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

Agenda

Observability Architecture

Dashboards

- Understanding Dashboards
- Creating Charts with Metrics Explorer
- Dashboard Construction
- Monitoring Query Language (MQL)

Uptime Checks

Google Cloud

Monitoring is all about keeping track of exactly what's happening with the resources we've spun up 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 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

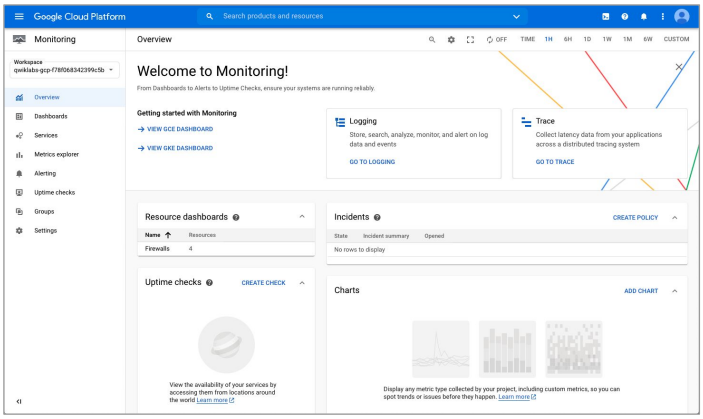
Uptime Checks

Google Cloud

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

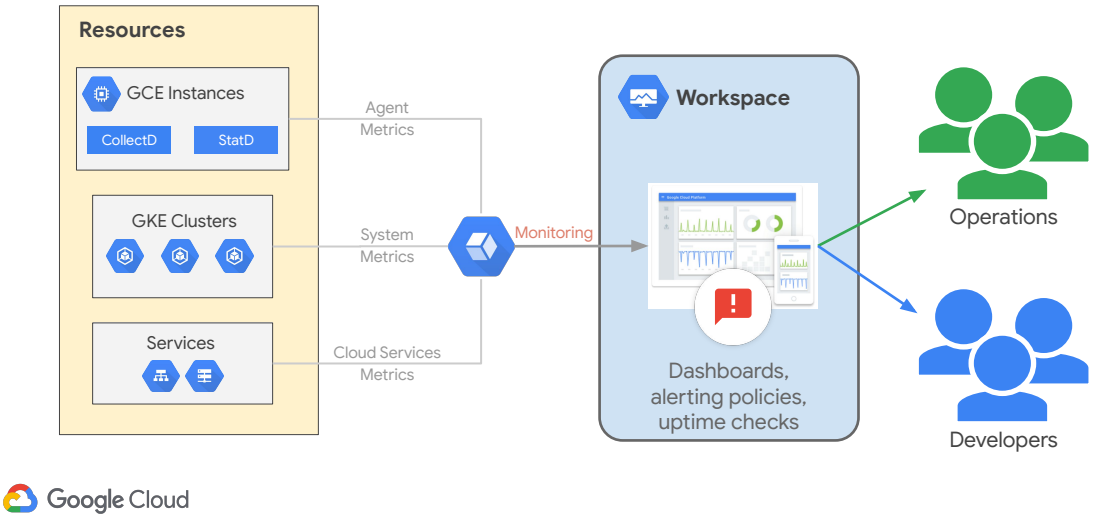
Google Cloud Monitoring uses Workspaces to organize monitoring information. A Workspace is a tool for monitoring resources contained in one or more Google Cloud projects.

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 project, and in up to 100 other projects, monitoring workspaces offer excellent cross-project visibility.

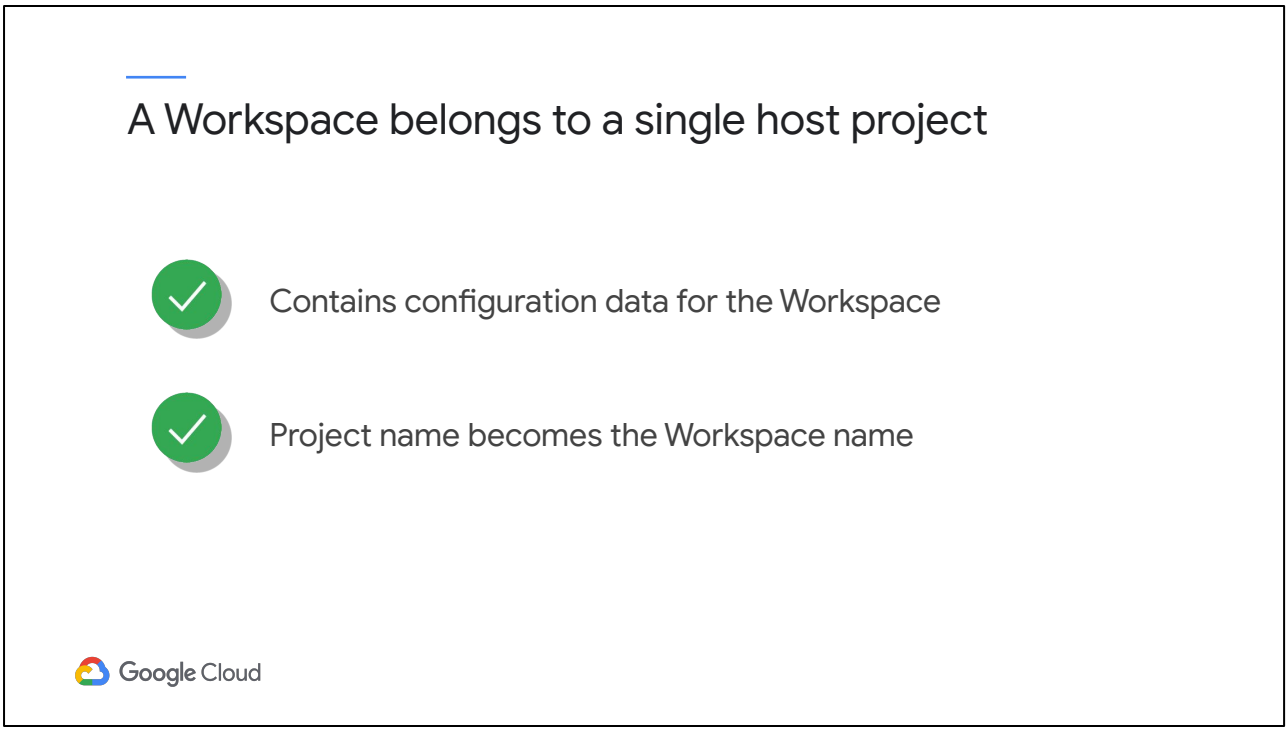
The monitored resources may be part of Google Cloud or AWS.

Organize your monitoring efforts with Workspaces



Google Cloud

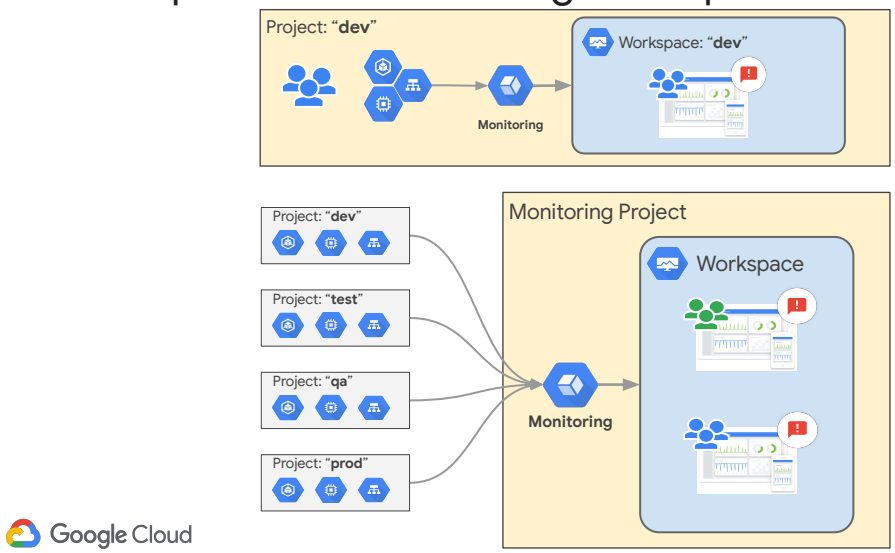
Monitoring Workspaces help organize your monitoring efforts. They serve as central, secured access hubs for monitoring information, dashboards, alerting policies, and uptime checks. This information is made available, IAM permitting, to both operations and developer personnel.



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.

Two options for monitoring Workspace architectures

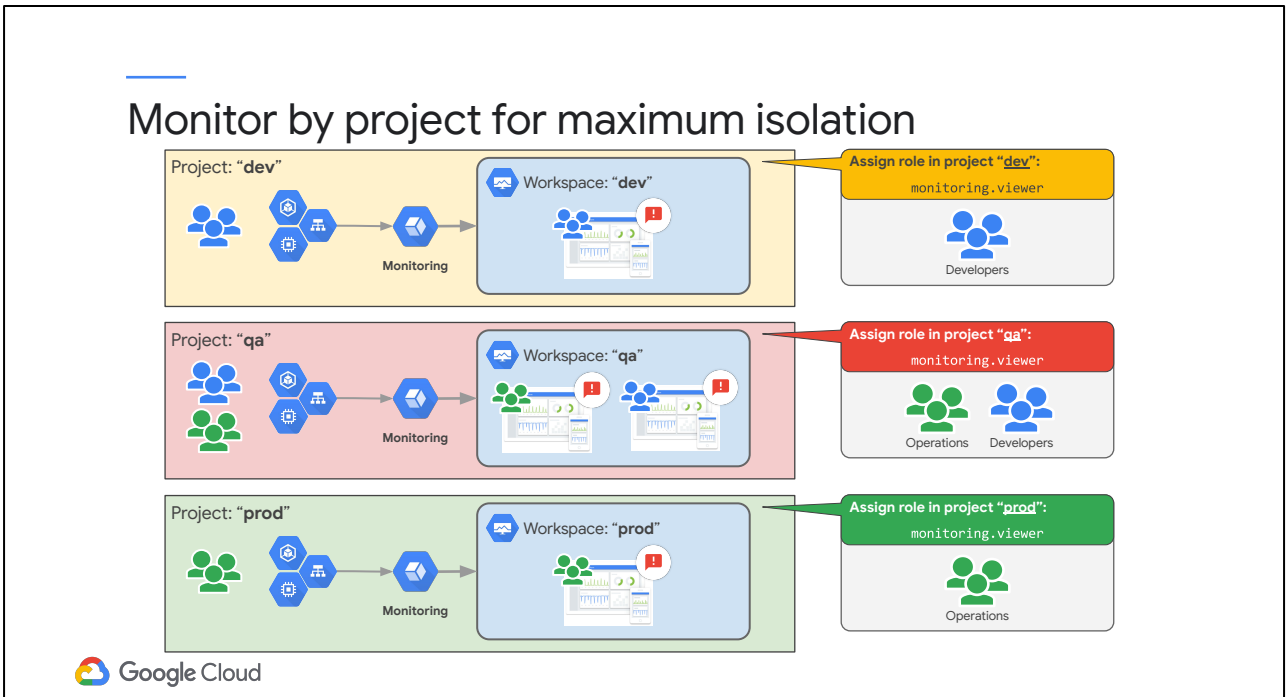


There are only two options on how a project with resources is monitored. Either the project contains the resources and all the monitoring, or a project contains resources and it's monitored by an externally configured "monitoring" project.

