



## Monitoring Critical Systems



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



## Agenda

Observability Architecture

Dashboards

Uptime Checks

Monitoring is all about keeping track of exactly what's happening with the resources we've spun up inside of Google's 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 Cloud 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 Google created default dashboards, 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.

---

# Agenda

Observability Architecture

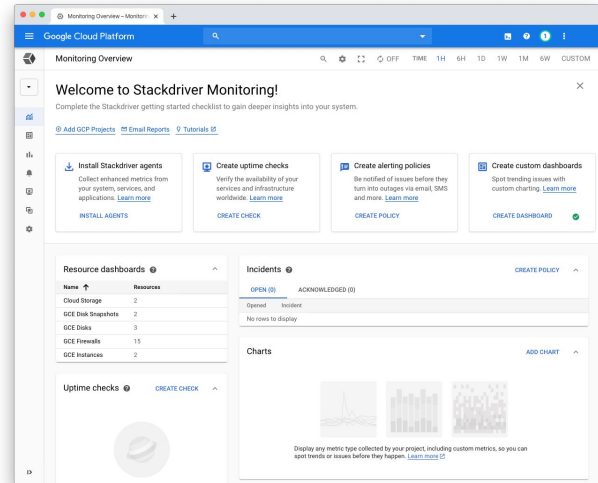
Dashboards

Uptime Checks

Let's start with the observability architecture.

# Monitoring is Configured via Workspaces

- ▶ Single Pane of Glass
- ▶ Cross-project visibility
- ▶ Monitor resources in Google Cloud and AWS



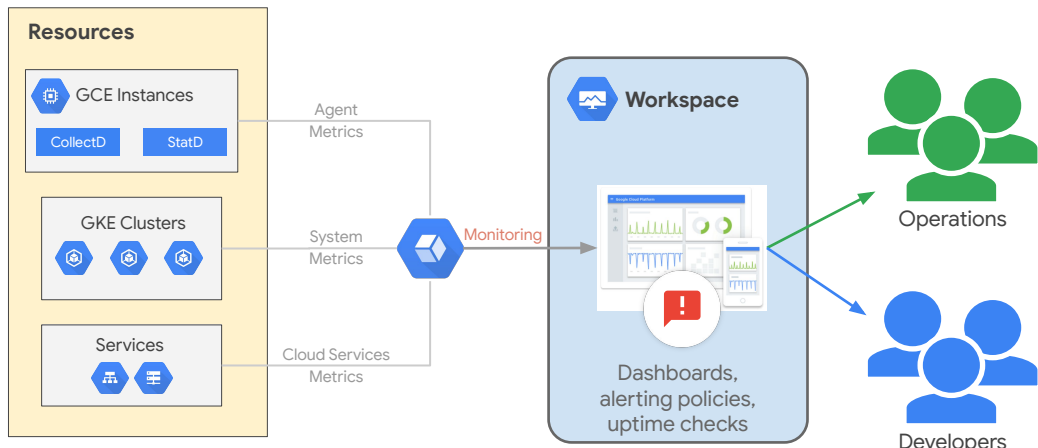
Google Cloud Monitoring uses Workspaces to organize monitoring information. A Workspace is a tool for monitoring resources contained in one or more Google Cloud projects.

It offers a unified view, or single pane of glass, through which those resources can be watched.

With the ability to monitor resources in the current, and in up to 100 other projects, monitoring workspaces offer excellent cross-project visibility.

The monitored resources may be part of Google Cloud or AWS.

## Organize Your Monitoring Efforts with Workspaces



Monitoring workspaces help organize your monitoring efforts. They serve as central, secured, access hubs for monitoring information, dashboards, alerting policies, and uptime checks. This information is made available, IAM permitting, to both operations and developer personnel.

## Centralize and Consolidate Resource Monitoring



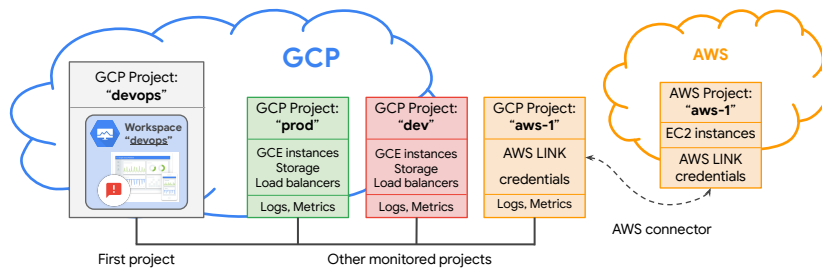
A project may belong to only one Workspace



Create a dedicated project to host cross-project Workspaces



But a Workspace may monitor up to 100 projects



A Workspace is wrapped by a Google Cloud host Project, which the Workspace always monitors.

However, you can configure a Workspace to monitor up to 100 other Google Cloud projects and AWS accounts.

This allows you to create dedicated monitoring projects to host cross-project workspaces, and thus, expand your single pane of glass outside of the current project.

## A Workspace Belongs to a Single Host Project



Contains configuration data for the Workspace



Project name becomes the Workspace name



A Workspace belongs to a single host project. The host project stores all of the configuration content for dashboards, alerting policies, uptime checks, notification channels, and group definitions that you configure. If you delete the host project, you also delete the Workspace.

The name of the Workspace is set to the name of the host project. This isn't configurable.

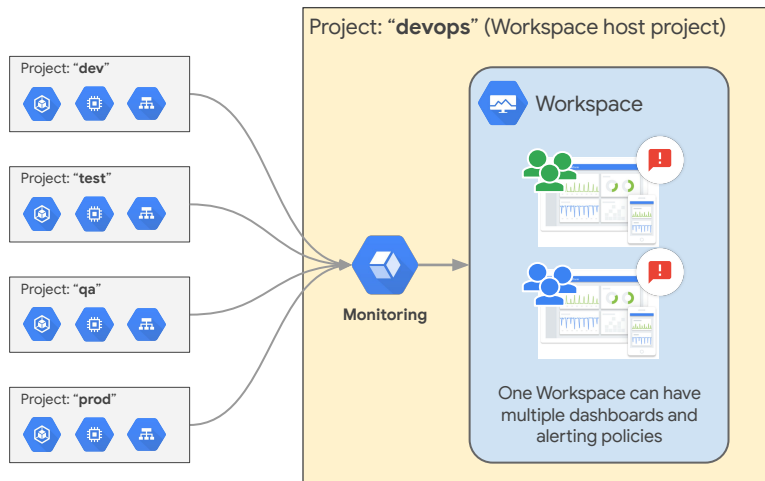
## One Workspace Can Monitor Multiple Projects



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



## One Workspace Can Monitor Multiple Projects



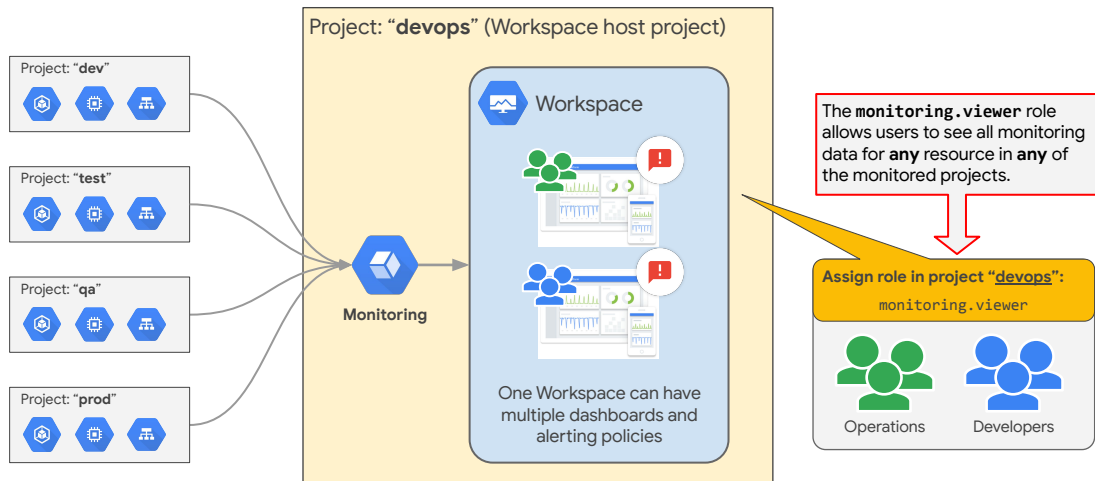
 Google Cloud

**Strategy A - Single monitoring Workspace for large units of projects, probably an application or application part.**

Advantages:

