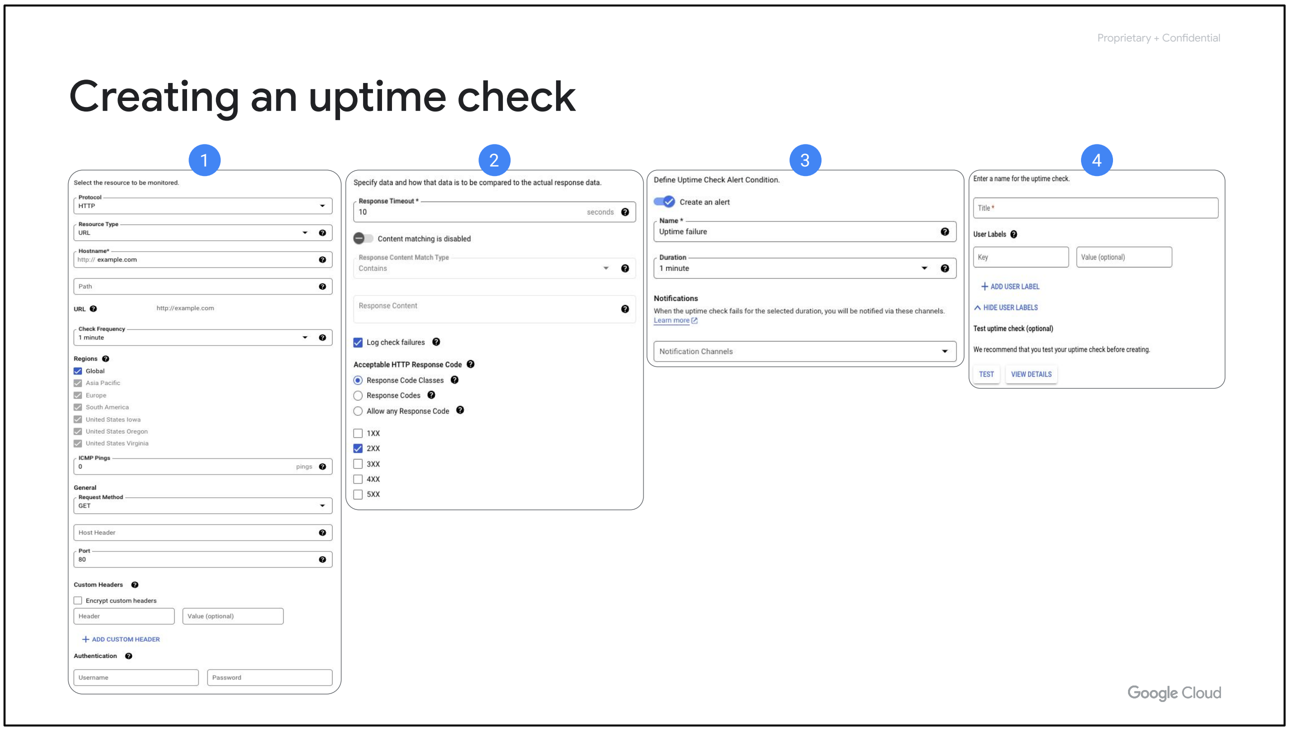
Google Cloud

Uptime checks can be configured to test the availability of your public services from locations around the world, as you can see on this slide. The type of uptime check can be set to HTTP, HTTPS, or TCP. The resource you can check are App Engine application, a Compute Engine instance, a URL of a host, or an AWS instance or load balancer.

For each uptime check, you can create an alerting policy and view the latency of each global location.

Uptime checks can help us ensure that our externally facing services are running and that we aren't burning our error budgets unnecessarily. Here is an example of an HTTP uptime check. The resource is checked every minute with a 10-second timeout. Uptime checks that do not get a response within this timeout period are considered failures.

So far, there is a 100% uptime with no outages.



Proprietary + Confidential

Creating an uptime check

1

Select the resource to be monitored.

Protocol: HTTP

Resource Type: URL

Hostname*: http:// example.com

Path

URL: http://example.com

Check Frequency: 1 minute

Regions
- [✓] Global
- [✓] Asia Pacific
- [✓] Europe
- [✓] South America
- [✓] United States Iowa
- [✓] United States Oregon
- [✓] United States Virginia

ICMP Pings: 0 pings

General

Request Method: GET

Host Header

Port: 80

Custom Headers
- [ ] Encrypt custom headers

Header | Value (optional)

+ ADD CUSTOM HEADER

Authentication

Username | Password

2

Specify data and how that data is to be compared to the actual response data.