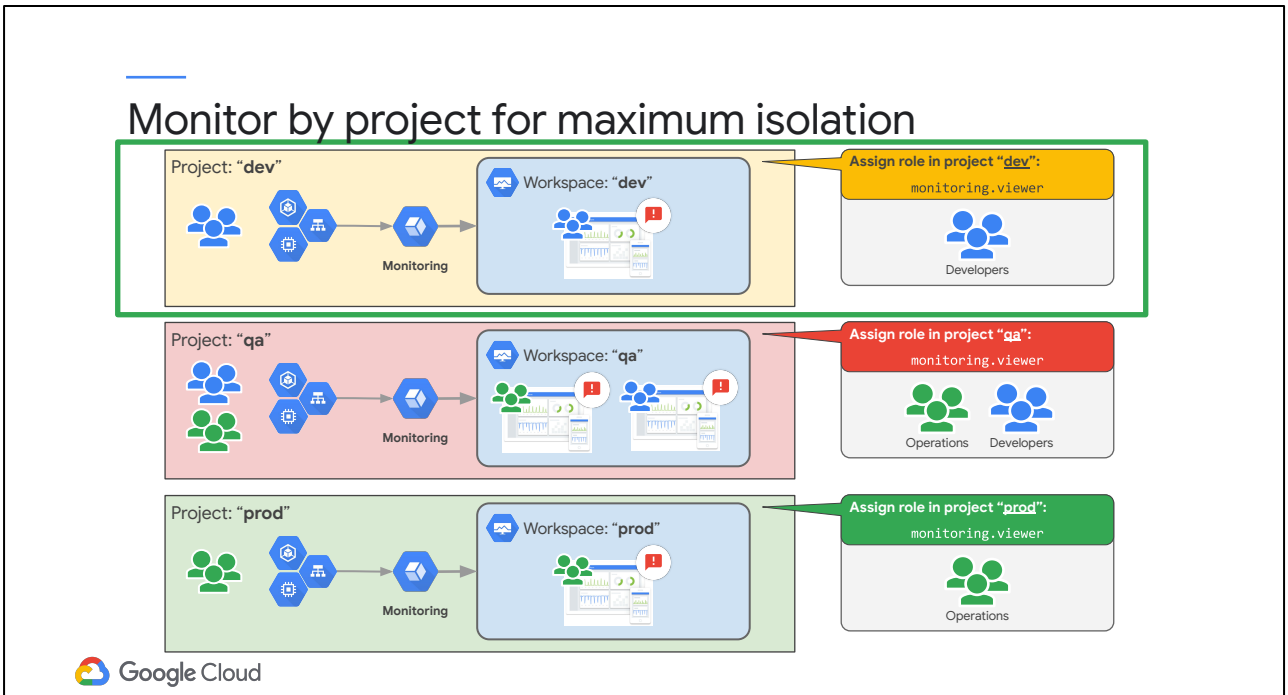
Lecture Notes:

Future changes in monitoring:
(1) Workspace not required for the same project (Option 1)
(2) Workspace will ONLY be required when you are approaching (Option 2)
(3) A project can be monitored by multiple monitoring workspaces simultaneously. Not available today.
(4) #3 above will allow for giving differential access to different users on monitoring workspace (workaround). User does not need access to the actual projects where resources are running to view the workspace for monitoring

Best Practices:
Monitoring project should ONLY contain monitoring workspace and nothing else.
Note: Host project here is different from Shared VPC host project.
One project can ONLY have 1 workspace. Workspace name = project name



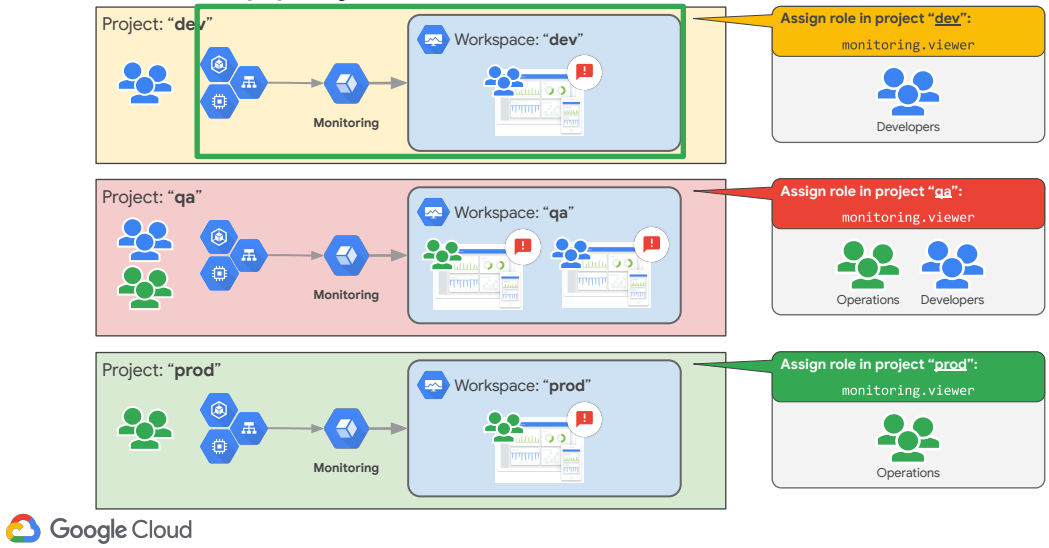
Strategy C: Every project is monitored locally, in that project.



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.

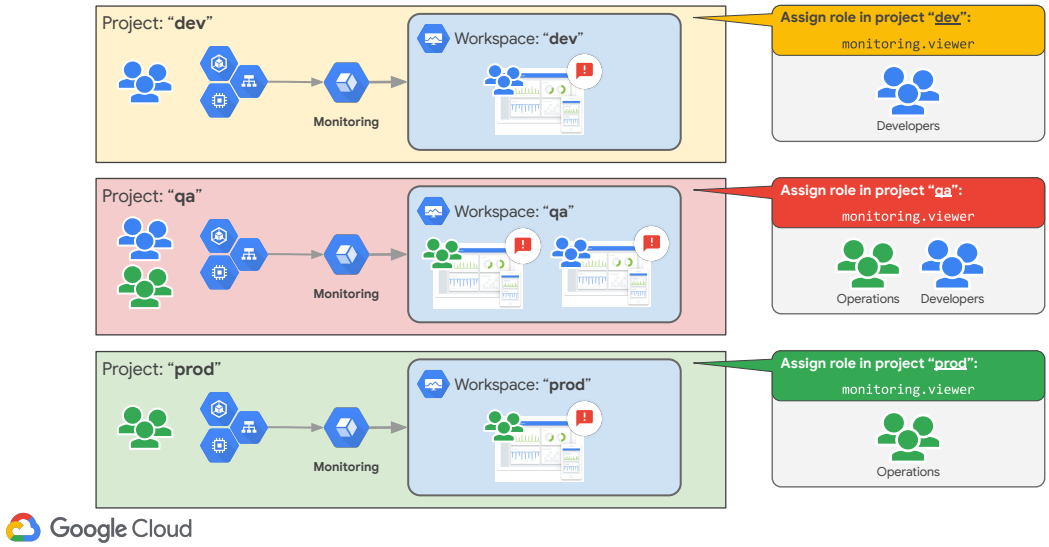
Monitor by project for maximum isolation



Google Cloud

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

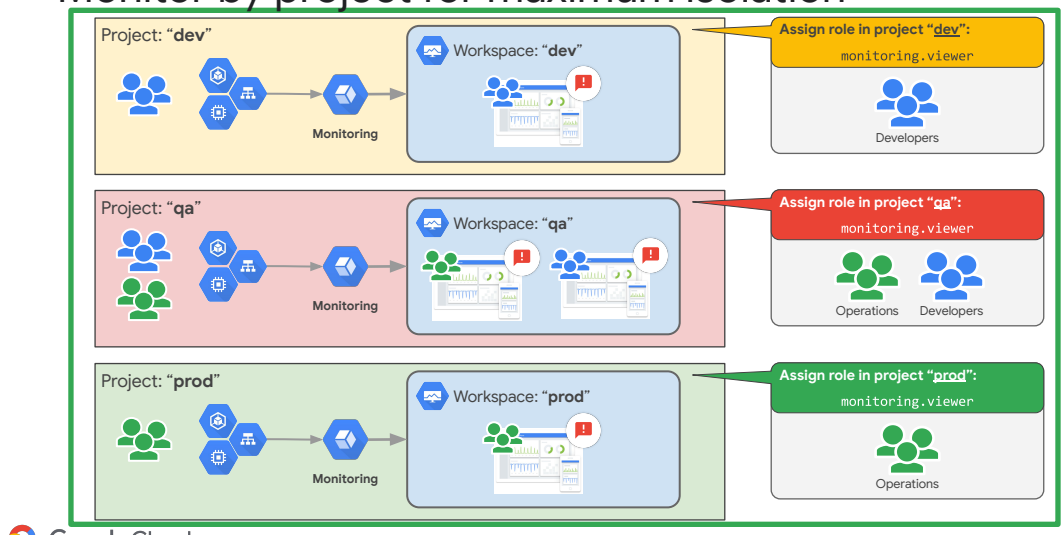
Monitor by project for maximum isolation



Google Cloud

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

Monitor by project for maximum isolation



Google Cloud

Disadvantages:

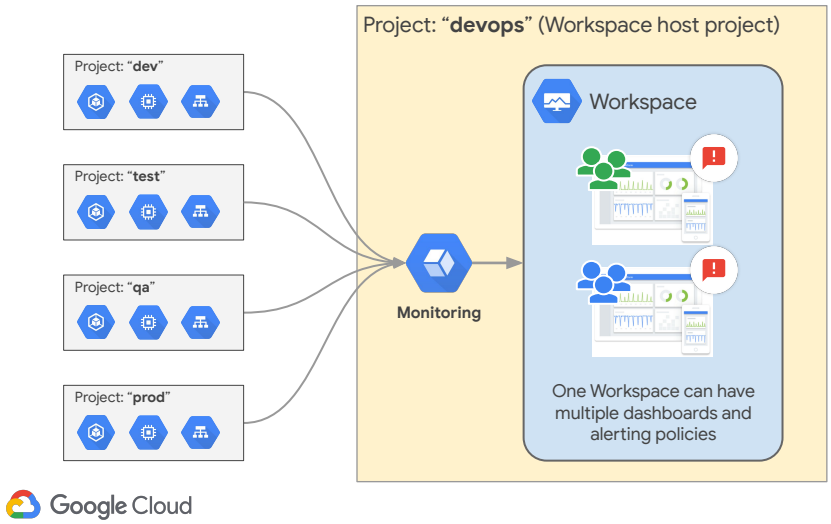
- If the application is larger than a single project, then you will be looking at a small slice of a bigger picture, and bringing that full picture into focus might be much harder to do.