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

## One Workspace Can Monitor Multiple Projects

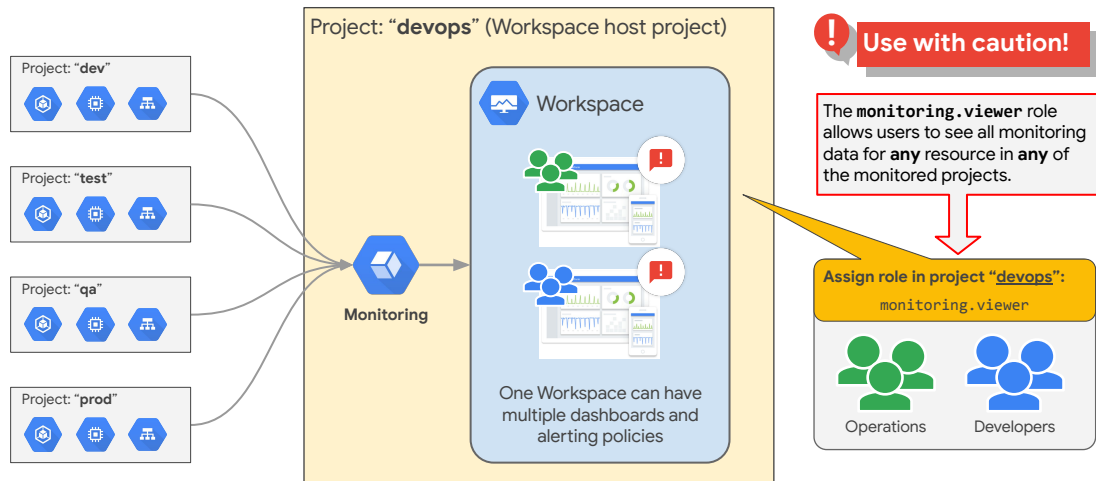


 Google Cloud

### Disadvantages:

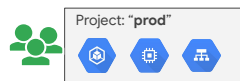
- Anyone with IAM permissions to access Monitoring will be able to see metrics for all environments
- Monitoring in prod is usually done by different teams; this approach wouldn't allow that delineation.

## One Workspace Can Monitor Multiple Projects



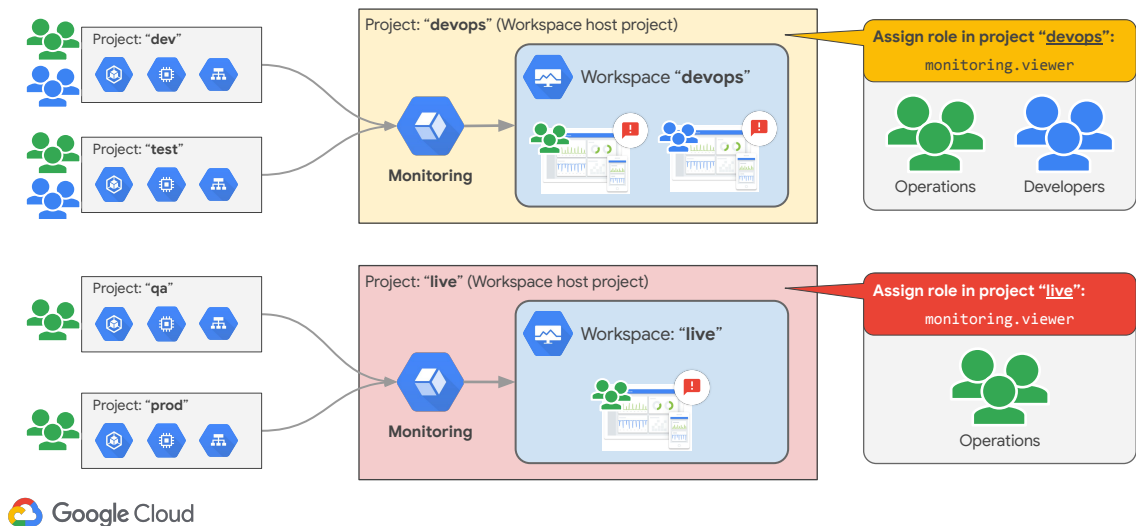
Although the metric data and log entries remain in the individual projects, any user who has been granted the role Monitoring Viewer (roles/monitoring.viewer) will have access to the dashboards and have access to all data by default. This means that a role assigned to one person on one project applies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

## Multiple Workspaces Can Limit Access



To give people different roles per-project, and to better control visibility to data, consider smaller, more selective, monitoring workspaces.

## Multiple Workspaces Can Limit Access



### Strategy B - Prod and Non-Prod monitoring Workspaces

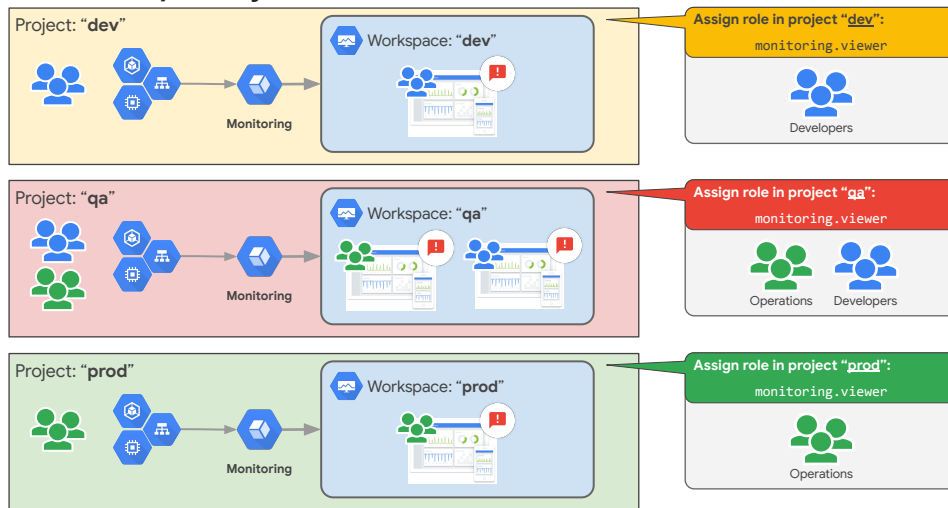
#### Advantages:

- Clear delineations between production and the other environments
- Lowers the maintenance burden of too many monitoring Workspaces (such as in Strategy C)
- Logical boundaries don't have to be production, non-production. This approach of small groups of projects being monitored centrally can apply to many different Google Cloud architectures.

#### Disadvantages:

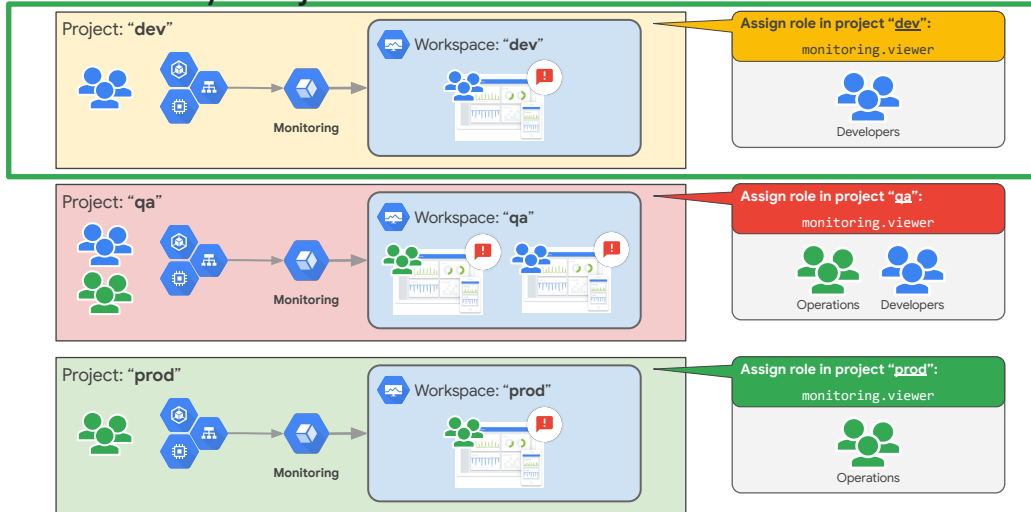
- Have to be careful of the monitored project groupings.
- This approach still provides multi-project access to monitoring data