Response Timeout *: 10 seconds

Content matching is disabled

Response Content Match Type: Contains

Response Content

- [✓] Log check failures

Acceptable HTTP Response Code
- [●] Response Code Classes
- [ ] Response Codes
- [ ] Allow any Response Code

- [ ] 1XX
- [✓] 2XX
- [ ] 3XX
- [ ] 4XX
- [ ] 5XX

3

Define Uptime Check Alert Condition.

[✓] Create an alert

Name *: Uptime failure

Duration: 1 minute

Notifications

When the uptime check fails for the selected duration, you will be notified via these channels.
Learn more

Notification Channels

4

Enter a name for the uptime check.

Title *

User Labels

Key | Value (optional)

+ ADD USER LABEL

HIDE USER LABELS

Test uptime check (optional)

We recommend that you test your uptime check before creating.

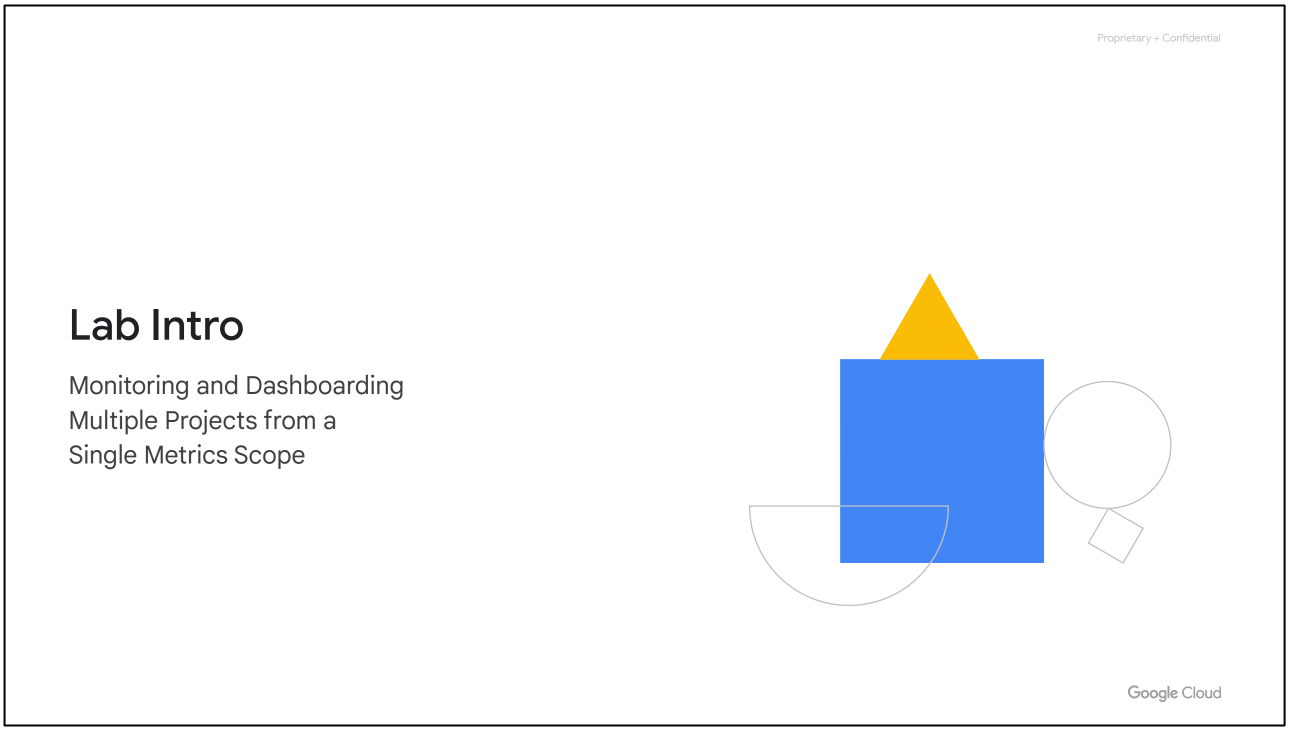
TEST | VIEW DETAILS

Google Cloud

Uptime checks are easy to create. In **Monitoring**, navigate to **Uptime Checks** and click **Create Uptime Check**.