One Workspace can monitor multiple projects

Google Cloud

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

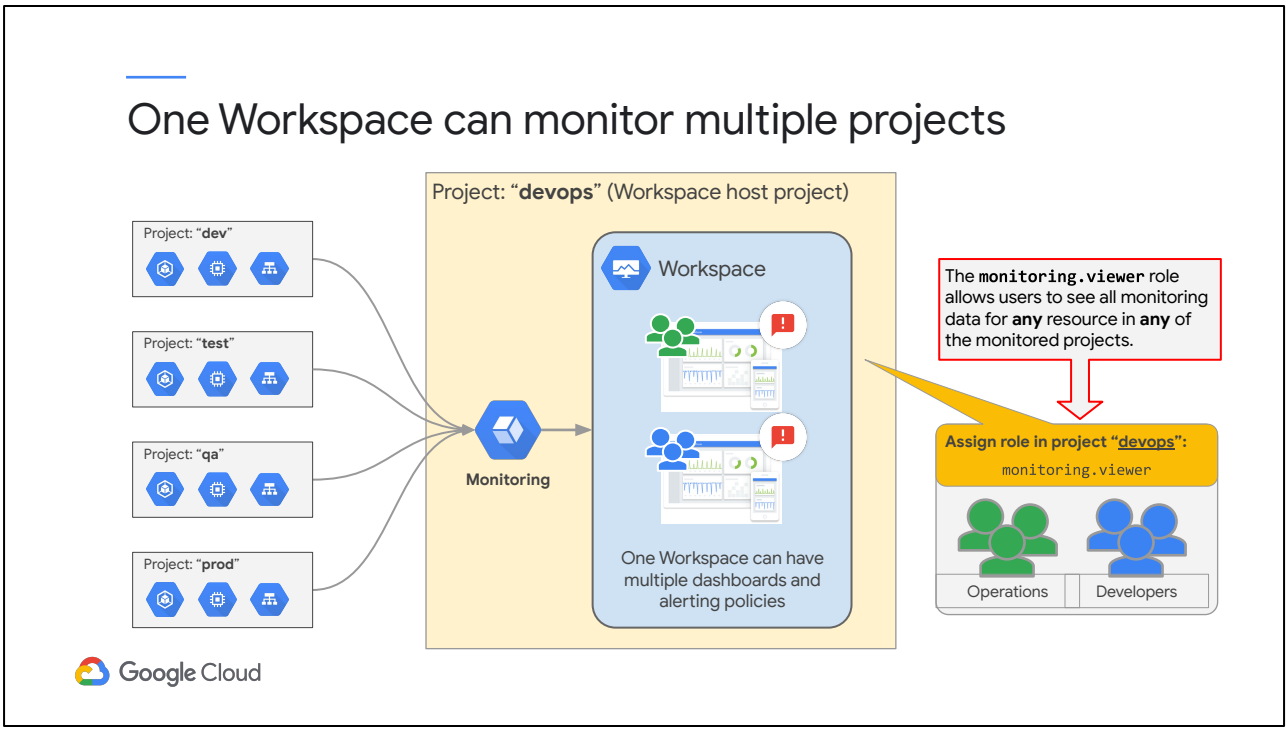


Google Cloud

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

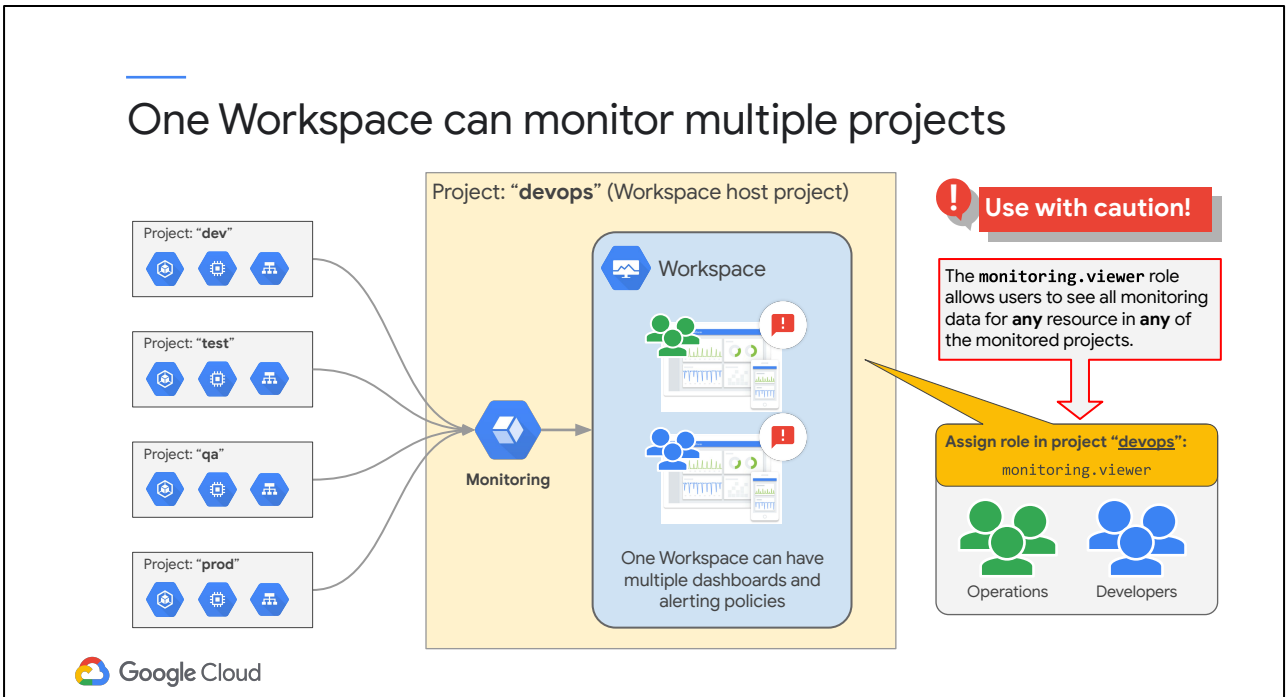
Advantages:

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



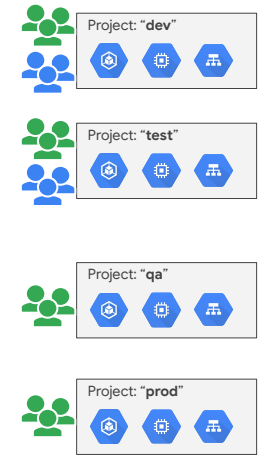
Disadvantages:

- 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.



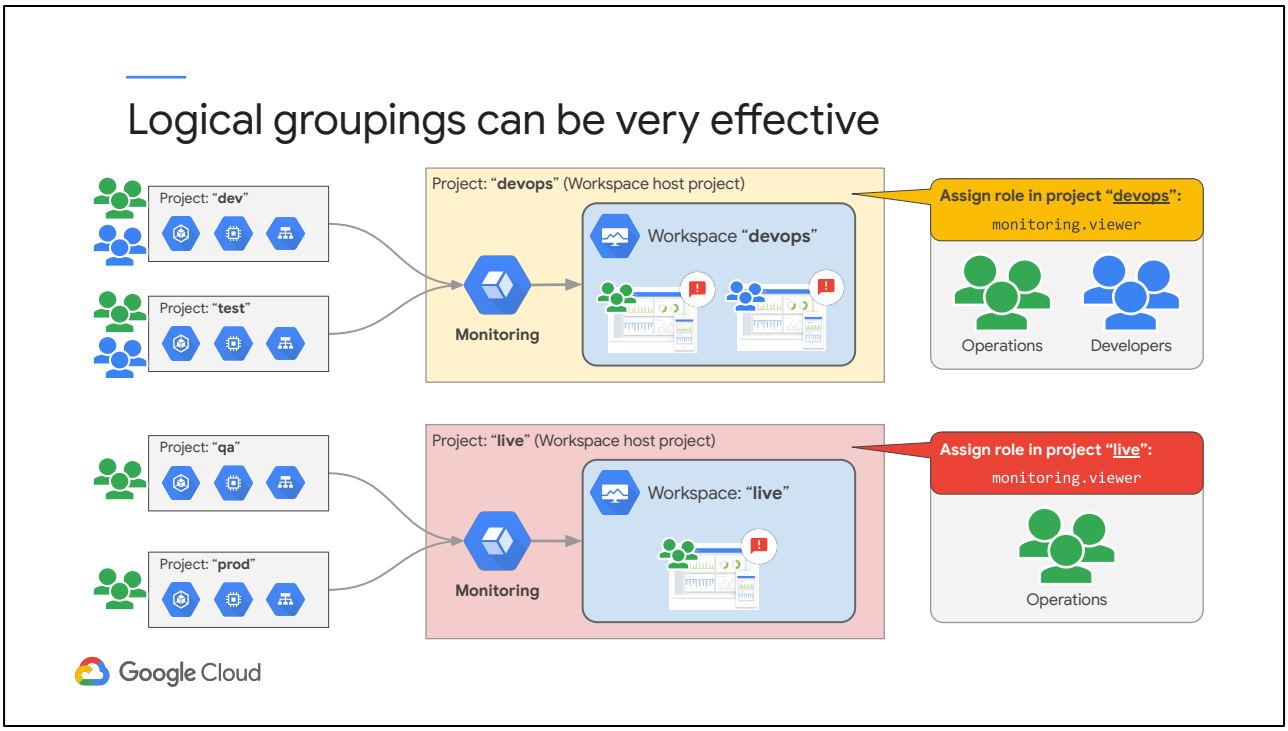
Although the metric data and log entries remain in the individual projects, any user who has been granted the role Monitoring Viewer (roles/monitoring.viewer) will have access to the dashboards and have access to all data by default. This means that a role assigned to one person on one project applies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

Logical groupings can be very effective



Google Cloud

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



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.

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
Monitoring Viewer	Gives you read-only access to the Monitoring console and API
Monitoring Editor	Gives you read-write access to the Monitoring console and API, and lets you write monitoring data to a Workspace
Monitoring Admin	Gives you full access to all Monitoring features

Google Cloud

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.

Past that, a Monitoring Viewer can get read-only access to the Monitoring console and API.

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

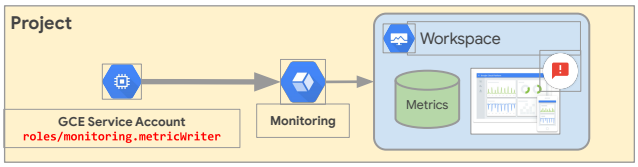
Monitoring Admin has full access to, and control over, all monitoring resources.

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: https://cloud.google.com/monitoring/access-control.

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.



Google Cloud

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.

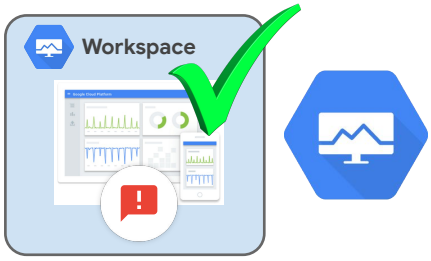
To allow it the write access it needs, grant the VM's service account the Monitoring Metric Writer role in the Workspace host project.

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.

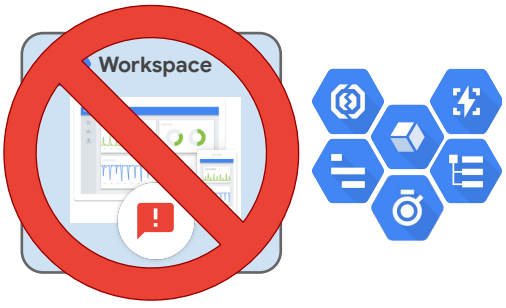
Lecture Notes:
In many cases, when you are creating custom service account, it is easy to forget different writer roles for logging, monitoring and ops suite. Common problem and should be aware of that.

Remember, Workspaces only affect Monitoring

- Only the Monitoring system relies upon Workspaces



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



Google Cloud

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 with Metrics Explorer
- Dashboard Construction
- Monitoring Query Language (MQL)

Uptime Checks

Google Cloud

Now that we've talked about the organization of Monitoring Workspaces, let's create and use some dashboards.

Dashboards: View and analyze metrics

Google Cloud

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.

Lecture Notes:
Before you build your own dashboard, be aware that Google has a lot of default/pre-defined dashboards in the last few years, depending on the products you are using.

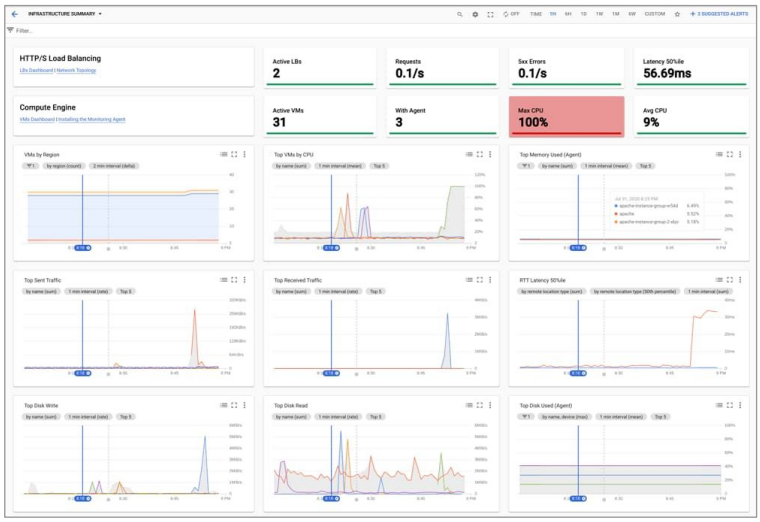
For creating user specific dashboards - use Data Studio and Looker. These are sophisticated tools to build dashboards.
Looker bought by Google. Much more advanced than Data Studio

Predefined Dashboards

Google Cloud

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.

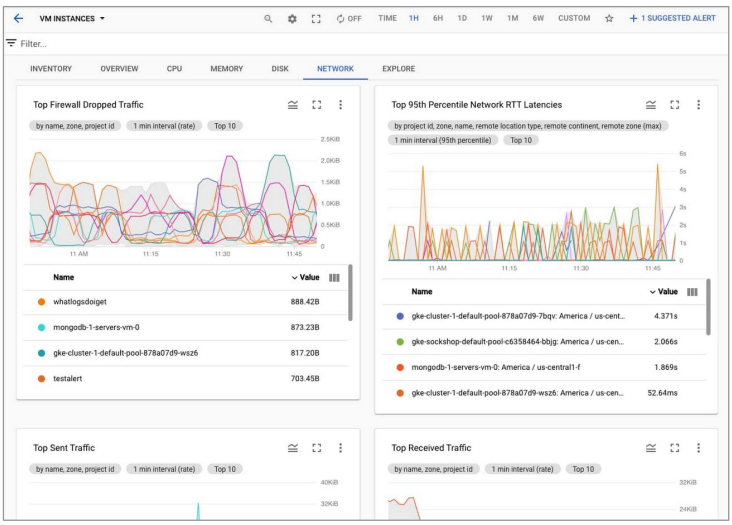
Monitor VM fleet health from infrastructure summary



Google Cloud

The single-pane-of-glass Infrastructure Summary dashboard lets you see aggregate fleet-wide statistics at a glance, and provides insight into the top VMs for a select group of key CPU, disk, memory, and network metrics. You can use the quick links in the top left to jump into detailed troubleshooting dashboards for load balancers, network, and VM instances. The filter bar enables you to narrow your view if you want to see a specific subset of VMs.

Troubleshoot issues with the VM instances dashboard



The VM Instances dashboard now includes agent visibility and installation, and its new tabs let you see fleet-wide information across key metrics.

The per-VM status of the Cloud Monitoring agent is now available in the main inventory page, and you can install the agent on a VM using our built-in wizard. Use the agent to track specified system and application metrics, including memory and disk metrics, advanced system metrics, metrics for workloads like MySQL, Apache, Java virtual machine, and others.