## Monitor by Project for Maximum Isolation



**Strategy C - Every project is monitored locally, in that project.**

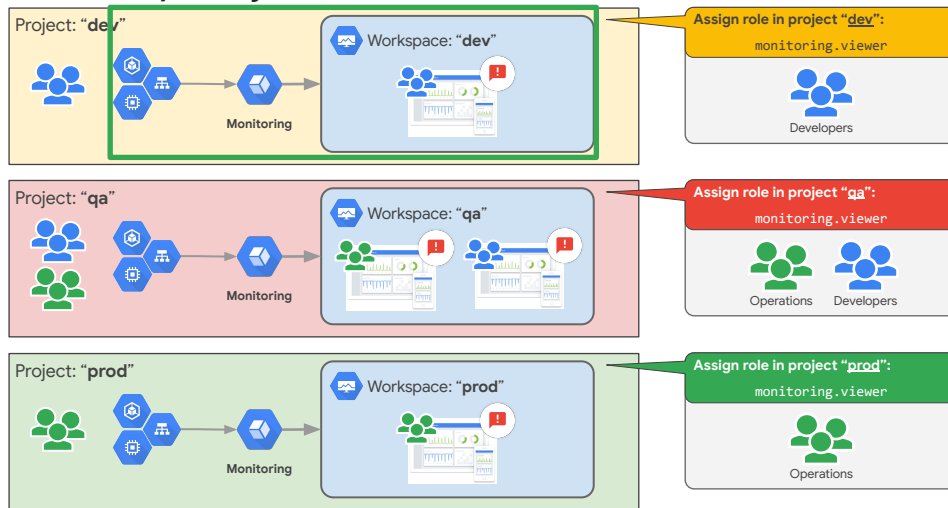
## Monitor by Project for Maximum Isolation



### Advantages:

- Clear and obvious separation for each project. If the project contains dev related resources, it's easy to provide access to the dev personnel.

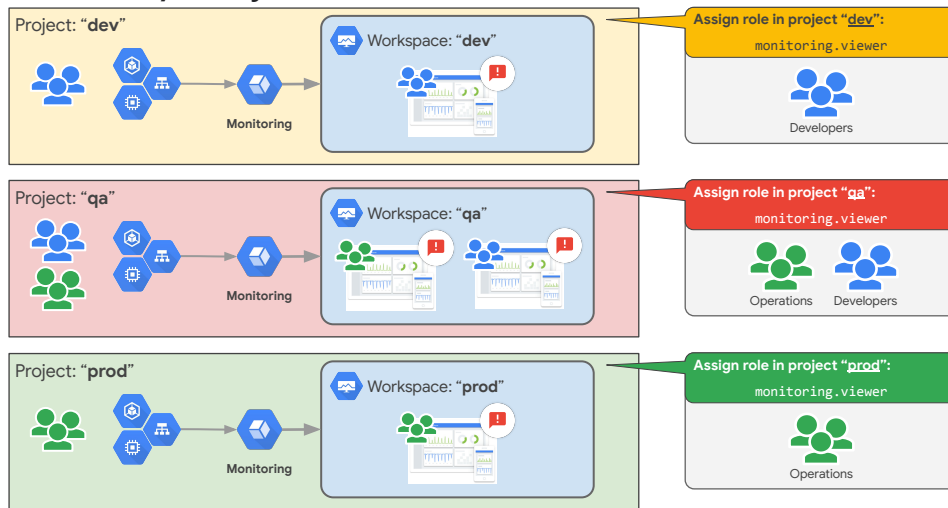
## Monitor by Project for Maximum Isolation



- Project resources and monitoring resources all in the same place.

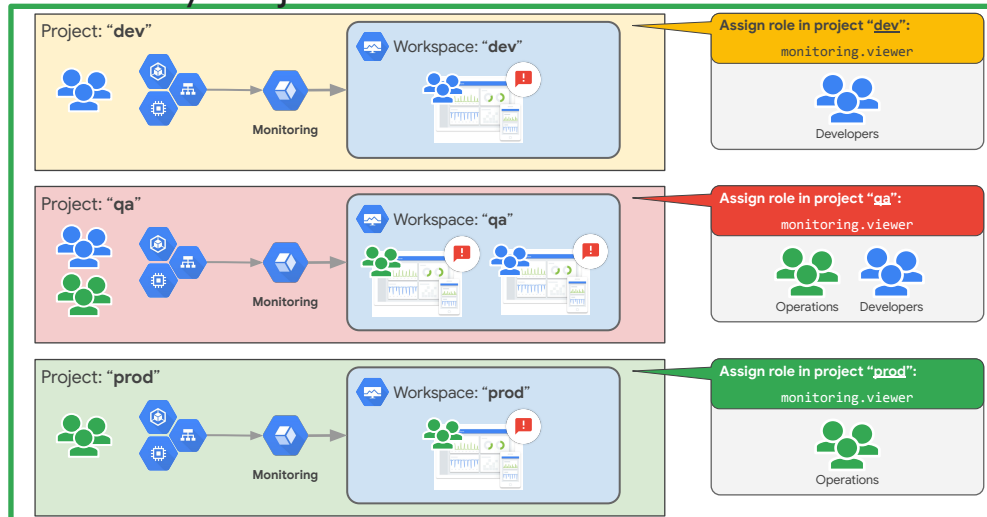


## Monitor by Project for Maximum Isolation



- Easy to automate, since monitoring becomes a standard part of the initial project setup.

## Monitor by Project for Maximum Isolation



### Disadvantages:

- If the application is larger than a single project, then you will be looking at a small slice of a bigger picture, and bringing that full picture into focus might be much harder to do

## IAM Roles Control User Access to Workspaces

- To initially create the Monitoring Workspace, a user will need the [Monitoring Editor](#) or [Monitoring Admin](#) role in the Workspace host project

Role Name	Description
<b>Monitoring Viewer</b>	Gives you read-only access to the Monitoring console and API
<b>Monitoring Editor</b>	Gives you read-write access to the Monitoring console and API, and lets you write monitoring data to a Workspace
<b>Monitoring Admin</b>	Gives you full access to all Monitoring features



There are a number of IAM security roles related to monitoring. The big three are viewer, editor, and admin.

To create the monitoring Workspace initially, a user will need the Monitoring Editor or Admin role in the Workspace's host project.

The Monitoring Viewer can get read-only access to the Monitoring console and API.

The Monitoring Editor has read-write access to the Monitoring console and APIs and can write monitoring data and configurations into the Workspace.

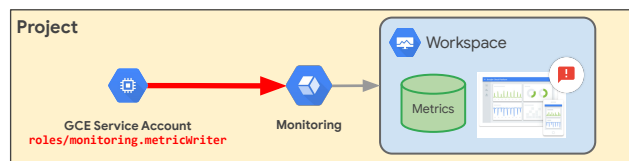
And the Monitoring Admin has full access to, and control over, all monitoring resources.

Past these big three roles, monitoring roles exist to provide and limit access to alert policies, dashboards, notification channels, service monitoring, and uptime checks. Check the [documentation](#) for more information.

## Services May Need Permission to Add Metric Data

- For example, the service account of a GCE instance with the monitoring agent installed
- Grant the service account the Monitoring Metric Writer role in the Workspace host project

Role Name	Description
Monitoring Metric Writer	Permits writing monitoring data to a Workspace. This does not permit read access to the Monitoring console. Typically this permission is used by service accounts.



Another critical security role is *metricWriter*. Services may need permission to add metric data to the monitoring Workspace.

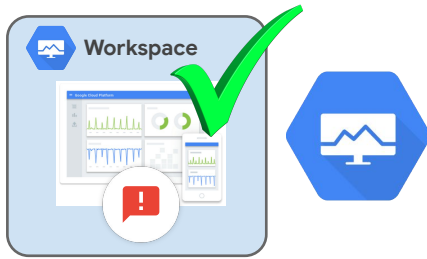
For example, take a Google Compute Engine VM running an agent that needs to stream metrics into the monitoring workspace.

To allow it the write access it needs, grant the VM's service account the Monitoring Metric Writer role in the Workspace host project.

Monitoring Metric Writer permits writing monitoring data to a Workspace. This does not permit read access to the Monitoring console. Typically, this permission is used by service accounts, as in this example.

