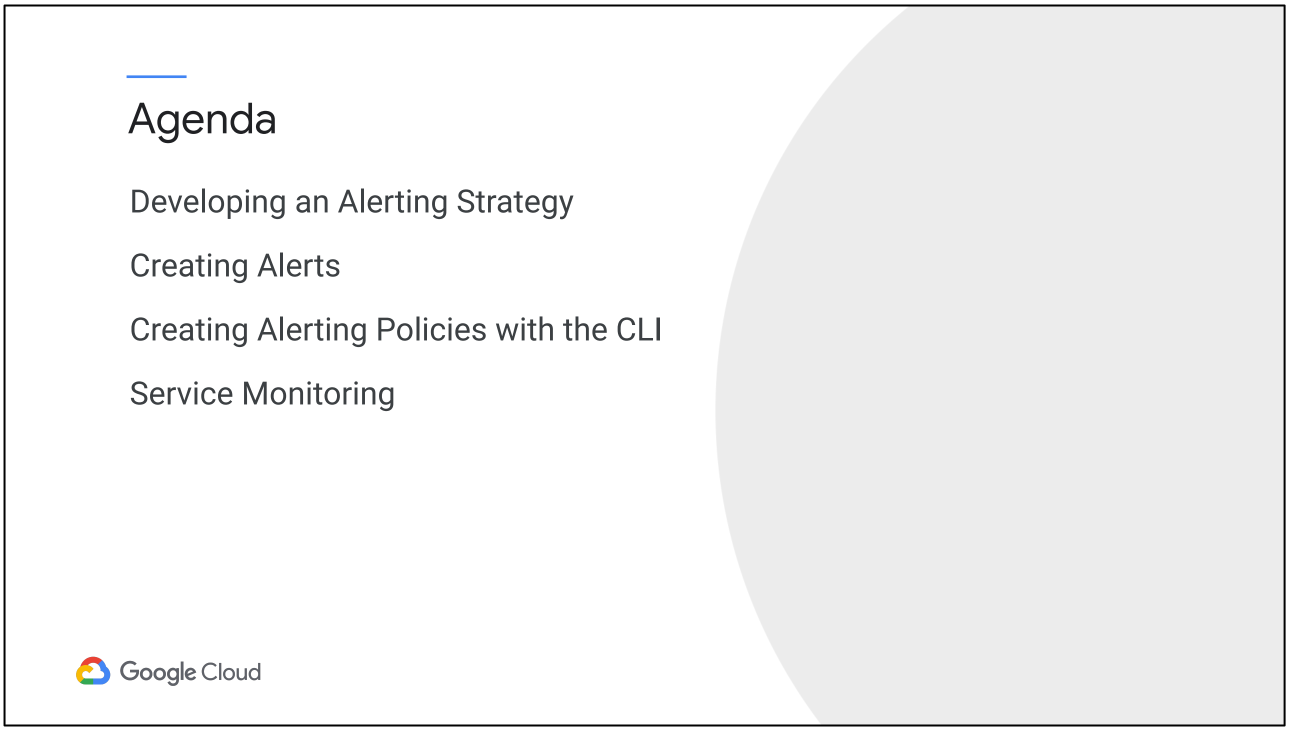
Alerting Policies

Alerting gives timely awareness to problems in your cloud applications so you can resolve the problems quickly.



Agenda

Developing an Alerting Strategy

Creating Alerts

Creating Alerting Policies with the CLI

Service Monitoring

Google Cloud

In this module, you will learn how to:

Develop alerting strategies

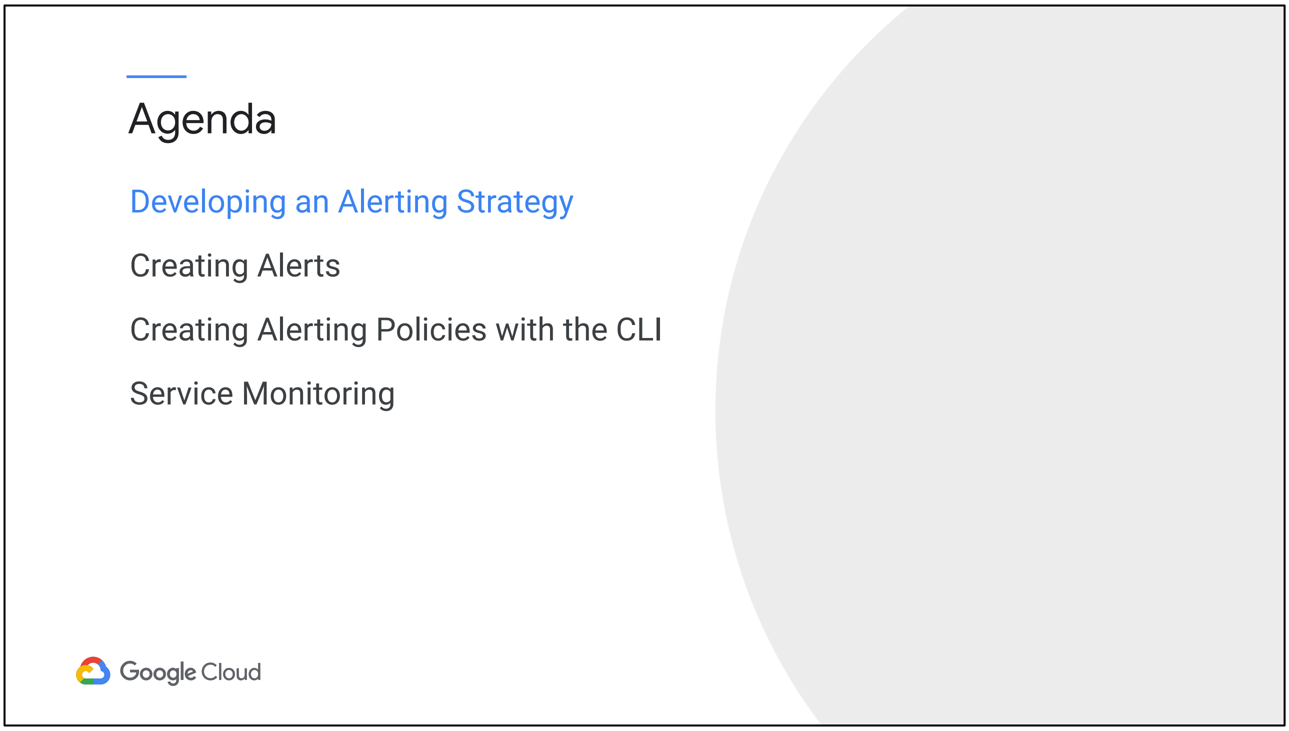
Define alerting policies

Add notification channels

Identify types of alerts and common uses for each

Construct and alert on resource groups

And manage alerting policies programmatically



Agenda

Developing an Alerting Strategy

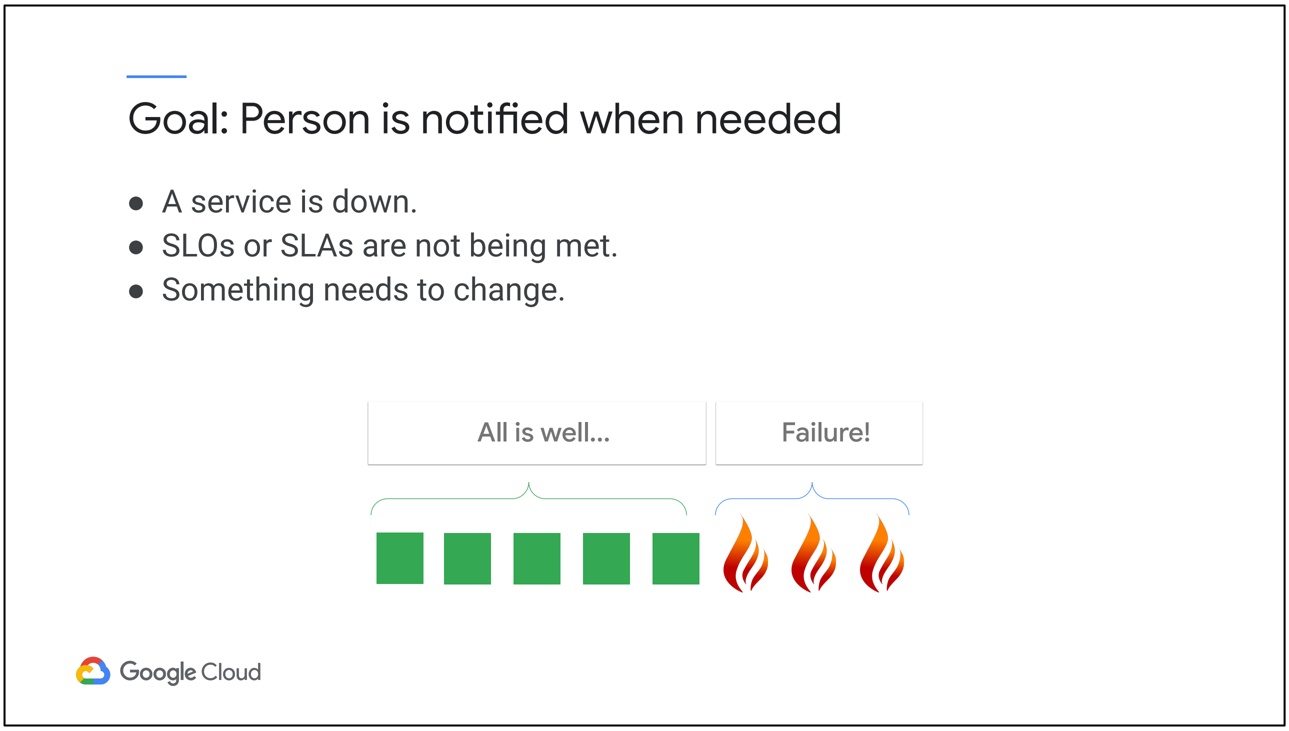
Creating Alerts

Creating Alerting Policies with the CLI

Service Monitoring

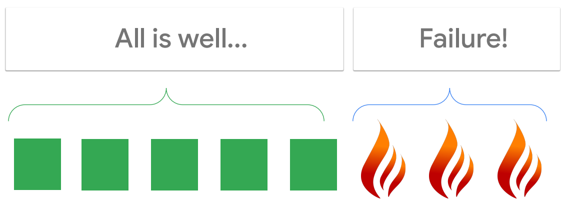
Google Cloud

Let's start by getting an alerting strategy in place.



Goal: Person is notified when needed

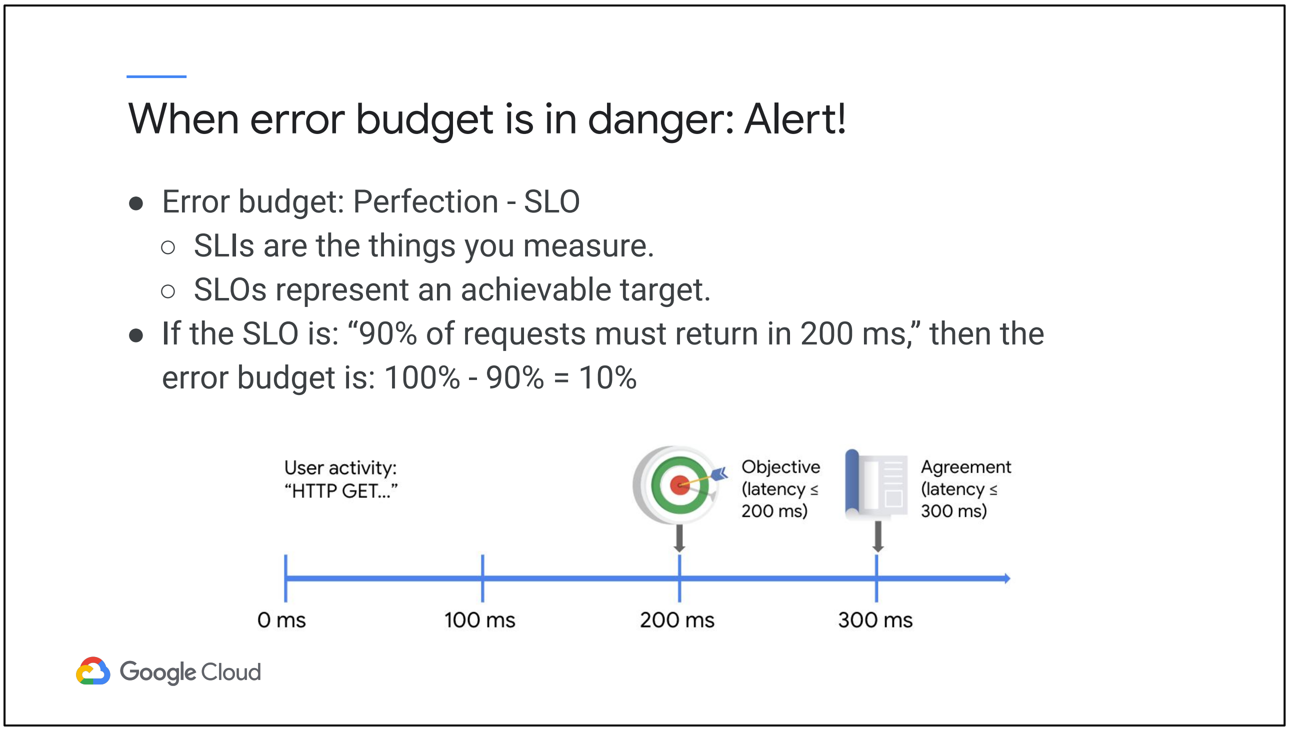
- A service is down.
- SLOs or SLAs are not being met.
- Something needs to change.



Google Cloud

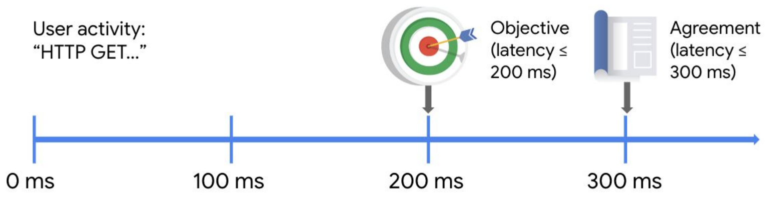
Let's start by defining our term. An alert is an automated notification sent by Google Cloud through some notification channel to an external application, ticketing system, or person.

Why is the alert being sent? Perhaps a service is down, or an SLO isn't being met. Regardless, an alert is generated when something needs to change.



When error budget is in danger: Alert!

- Error budget: Perfection - SLO
  - SLIs are the things you measure.
  - SLOs represent an achievable target.
- If the SLO is: “90% of requests must return in 200 ms,” then the error budget is: 100% - 90% = 10%

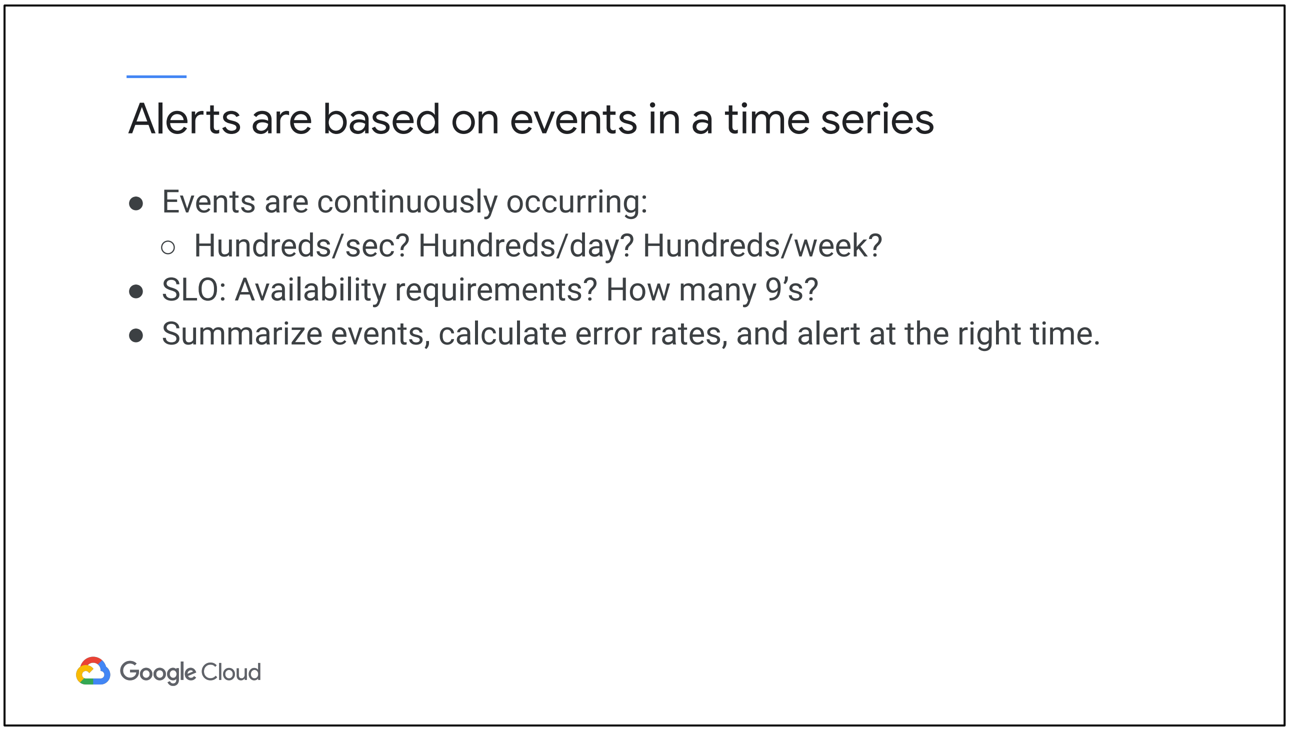

User activity: “HTTP GET...”
Objective (latency ≤ 200 ms)
Agreement (latency ≤ 300 ms)
0 ms
100 ms
200 ms
300 ms

Google Cloud

A great time to generate alerts is when a system appears to be on track to spend all of its error budget before the allocated time window.

Remember from our SLI/SLO discussion in the last module that an Error budget is Perfection - SLO. SLIs are things that are measured, and SLOs represent achievable targets.

If the SLO target is "90% of requests must return in 200 ms," then the error budget is 100% - 90%=10%.