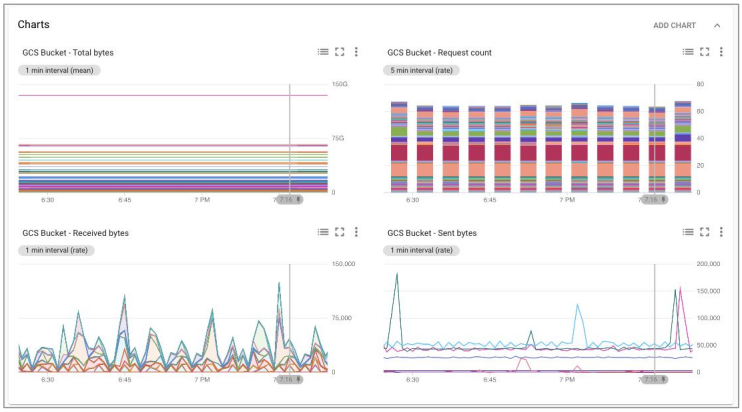
View top VMs across key metrics for CPU, disk, memory, and network

Dedicated tabs for CPU, disk, memory, and network show you outlier VMs for key metrics in each category, so you can visually inspect for anomalies and quickly drill into problem areas and VMs. Filtering allows you to narrow down the set of VMs being displayed in any tab for detailed analysis.

The “Explore” tab gives you insight into the advanced metrics you’re currently collecting in Cloud Monitoring, and quick links to information on how to send additional metrics, so you can see even more metrics in one place.

Easily enable suggested alerts for excessive utilization (memory, disk, network, etc), and receive alert notifications across a variety of channels (email, SMS, Slack, PagerDuty, Cloud Console mobile app, Cloud Pub/Sub, and webhooks).

Dashboards broken into charts



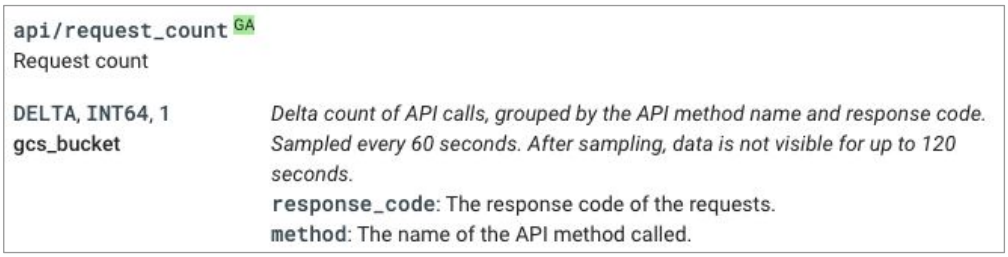
Google Cloud

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



Google Cloud

- Good documentation here.
- Over 1000 metric types and you can create your own.

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/)* 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. Unit of 1 means it is a count

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.
- And CUMULATIVE: a value accumulated over time. This might be the total

Lecture Notes: Before you start with monitoring, you need to understand available metrics and research which are the right metrics to monitor. All these existing metrics are free

- 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.

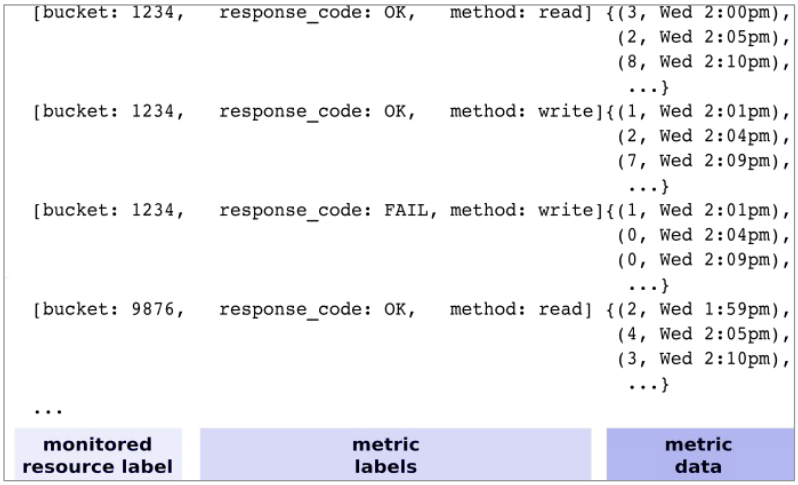
Monitored resource data stored in time series

- Time series are measurements collected over time and stored in a monitoring relational database
  - Metric type, the resource, and (measurement, timestamp)
- One time series is created for each value of:
  - Resource name, metric type, and label

Google Cloud

Cloud Monitoring stores regular measurements collected over time in a monitoring relational database. Each time series contains information on the metric, the resource, and a collection of (measurement, timestamp) pairs. A single time series is created for each combination of metric, resource, and label value.

Time series example



Google Cloud

Here, we see some possible time series for our `storage.googleapis.com/api/request_count` example described above.

In the illustration, the value `bucket: xxxx` represents the value of the bucket_name label in the monitored-resource type, and `response_code` and `method` are labels in the metric type. There is one time series for each combination of values in the resource and metric labels; the illustrations shows some of them.

Agenda

Observability Architecture

Dashboards

- Understanding Dashboards
- Creating Charts with Metrics Explorer
- Dashboard Construction
- Monitoring Query Language (MQL)

Uptime Checks

Google Cloud

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

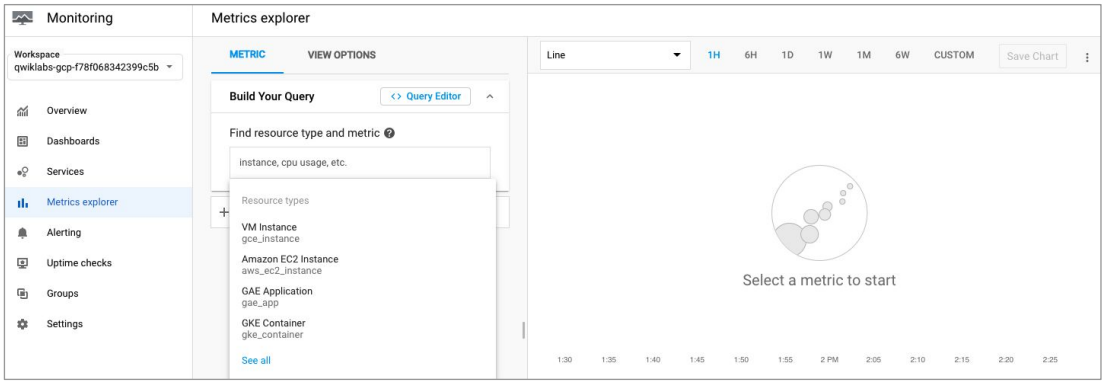
Google Cloud

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 than 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



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 dashboard.
- Share charts by their URL.
- View the configuration for charts as JSON.

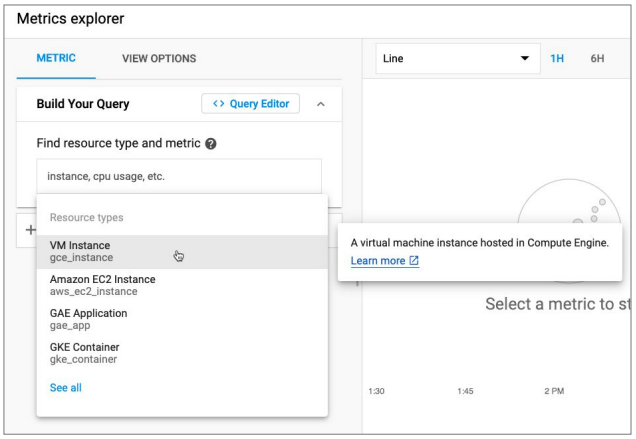
Most importantly, you can use Metrics Explorer as a tool to explore data that you don't need to display long term on a dashboard.

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
  - Automatically filtered



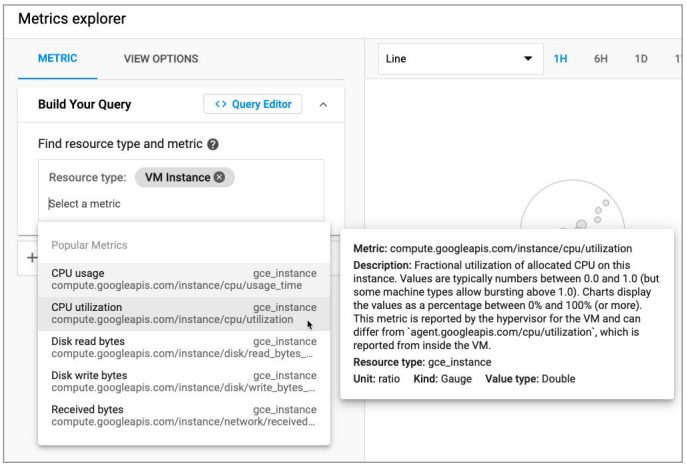
Google Cloud

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*),

Choose a Metric

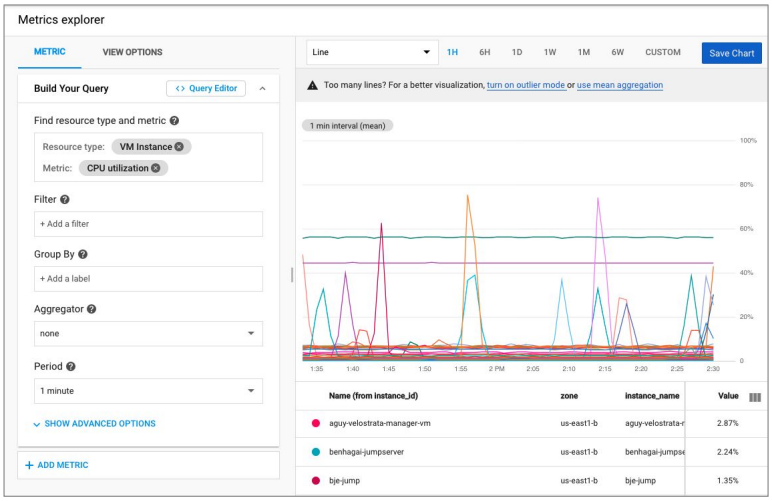
- Start with a Resource
  - Automatically filtered
- Next, pick the desired metric for that resource



Google Cloud

and the *metric type* (also called the *metric descriptor*, or just *metric*).

Metric example



Google Cloud

Let's look at an example.

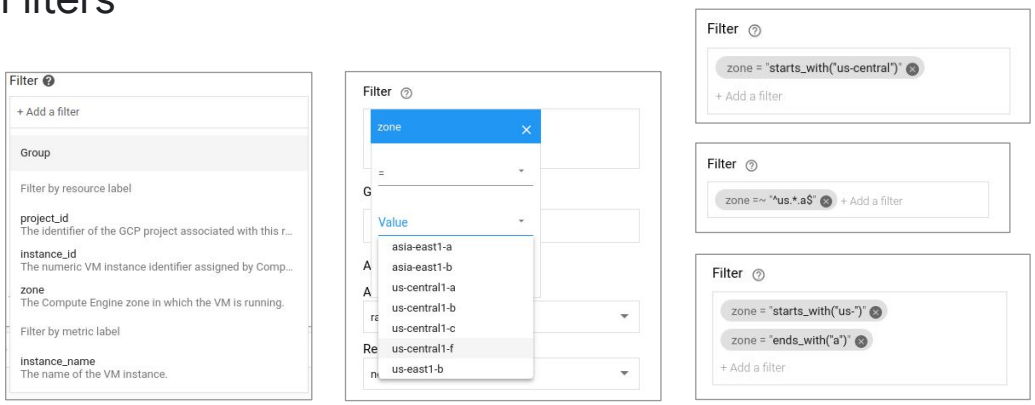
First, you set the resource type to GCE VM Instance. So, we're looking at metrics related to Compute Engine virtual machines.

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



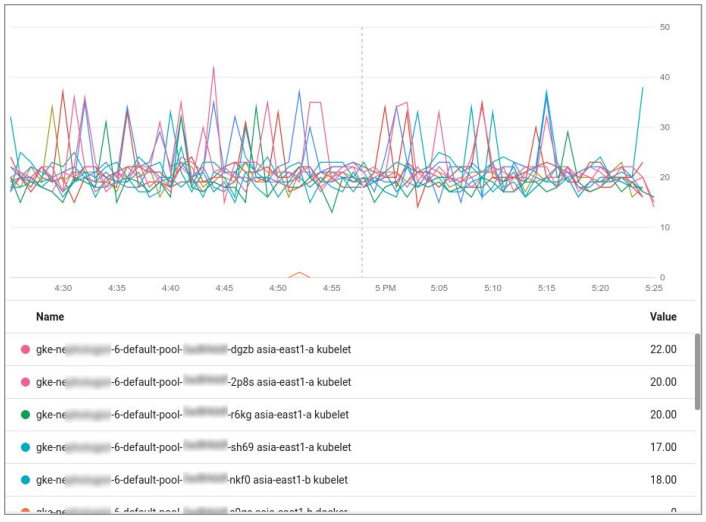
Google Cloud

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.