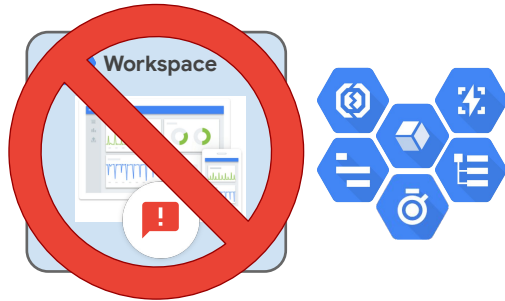
## Remember, Workspaces Only Affect Monitoring

- Only the Monitoring system relies upon Workspaces



 Google Cloud

- The other tools in this course are:
  - Configured on a per-project basis
  - Have their own Cloud IAM roles



Remember, Monitoring Workspaces only affect and control Google Cloud resources related to monitoring.

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

---

# Agenda

Observability Architecture

## Dashboards

- [Understanding Dashboards](#)
- Creating Charts
- Dashboard Construction

Uptime Checks

Not that we've talked about the organization of Monitoring Workspaces, let's create and use some dashboards.

## Dashboards: View and Analyze Metrics



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

Dashboards are assembled from one or more individual charts, laid out a particular way. Here we see a portion of a dashboard displaying two charts: Disk I/O and network traffic.

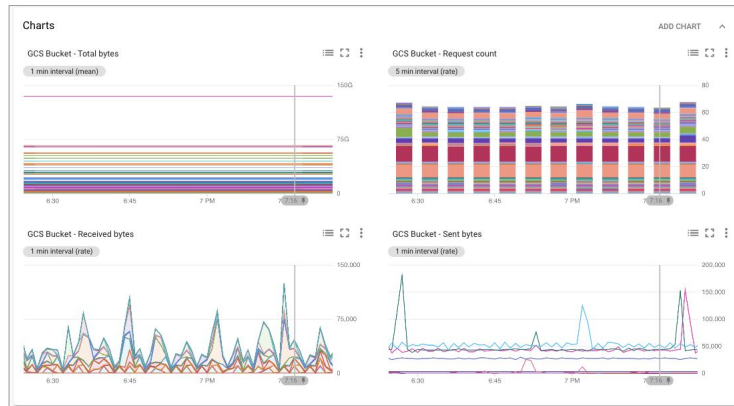
## Predefined Dashboards

☆	App Engine	Google Cloud Platform
☆	BigQuery	Google Cloud Platform
☆	Cloud Pub/Sub	Google Cloud Platform
☆	Cloud SQL	Google Cloud Platform
☆	Cloud Storage	Google Cloud Platform
☆	GCE Disk Snapshots	Google Cloud Platform
☆	GCE Disks	Google Cloud Platform
☆	GCE Firewalls	Google Cloud Platform
☆	GCE Instances	Google Cloud Platform
☆	Google Cloud Load Balancers	Google Cloud Platform
☆	Kubernetes Engine (new)	Google Cloud Platform
☆	VPN	Google Cloud Platform

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 Kubernetes 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. If nothing else, these dashboards form a great monitoring foundation on which you can build.



## Dashboards Broken Into Charts



You assemble Dashboards from individual charts. A chart takes a metric, that's raw signal data, and it breaks it into windows of time (alignment). It does math to each aligned window to reduce it to a single value, and it graphs the resulting points into some chart type. Ultimately, you get a picture that radiates some sort of useful information.

Take a look at the top-right chart. There you see a stacked bar chart displaying request count **rates** collected over **5-minute intervals**.

If you look at the other charts, you'll see they each identify the information displayed, the alignment period, and the math used to reduce each alignment windows into plottable values, rate and mean in these examples.

## Deconstructing a Metric

<code>api/request_count</code> <small>GA</small>	
Request count	
<b>DELTA, INT64, 1</b>	<i>Delta count of API calls, grouped by the API method name and response code. Sampled every 60 seconds. After sampling, data is not visible for up to 120 seconds.</i>
<b>gcs_bucket</b>	
	<b>response_code:</b> The response code of the requests.
	<b>method:</b> The name of the API method called.

Charts get their raw data from metrics. Google has upwards of 1,000 metrics they have pre-created, and you can augment that list by using the API and creating your own where needed.

Here we see an example metric from Google's documentation. This is from the Cloud Storage ([storage.googleapis.com/](https://storage.googleapis.com/)) metrics, so it's related to storage buckets.

At the top-left, we see the actual, and the human-readable versions of the metric name.

Below that is the metric descriptor: DELTA, INT64, 1. Metric descriptors list the metric kind, value type, and unit.

The available metric kinds are:

- **GAUGE:** each data point is an instantaneous measurement of the value. Think of the fuel gauge in your car.
- **DELTA:** which reports the change in value over the time interval. Think a car gauge which showed changes in your fuel mileage.
- **CUMULATIVE:** a value accumulated over time. This might be the total miles on your car.

The value type options are: BOOL, INT64, DOUBLE, STRING, and DISTRIBUTION

The last part of the metric descriptor is the unit in which the value is returned. Units are only valid for the value types INT64, DOUBLE, or DISTRIBUTION, and they are based on *The Unified Code for Units of Measure* standard. Examples include bit, second (s), min, hour (h), etc. In this example, the 1 represents a unitary dimensionless unit, typically used when none of the basic units are appropriate.

Below that, *gcs\_bucket* is the monitored resource, in this case, a Cloud Storage bucket.

The description to the right tells us a bit more about the metric, what it represents, how often it's sampled, and how it's being calculated.

At the bottom are the labels, in this case: *response\_code* and *method*. Labels and can be used for group-by or filtering operations. In this case, for example, we could use the API method to differentiate bucket request types.

---

# Agenda

Observability Architecture

## Dashboards

- Understanding Dashboards
- [Creating Charts](#)
- Dashboard Construction

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.

---

## A Note on Security

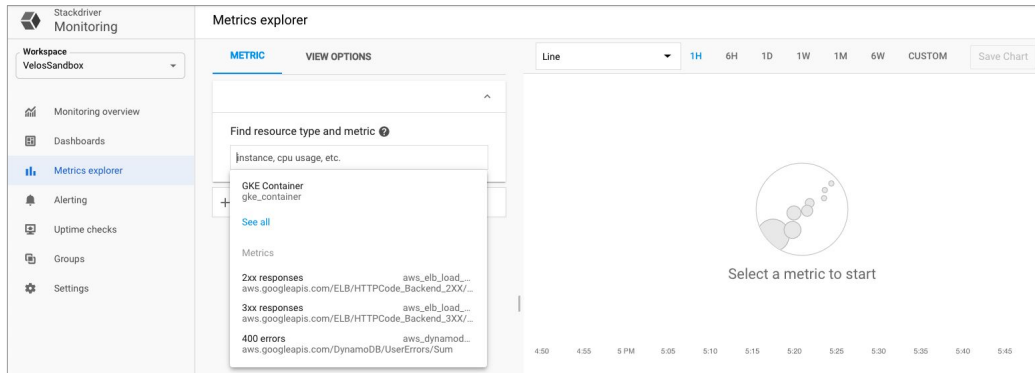
- roles/**monitoring.dashboardEditor**: can edit dashboard settings
  - roles/**monitoring.dashboardViewer**: can view dashboard settings
  - roles/**monitoring.editor**: can create dashboards and add charts
  - roles/**monitoring.viewer**: can view charts and dashboards
- 
- Note, the monitoring editor and viewer can do a lot more, we are just focusing on the dashboards and charts at this point

A note on security IAM roles related to charts and dashboards:

- roles/**monitoring.dashboardEditor**: can be used to edit dashboard settings
- roles/**monitoring.dashboardViewer**: can view dashboard settings
- roles/**monitoring.editor**: can create dashboards and add charts
- roles/**monitoring.viewer**: can view charts and dashboards

The monitoring editor and viewer roles can do a lot more what's stated on this slide, in terms of monitoring related activities, but we are focusing just on the dashboard and chart abilities at this point.

# Start with the Metrics Explorer



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 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 dashboard.

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

- A configuration region, where you pick the metric and its options.
- And the chart displaying the selected metric.