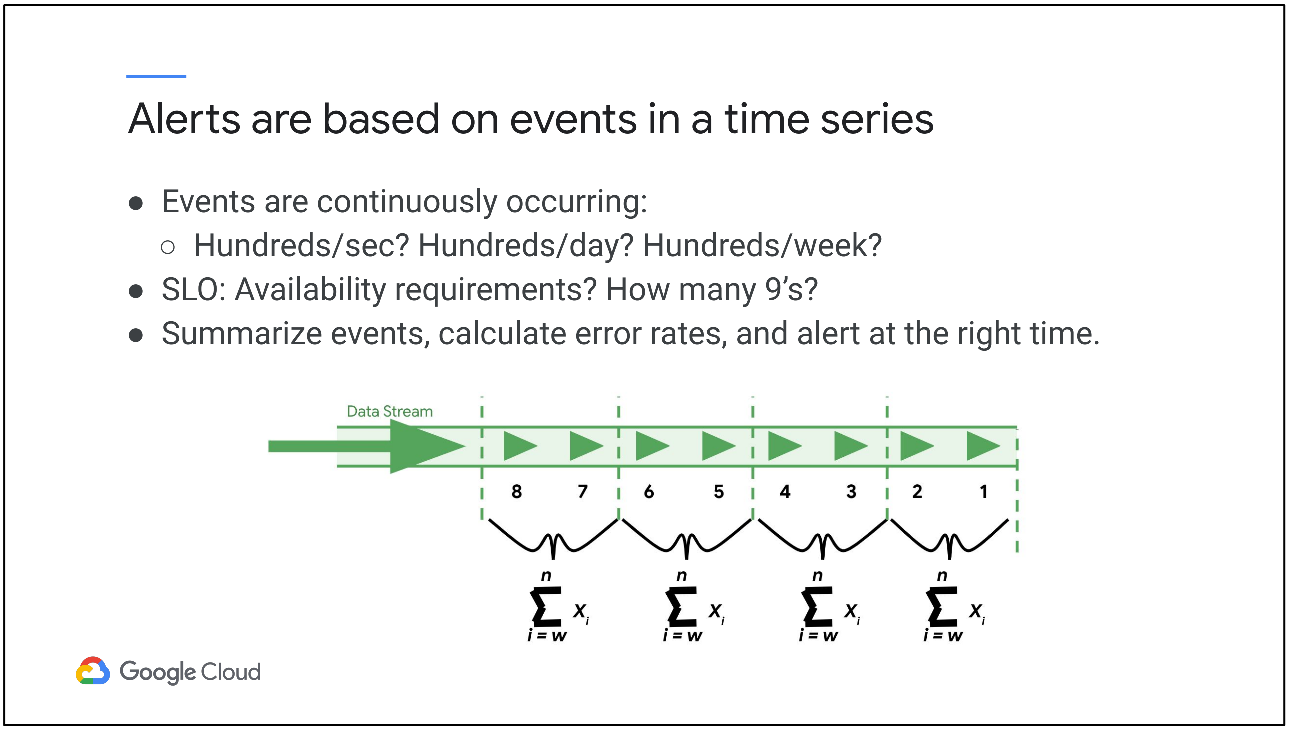
Alerts are based on events in a time series

- Events are continuously occurring:
    - Hundreds/sec? Hundreds/day? Hundreds/week?
- SLO: Availability requirements? How many 9's?
- Summarize events, calculate error rates, and alert at the right time.

Google Cloud

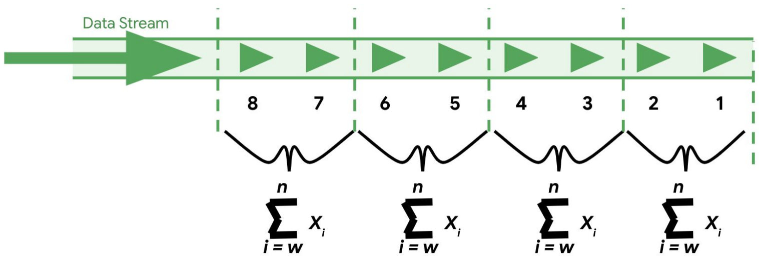
The events are processed through a time series: a series of event data points broken into successive, equally spaced windows of time. Based on need, each window's duration and the math applied to the member data points inside of each window are both configurable.

Because of the time series, events can be summarized, error rates can be calculated, and alerts can be triggered where appropriate.



Alerts are based on events in a time series

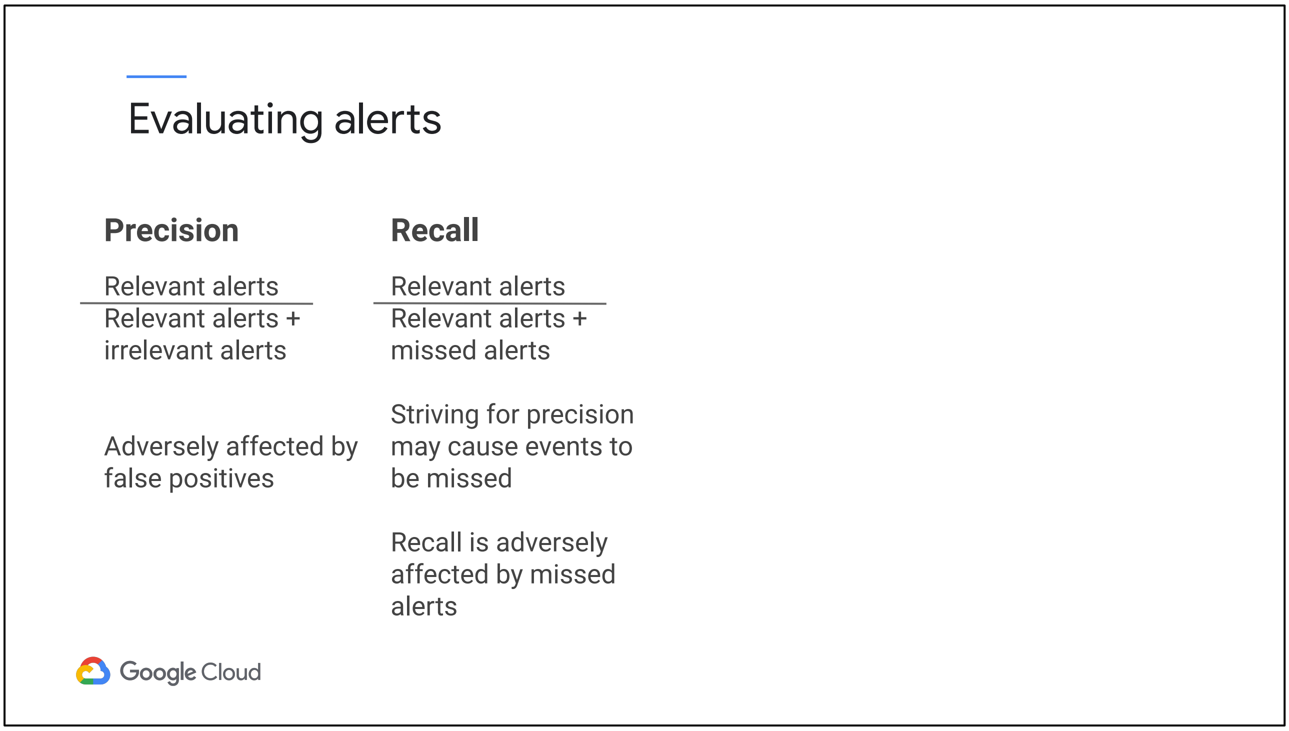
- Events are continuously occurring:
  - Hundreds/sec? Hundreds/day? Hundreds/week?
- SLO: Availability requirements? How many 9's?
- Summarize events, calculate error rates, and alert at the right time.



Google Cloud

The events are processed through a time series: a series of event data points broken into successive, equally spaced windows of time. Based on need, each window's duration and the math applied to the member data points inside of each window are both configurable.

Because of the time series, events can be summarized, error rates can be calculated, and alerts can be triggered where appropriate.



Evaluating alerts

Precision

$$\frac{\text{Relevant alerts}}{\text{Relevant alerts + irrelevant alerts}}$$

Adversely affected by false positives

Recall

$$\frac{\text{Relevant alerts}}{\text{Relevant alerts + missed alerts}}$$

Striving for precision may cause events to be missed

Recall is adversely affected by missed alerts

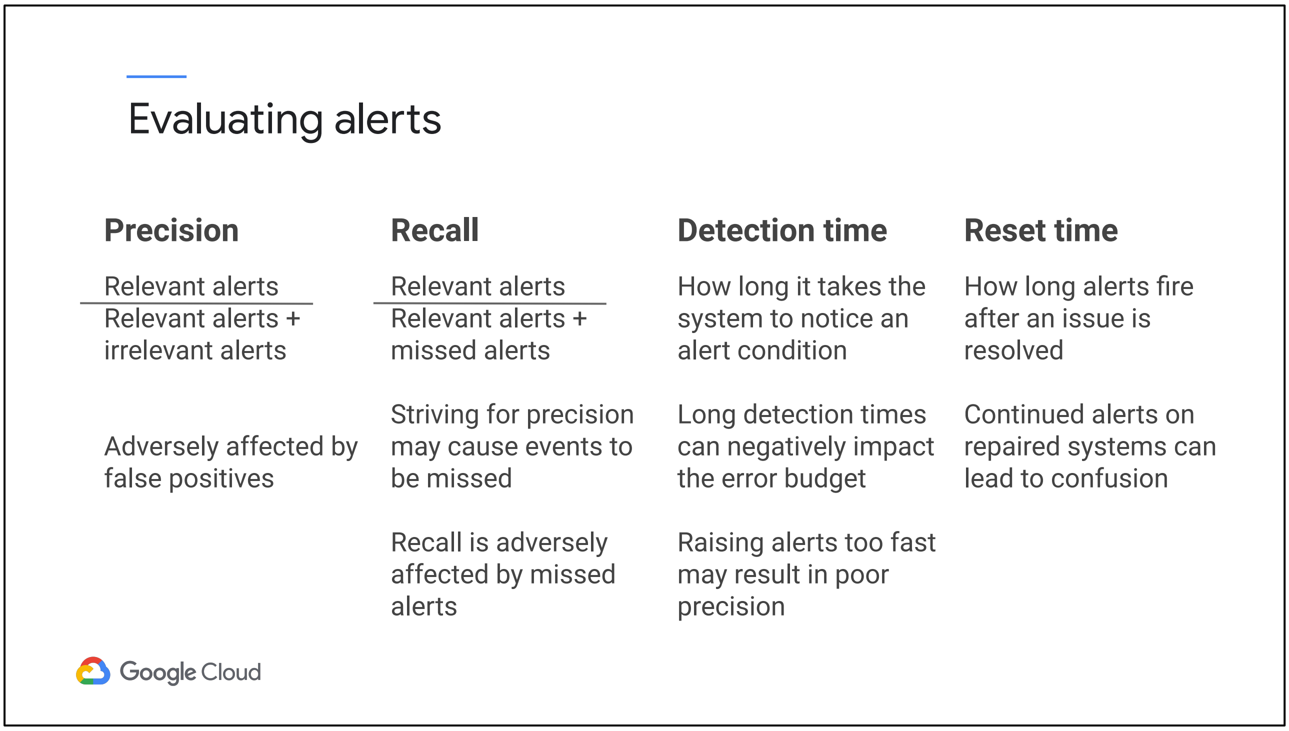
Google Cloud

Several attributes should be considered when attempting to measure the accuracy or effectiveness of a particular alerting strategy.

Precision is the proportion of alerts detected that were relevant to the sum of relevant and irrelevant alerts. It is decreased by false alerts.

Recall is the proportion of alerts detected that were relevant to the sum of relevant alerts and missed alerts. It is decreased by missing alerts.

Precision can be seen as a measure of exactness, whereas recall is a measure of completeness.



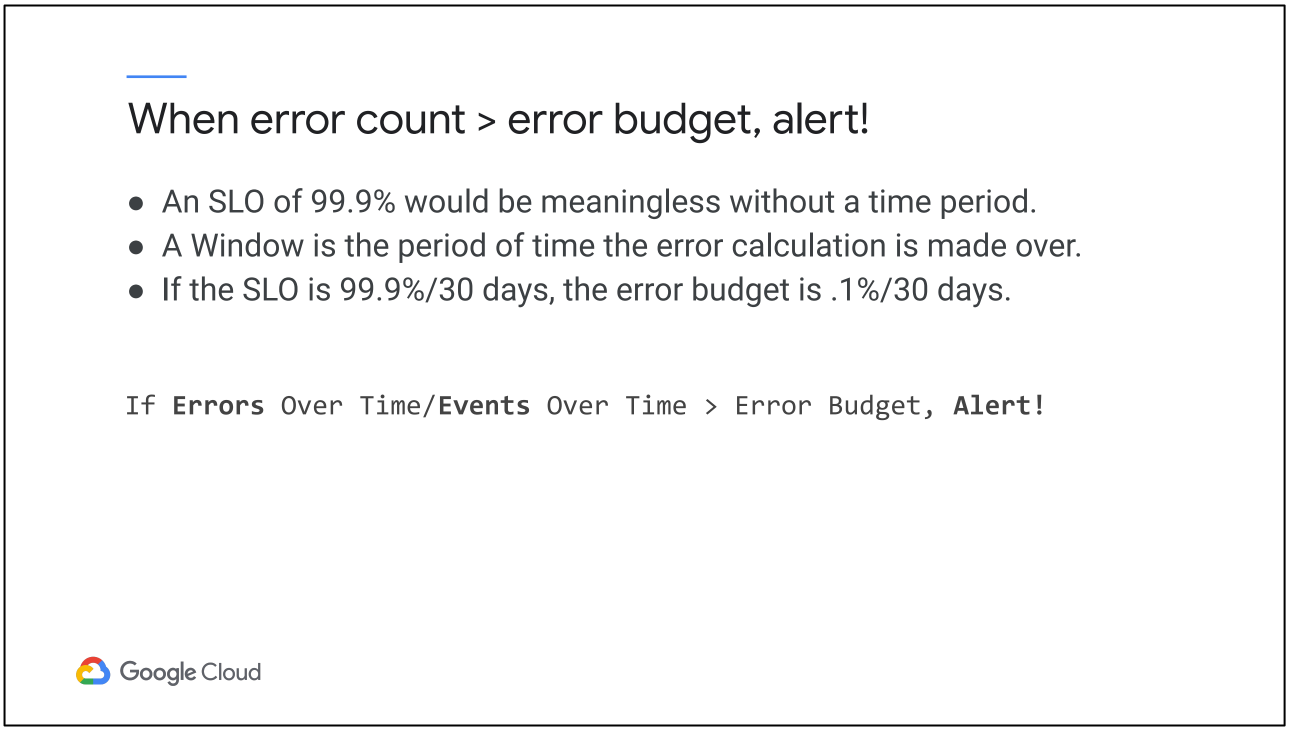
Evaluating alerts

Precision	Recall	Detection time	Reset time
$\frac{\text{Relevant alerts}}{\text{Relevant alerts + irrelevant alerts}}$	$\frac{\text{Relevant alerts}}{\text{Relevant alerts + missed alerts}}$	How long it takes the system to notice an alert condition	How long alerts fire after an issue is resolved
Adversely affected by false positives	Striving for precision may cause events to be missed	Long detection times can negatively impact the error budget	Continued alerts on repaired systems can lead to confusion
	Recall is adversely affected by missed alerts	Raising alerts too fast may result in poor precision	

Google Cloud

Detection time can be defined as how long it takes the system to notice an alert condition. Long detection times can negatively affect the error budget, but alerting too fast may generate false positives.

Reset time measures how long alerts fire after an issue has been resolved. Continued alerts on repaired systems can lead to confusion.



When error count > error budget, alert!

- An SLO of 99.9% would be meaningless without a time period.
- A Window is the period of time the error calculation is made over.
- If the SLO is 99.9%/30 days, the error budget is .1%/30 days.

```
If Errors Over Time/Events Over Time > Error Budget, Alert!
```

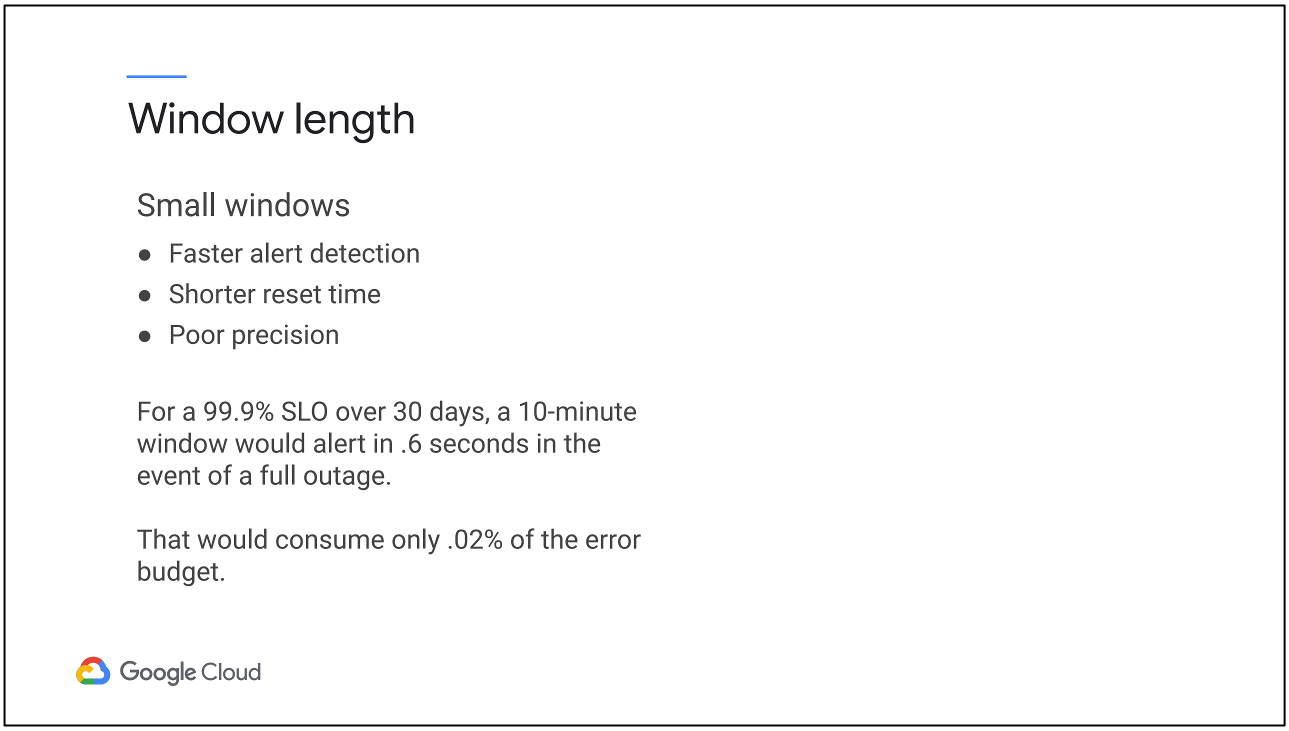
Google Cloud

Error budgeting 101 would state that when the error count is greater than the error budget, an alert should be generated.

An SLO of 99.9% would be meaningless without a time period over which errors and events were counted.

A Window is the period of time the error calculation is made over.

So if the SLO is 99.9%, and the window is 30 days, the error budget is .1% for every 30 days.



Window length

Small windows

- Faster alert detection
- Shorter reset time
- Poor precision

For a 99.9% SLO over 30 days, a 10-minute window would alert in .6 seconds in the event of a full outage.

That would consume only .02% of the error budget.