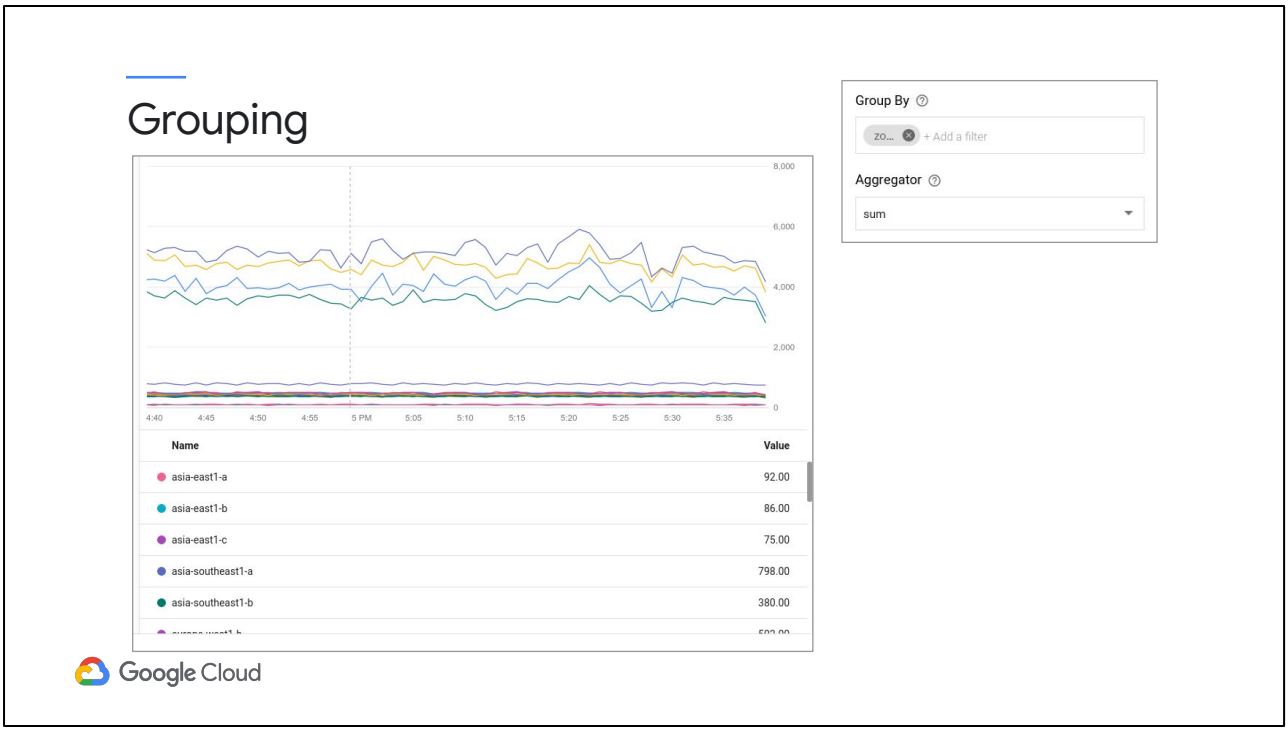
Better



zone=~"asia-*"

Google Cloud

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.



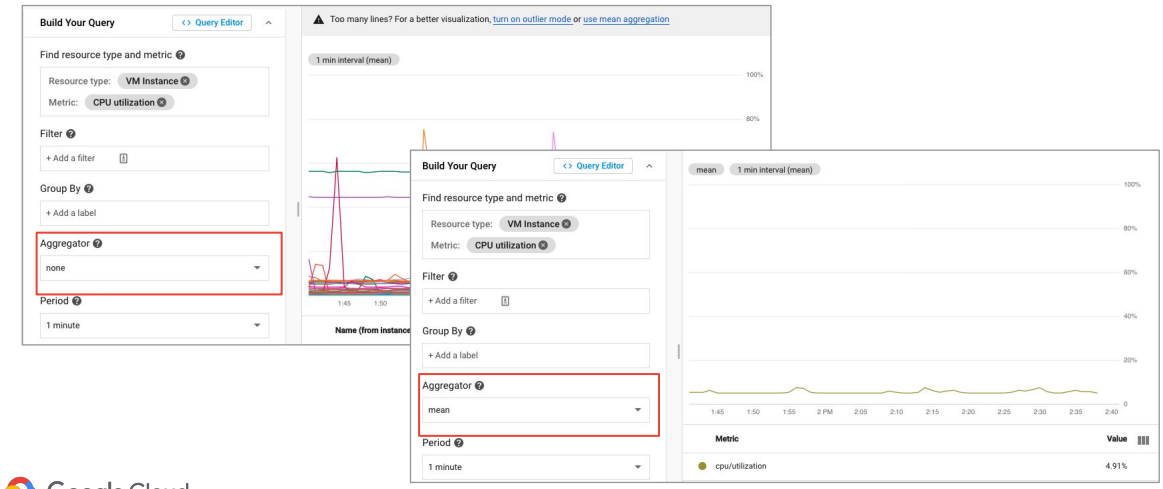
Grouping

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



Google Cloud

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.

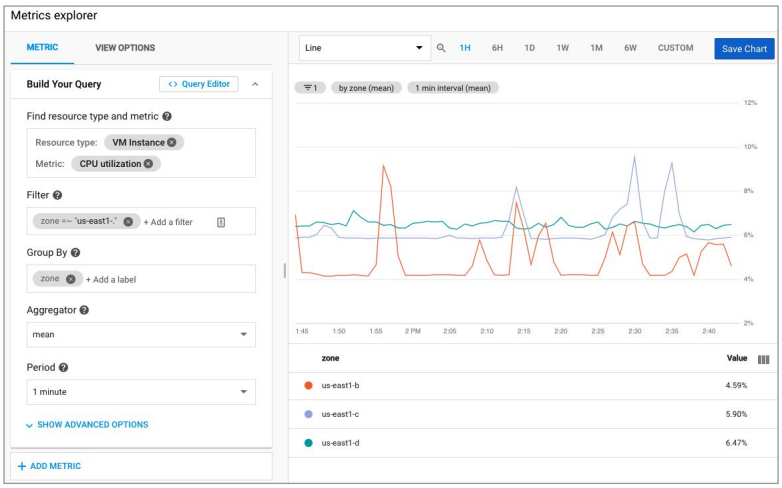
For more information about these dependencies, see the Metrics, time series, and resources documentation.

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)?

Lecture Notes:
Sometimes the Aggregator gets automatically selected (based on earlier selections), and might not be what you desire.

Aggregation, filtered, and grouped



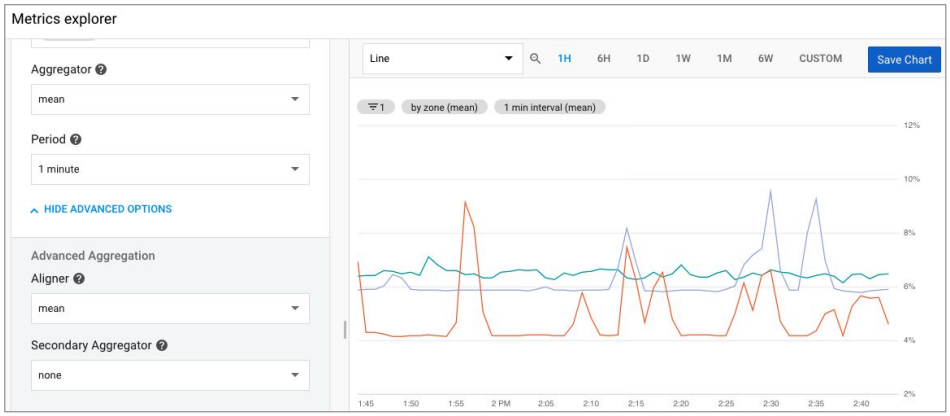
Google Cloud

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.

Lecture Notes:
In some cases where you come across a nice dashboard you would like to create similar, you can click on link "open in metrics explorer" - that should open up the dashboard on metrics explorer and you can find out how that was created

Aligner (break data into regular time buckets)



Google Cloud

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.

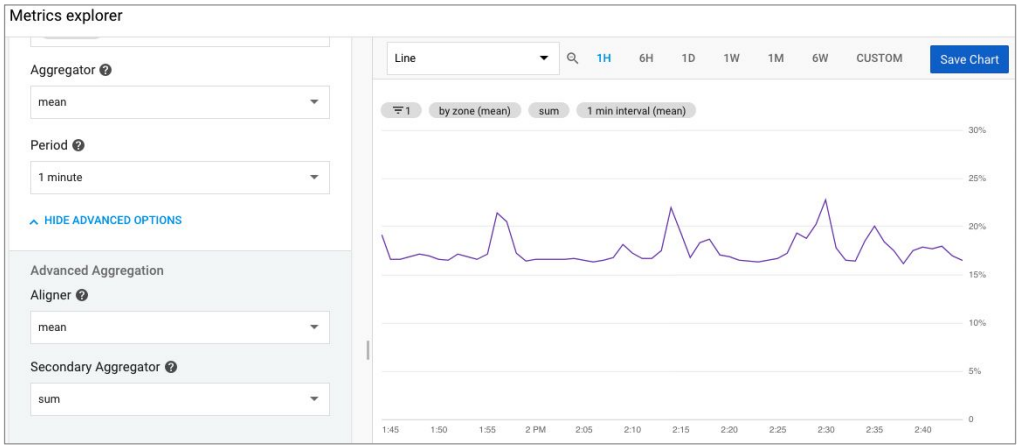
Alignment is a prerequisite to aggregation across time series, and monitoring does it automatically, by using default values. You can override these defaults by using the alignment options, **Aligner** and **Alignment Period.**

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.

Lecture Notes:
Scenarios where you might like to align for periods longer than the shortest duration is to watch trends over time (smaller noises flattens out when you increase the period)

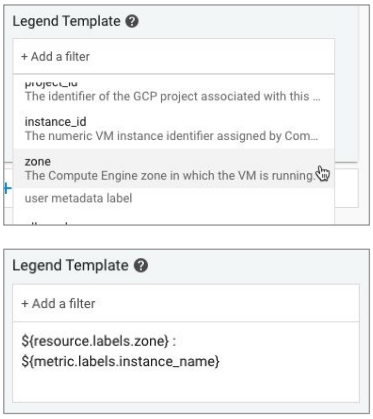
Secondary aggregation

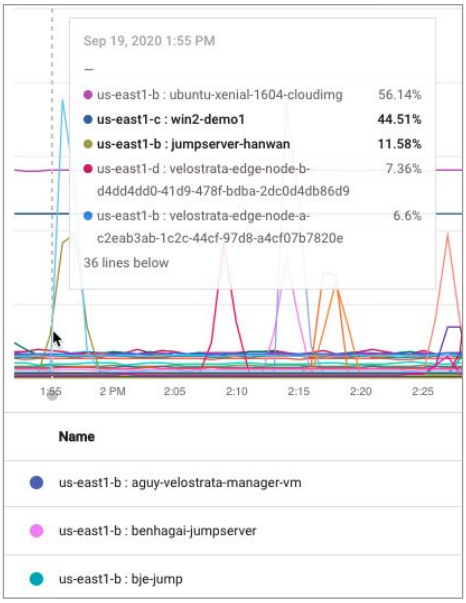


Google Cloud

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





Google Cloud

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.

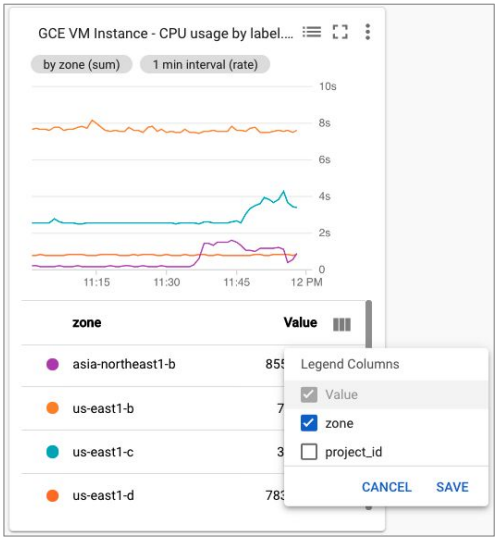
The Legend Template field accepts both plain text and label patterns. The syntax can vary in the patterns but the dynamic replacement sections use the ${ variable } notation.