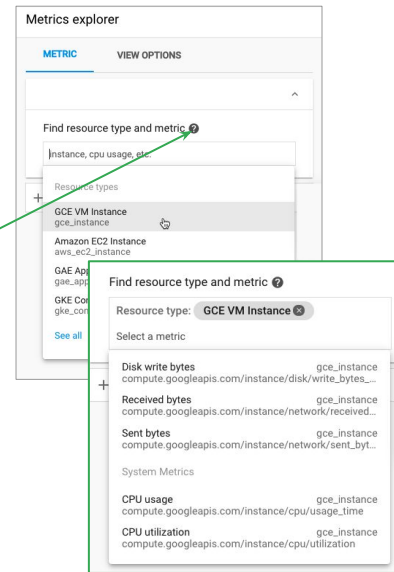
## Choose a Metric

- Start with a Resource
  - Metrics automatically filtered
- Next, pick the desired metric for that resource
- Can also use Direct Filter Mode (Hit ? icon)

Resource type, metric, and filter ?

```
metric.type="agent.googleapis.com/cpu/utilization"  
resource.type="gce_instance"
```

[Standard mode](#)

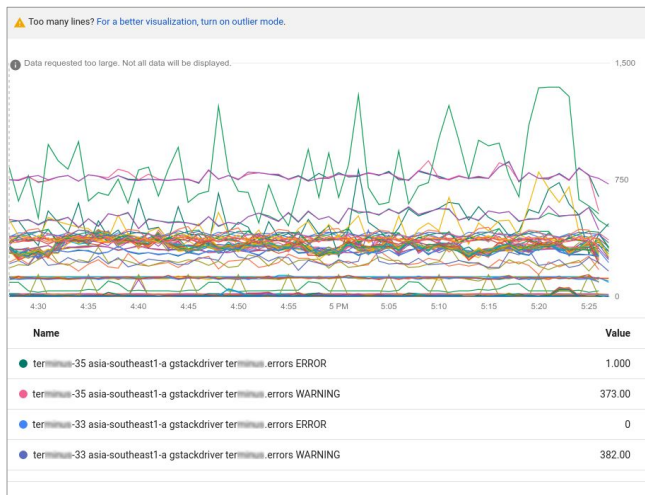


You define a chart by specifying both what data should display and how the chart should display it.

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*).

If you prefer, you can also use Direct Filter Mode by hitting the question mark icon as seen in the diagram. When Direct filter mode is enabled, the Find resource type and metric option is replaced with an editable text box labeled Resource type, metric, and filter.

## Metric Example



Find resource type and metric ⓘ

Resource type:

Metric:

Filter ⓘ

+ Add a filter

Group By ⓘ

+ Add a label

Aggregator ⓘ

SHOW ADVANCED OPTIONS

Let's look at an example.

First, you set the resource type to GCE VM Instance. So we're looking at metrics related to Compute Engine virtual machines.

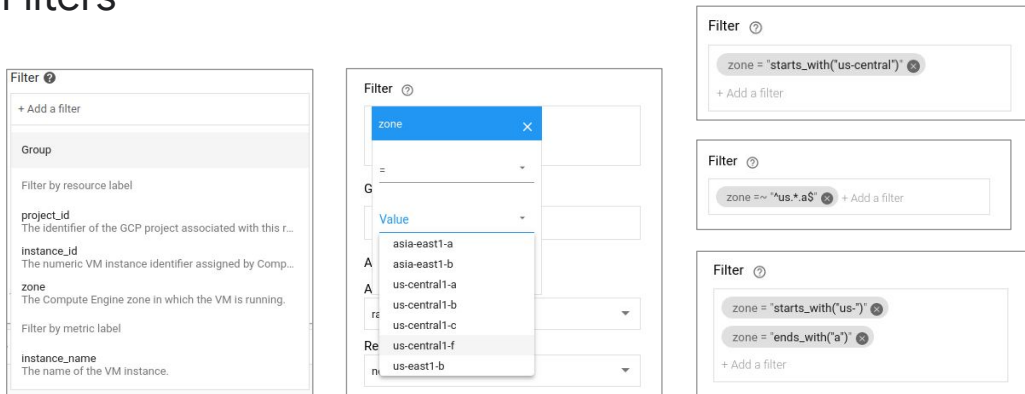
Next, you pick the Logging Agent Log Entry Count metric. If you check the [agent's metric documentation](#), you'll see that this is a cumulative count of log entries written by the logging agent, sampled every minute.

You'd also see that it has a single response code label, and you can see that reflected in the diagram by the ERROR and WARNING entries.

The example is a good start, but there are so many lines being displayed. A single chart can plot up to 300 individual lines, but it's tough to see detail through that much clutter. How about we filter some of them out, and group others.



## Filters



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 containing 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.

In this example, we will filter by zone. The zone can then be compared to a direct value, like "`=asia-east1-a`," or by using the "`=~`" operator, to any valid regular expression, as you see in the right-hand examples. If you'd like to see the fully supported filter syntax, please check the [documentation](#).

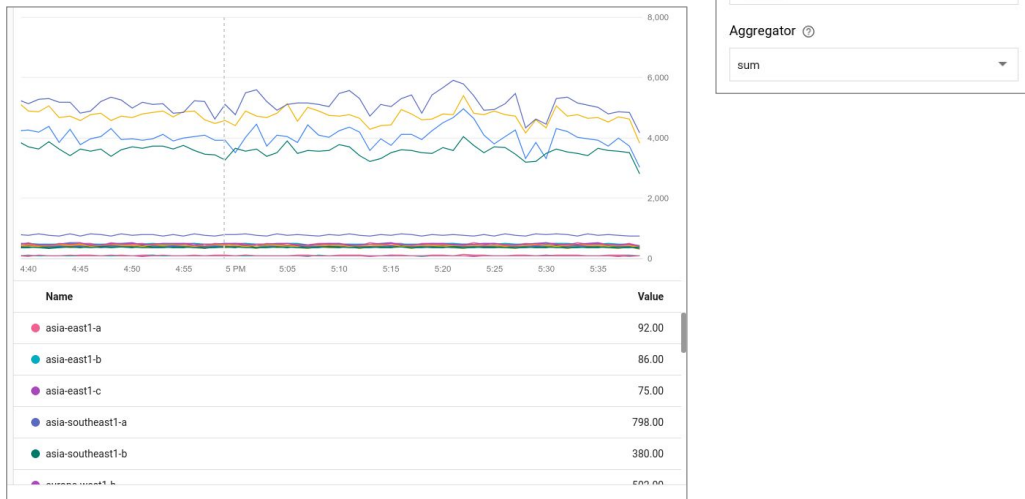
## Better



zone=~"asia-\*."

That's a little better. Now, instead of all the VM log counts, we've restricted our chat to just displaying those VMs in the Asia based regions.

## Grouping



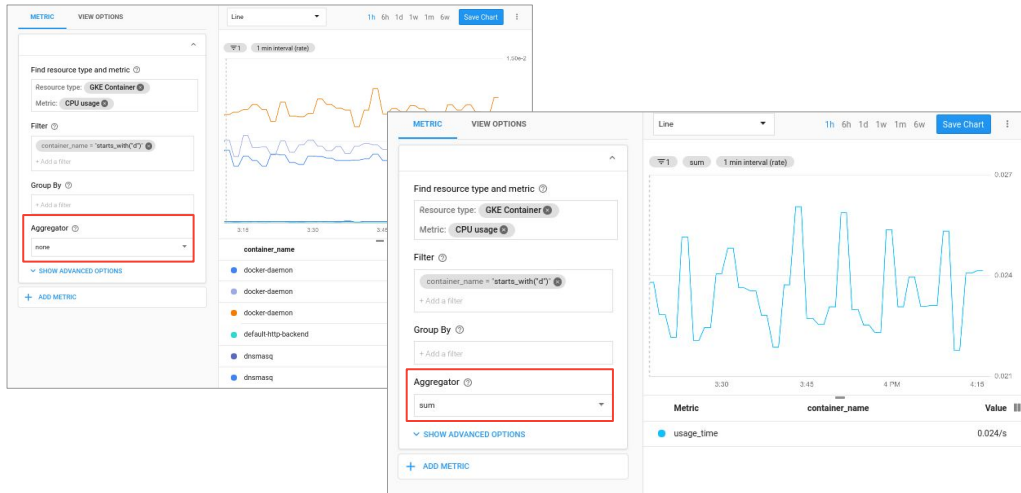
## Grouping

Like filtering, grouping is a way to reduce the amount of data you are manipulating. Filtering works by excluding some time series, and grouping works by identifying sets of time series that all meet some criterion, and then combining, or *aggregating*, the members of the sets together.

The **Group By** option lets you group time series by resource and metric label, and then combine the data within those groups. This creates a single new time series for each combination of group-by values, and the new time series represents all the members of the group.

To continue our agent log entry count example, let's take the VMs, which have been filtered to the Asia regions, and group them by zone. You can see the results on the slide.

# Aggregation



## Aggregation

The **Aggregator** lets you combine time series by using common math functions. The result is fewer lines on the chart displaying the metric, which can improve the chart's performance.