Google Cloud

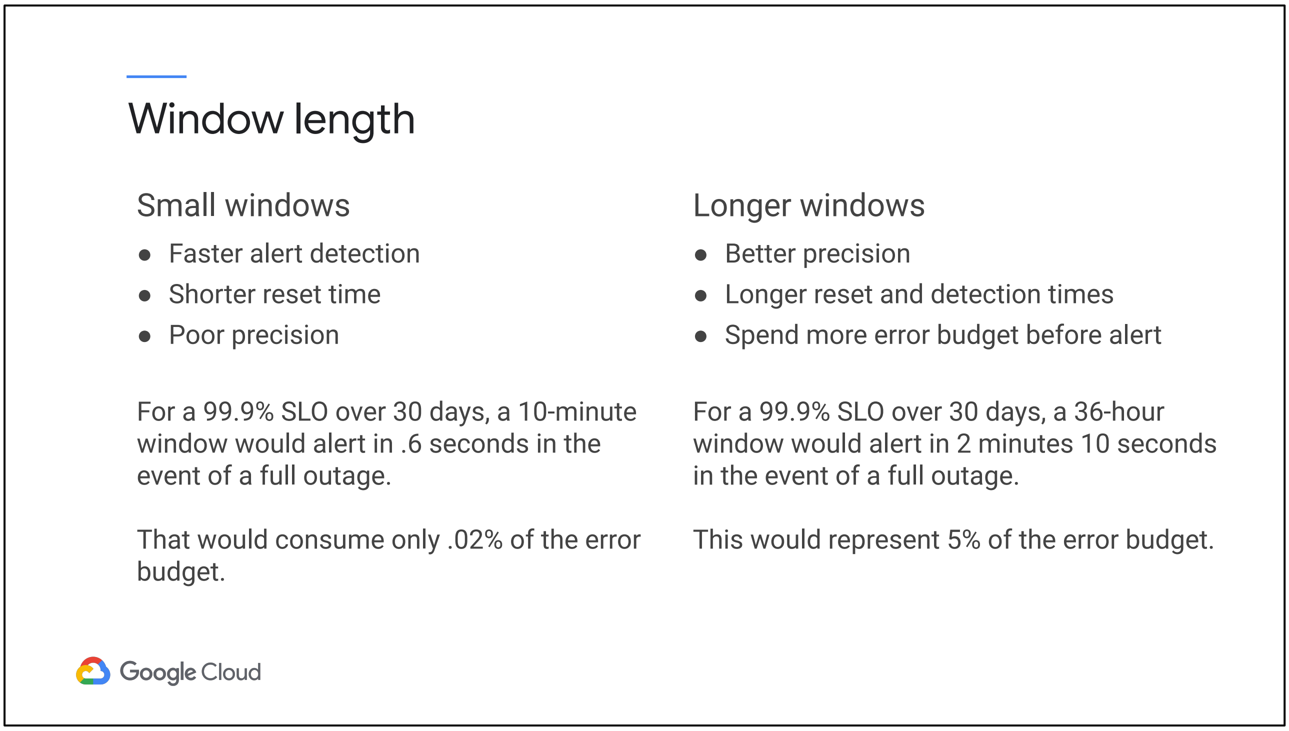
One of the alerting decisions you and your team will have to make is window length. The window is a regular-length subdivision of the SLO's total time.

Imagine you set a Google Cloud spend budget of $1,000 a month. When would you like to receive an alert: When the $1,000 is spent? Or when the predicted spend is trending past the $1,000? Of course, the latter.

Now, the same concept, but this time imagine a 99.9% SLO over 30 days. You don't want to get an alert when your error budget is already gone, because by then it's too late to do anything about the problem.

One option would be small windows. Smaller windows tend to yield faster alert detections and shorter reset times, but they also tend to decrease precision because of their tendency toward false positives.

In our 99.9% SLO over 30 days, a 10-minute window would alert in .6 seconds in the event of a full outage and would consume only .02% of the error budget.



Window length

Small windows

- Faster alert detection
- Shorter reset time
- Poor precision

For a 99.9% SLO over 30 days, a 10-minute window would alert in .6 seconds in the event of a full outage.

That would consume only .02% of the error budget.

Longer windows

- Better precision
- Longer reset and detection times
- Spend more error budget before alert

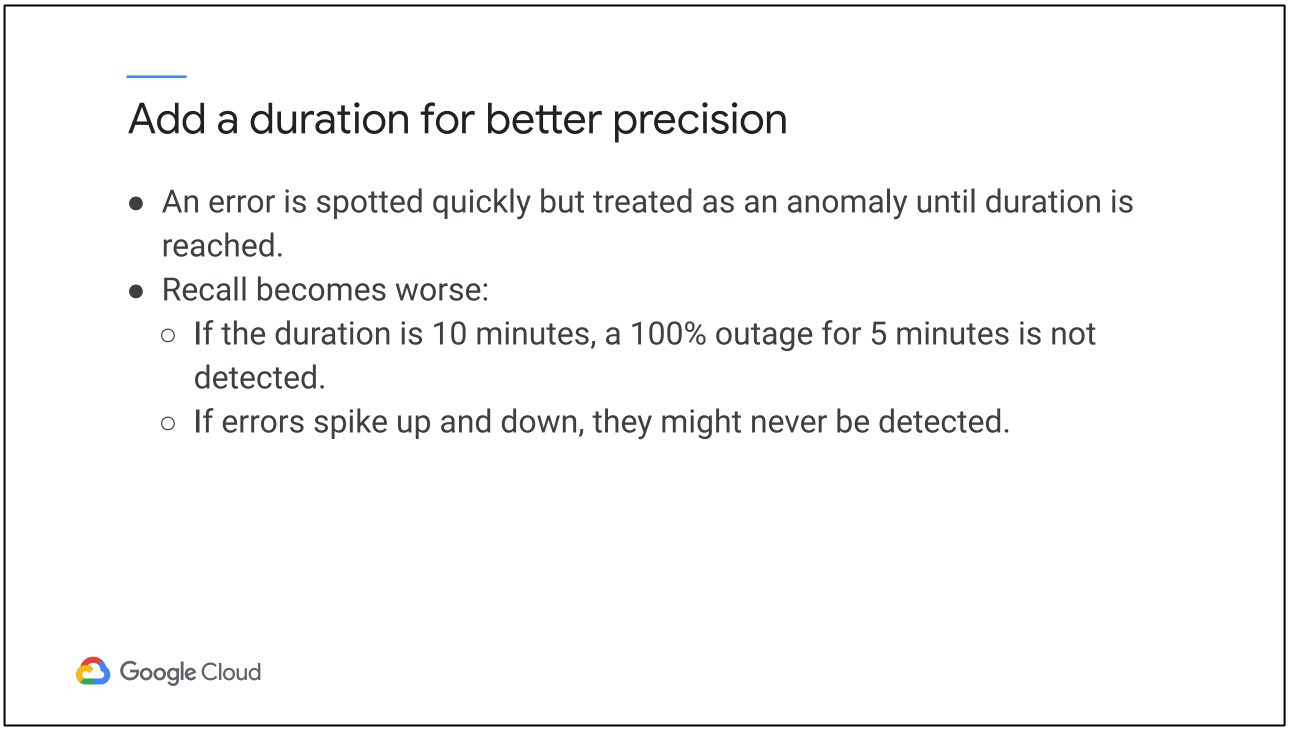
For a 99.9% SLO over 30 days, a 36-hour window would alert in 2 minutes 10 seconds in the event of a full outage.

This would represent 5% of the error budget.

Google Cloud

Longer windows tend to yield better precision, because they have longer to confirm that an error is really occurring, but reset and detection times are also longer. That means you spend more error budget before the alert triggers.

In our same 99.9% SLO over 30 days, a 36-hour window would alert in 2 minutes 10 seconds in the event of a full outage, but would consume a full 5% of the error budget.



Add a duration for better precision

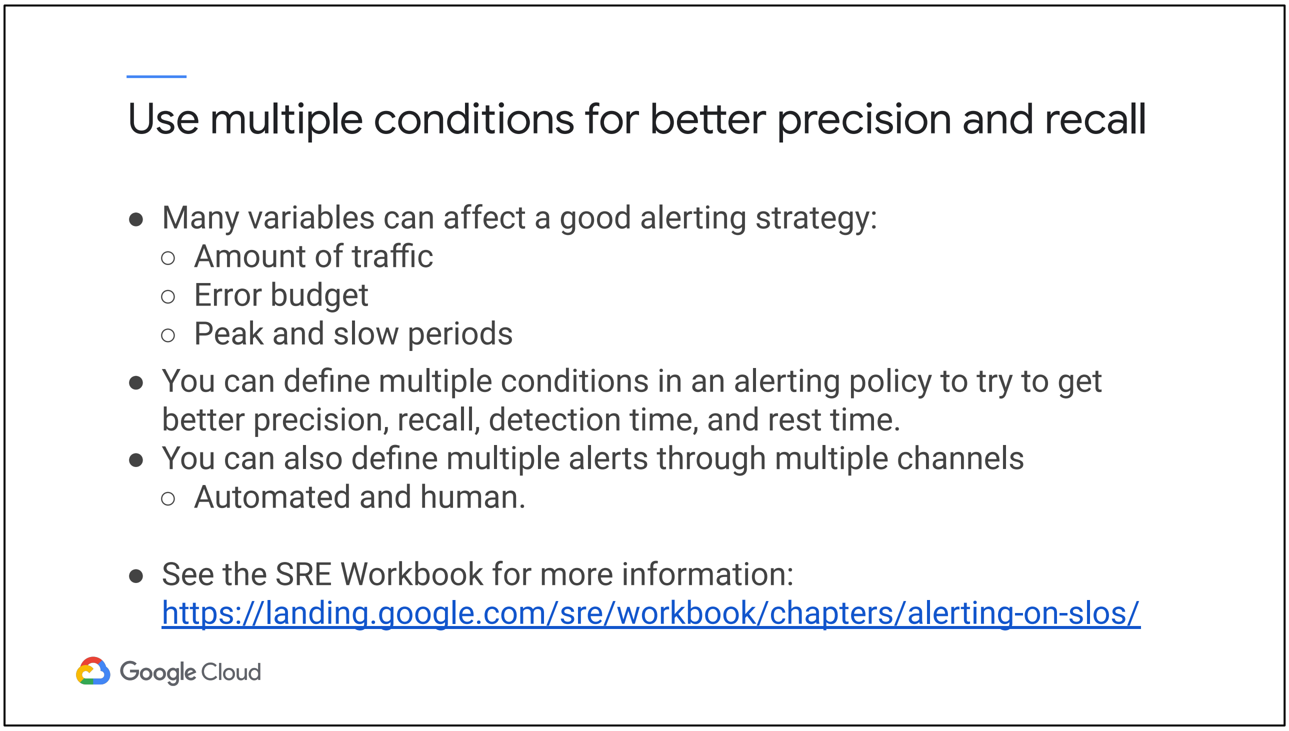
- An error is spotted quickly but treated as an anomaly until duration is reached.
- Recall becomes worse:
  - If the duration is 10 minutes, a 100% outage for 5 minutes is not detected.
  - If errors spike up and down, they might never be detected.

Google Cloud

One step toward faster detection and higher precision is the addition of a duration. The error is spotted quickly but treated as an anomaly until the duration reached. This is what you do when your car starts making a sound. You don't immediately freak out, but you pay attention and try to determine whether it's a real issue or a fluke.

The downside is that precision typically has an inverse relationship to recall. As the precision goes up, as you avoid false positives, you let the problem continue to happen.

If the "pay attention but don't alert yet" duration is 10 minutes, a 100% outage for 5 minutes would not be detected. As a result, if errors spike up and down, they may never be detected.



Use multiple conditions for better precision and recall

- Many variables can affect a good alerting strategy:
  - Amount of traffic
  - Error budget
  - Peak and slow periods
- You can define multiple conditions in an alerting policy to try to get better precision, recall, detection time, and rest time.
- You can also define multiple alerts through multiple channels
  - Automated and human.

- See the SRE Workbook for more information: https://landing.google.com/sre/workbook/chapters/alerting-on-slos/

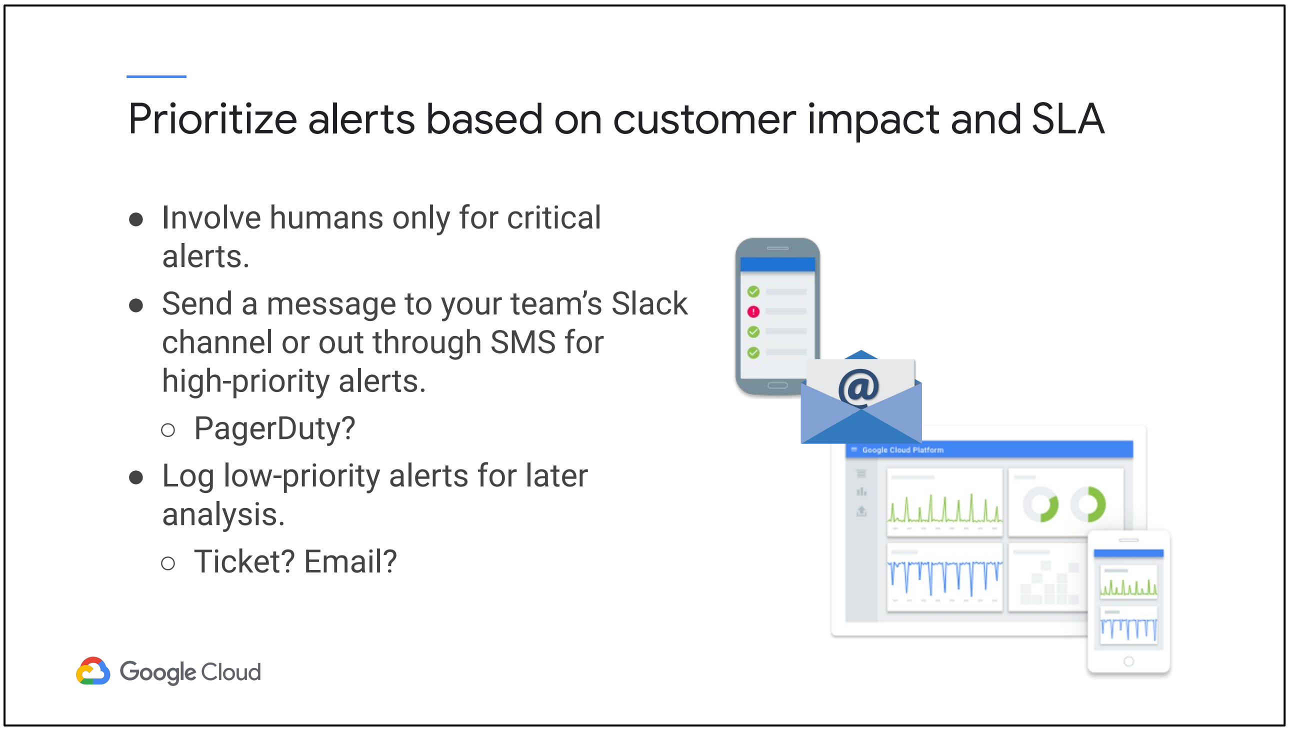
Google Cloud

So how do we get good precision and good recall? Multiple Conditions.

There are many variables that affect a good alerting strategy, including the amount of traffic, the error budget, peak and slow periods, to name a few.

The fallacy is believing that you have to choose a single option. Define multiple conditions in an alerting policy to get better precision, recall, detection time, and rest time.

You can also define multiple alerts through multiple channels. Perhaps a short window condition generates an alert, but it takes the form of a Pub/Sub message to a Google Cloud Run container, which then uses complex logic to check multiple other conditions before deciding if a human gets a notification.



Prioritize alerts based on customer impact and SLA

- Involve humans only for critical alerts.
- Send a message to your team's Slack channel or out through SMS for high-priority alerts.
  - PagerDuty?
- Log low-priority alerts for later analysis.
  - Ticket? Email?

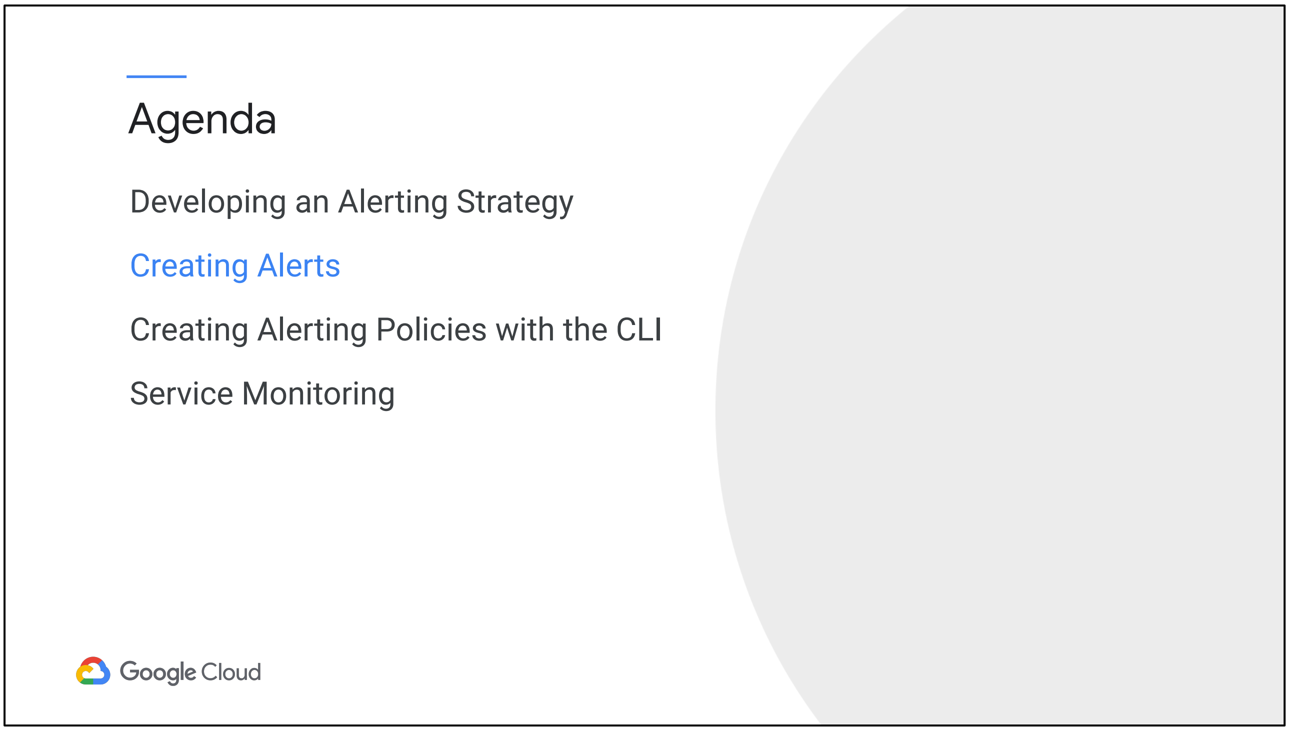
Google Cloud

And alerts should always be prioritized based on customer impact and SLA.

Don't involve the humans unless the alert meets some threshold for criticality.

High priority alerts might go to Slack, SMS, or maybe even a third-party solution like PagerDuty.

Low-priority alerts might be logged, sent through email, or inserted into a support ticket management system.