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

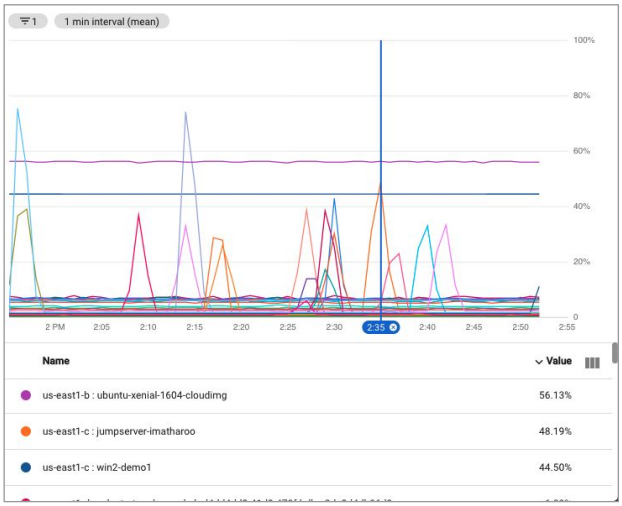
- Select columns and sort the rows



Google Cloud

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

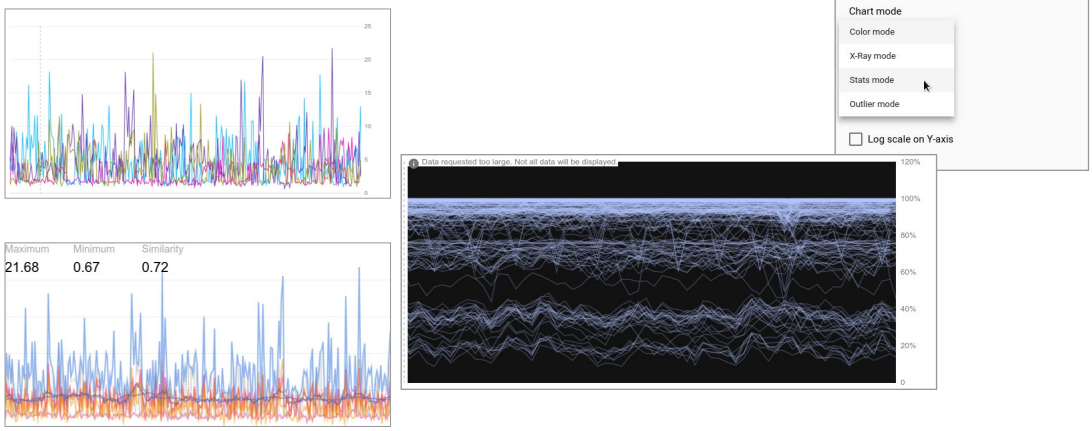
Examine a moment in time by clicking timeline



Google Cloud

Clicking the timeline across the bottom of the cart allows for the selection of a moment in time, and the legend will then display the details about that moment.

View options, Chart mode



Google Cloud

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:

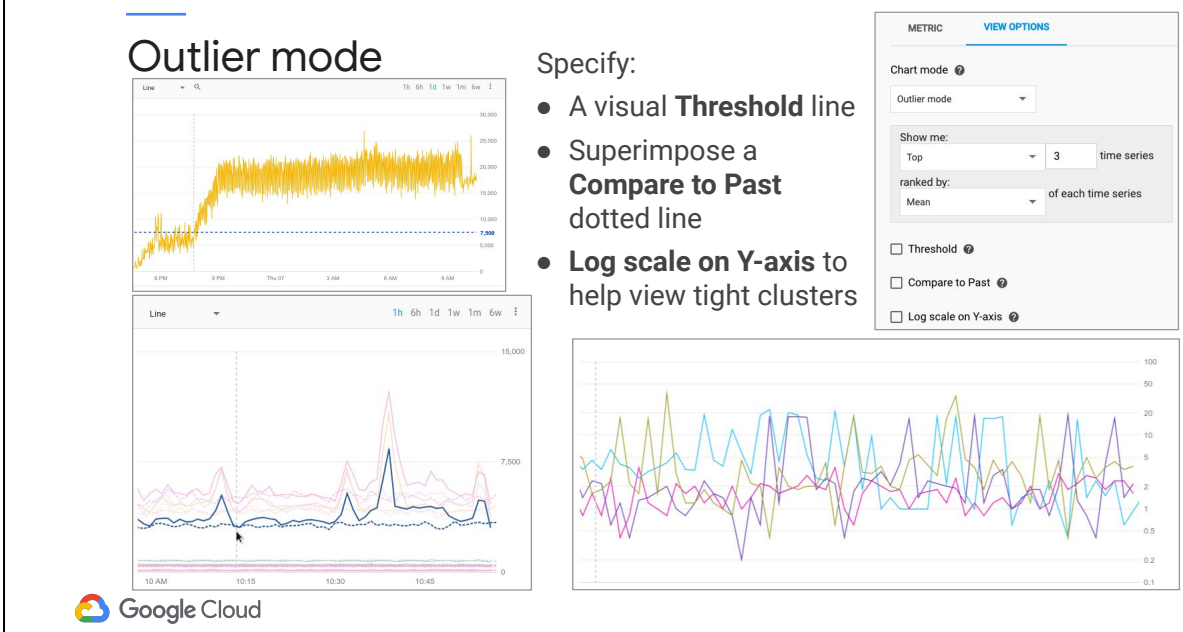
- Color
- Statistics
- X-Ray
- Outlier

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.

Lecture Notes:
Very important are:
Statistics mode - shows different statistically significant figures
Outlier mode - shows outliers. Filters out others



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.

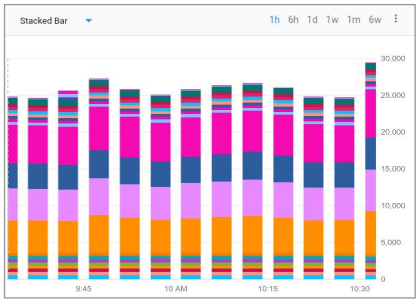
In Outlier mode, you can select the number of time series to show, whether you want extreme high or low values, and the method by which the time series are ranked.

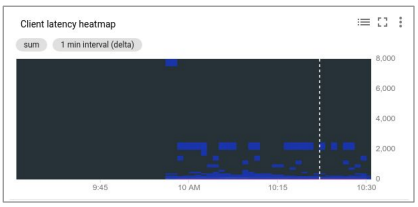
Additionally, you can add a visual threshold line at a particular value.

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





Google Cloud

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 with Metrics Explorer
- Dashboard Construction
- Monitoring Query Language (MQL)

Uptime Checks

Google Cloud

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

Sharing charts and dashboards

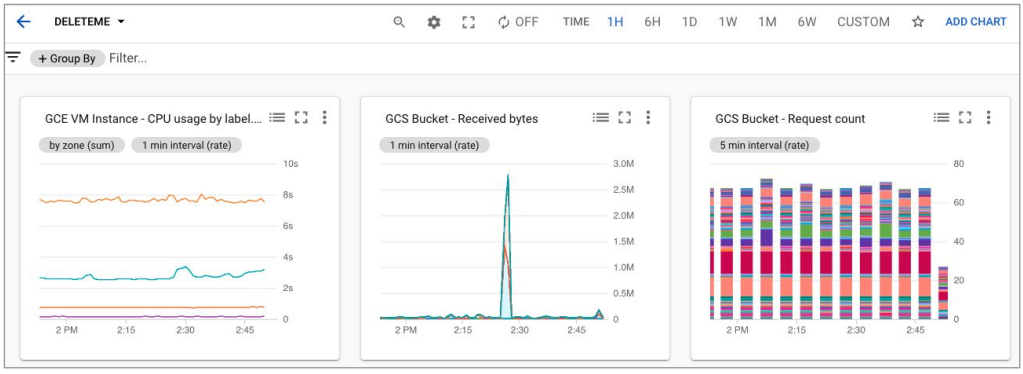
- Share from Metrics Explorer by URL or as JSON
  - URL is nice for quick shares, or charts that aren't repeatedly used
    - Bookmarks?
  - JSON is good if you need to make the chart part of IaC automation

Google Cloud

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.

Putting together the dashboard



Google Cloud

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.

Lecture Notes:
As of April 2021, this is obsolete.

To add a chart to a dashboard, go to the chart and add it to the dashboard

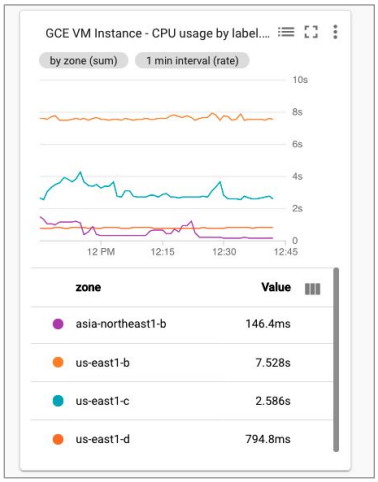
Time to build the dashboard

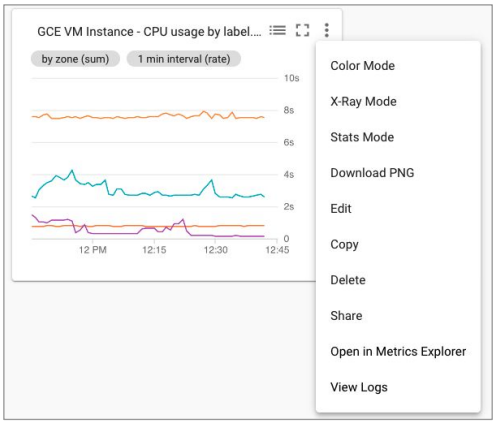
- Create the Dashboard and Save Chart to it

Google Cloud

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

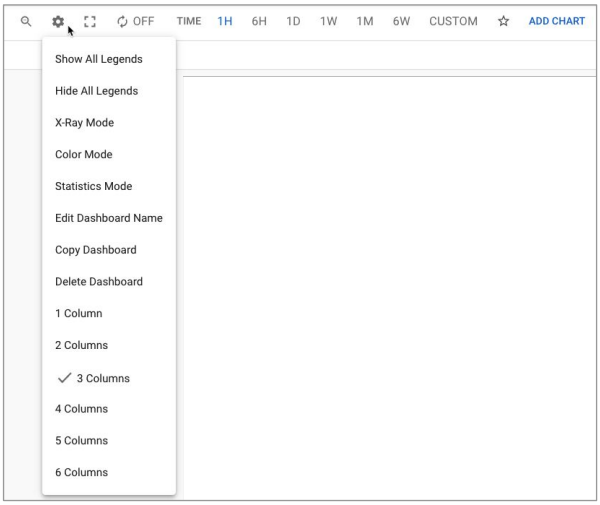




Google Cloud

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



Google Cloud

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

- Understanding Dashboards
- Creating Charts with Metrics Explorer
- Dashboard Construction
- Monitoring Query Language (MQL)

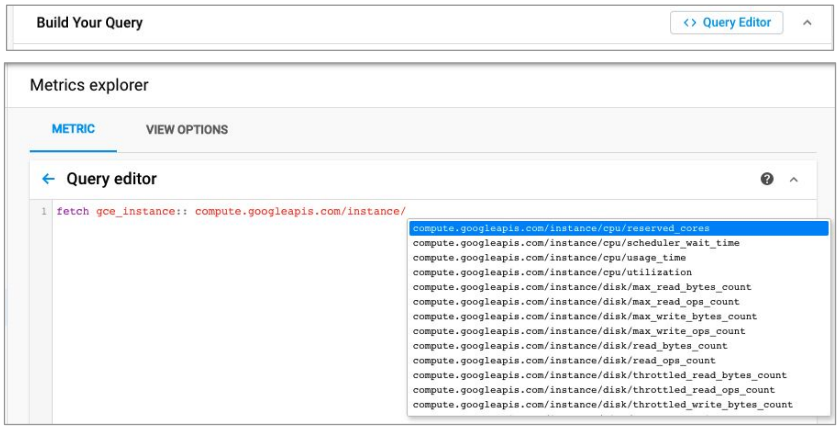
Uptime Checks

Google Cloud

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 Monitoring Query Language (MQL)

MQL is used to query the time series database

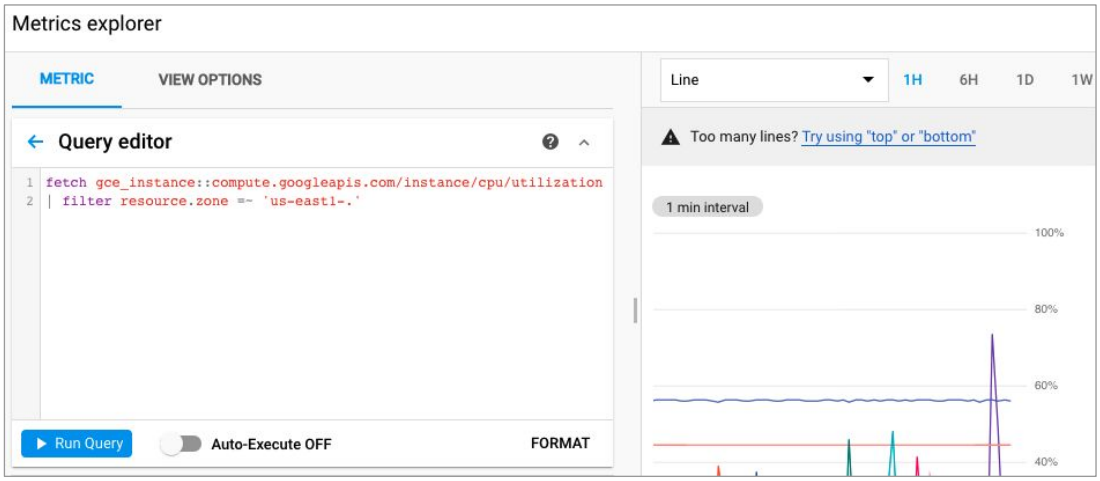
- Use the Query Editor



Google Cloud

Clicking the **Query Editor** button at the top of the Metrics Explorer launches the query interface. Queries can be dynamically created using MQL syntax with the help of the editor's auto complete.