Click in the **Aggregator** field to see a list of the available aggregation functions, or *reducers*, that can be used to combine the time series.

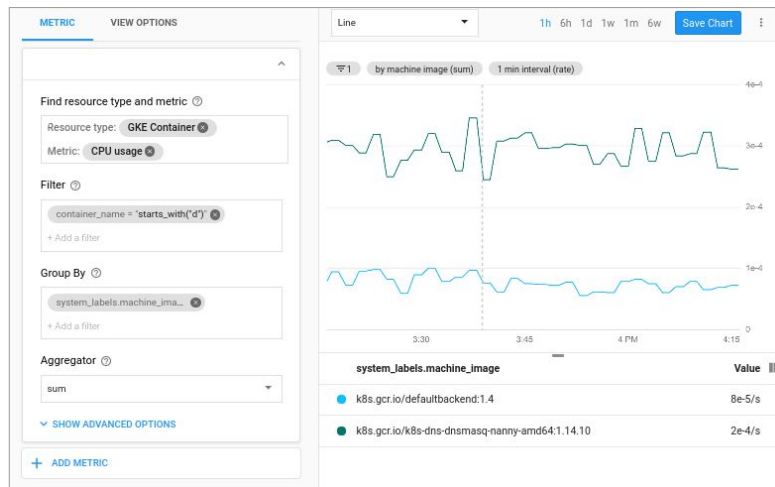
The available reducing functions depend on the type of values the metric captures, but they commonly include choices like mean, max or min, standard deviation, assorted percentile values, and so forth.

For more information about these dependencies, see the [Metrics, time series, and resources](#) documentation.

We're using a slightly different example on this slide. In the first screenshot, we see CPU usage for all the GKE containers with a container name that starts with a "d." Notice, the Aggregator is set to none.

In the second example, the aggregator function is set to **sum**. Take a moment in time, grab all the aligned data points for that moment, and add them together. See how all the lines have been combined (or aggregated)?

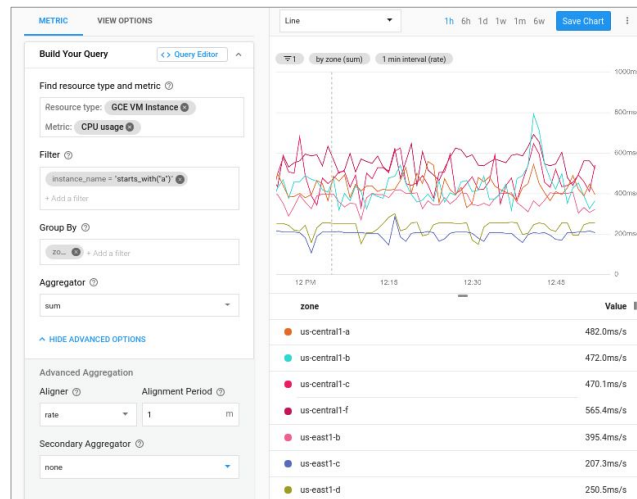
## Aggregation, filtered, and grouped



Here, we have an extension of that same example.

- The resource is set to GKE Container.
- CPU usage is the chosen metric.
- The time series were all filtered to containers with a name starting with "d."
- Then the containers were grouped by machine image.
- And lastly, the CPU usages for the containers in each of the two resulting groups were summed.

## Aligner (break data into regular time buckets)



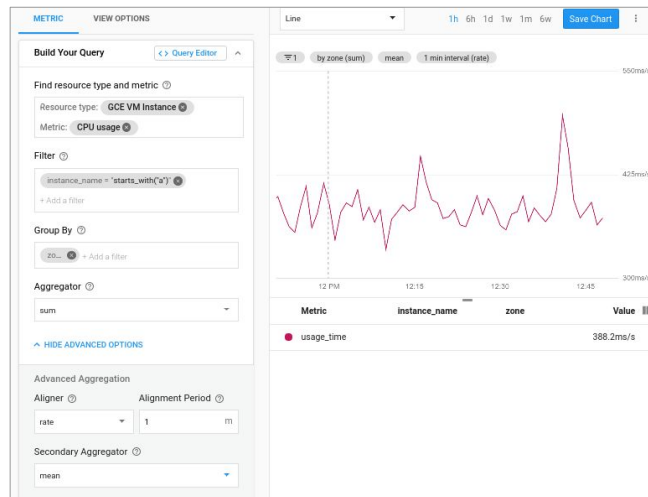
In the advanced options for the Metrics Explorer, you can directly access the alignment being used by the chart. A time series is a set of data points in temporal order. To align a time series is to break the data points into regular buckets of time, the *alignment period*. Multiple time series must 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, **Aligner** and **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. The data in each period is summarized so that a single value represents that period. The default alignment period, and 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 aligner is a function that determines how to summarize the data in each alignment period. Aligners include the sum, the mean, and so forth. Valid aligner choices depend on the kind and type of metric data a time series stores.

## Secondary Aggregation



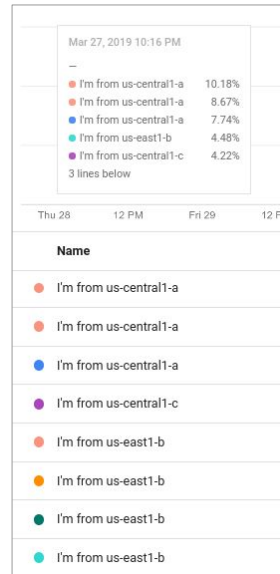
When you have multiple time series that already represent "group by" aggregations, then the **Secondary Aggregator** can do exactly that, aggregate the groups a second time.

## Legend template

Legend Template ?

+ Add a filter

I'm from \${resource.labels.zone}



The Legend Template field lets you customize a description for the time series on your chart. By default, these descriptions 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. You can use this field to build a template for the descriptions.

The Legend Template field accepts both plain text and label patterns. The syntax can vary in the patterns but the dynamic replacement sections use the `${ variable }` notation.

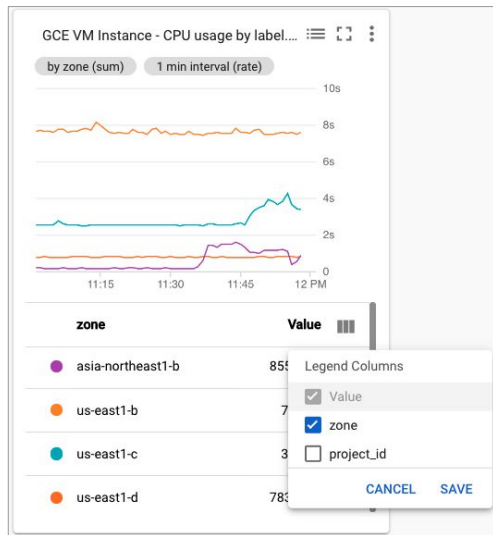
If you know the variable names, you can manually type them into the template field. You can also select variables for the available labels by using the **+ Add a filter** widget in the field. This approach ensures that the variable syntax is correct.

In this example, you see the mix of static text, "I'm from", and the variable, `${resource.labels.zone}`. You can see the resulting output in the chart legend.



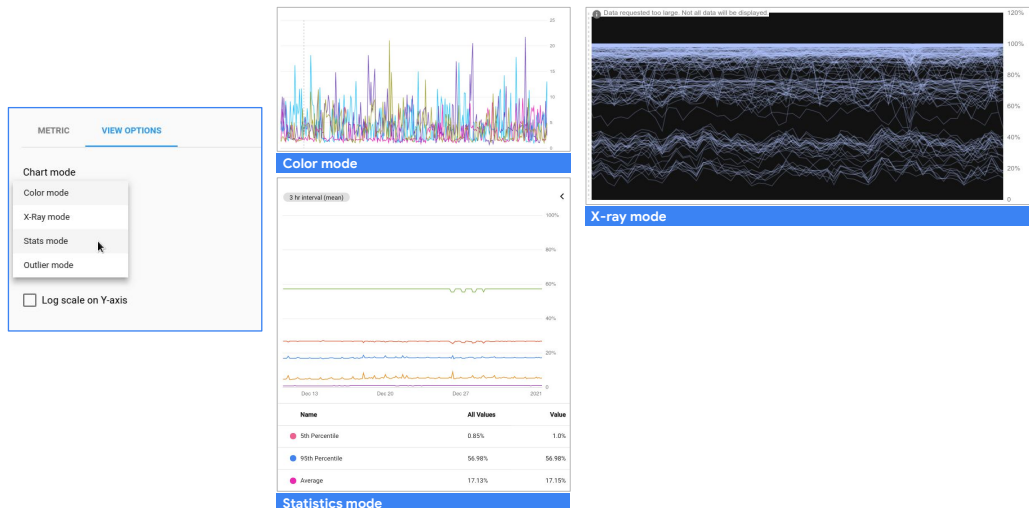
## Legend Configuration

- Select and sort the rows



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

## View options, Chart mode



The **View options** tab in the chart-definition interface lets you specify how data is presented in the chart. One of the options is Chart Mode.