Agenda

Developing an Alerting Strategy

Creating Alerts

Creating Alerting Policies with the CLI

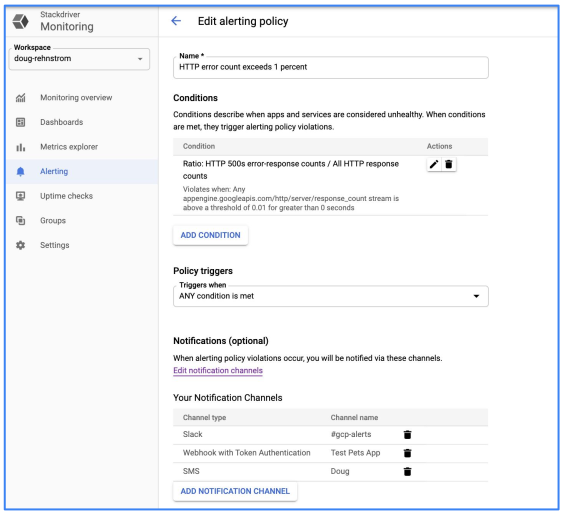
Service Monitoring

Google Cloud

Okay, we've discussed some of the alerting concepts and strategies. Now, let's run through the Google Cloud mechanics of creating alerts.

Use alerting policies to define alerts

- An alerting policy has:
  - A name
  - One or more conditions
  - Notifications
  - Documentation



Workspace: doug-rehnstrom

Edit alerting policy

Name *: HTTP error count exceeds 1 percent

Conditions

Conditions describe when apps and services are considered unhealthy. When conditions are met, they trigger alerting policy violations.

Condition	Actions
Ratio: HTTP 500s error-response counts / All HTTP response counts Violates when: Any appengine.googleapis.com/http/server/response_count stream is above a threshold of 0.01 for greater than 0 seconds	

Policy triggers

Triggers when: ANY condition is met

Notifications (optional)

When alerting policy violations occur, you will be notified via these channels.
Edit notification channels

Your Notification Channels

Channel type	Channel name
Slack	#gcp-alerts
Webhook with Token Authentication	Test Pets App
SMS	Doug

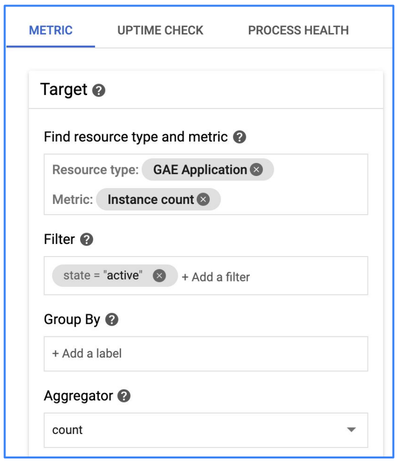
Google Cloud

Google Cloud defines alerts using Alerting Policies.

An alerting policy has:
A name
One or more alert conditions
Notifications
And documentation

For the name, use something descriptive so you can recognize alerts after the fact. Organizational naming conventions can be a great help.

Conditions: What's watched and when to alert


METRIC | UPTIME CHECK | PROCESS HEALTH

Target

Find resource type and metric

Resource type: GAE Application

Metric: Instance count

Filter

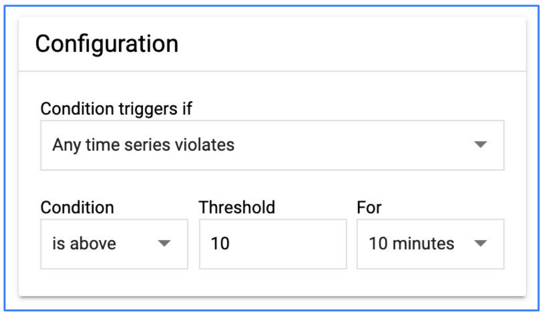
state = "active"   + Add a filter

Group By

+ Add a label

Aggregator

count


Configuration

Condition triggers if

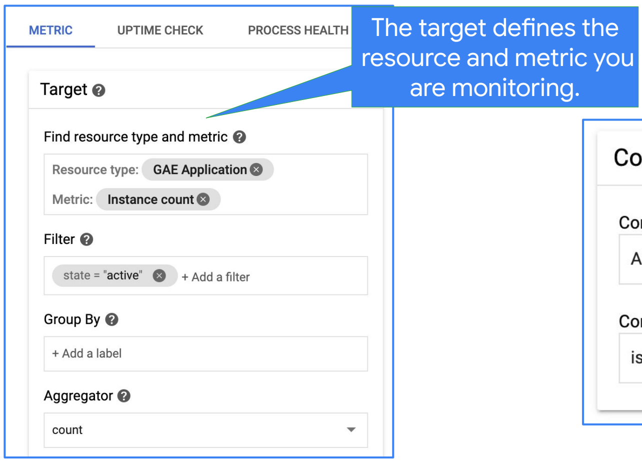
Any time series violates

Condition	Threshold	For
is above	10	10 minutes

Google Cloud

The alert condition is where you'll be spending the most alerting policy time and making the most decisions. This is where you decide what's being monitored and under what condition an alert should be generated.

Conditions: What's watched and when to alert


METRIC
UPTIME CHECK
PROCESS HEALTH
The target defines the resource and metric you are monitoring.

Target

Find resource type and metric

Resource type: GAE Application

Metric: Instance count

Filter

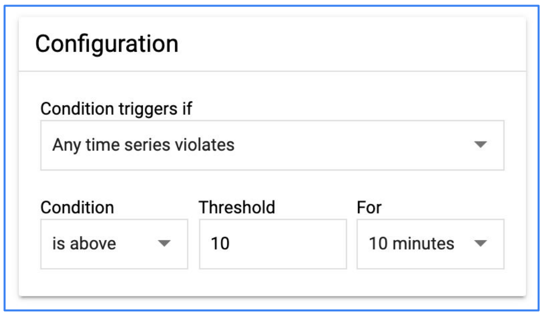
state = "active" + Add a filter

Group By

+ Add a label

Aggregator

count



Configuration

Condition triggers if

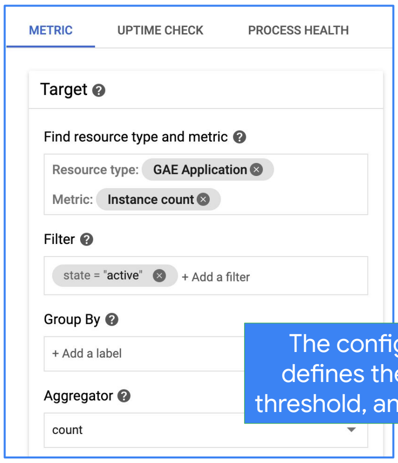
Any time series violates

Condition	Threshold	For
is above	10	10 minutes

Google Cloud

You start with a target resource and metric you want the alert to monitor. You can filter, group by, and aggregate to the exact measure you require.

Conditions: What's watched and when to alert


METRIC | UPTIME CHECK | PROCESS HEALTH

Target

Find resource type and metric

Resource type: GAE Application

Metric: Instance count

Filter

state = "active" + Add a filter

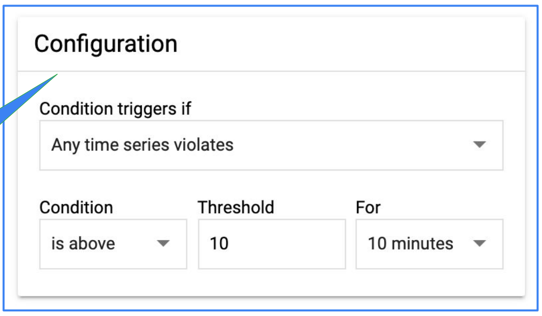
Group By

+ Add a label

Aggregator

count

The configuration defines the trigger, threshold, and duration.


Configuration

Condition triggers if

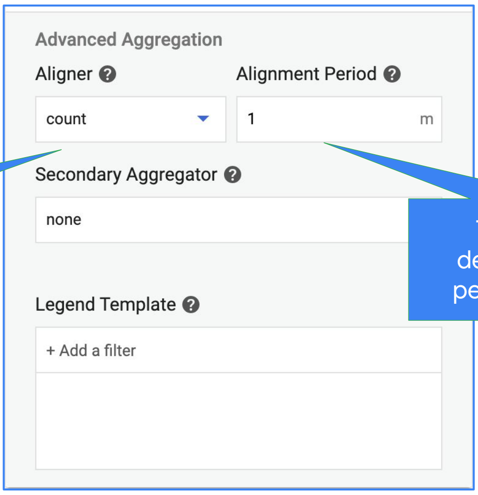
Any time series violates

Condition	Threshold	For
is above	10	10 minutes

Google Cloud

Then the yes-no decision logic for triggering the alert notification is configured. It includes the trigger condition, threshold, and duration.

Set aligner, period, and secondary aggregator


Advanced Aggregation

Aligner: count

Alignment Period: 1 m

Secondary Aggregator: none

Legend Template

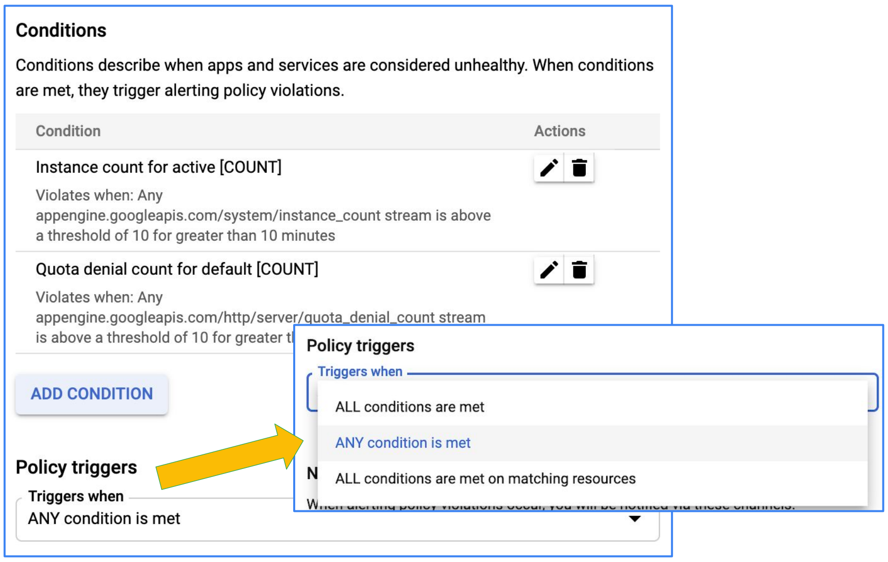
Aligners include count, min, max, mean, and sum.

The alignment period determines how often to perform the aggregation.

Google Cloud

If needed, you can set an aligner, alignment period, and even a secondary aggregator. Aligners are the math applied to each alignment period in the time series. The alignment period determines how frequently the aligner is applied.

Use multiple conditions


Conditions

Conditions describe when apps and services are considered unhealthy. When conditions are met, they trigger alerting policy violations.

Condition	Actions
Instance count for active [COUNT] Violates when: Any appengine.googleapis.com/system/instance_count stream is above a threshold of 10 for greater than 10 minutes	
Quota denial count for default [COUNT] Violates when: Any appengine.googleapis.com/http/server/quota_denial_count stream is above a threshold of 10 for greater t	

ADD CONDITION

Policy triggers

Triggers when

ANY condition is met

Policy triggers

Triggers when

- ALL conditions are met
- ANY condition is met
- ALL conditions are met on matching resources

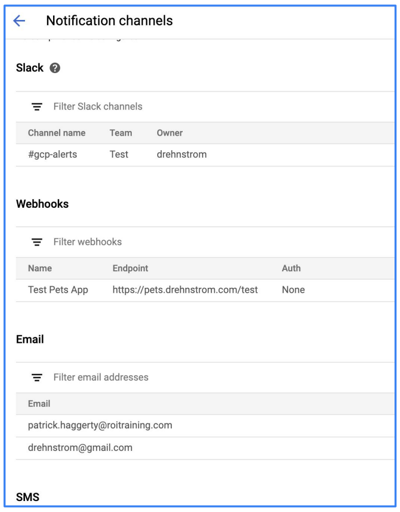
Google Cloud

To try to maximize both precision and recall within a single alert, you can create multiple conditions. The policy trigger is used to determine how more than one trigger will relate to one another and to the alert triggering itself.

Select notification channels

Supported notification channels include:

- Email
- SMS
- Slack
- GCP Mobile app
- PagerDuty
- Webhooks
- Pub/Sub



Notification channels

Slack

Filter Slack channels

Channel name	Team	Owner
#gcp-alerts	Test	drehnstrom

Webhooks

Filter webhooks

Name	Endpoint	Auth
Test Pets App	https://pets.drehnstrom.com/test	None

Email

Filter email addresses

Email
patrick.haggerty@roitraining.com
drehnstrom@gmail.com

SMS

Google Cloud

The notification channel, or channels, decides how the alert is sent to the recipient.

Email alerts are easy and informative, but they can become notification spam if you aren't careful.

SMS is a great option for fast notifications, but choose the recipient carefully.

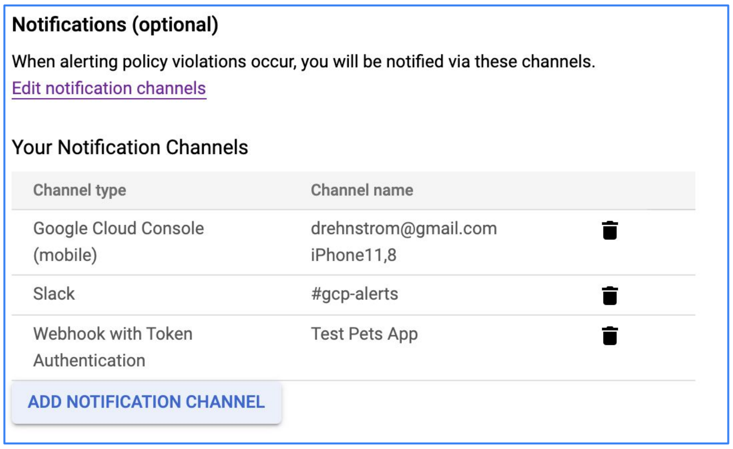
Slack is very popular in support circles.

Google Cloud's mobile app is a valid option.

PagerDuty is a third party on-call management and incident response service.

Webhooks and Pub/Sub are excellent options when alerting to external systems or code.

Zero to many notification channels



Notifications (optional)

When alerting policy violations occur, you will be notified via these channels.
Edit notification channels

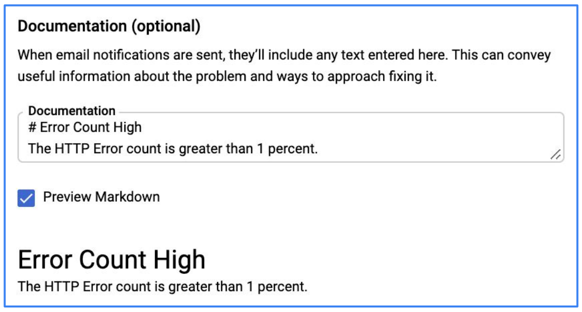
Your Notification Channels

Channel type	Channel name	
Google Cloud Console (mobile)	drehnstrom@gmail.com iPhone11,8	🗑
Slack	#gcp-alerts	🗑
Webhook with Token Authentication	Test Pets App	🗑

An alert may have zero to many notification options selected, and they each can be of a different type.

Include documentation for added clarity

- Make it easy for the team to understand what is wrong.
- Use markdown to format messages.


Documentation (optional)

When email notifications are sent, they'll include any text entered here. This can convey useful information about the problem and ways to approach fixing it.

Documentation
```
# Error Count High
The HTTP Error count is greater than 1 percent.
```

☑ Preview Markdown

Error Count High

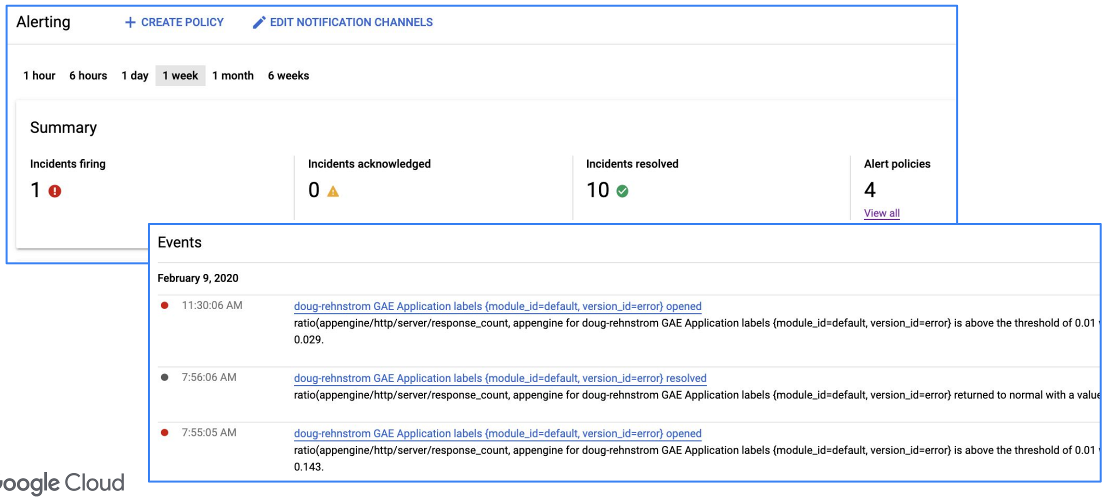
The HTTP Error count is greater than 1 percent.

Google Cloud

The documentation option is designed to give the alert recipient additional information they might find helpful. The default alert content will already contain information about which alert is failing and why, so think of this more like an easy button. If there's a standard solution to this particular alert, adding a reference to it here might be a good example of proper documentation inclusion.

Then again, if it was that easy, automate it!

Alerting UI summarizes incidents and events



Alerting　+ CREATE POLICY　EDIT NOTIFICATION CHANNELS

1 hour　6 hours　1 day　**1 week**　1 month　6 weeks

Summary

Incidents firing	Incidents acknowledged	Incidents resolved	Alert policies
1 ●	0 ▲	10 ✓	4 View all

Events

February 9, 2020

● 11:30:06 AM　doug-rehnstrom GAE Application labels {module_id=default, version_id=error} opened
ratio(appengine/http/server/response_count, appengine for doug-rehnstrom GAE Application labels {module_id=default, version_id=error} is above the threshold of 0.01
0.029.