Give the uptime check a name/title that is descriptive. Select the check type protocol, the resource type, and appropriate information for that resource type. A URL, for example, would need a hostname and an optional page path.

A number of optional advanced options are available, including logging failures, narrowing the locations in the world from where test connections are made, the addition of custom headers, check timeout, and authentication. The interface also makes it easy to create an alert for failing uptime checks.

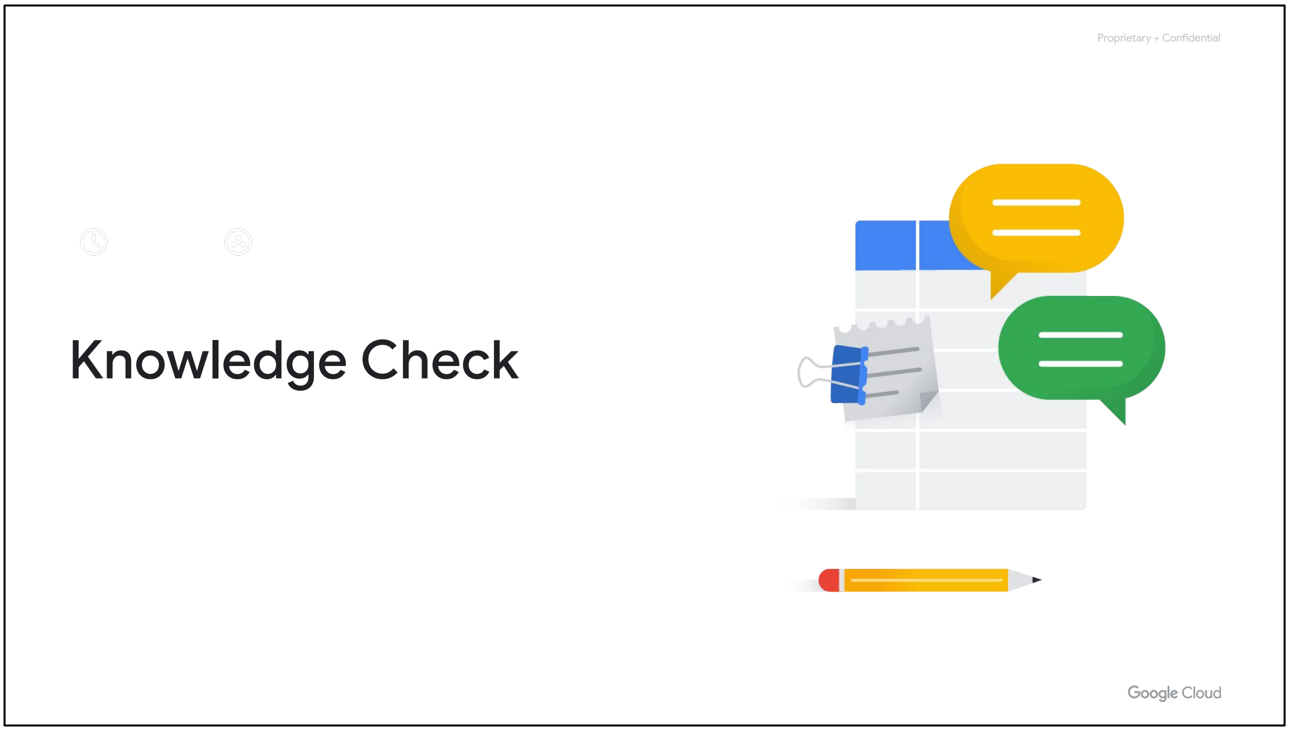


Lab Intro

Monitoring and Dashboarding Multiple Projects from a Single Metrics Scope

Google Cloud

Cloud Monitoring empowers users with the ability to monitor multiple projects from a single metrics scope. In this exercise, you start with three Google Cloud projects: two have monitorable resources, and you will use the third one to host a metrics scope. You attach the two resource projects to the metrics scope, build uptime checks, and construct a centralized dashboard.



Knowledge Check

Quiz | Question 1

Question

You want to figure out a way to analyze error rate in your load balancing environment. Which interface helps you view a chart with a ratio of 500 responses to all responses?

A. Metrics Explorer

B. MQL

C. Uptime check

D. Liveness probe

Quiz | Question 1

Answer

You want to figure out a way to analyze error rate in your load balancing environment. Which interface helps you view a chart with a ratio of 500 responses to all responses

A. Metrics Explorer

[✓ B. MQL]

C. Uptime check

D. Liveness probe

Quiz | Question 2

Question

You want to be notified if your application is down. What Google Cloud tool makes this easy?