Charts provide multiple viewing modes, though not all of them may be available for every chart. For a given chart, the possible modes are:

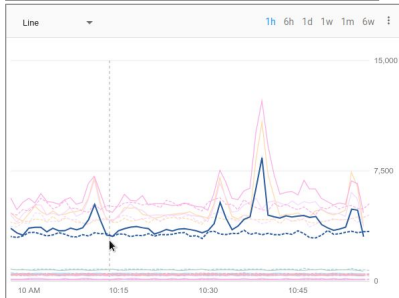
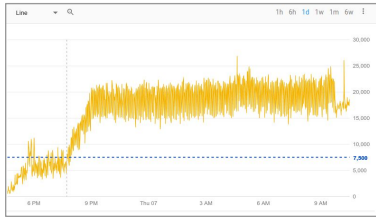
- Color
- Statistics
- X-Ray
- Outlier

Color mode is the default, and it is the display mode you most frequently encounter.

Statistics mode displays common statistical measures for the data in a chart. When you select statistics mode, the chart displays a banner that shows the maximum and minimum values as well as a measure of similarity.

X-ray mode displays all the graph lines with a translucent gray color. Each line is faint, and where lines overlap or cross, the points appear brighter. Overlapping lines create bands of brightness, which indicate the normal behavior within a metrics group.

## Outlier mode



Specify:

- A visual **Threshold** line
- Superimpose a **Compare to Past** dotted line
- **Log scale on Y-axis** to help view tight clusters

METRIC

VIEW OPTIONS

Chart mode ?

Outlier mode

Show me:

Top

3

time series

ranked by:

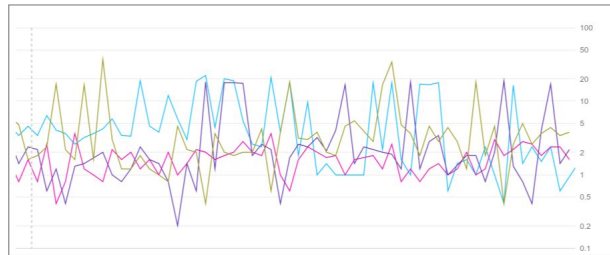
Mean

of each time series

☐ Threshold ?

☐ Compare to Past ?

☐ Log scale on Y-axis ?



Displaying a large number of lines on a single chart can obscure the interesting ones. We've addressed this with filtering and grouping, but another option might be Outlier mode. Outlier mode shows you the anomalous lines, the outliers if you will, rather than the highly representative ones.

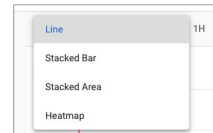
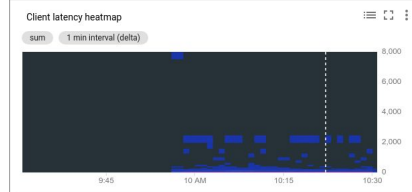
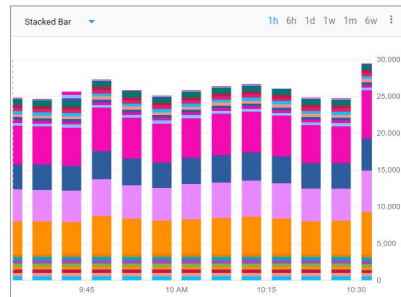
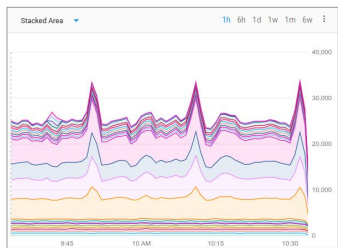
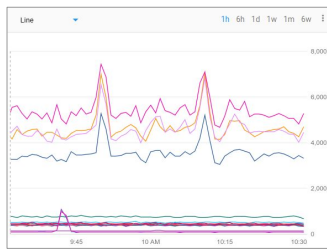
In Outlier mode, you can select the number of time series to show, whether you want extreme high or low values, and the method by which the time series are ranked.

Additionally, you can add a visual threshold line at a particular value.

You can use **Compare to Past** option to select a time range from the past, and then superimpose the past series as a dotted line over the current data on a line chart.

You can also rescale the chart's Y-values logarithmically. This rescaling is useful when values cluster tightly within a small range.

## Chart Type



In addition to the options on the **View options** tab, you can also specify how the chart graphs the data. The default style is a line chart, but stacked bar, stacked area, and heatmap are also available.

---

# Agenda

Observability Architecture


## Dashboards

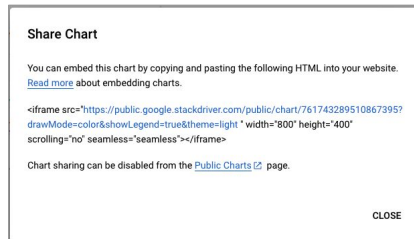
- Understanding Dashboards
- Creating Charts
- **Dashboard Construction**

Uptime Checks

Speaking of dashboards, let's spend a moment and talk about how they are constructed.

## Sharing Charts and Dashboards

- Share from Metrics Explorer by URL
- Use the  to share a chart from a Dashboard
  - *Shared through iframe*

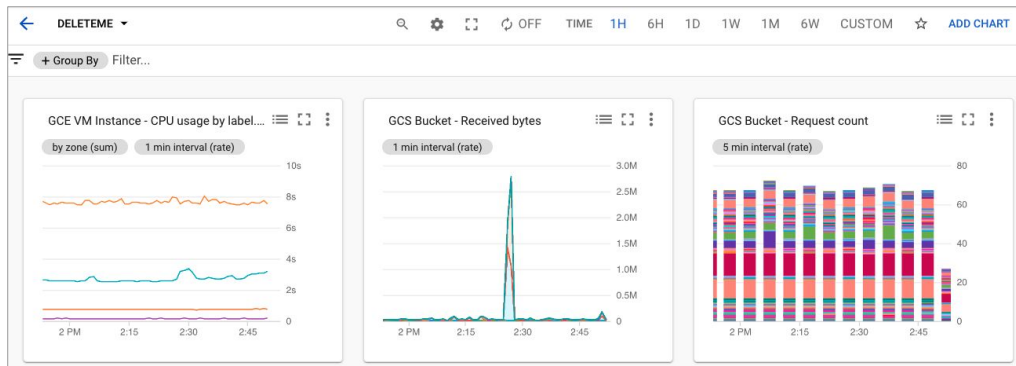


It's easy to share charts from the Metrics Explorer.

For quick, one-off charts, simply share them by URL. The URL contains all the selected options from the current chart and dynamically recreates those selections whenever you visit it.

For longer-term usage, it might be easier to place the chart on a dashboard or on some external page using an iframe. Dashboards also add the ability to view multiple charts from a single location.

## Putting Together the Dashboard

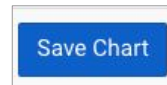


In addition to predefined dashboards, Cloud Monitoring lets you define custom dashboards. With custom dashboards, you have complete control over the charts that are displayed and the configuration. You can create these dashboards through the [Google Cloud Console](#) or by using the [Dashboard endpoint](#) in the Cloud Monitoring API.