Here's an example we saw earlier in the course



Google Cloud

Here's an example similar to one we saw earlier in the course. It fetches all the CPU utilizations for the Compute Engine VMs in the us-east1 zone, no matter the region.

Though we can't see most of the graph, just like earlier, it is showing a lot of lines. It really needs a filter, group, or something like that.

Lecture Notes:
MQL allows you to join, do maths, etc. that you can't from UI

A query is a list of table operations piped together

- Start with a section operation (***fetch***) and pipe the results through multiple operations
  - ***fetch*** retrieves time series from the Cloud Monitoring data store
  - ***fetch resource_type::metric_type***
  - ***fetch resource_type*** | ***metric metric_type*** also works
- Subset with ***filter***, aggregate with ***group_by***, locate outliers with ***top*** and ***bottom***
- Combine multiple queries with ***{ ; }*** and ***join***
- Use functions like ***value*** to compute ratios and other values
- In the chart, each line represents a resultant time series

Google Cloud

A query is a list of table operations (table_op) piped (|) together. It starts with a selection operation, usually **fetch**, and pipes the results through multiple follow-on operations.

fetch retrieves the time series from the Cloud Monitoring data store and its syntax is: fetch resource_type::metric_type.

You can combine multiple queries with { ; } and join, and you can use functions like value to compute ratios and other values.

When charted, each line represents one of the returned time series.

Here's an example of filtering our time series

Google Cloud

Here's the example we saw at the beginning of this section. The **fetch** returns the time series containing all of the current project's CPU Utilizations. Each one is then filtered in turn, and only the ones with the resource.zone label with values that contain the regular expression us-east1-. are kept.

Now let's add a group_by

Google Cloud

Continuing our example, here we're piping our filtered time series into a group_by. group_by takes two arguments separated by a comma:

- A map, in this case identifying the label we are using to group.
- And an argument specifying how the times series in each group should be combined. Leaving the second argument out will default to sum.

Find outliers with top, top_by, bottom, bottom_by

Google Cloud

To find outliers, use top and bottom. They each take two arguments:

- How many series.
- Over what sort of window.

top_by and bottom_by first group_by, and then grab the top/bottom. They take three arguments:

- A map specifying the labels to group by.
- How many time series to grab.
- Over what sort of window.

top_by and bottom_by group first, and pick some from the group. Group first by zone, then grab the top 1, by max value within the last 10 minutes.

A powerful MQL feature is the ability to create ratios

- Ratios are used frequently in charting and in creating Alerts
  - Error Requests / All requests
  - Reads / Writes
  - Success / Fail
- MQL provides multiple ways to create ratios
  - ***outer_join*** and ***div*** or it's shortened form ***ratio***
  - ***group_by*** and **/** or it's shortened form ***filter_ratio_by***

Google Cloud

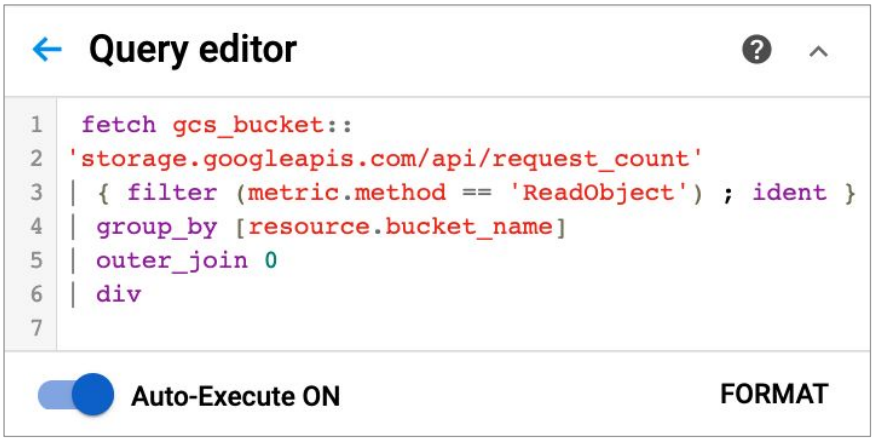
One of the powerful features that direct MQL writing supports, but that the standard Metrics Explorer interface doesn't, is the ability to create ratio metrics. Ratios are used frequently in charting and in creating Alerts. Examples might include:

- Error Requests/All requests
- Reads/Writes
- Success/Fail

MQL supports multiple ways to do ratios and we will examine several, including:

- outer_join and div or its shortened form ratio.
- group_by and/or its shortened form filter_ratio_by.

Let's do a ratio for reads / all access for Cloud Storage



Google Cloud

Let's create a ratio for Cloud Storage that reads as a fraction of all operations. We start by fetching the ***request_count*** metric we used as an example earlier in this section.

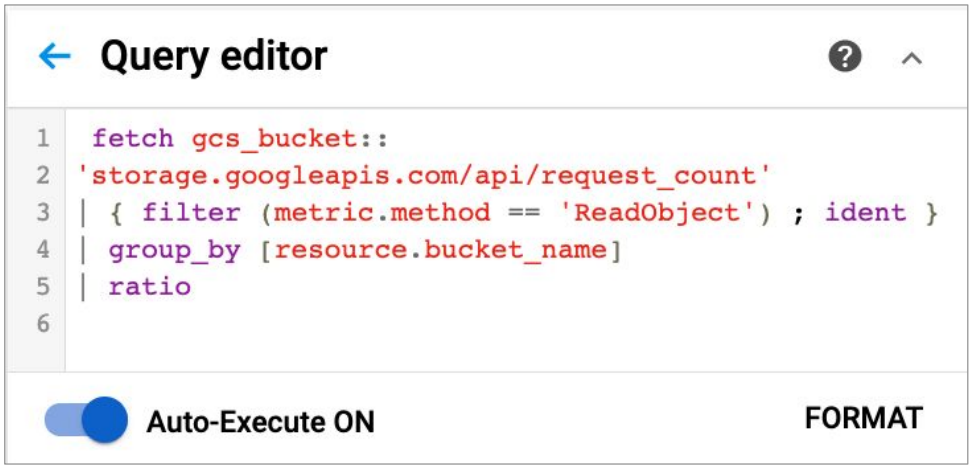
Next, we go to work on the ratio. For the numerator, let's create a table containing only the entries where the method label is ReadObject. For the denominator, we use the ident(ity) operator, which simply returns the original table with all the data.

We take the two tables and group each by bucket_name. Since we don't specify an aggregation function, sum will be used.

When we pipe the two tables through the outer_join, we end up pairing time series with matching labels (bucket_names). The paired time series are zipped up by matching the timestamp of each point. Where a value is missing, 0 will be used as a stand in.

Finally, we use the div function to divide the two values for each input point to produce a single output.

Since ratios are so common, MQL provides ***ratio***



Google Cloud

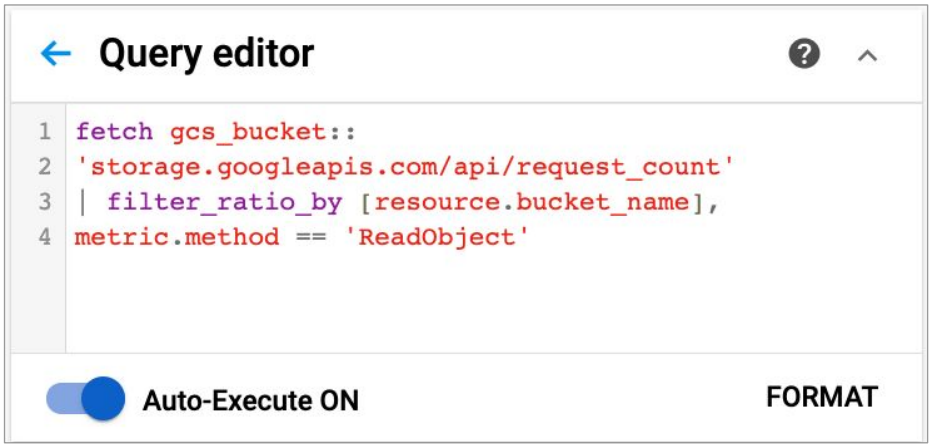
In the last example, we could have replaced div with any function. Since ratios are so common, we can simplify the last example by replacing the last two steps with the ratio operation.

If the ratio is computed from a single time series

Google Cloud

If the ratio is computed from a single time series, then leveraging the group_by's aggregation function will also do the job.

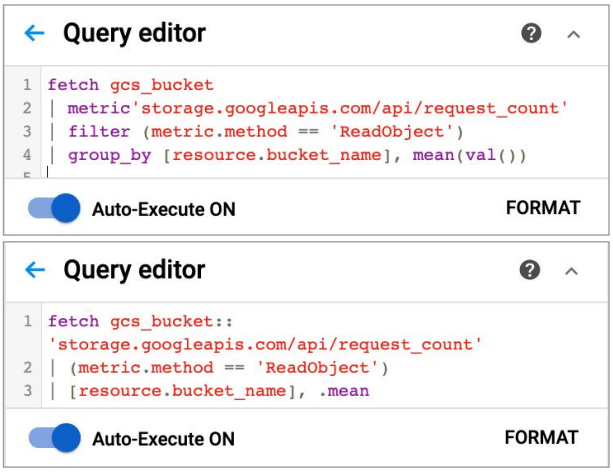
Single time series ratio, simplified form



Google Cloud

Finally, we see the ratio in its simplest form. Filter_ratio_by takes in the group_by map as a first argument, and the test to use to determine the values in the numerator table as the second. Remember, the default aggregation function it's using is sum.

A final note on strict and concise queries



Google Cloud

Here, we see the exact same query written two ways. In the first example, we are using strict syntax, specifying all the functions by name. In the second, we see concise format. Concise format takes advantage of shorthand notations and optional elements. Notice ( ) implies a filter and the [ ] map implies a group_by.

.function is short for function(val)).

When saving queries as charts they will always be converted into strict form.

Agenda

Observability Architecture

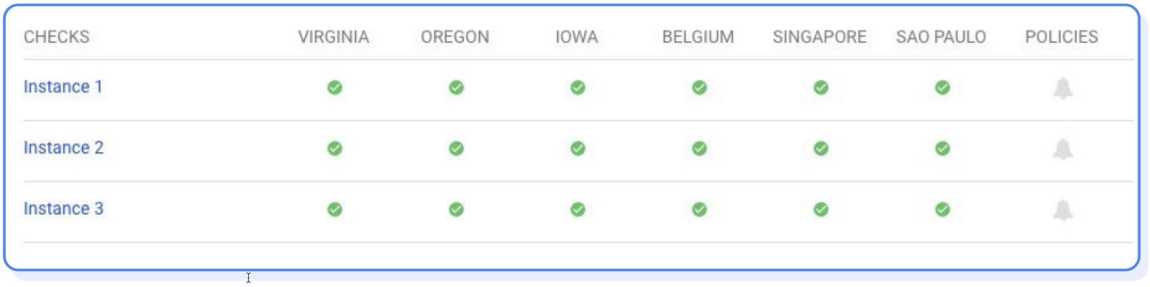
Dashboards

Uptime Checks

Google Cloud

Another monitoring component we discussed briefly in an earlier module of this course is uptime checks.