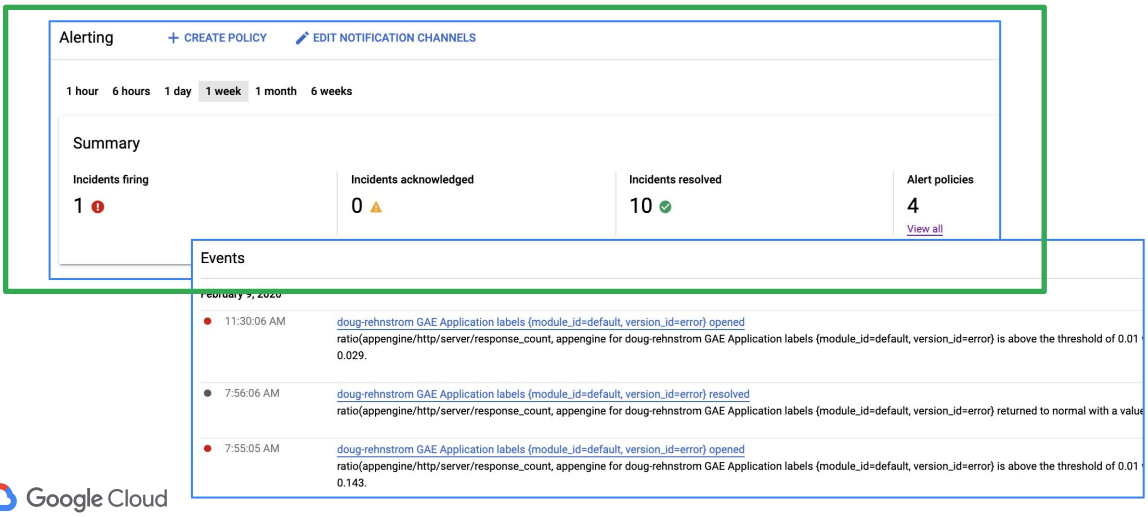
● 7:56:06 AM　doug-rehnstrom GAE Application labels {module_id=default, version_id=error} resolved
ratio(appengine/http/server/response_count, appengine for doug-rehnstrom GAE Application labels {module_id=default, version_id=error} returned to normal with a value

● 7:55:05 AM　doug-rehnstrom GAE Application labels {module_id=default, version_id=error} opened
ratio(appengine/http/server/response_count, appengine for doug-rehnstrom GAE Application labels {module_id=default, version_id=error} is above the threshold of 0.01
0.143.

Google Cloud

When one or more alert policies have been created, the Alerting UI provides a summary of incidents and alerting events. An event occurs when the conditions for an alerting policy are met. When an event occurs, Cloud Monitoring opens an incident.

Alerting UI summarizes incidents and events



Alerting + CREATE POLICY EDIT NOTIFICATION CHANNELS

1 hour 6 hours 1 day 1 week 1 month 6 weeks

Summary

Incidents firing	Incidents acknowledged	Incidents resolved	Alert policies
1	0	10	4 View all

Events

February 9, 2020

● 11:30:06 AM doug-rehnstrom GAE Application labels {module_id=default, version_id=error} opened
ratio(appengine/http/server/response_count, appengine for doug-rehnstrom GAE Application labels {module_id=default, version_id=error} is above the threshold of 0.01
0.029.

● 7:56:06 AM doug-rehnstrom GAE Application labels {module_id=default, version_id=error} resolved
ratio(appengine/http/server/response_count, appengine for doug-rehnstrom GAE Application labels {module_id=default, version_id=error} returned to normal with a value

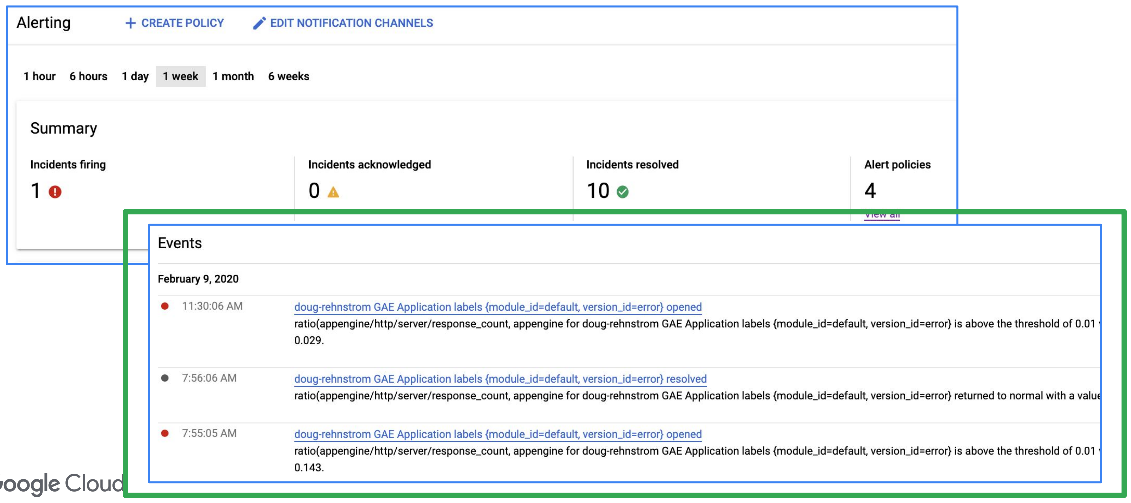
● 7:55:05 AM doug-rehnstrom GAE Application labels {module_id=default, version_id=error} opened
ratio(appengine/http/server/response_count, appengine for doug-rehnstrom GAE Application labels {module_id=default, version_id=error} is above the threshold of 0.01
0.143.

Google Cloud

In the Alerting window, the Summary pane lists the number of incidents, and the Incidents pane displays the 10 most recent incidents. Each incident is in one of three states:

- Open incidents. If an incident is open, the alerting policy's set of conditions is currently being met, or there is no data to indicate that the condition is no longer met. This usually indicates a new or unhandled alert.
- Acknowledged incidents. A tech spots a new open alert, but before they start to investigate, they mark it as acknowledged as a signal to others that someone is dealing with the issue.
- Closed incidents. If an incident is closed, the alert policy conditions are no longer being met. An incident is listed as closed if there is no data to indicate whether the condition still exists and the incident has expired.

Alerting UI summarizes incidents and events



Summary

Incidents firing	Incidents acknowledged	Incidents resolved	Alert policies
1	0	10	4

Events

February 9, 2020

● 11:30:06 AM doug-rehnstrom GAE Application labels {module_id=default, version_id=error} opened
ratio(appengine/http/server/response_count, appengine for doug-rehnstrom GAE Application labels {module_id=default, version_id=error} is above the threshold of 0.01
0.029.

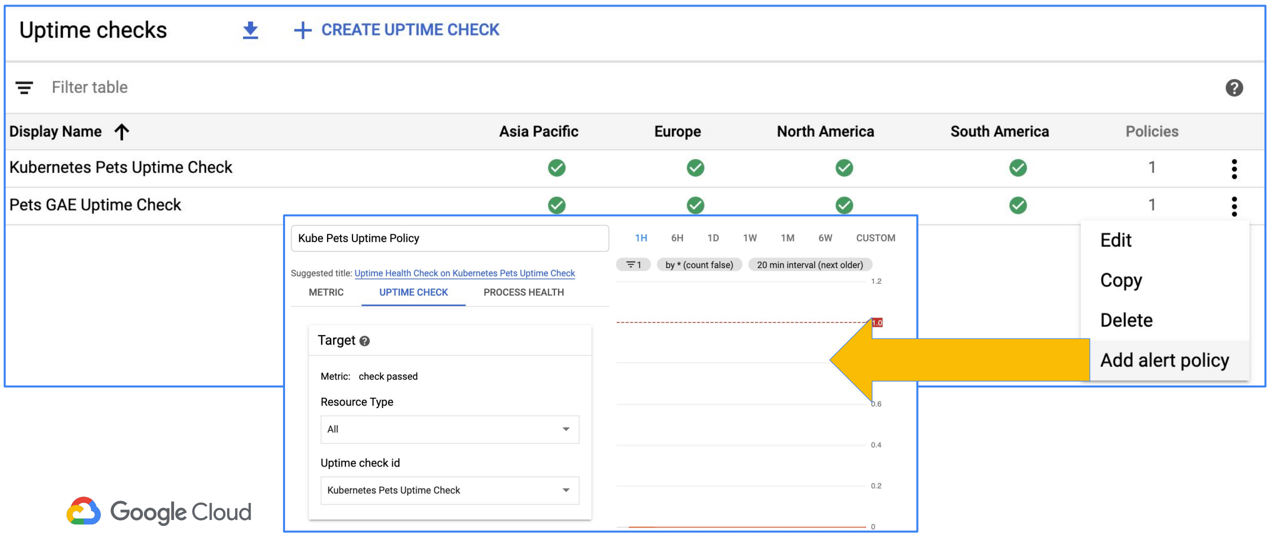
● 7:56:06 AM doug-rehnstrom GAE Application labels {module_id=default, version_id=error} resolved
ratio(appengine/http/server/response_count, appengine for doug-rehnstrom GAE Application labels {module_id=default, version_id=error} returned to normal with a value

● 7:55:05 AM doug-rehnstrom GAE Application labels {module_id=default, version_id=error} opened
ratio(appengine/http/server/response_count, appengine for doug-rehnstrom GAE Application labels {module_id=default, version_id=error} is above the threshold of 0.01
0.143.

Google Cloud

The Events pane of the Alerting dashboard displays the most recent events and includes a graphical indicator of the alert status, a link to event details, a quick description, and a timestamp.

Attach alerts to uptime checks



Display Name	Asia Pacific	Europe	North America	South America	Policies
Kubernetes Pets Uptime Check	✓	✓	✓	✓	1
Pets GAE Uptime Check	✓	✓	✓	✓	1

Kube Pets Uptime Policy

Suggested title: Uptime Health Check on Kubernetes Pets Uptime Check

Target

Metric: check passed

Resource Type: All

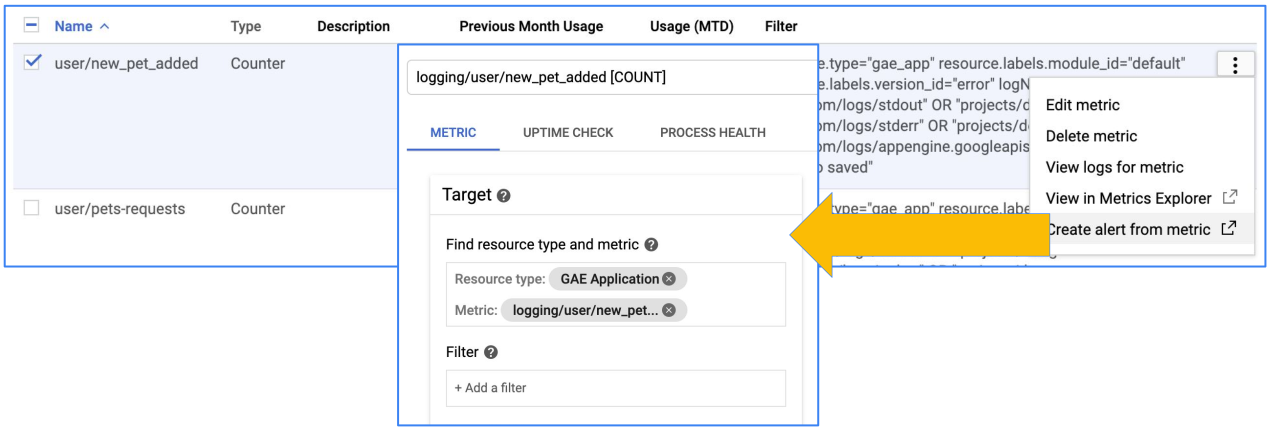
Uptime check id: Kubernetes Pets Uptime Check

Google Cloud

An uptime check is a request sent to an externally accessible site or service to see if it responds, or is up. You can use uptime checks to determine the availability and latency of a VM instance, an App Engine service, a URL, or an AWS load balancer.

You can monitor the availability of a resource by creating an alerting policy that creates an incident if the uptime check fails. You also have the option to observe the results of uptime checks in the Monitoring uptime-check dashboards.

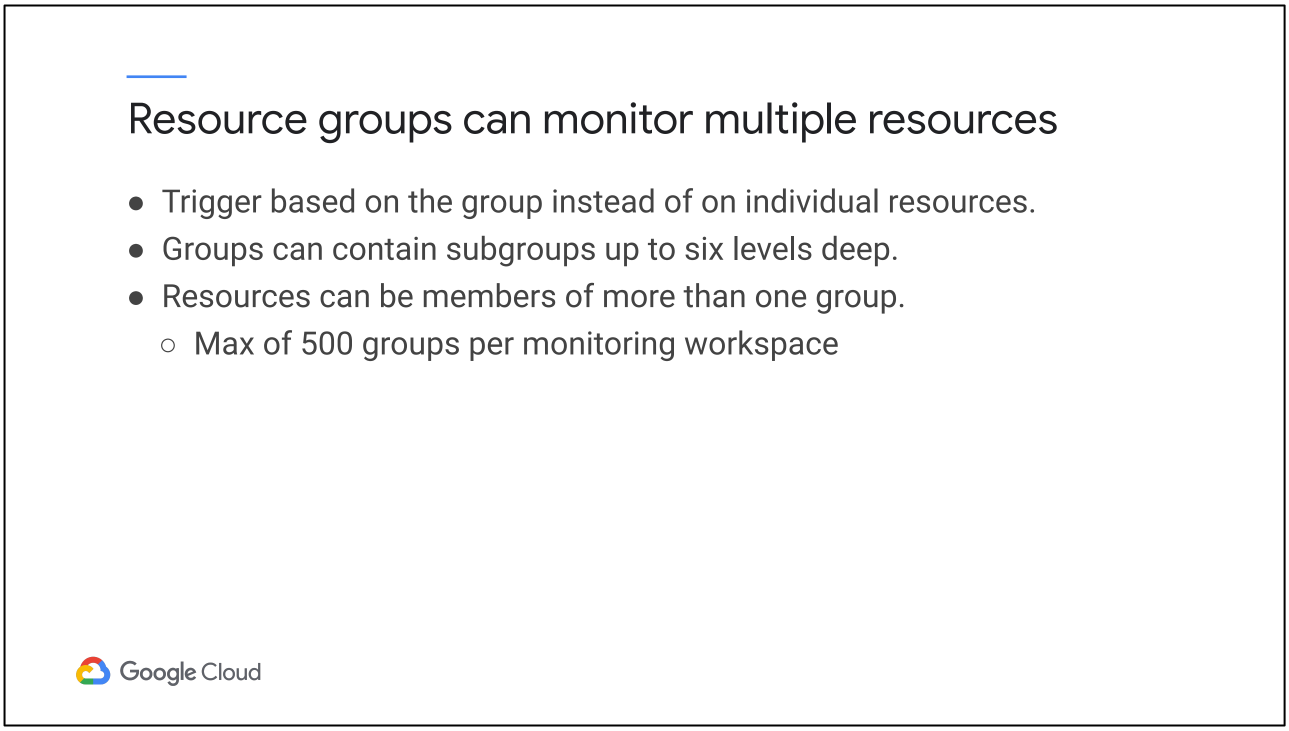
Attach alerts to logs-based metrics



Google Cloud

Logs-based metrics are Cloud Monitoring metrics based on the content of log entries. For example, the metrics can record the number of log entries containing particular messages, or they can extract latency information reported in log entries. You can use logs-based metrics in Cloud Monitoring charts and alerting policies.

As we covered earlier in this module, an alerting policy describes a set of conditions that you want to monitor. When you create an alerting policy, you must also specify its conditions: what is monitored and when to trigger an alert. The logs-based metric serves as the basis for an alerting condition.


Resource groups can monitor multiple resources

- Trigger based on the group instead of on individual resources.
- Groups can contain subgroups up to six levels deep.
- Resources can be members of more than one group.
  - Max of 500 groups per monitoring workspace

Google Cloud

Groups provide a mechanism for alerting on the behavior of a set of resources, rather than on individual resources. For example, you can create an alerting policy that is triggered if some number of resources in the group violates a particular condition (for example, CPU load), rather than having each resource inform you of violations individually.

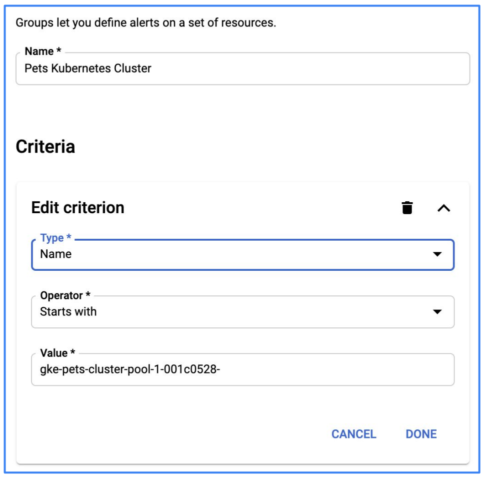
Groups can contain subgroups, up to six levels deep. One application for groups and subgroups is the management of physical or logical topologies. For example, with groups, you can separate your monitoring of production resources from your monitoring of test or development resources. You can also create subgroups to monitor your production resources by zone

Resources can belong to multiple groups, and a given monitoring workspace can have up to 500 groups.

Use multiple criteria to create resource groups

Criteria can include:

- Resource name
- Resource type
- Tags and labels
- Security groups
- Regions
- App Engine apps and services



Groups let you define alerts on a set of resources.

Name *
Pets Kubernetes Cluster

Criteria

Edit criterion

Type *
Name

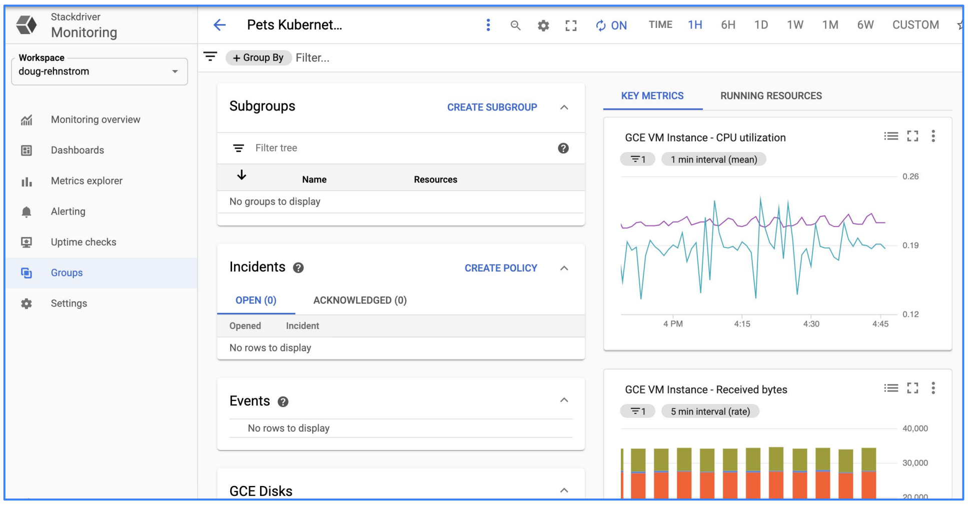
Operator *
Starts with

Value *
gke-pets-cluster-pool-1-001c0528-

Google Cloud

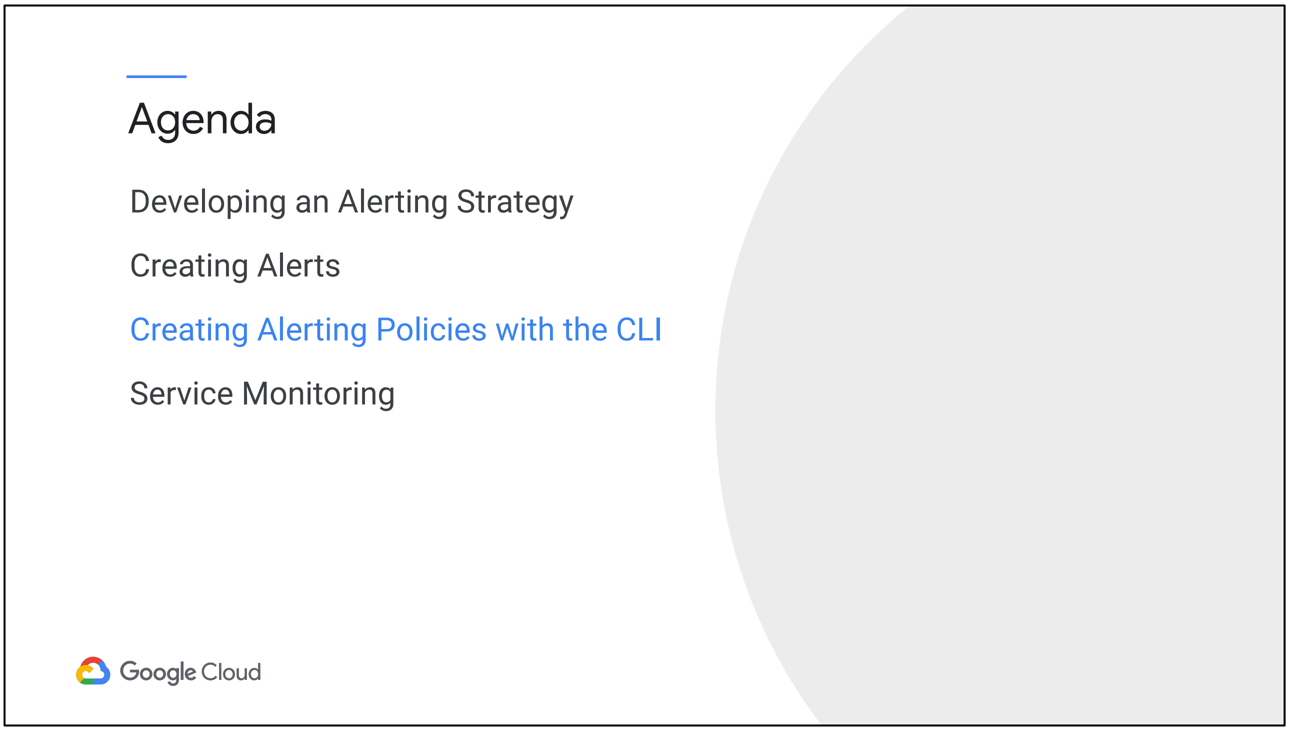
You define the one-to-many membership criteria for your groups. A resource belongs to a group if the resource meets the membership criteria of the group. Membership criteria can be based on resource name or type, network tag, resource label, security group, region, or App Engine app or service. Resources can belong to multiple groups.