A. Uptime check

B. Health check

C. Readiness probe

D. Liveness probe

Quiz | Question 2

Answer

You want to be notified if your application is down. What Google Cloud tool makes this easy?

[✓ A. Uptime check]
B. Health check
C. Readiness probe
D. Liveness probe

Quiz | Question 3

Question

What is the name of the project that hosts a metrics scope?

A. Hosting project

B. Monitored project

C. Evaluating project

D. Scoping project

Quiz | Question 3

Answer

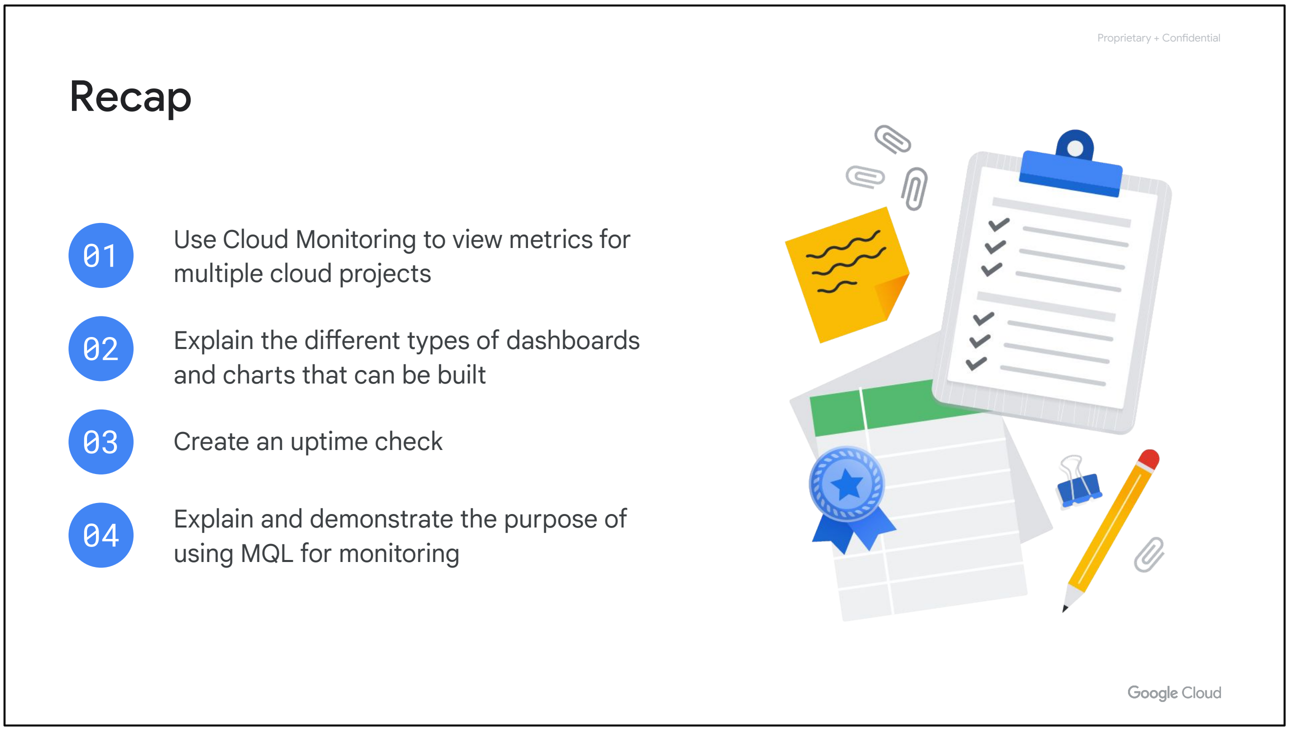
What is the name of the project that hosts a metrics scope?

A. Hosting project

B. Monitored project

C. Evaluating project

D. Scoping project ✓



Proprietary + Confidential

Recap

01 Use Cloud Monitoring to view metrics for multiple cloud projects

02 Explain the different types of dashboards and charts that can be built

03 Create an uptime check

04 Explain and demonstrate the purpose of using MQL for monitoring

Google Cloud

In this module, you learned how to:

- Use Cloud Monitoring to view metrics for multiple cloud projects
- Explain the different types of dashboards and charts that can be built
- Create an uptime check
- Explain and demonstrate the purpose of using MQL for monitoring