---

## Time to Build the Dashboard

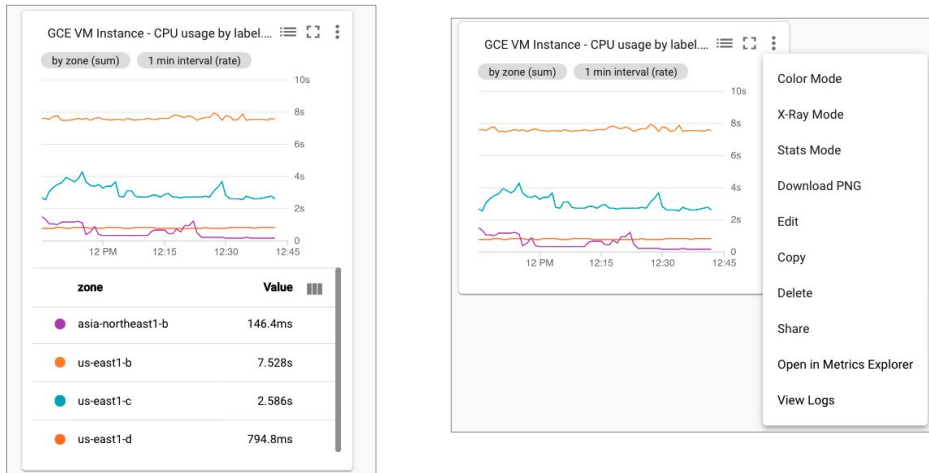
- Create the Dashboard and Save Chart to it



Dashboards can be created from the Monitoring | Dashboards page by clicking "Create Dashboard," or directly out of the Metrics Explorer by clicking "Save Chart."

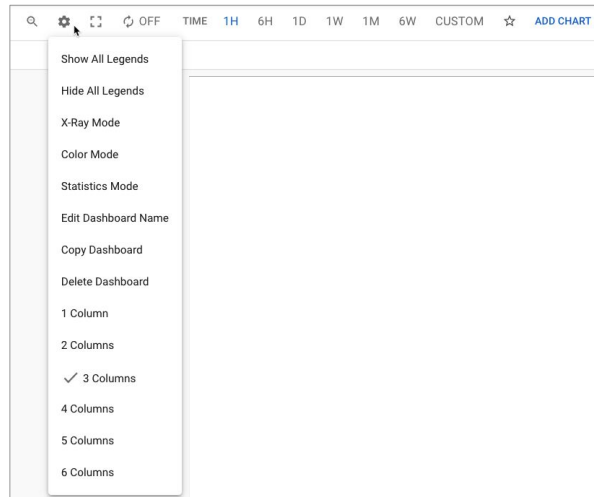


## Dashboarded Charts Can Be Tweaked



Once a chart is included in a dashboard, it's easy to tweak its mode, edit or view it in Metrics Explorer, view its Logs, and more.

## Change Default Configs for All Charts



Instead of editing the properties of the charts one at a time, some features, such as chart mode, can be changed for all the charts in a dashboard.

Dashboards can also be renamed, copied, and deleted.

The number of columns displayed in a chart may also be selected.



## Agenda

Observability Architecture

Dashboards

Uptime Checks

Another monitoring component we discussed briefly in an earlier module of this course is Uptime Checks.

## Error Budgets

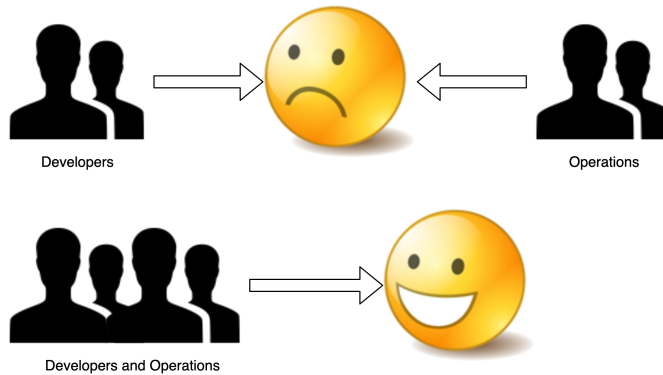
- Zero error tolerance means zero updates
- Error Budgets
  - Quantify the organization's tolerance for errors
  - Operate like a currency or allowance
  - Allow Devs to take calculated risks
  - Excuse the occasional error
  - Replenish over time



Error budgets quantify the organization's tolerance for errors. Instead of expecting a mythical zero-error environment, the organization allows a certain number of errors to occur so that the applications may be updated and properly maintained. Systems with zero errors also have zero updates, which leads to out of date applications, lack of innovation, and increased security risks. On the other hand, an organization that creates error budgets manages their downtime and spends it like currency.

Think of the budget as a rainwater barrel. When problems occur they reduce the available error budget. The budget (like the barrel) is replenished gradually over time. Application updates may incur an outage, which will in turn consume some of the error budget. If the application has sufficient error budget, the developers may take a risk and update the application. Conversely, they may decide to delay updates to applications that have little remaining error budget until the error budget replenishes.

## Error Budgets and Unifying Goals



Developers make changes to gain features.

Operations folk like stability because it's easier to keep things working.

Error budgets provide wiggle room. Developers test new features and the error budget provides them room for failure. Ops no longer has to choose uptime over improvements. Everyone wins.

## Uptime Checks Check Public Service Availability

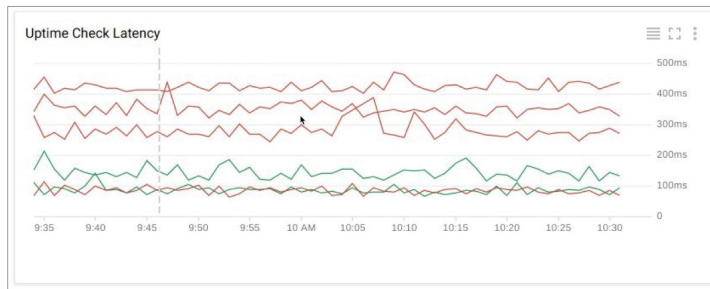
CHECKS	VIRGINIA	OREGON	IOWA	BELGIUM	SINGAPORE	SAO PAULO	POLICIES
Instance 1	✓	✓	✓	✓	✓	✓	
Instance 2	✓	✓	✓	✓	✓	✓	
Instance 3	✓	✓	✓	✓	✓	✓	

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 to be checked can be an 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 make sure our externally facing services are running and that we aren't burning our error budgets unnecessarily.

## Uptime Check Example



Uptime ⓘ	Outages ⓘ
100.000%	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

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.

---

## Learning From Mistakes



- Breakage is inevitable
- Find it, fix it, learn from it
- Uptime checks are about finding it

As discussed in the SLI/SLO part of our course, breakage is inevitable. The trick is to find the issue, fix it, and learn from the process. Uptime checks are one of the ways we find issues.



---

## What makes a good Uptime Check?

- Protocol, host, and port are appropriate
- Response checked for specific content
- Check frequency proportional to criticality

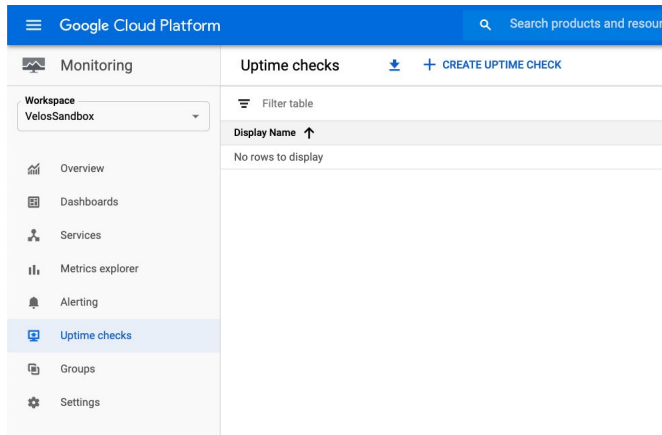


Good uptime checks use the appropriate protocol, host, and port.

They check the response for specific content, so a blank page or error page won't be registered as a successful response.

And they check in proportion to the criticality of the service being checked.

# Creating an Uptime Check



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

## Creating an Uptime Check

**New uptime check**

Title \*

Check Type: HTTP

Resource Type: URL

Hostname \*

Path

Check every: 1 minute

☒ Log check failures

**General**

Host Header

Port: 80

Response Content Match Type: Contains

Response Content

**Locations**

- ☒ Global
- ☒ Europe
- ☒ Asia Pacific
- ☒ South America
- ☒ United States

**Custom Headers**

☐ Encrypt custom headers

Header	Value (optional)
--------	------------------

+ ADD CUSTOM HEADER

**Healthcheck**

Timeout

**Authentication**

Username Password

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.

## Lab Intro

Monitoring and Dashboarding  
Multiple Projects from a  
Single Workspace



Google Cloud Monitoring empowers users with the ability to monitor 1..100 projects from a single monitoring Workspace. In this lab, you will start with three Cloud projects, two with monitorable resources, and the third you will configure as a "single pane of glass" monitoring Workspace. You will attach the two resource projects to the monitoring Workspace, build uptime checks, and build a centralized dashboard