Monitor all resources in a group together



Google Cloud

After the group is created, all the resources in the group can be monitored together as a unit.



Agenda

Developing an Alerting Strategy

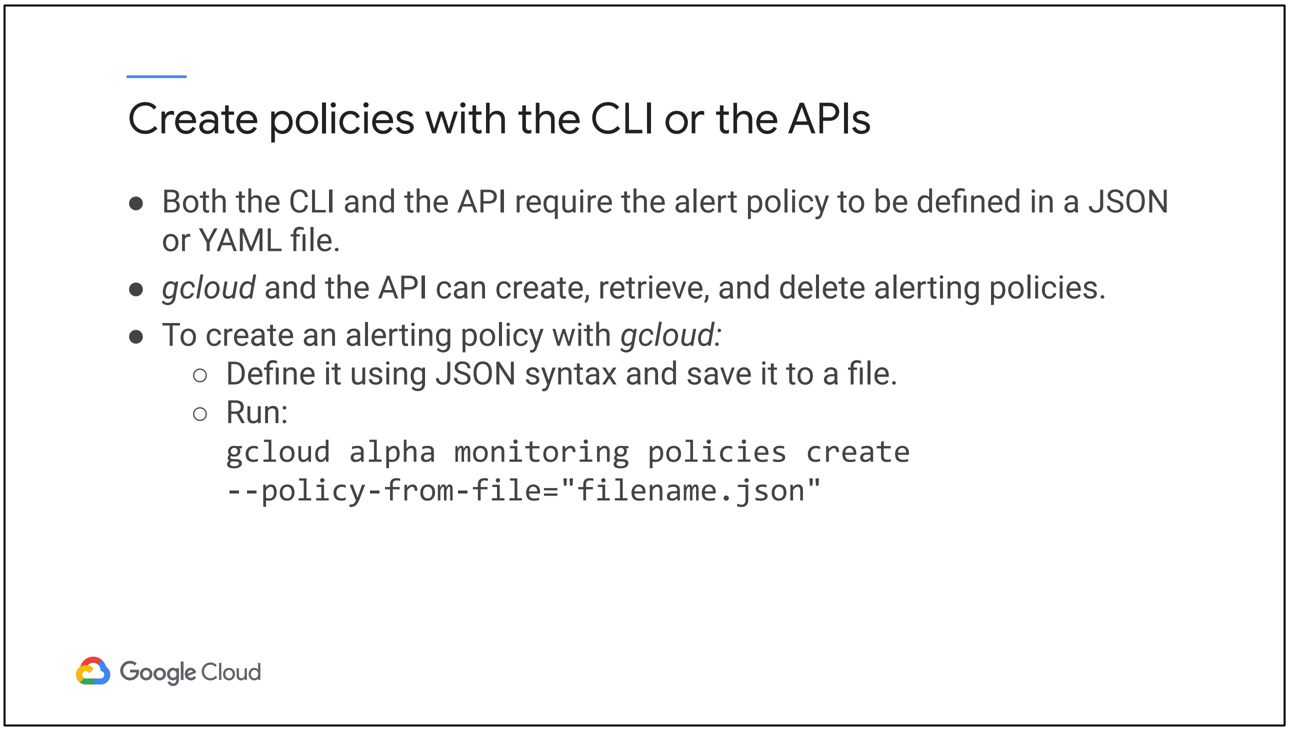
Creating Alerts

Creating Alerting Policies with the CLI

Service Monitoring

Google Cloud

Alerting policies can be created both in the Google Cloud Console and by using the CLI or API.



Create policies with the CLI or the APIs

- Both the CLI and the API require the alert policy to be defined in a JSON or YAML file.
- *gcloud* and the API can create, retrieve, and delete alerting policies.
- To create an alerting policy with *gcloud:*
  - Define it using JSON syntax and save it to a file.
  - Run:
    ```
    gcloud alpha monitoring policies create
    --policy-from-file="filename.json"
    ```

Google Cloud

https://cloud.google.com/sdk/gcloud/reference/alpha/monitoring/channels/create

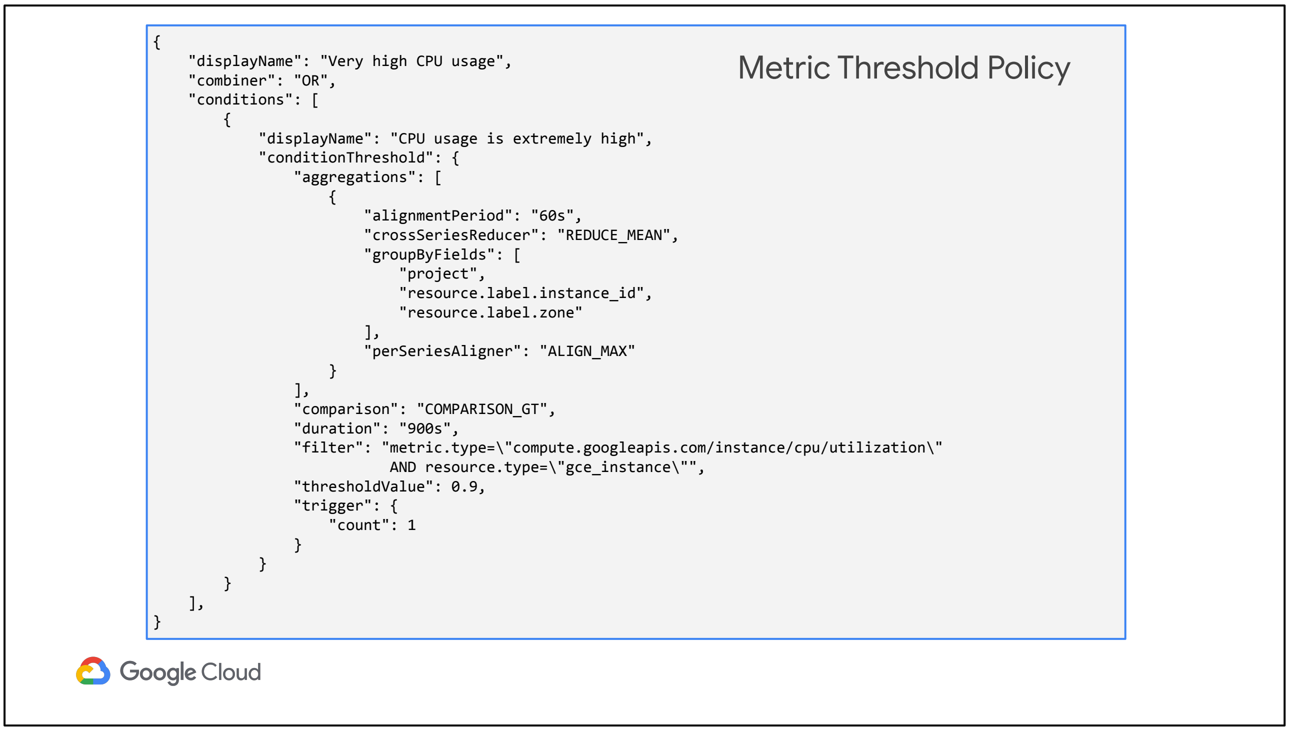
Creating alerts from the CLI or the API starts with an alert policy definition in either a JSON or YAML format. One neat learning trick when learning the correct file format is to create an alert using the Google Cloud console, then use the *gcloud monitoring policies list* and then *describe* commands to see the corresponding definition file.

The alerting API and *gcloud* can create, retrieve, and delete alerting policies.

As an example, to create an alerting *gcloud*

Define it using JSON syntax and save to a file

Then run:
gcloud alpha monitoring policies create --policy-from-file="filename.json"



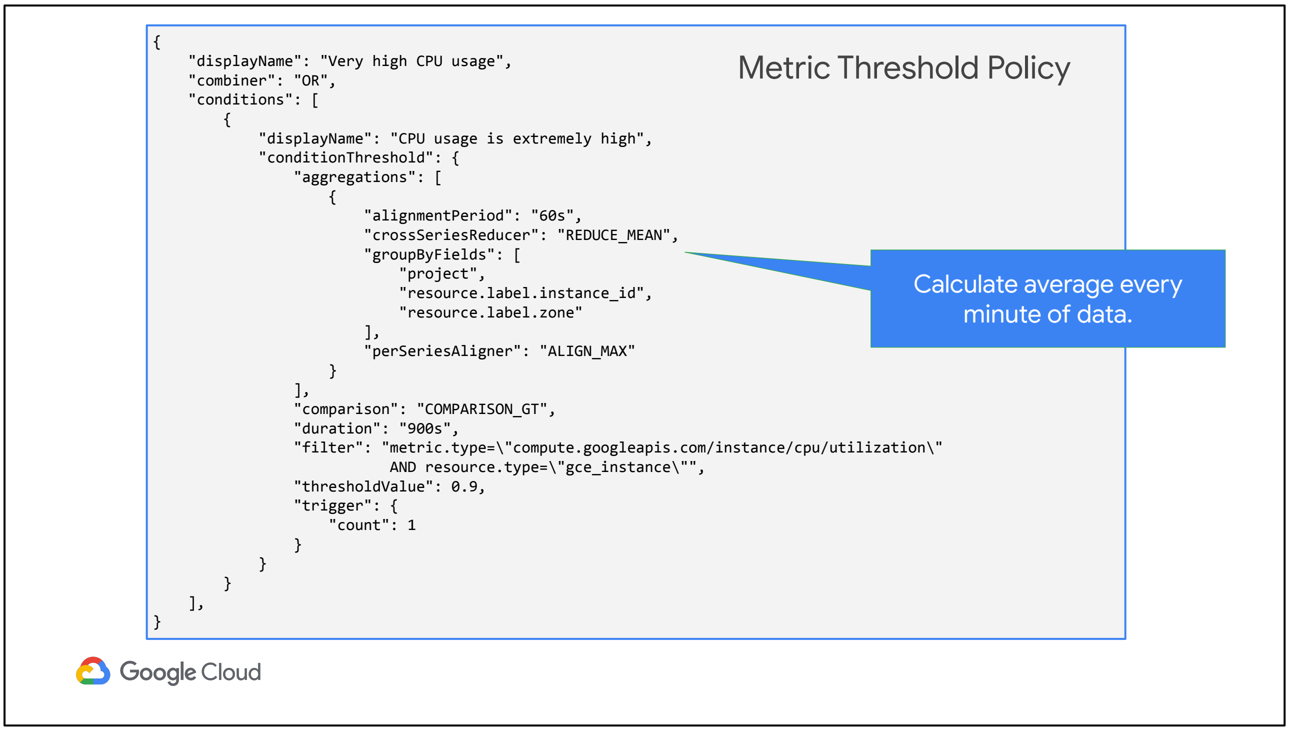
Metric Threshold Policy

```
{
    "displayName": "Very high CPU usage",
    "combiner": "OR",
    "conditions": [
        {
            "displayName": "CPU usage is extremely high",
            "conditionThreshold": {
                "aggregations": [
                    {
                        "alignmentPeriod": "60s",
                        "crossSeriesReducer": "REDUCE_MEAN",
                        "groupByFields": [
                            "project",
                            "resource.label.instance_id",
                            "resource.label.zone"
                        ],
                        "perSeriesAligner": "ALIGN_MAX"
                    }
                ],
                "comparison": "COMPARISON_GT",
                "duration": "900s",
                "filter": "metric.type=\"compute.googleapis.com/instance/cpu/utilization\"
                          AND resource.type=\"gce_instance\"",
                "thresholdValue": 0.9,
                "trigger": {
                    "count": 1
                }
            }
        }
    ],
}
```

Google Cloud

Now, let's examine some policy file examples. Our first example is a metric threshold policy.

A metric threshold policy detects when some value crosses a specified boundary. Threshold policies let you know that something is approaching an important point, so you can take some action. For example, when available disk space falls below 10 percent of total disk space, your system may soon run out of disk space.



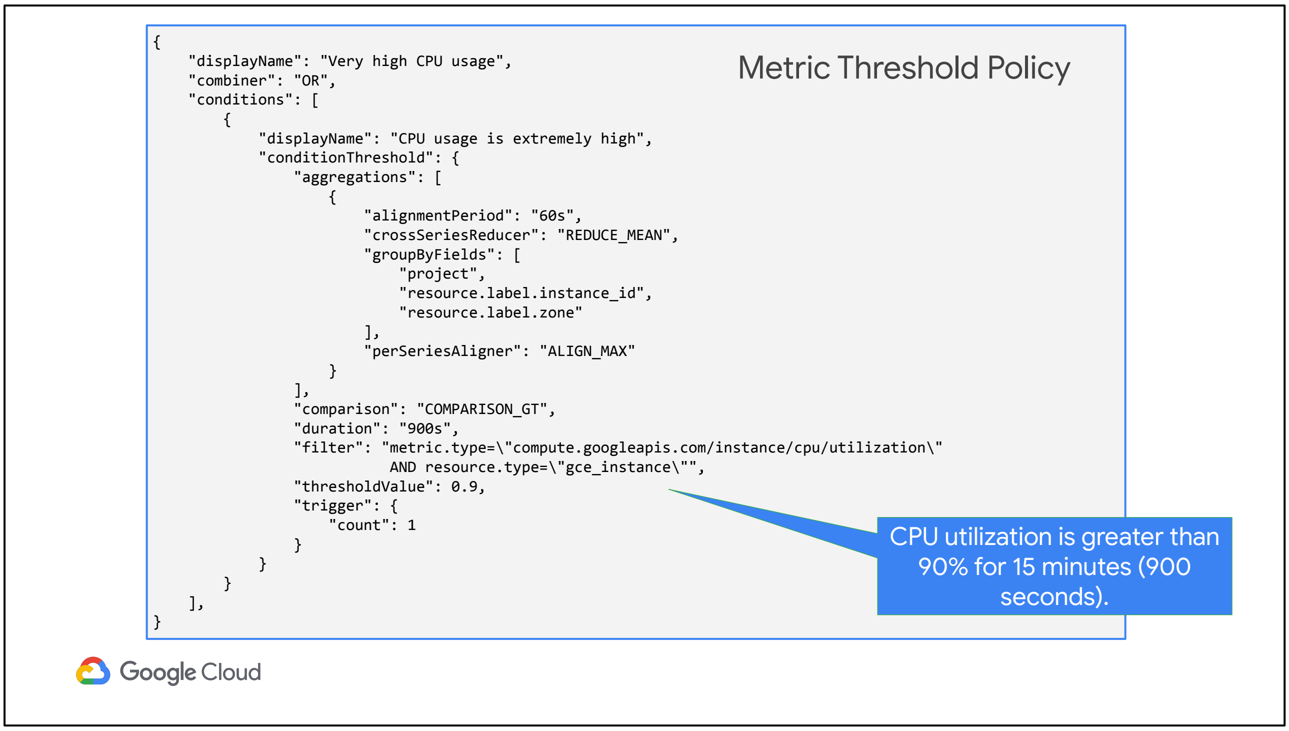
Metric Threshold Policy

```
{
    "displayName": "Very high CPU usage",
    "combiner": "OR",
    "conditions": [
        {
            "displayName": "CPU usage is extremely high",
            "conditionThreshold": {
                "aggregations": [
                    {
                        "alignmentPeriod": "60s",
                        "crossSeriesReducer": "REDUCE_MEAN",
                        "groupByFields": [
                            "project",
                            "resource.label.instance_id",
                            "resource.label.zone"
                        ],
                        "perSeriesAligner": "ALIGN_MAX"
                    }
                ],
                "comparison": "COMPARISON_GT",
                "duration": "900s",
                "filter": "metric.type=\"compute.googleapis.com/instance/cpu/utilization\"
                           AND resource.type=\"gce_instance\"",
                "thresholdValue": 0.9,
                "trigger": {
                    "count": 1
                }
            }
        }
    ],
}
```

Calculate average every minute of data.