Uptime Checks check public service availability



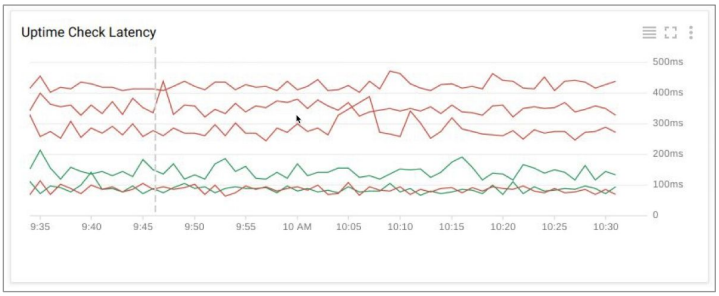
Google Cloud

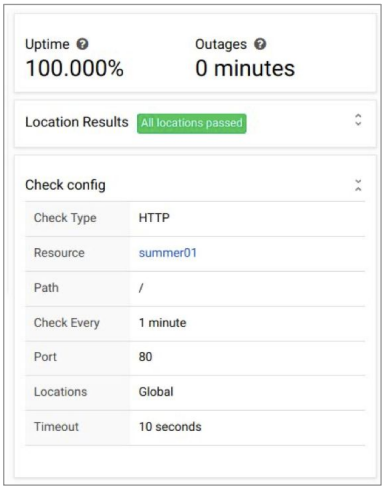
Uptime checks can be configured to test the availability of your public services from locations around the world, as you can see on this slide. The type of uptime check can be set to HTTP, HTTPS, or TCP. The resource to be checked can be an App Engine application, a Compute Engine instance, a URL of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





Google Cloud

Here is an example of an HTTP uptimecheck. 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

Google Cloud

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

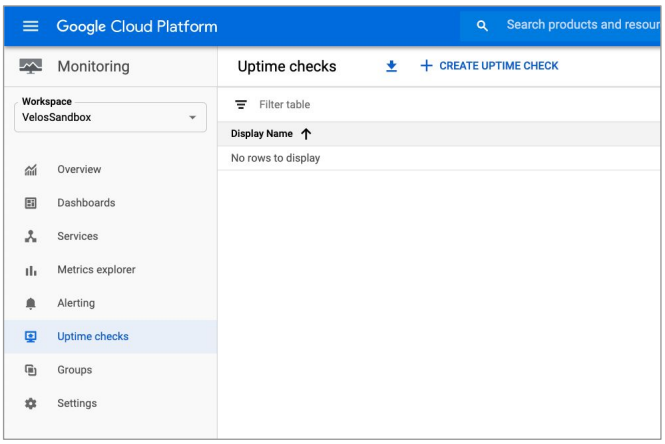
Google Cloud

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.

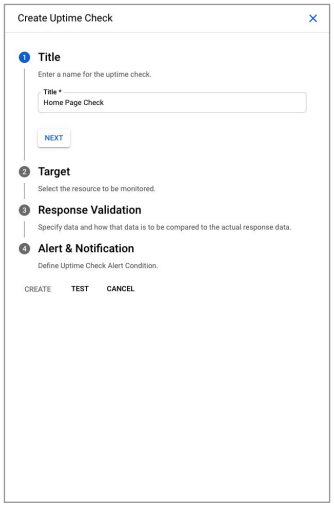
Creating an Uptime Check

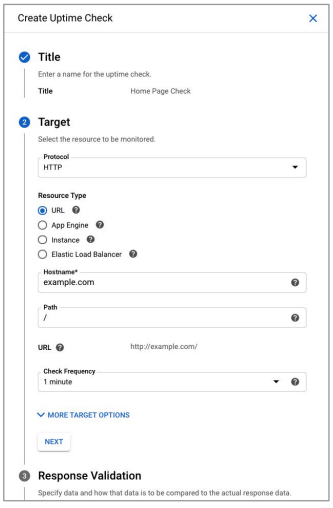


Google Cloud

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

Creating an Uptime Check

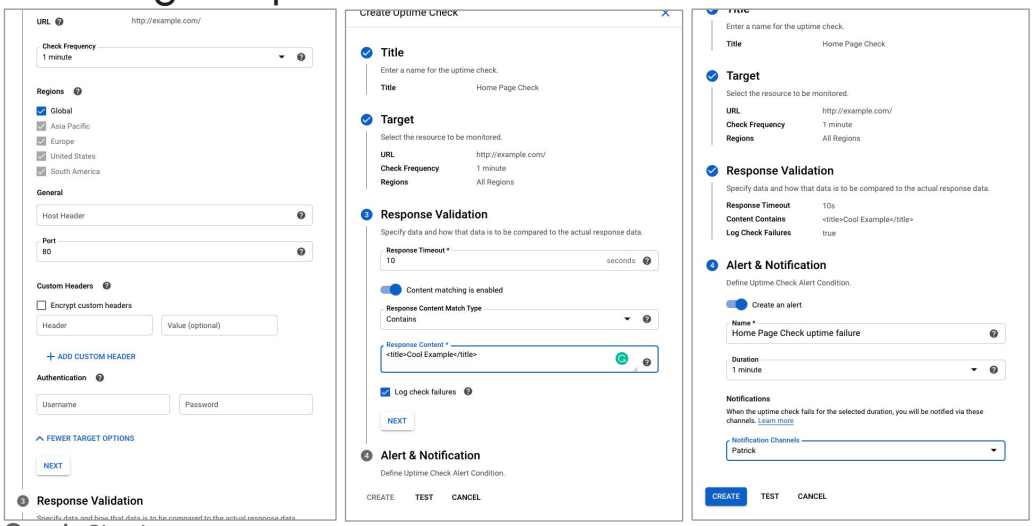




Google Cloud

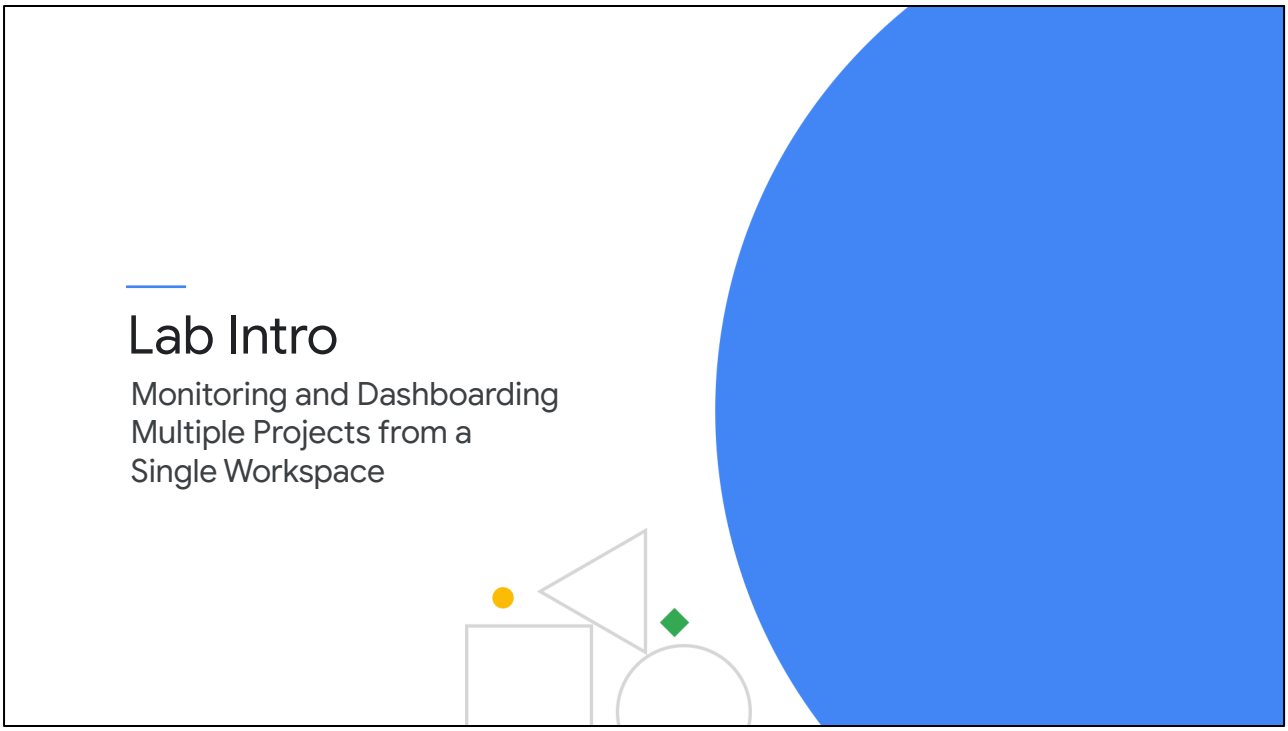
Give the uptime check a name/title that is descriptive. Select the check type protocol, the resource type, and appropriate information for that resource type. A URL, for example, would need a hostname and an optional page path.

Creating an Uptime Check



Google Cloud

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.



Google Cloud Monitoring empowers users with the ability to monitor 1..100 projects from a single monitoring Workspace. In this lab, you start with three Google 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.

Quiz

You're setting up a load balancer for an application you deployed to Compute Engine. To ensure the load balancer only sends requests to machines that are working, you would use what Google Cloud tool?

A. Uptime check
B. Health check
C. Readiness probe
D. Liveness probe

Quiz

You're setting up a load balancer for an application you deployed to Compute Engine. To ensure the load balancer only sends requests to machines that are working, you would use what Google Cloud tool?

A. Uptime check
B. Health check
C. Readiness probe
D. Liveness probe

Quiz

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

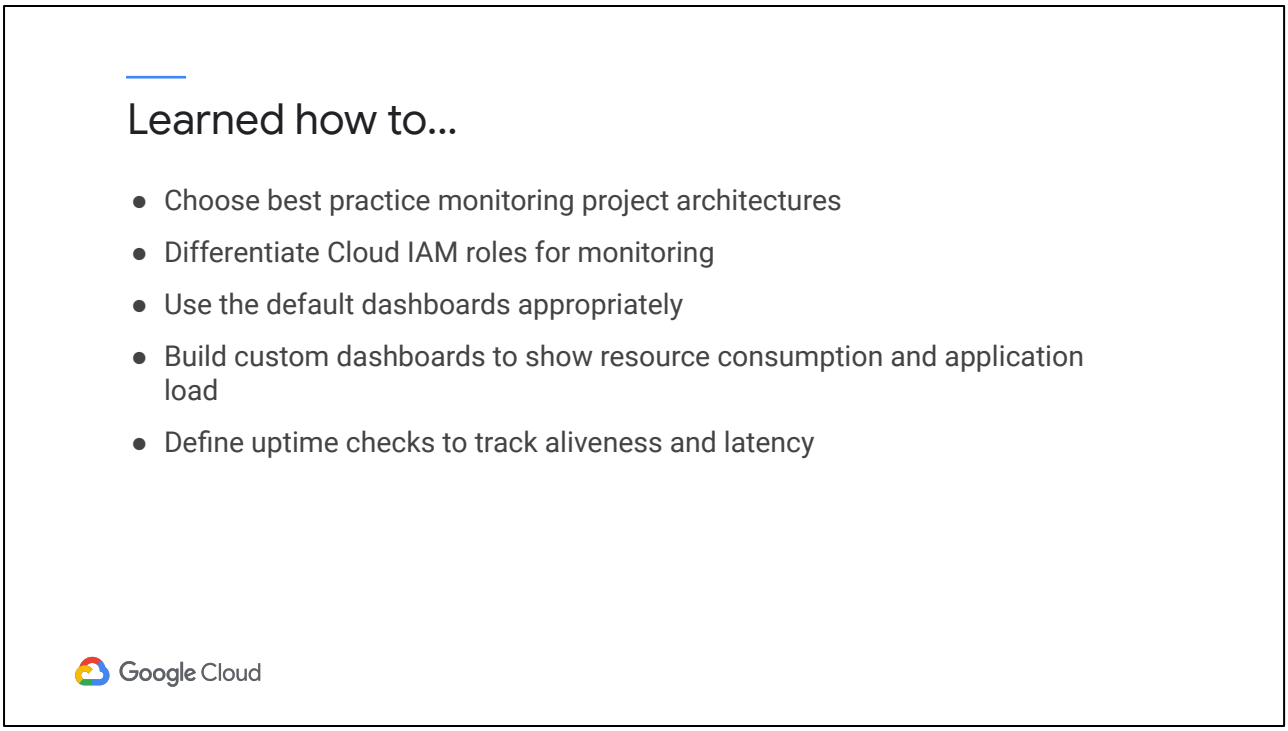
What below is a feature of Google Cloud monitoring dashboards?

A. Automated predefined dashboards based on project resources
B. Can monitor resources in AWS and on-premises as well as Google Cloud
C. Can create workspaces to monitor resources in different projects
D. All of the above

Quiz

What below is a feature of Google Cloud monitoring dashboards?

A. Automated predefined dashboards based on project resources
B. Can monitor resources in AWS and on-premises as well as Google Cloud
C. Can create workspaces to monitor resources in different projects
D. All of the above



Very nice. In this module, you learned how to:

- Choose best practice monitoring project architectures
- Differentiate Cloud IAM roles for monitoring
- Use the default dashboards appropriately
- Build custom dashboards to show resource consumption and application load
- And define uptime checks to track aliveness and latency.

Great job and keep up the good work.

Google Cloud