Google Cloud

This sample policy uses average CPU usage as an indicator of the health of a group of VMs. It averages by instance ID per zone,

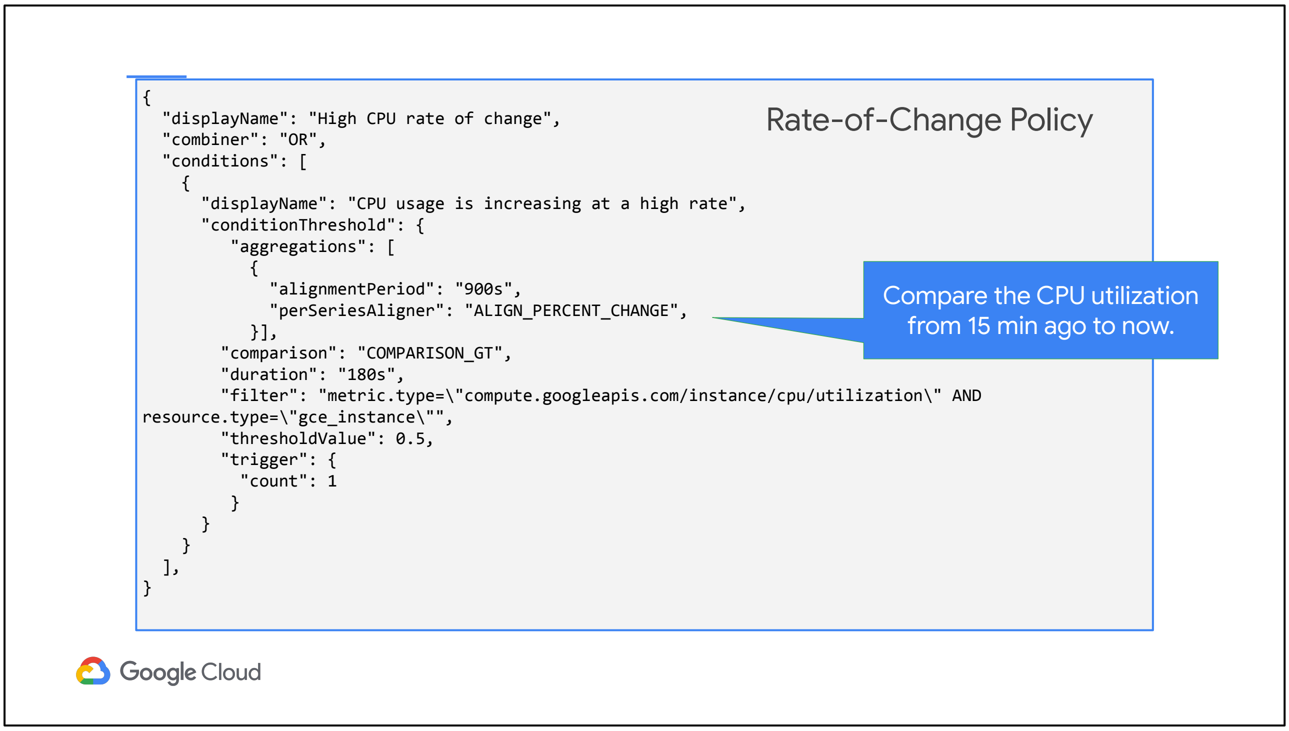


Metric Threshold Policy

```
{
    "displayName": "Very high CPU usage",
    "combiner": "OR",
    "conditions": [
        {
            "displayName": "CPU usage is extremely high",
            "conditionThreshold": {
                "aggregations": [
                    {
                        "alignmentPeriod": "60s",
                        "crossSeriesReducer": "REDUCE_MEAN",
                        "groupByFields": [
                            "project",
                            "resource.label.instance_id",
                            "resource.label.zone"
                        ],
                        "perSeriesAligner": "ALIGN_MAX"
                    }
                ],
                "comparison": "COMPARISON_GT",
                "duration": "900s",
                "filter": "metric.type=\"compute.googleapis.com/instance/cpu/utilization\"
                          AND resource.type=\"gce_instance\"",
                "thresholdValue": 0.9,
                "trigger": {
                    "count": 1
                }
            }
        }
    ],
}
```

CPU utilization is greater than 90% for 15 minutes (900 seconds).

then it grabs value with the highest CPU usage and compares it to the 90% threshold. If it’s over 90% for more than 15 minutes, an alert is triggered.



Rate-of-Change Policy

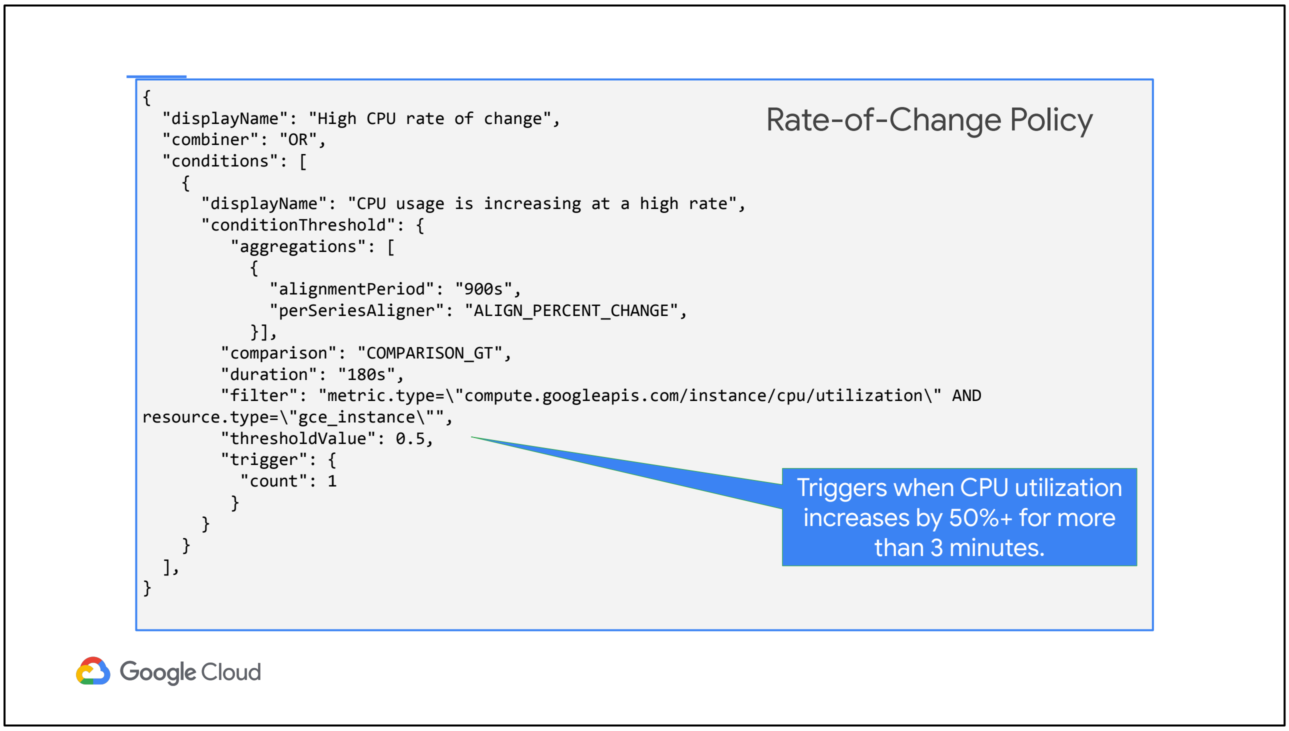
```
{
  "displayName": "High CPU rate of change",
  "combiner": "OR",
  "conditions": [
    {
      "displayName": "CPU usage is increasing at a high rate",
      "conditionThreshold": {
         "aggregations": [
           {
             "alignmentPeriod": "900s",
             "perSeriesAligner": "ALIGN_PERCENT_CHANGE",
           }],
        "comparison": "COMPARISON_GT",
        "duration": "180s",
        "filter": "metric.type=\"compute.googleapis.com/instance/cpu/utilization\" AND
resource.type=\"gce_instance\"",
        "thresholdValue": 0.5,
        "trigger": {
          "count": 1
         }
      }
    }
  ],
}
```

Compare the CPU utilization from 15 min ago to now.

Google Cloud

In our next example, let's look at a rate of change policy. In this example, we want to trigger an alert if the rate of CPU use is increasing rapidly.

This time we filter for the Compute Engine instance CPU utilization metric. The align percent change aligner averages CPU utilization over a ten-minute window. Here, we take the window average from 15 minutes (900 seconds) ago, and we compare it against the window for now, calculating a percent change.



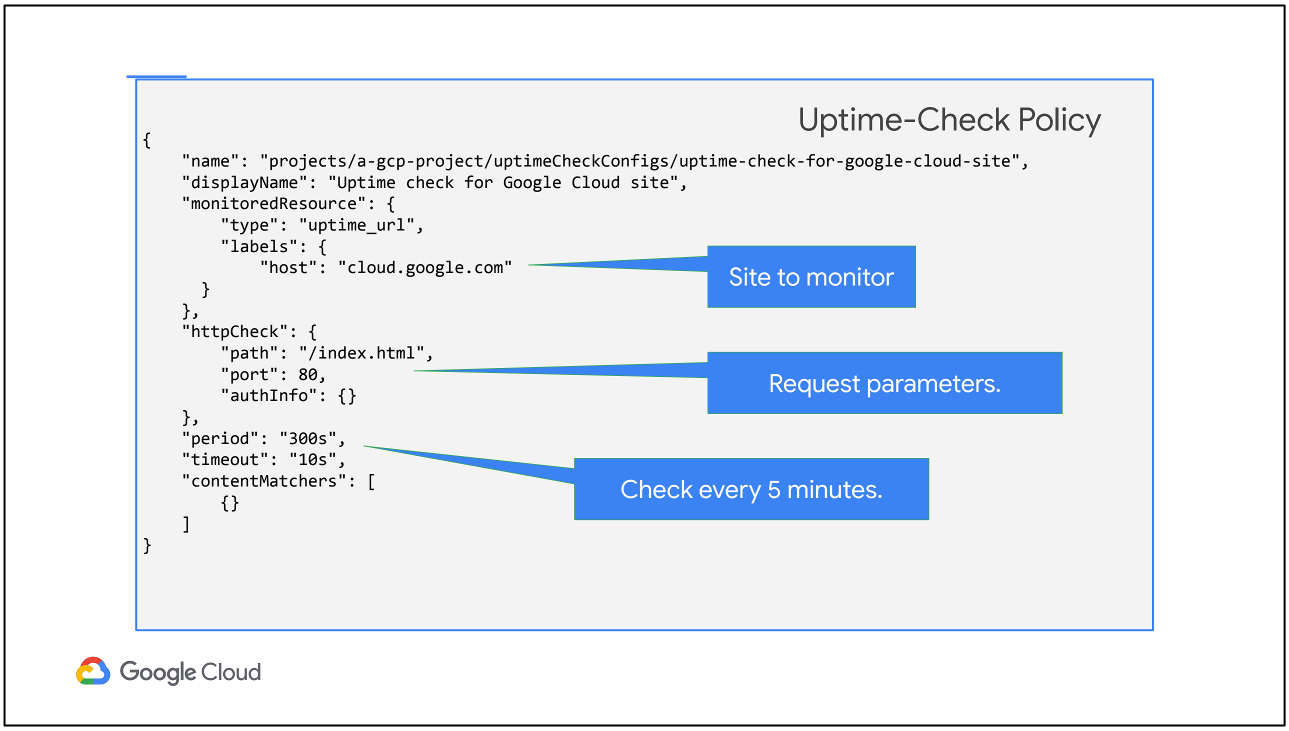
Rate-of-Change Policy

```
{
  "displayName": "High CPU rate of change",
  "combiner": "OR",
  "conditions": [
    {
      "displayName": "CPU usage is increasing at a high rate",
      "conditionThreshold": {
         "aggregations": [
           {
             "alignmentPeriod": "900s",
             "perSeriesAligner": "ALIGN_PERCENT_CHANGE",
           }],
        "comparison": "COMPARISON_GT",
        "duration": "180s",
        "filter": "metric.type=\"compute.googleapis.com/instance/cpu/utilization\" AND
resource.type=\"gce_instance\"",
        "thresholdValue": 0.5,
        "trigger": {
          "count": 1
         }
      }
    }
  ],
}
```

Triggers when CPU utilization increases by 50%+ for more than 3 minutes.

Google Cloud

If that percent change is over 50% for 3 minutes (180 seconds) or more, trigger an alert.



Uptime-Check Policy

```
{
    "name": "projects/a-gcp-project/uptimeCheckConfigs/uptime-check-for-google-cloud-site",
    "displayName": "Uptime check for Google Cloud site",
    "monitoredResource": {
        "type": "uptime_url",
        "labels": {
            "host": "cloud.google.com"
      }
    },
    "httpCheck": {
        "path": "/index.html",
        "port": 80,
        "authInfo": {}
    },
    "period": "300s",
    "timeout": "10s",
    "contentMatchers": [
        {}
    ]
}
```

Site to monitor
Request parameters.
Check every 5 minutes.
Google Cloud

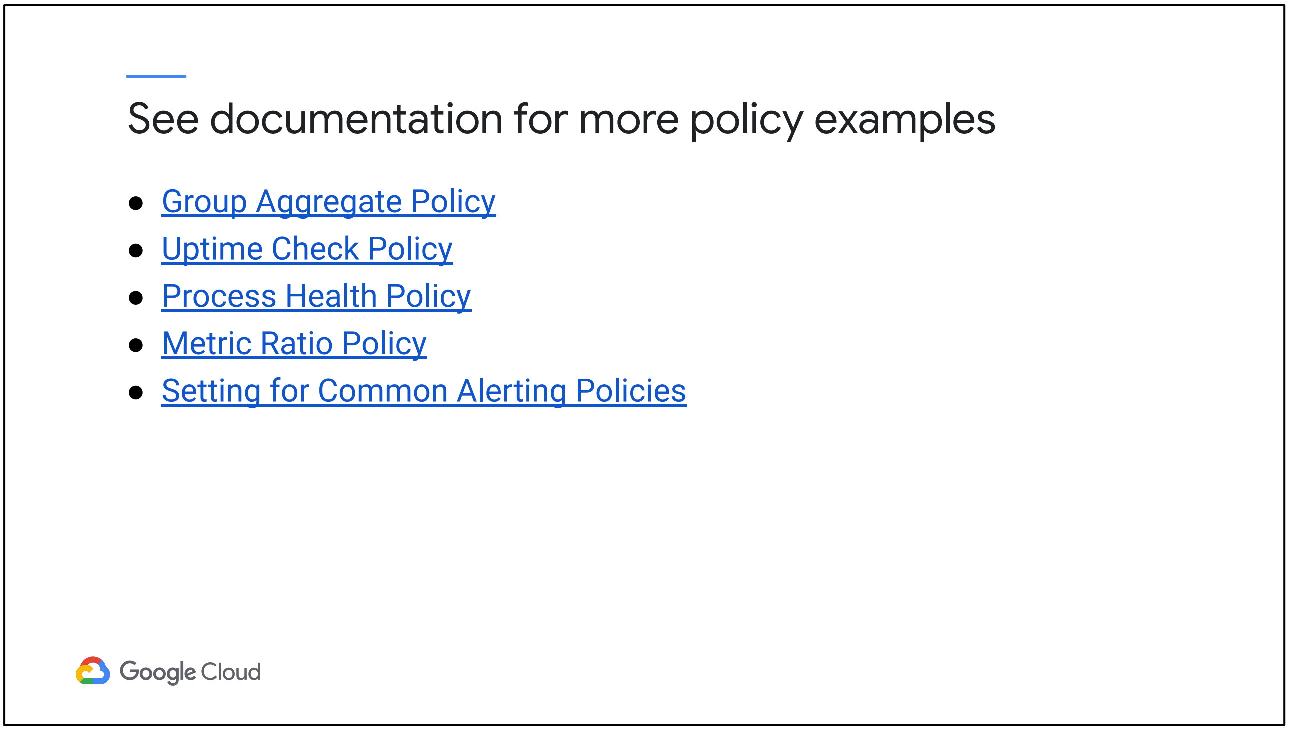
Lastly, let's examine an uptime check, and its corresponding alert policy, starting with the uptime check itself.

Here we have an HTTPS uptime check on the Google Cloud home page. It checks availability every 5 minutes (300 seconds) by pinging the index.html page sitting on port 443 of cloud.google.com. The page has 10 seconds to respond, or the check will fail.

To create the corresponding alerting policy for the last slide's uptime check, refer to the uptime check by its UPTIME_CHECK_ID. This ID is set when the check is created, it appears as the last component of the name field, and it is visible in the UI as the Check ID in the configuration summary. The ID is derived from the displayName, and can be verified by listing the uptime checks and looking at the name value.

The ID for the uptime check previously described is uptime-check-for-google-cloud-site.

This alerting policy example triggers if the uptime check fails.



See documentation for more policy examples

- Group Aggregate Policy
- Uptime Check Policy
- Process Health Policy
- Metric Ratio Policy
- Setting for Common Alerting Policies

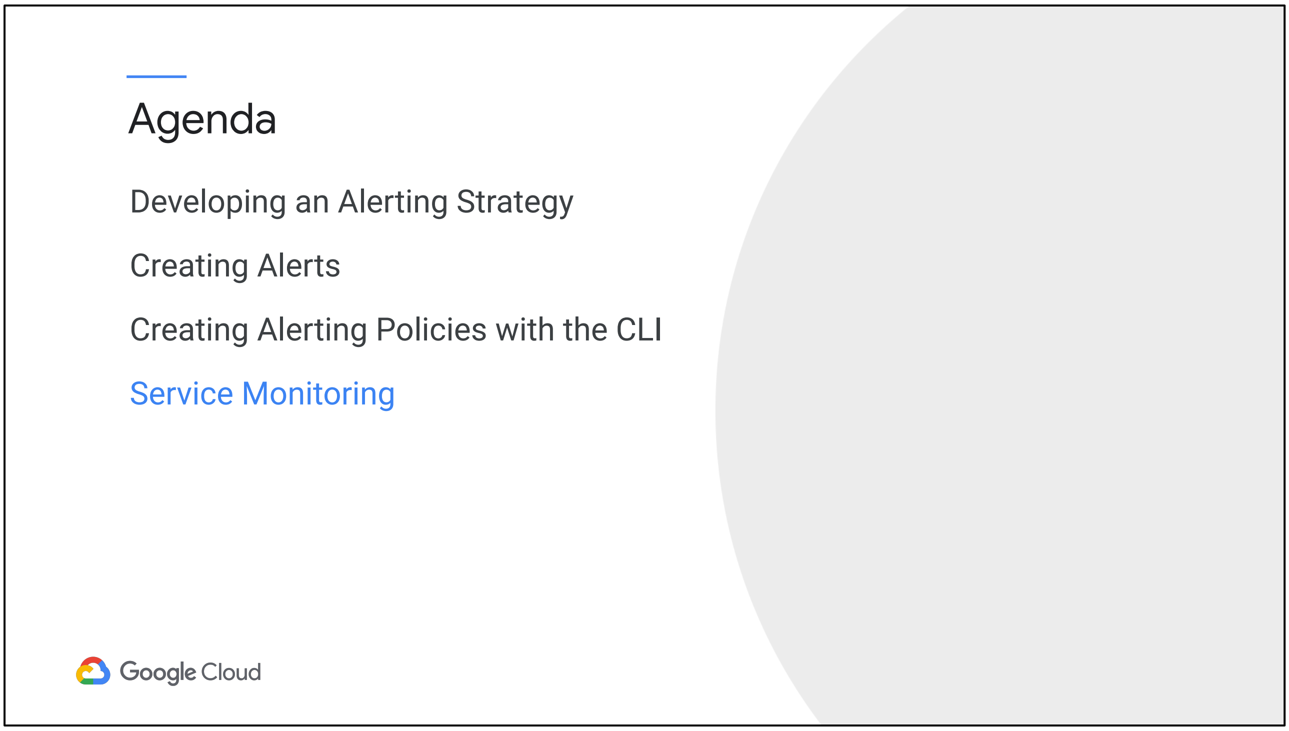
Google Cloud

For some more policy examples, visit the links listed on this page.

Lab Intro

Alerting in Google Cloud

In this lab, you deploy an application to GCP and then create alerting policies that notify you if the application is not up or is generating errors.



Agenda

Developing an Alerting Strategy

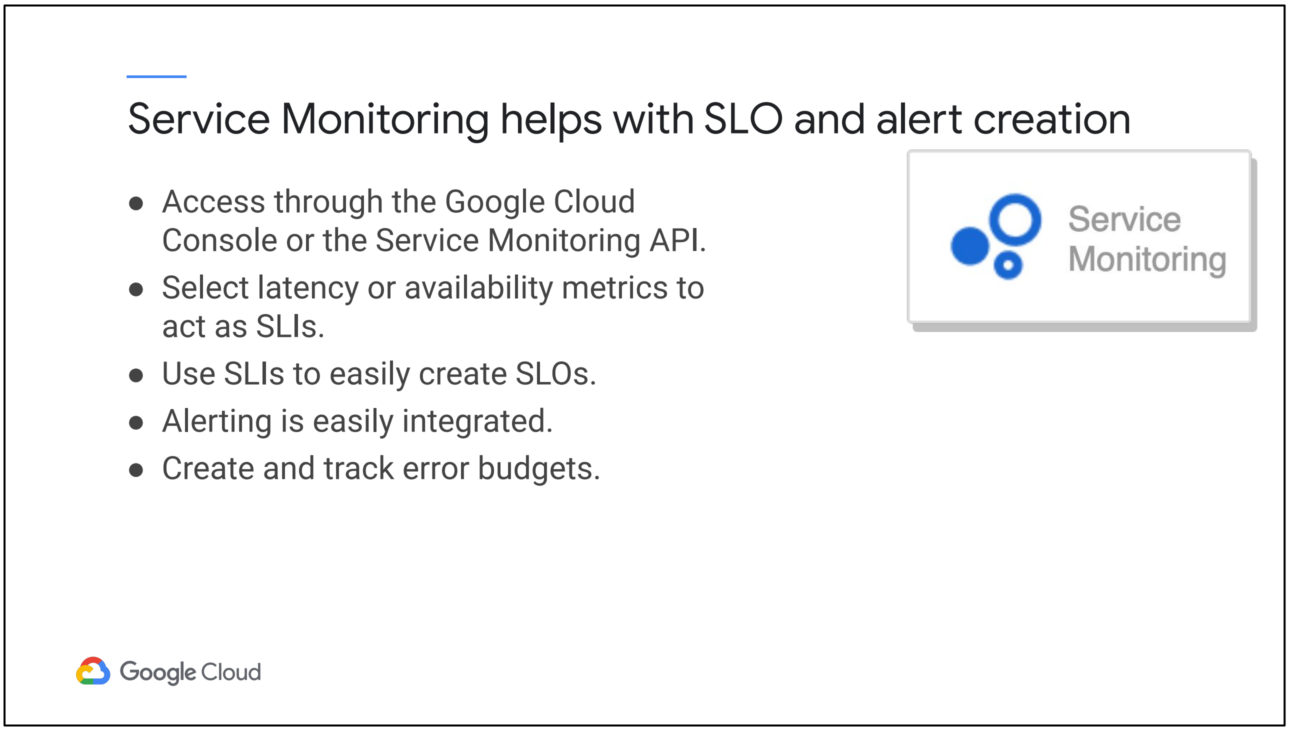
Creating Alerts

Creating Alerting Policies with the CLI

Service Monitoring

Google Cloud

Now that we've examined alerts, their use, and their creation, let's see how Google's new Service Monitoring console and API can help.



Service Monitoring helps with SLO and alert creation

- Access through the Google Cloud Console or the Service Monitoring API.
- Select latency or availability metrics to act as SLIs.
- Use SLIs to easily create SLOs.
- Alerting is easily integrated.
- Create and track error budgets.

Service Monitoring

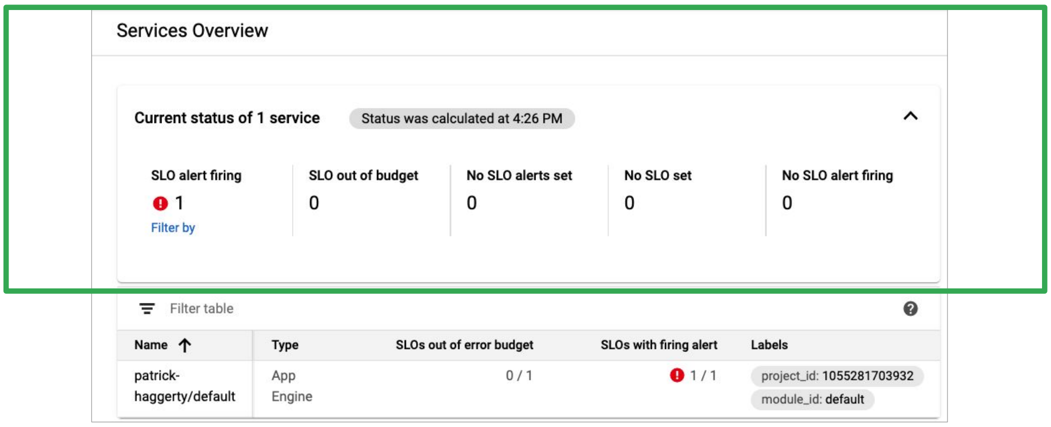
Google Cloud

Modern applications are composed of multiple services, and when something fails, it often seems like many things fail at once. To help manage this complexity, Service Monitoring helps with SLO and Alert creation.

Accessible through the Google Cloud console and via an API, Service Monitoring supports latency and availability based SLI metrics. SLO performance goals can be specified and, when combined with compliance periods, automated alerting is easily configurable.

Service Monitoring also calculates and reports error budgets to help with change planning.

Consolidated services overview



Services Overview

Current status of 1 service — Status was calculated at 4:26 PM

SLO alert firing	SLO out of budget	No SLO alerts set	No SLO set	No SLO alert firing
❗ 1 Filter by	0	0	0	0

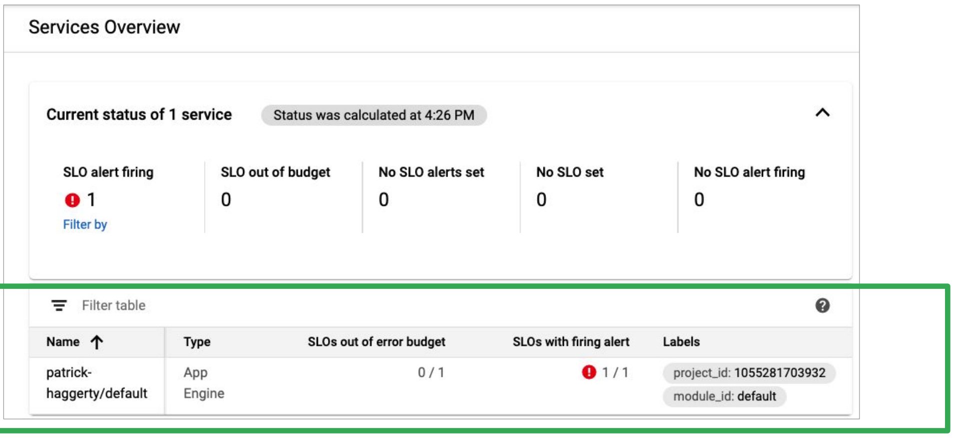
Filter table

Name ↑	Type	SLOs out of error budget	SLOs with firing alert	Labels
patrick-haggerty/default	App Engine	0 / 1	❗ 1 / 1	project_id: 1055281703932 module_id: default

Google Cloud

The Service Monitoring consolidated services overview page is your point of entry. Near the top of the page, a summary of your alerts and SLOs is displayed.

Consolidated services overview



Services Overview

Current status of 1 service — Status was calculated at 4:26 PM

SLO alert firing	SLO out of budget	No SLO alerts set	No SLO set	No SLO alert firing
❗ 1 Filter by	0	0	0	0

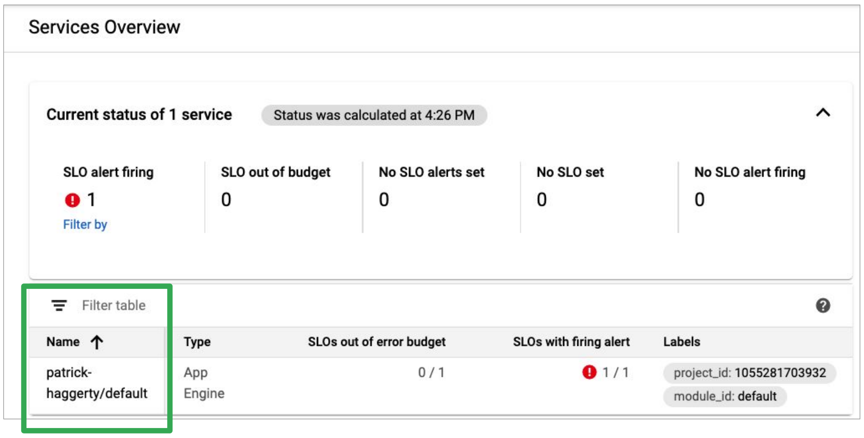
Filter table

Name ↑	Type	SLOs out of error budget	SLOs with firing alert	Labels
patrick-haggerty/default	App Engine	0 / 1	❗ 1 / 1	project_id: 1055281703932 module_id: default

Google Cloud

Below that is a summary view of the health of your various services. Here you can see the service name, type, SLO status, and whether any SLO-related alerts are firing.

Consolidated services overview



Services Overview

Current status of 1 service — Status was calculated at 4:26 PM

SLO alert firing	SLO out of budget	No SLO alerts set	No SLO set	No SLO alert firing
● 1 Filter by	0	0	0	0

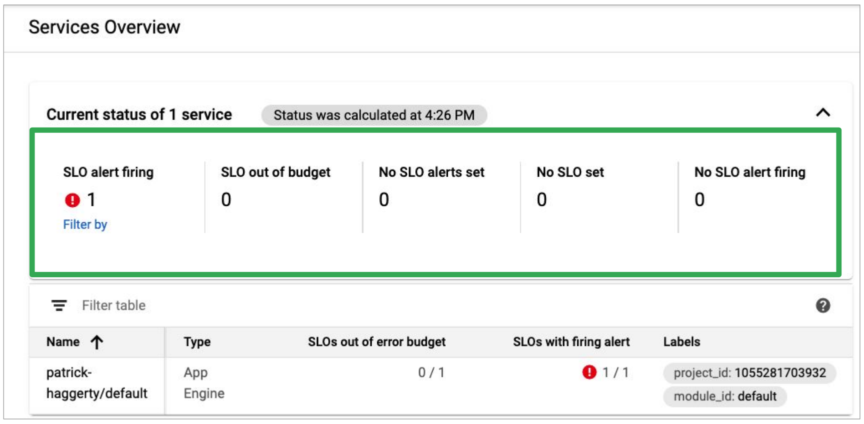
Filter table

Name ↑	Type	SLOs out of error budget	SLOs with firing alert	Labels
patrick-haggerty/default	App Engine	0 / 1	● 1 / 1	project_id: 1055281703932 module_id: default

Google Cloud

To monitor or view details for a specific service, click the service name.

Consolidated services overview



Current status of 1 service   Status was calculated at 4:26 PM

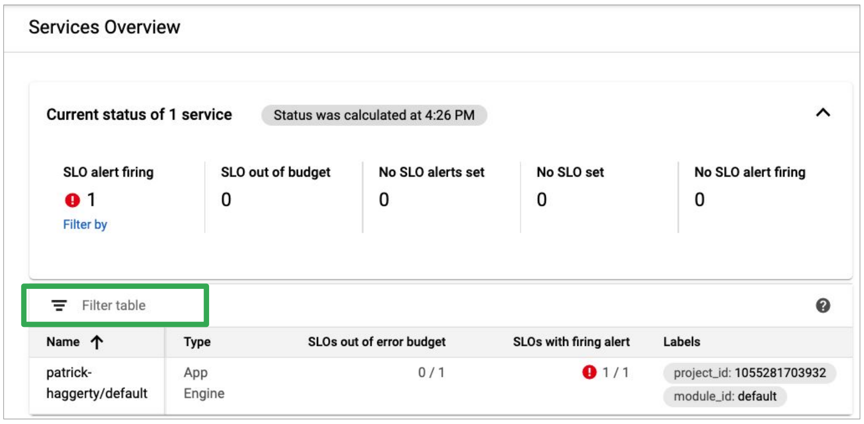
SLO alert firing	SLO out of budget	No SLO alerts set	No SLO set	No SLO alert firing
1 Filter by	0	0	0	0

Name	Type	SLOs out of error budget	SLOs with firing alert	Labels
patrick-haggerty/default	App Engine	0 / 1	1 / 1	project_id: 1055281703932 module_id: default

Google Cloud

You can apply filters to control which services are displayed in the table. First, you can click a **Filter by** link in the SLO status section to display only the applicable services in the table. For example, you can filter the table to show only the services that currently have SLO alerts firing.

Consolidated services overview


Services Overview

Current status of 1 service — Status was calculated at 4:26 PM

SLO alert firing	SLO out of budget	No SLO alerts set	No SLO set	No SLO alert firing
1	0	0	0	0

Filter by

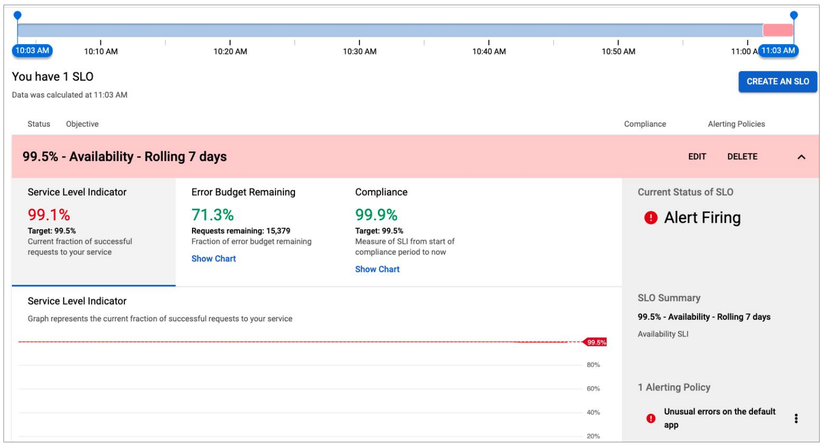
Filter table

Name	Type	SLOs out of error budget	SLOs with firing alert	Labels
patrick-haggerty/default	App Engine	0 / 1	1 / 1	project_id: 1055281703932 module_id: default

Google Cloud

You can also filter by entering a value in the **Filter table** in the upper-left corner of the table to apply additional conditions.

SLO details, compliance, and alerting in central UI



You have 1 SLO

Data was calculated at 11:03 AM

Status	Objective	Compliance	Alerting Policies

99.5% - Availability - Rolling 7 days

Service Level Indicator
99.1%
Target: 99.5%
Current fraction of successful requests to your service

Error Budget Remaining
71.3%
Requests remaining: 15,379
Fraction of error budget remaining

Compliance
99.9%
Target: 99.5%
Measure of SLI from start of compliance period to now

Current Status of SLO
Alert Firing

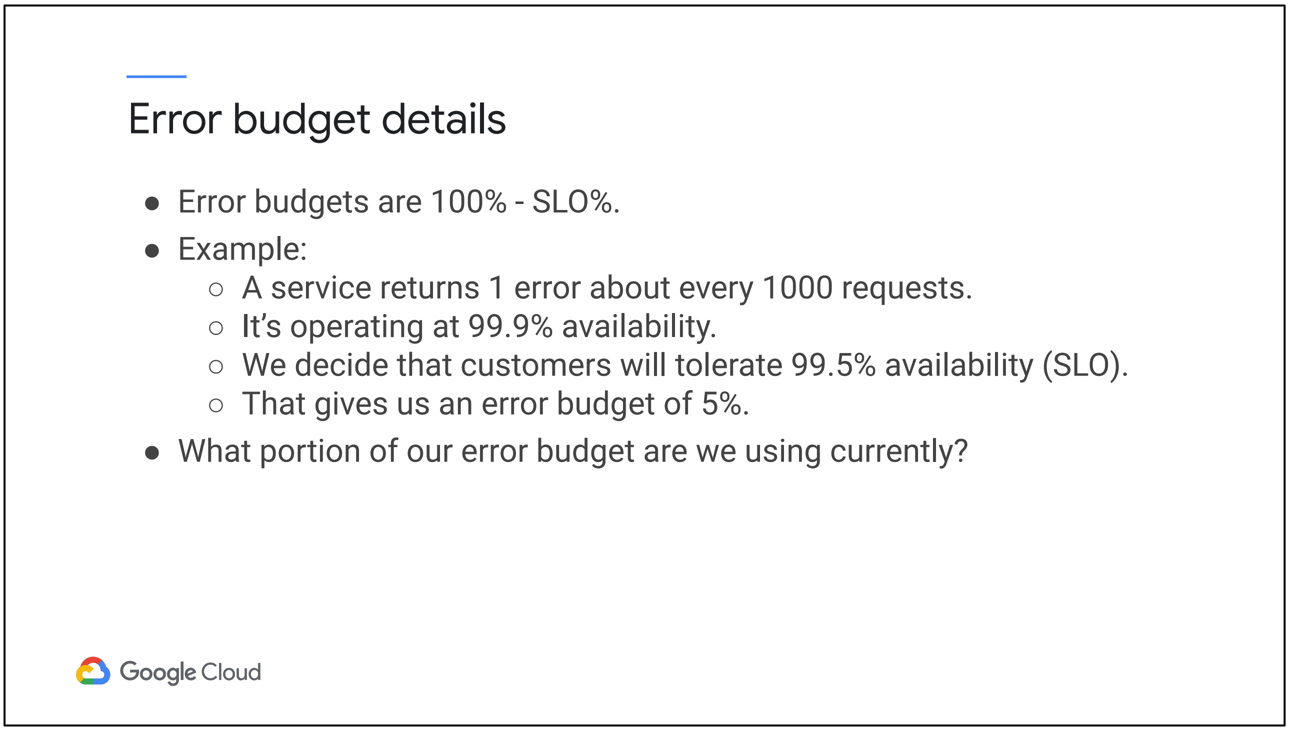
Service Level Indicator
Graph represents the current fraction of successful requests to your service

SLO Summary
99.5% - Availability - Rolling 7 days
Availability SLI

1 Alerting Policy
Unusual errors on the default app

Google Cloud

Click an individual service on the Services Overview page to view its details. There you can see existing SLOs and, by expanding them, their details. The SLI's current status, the error budget remaining, and the current level of SLO compliance are all displayed. If alerts have been set, their status is also displayed.



Error budget details

- Error budgets are 100% - SLO%.
- Example:
  - A service returns 1 error about every 1000 requests.
  - It’s operating at 99.9% availability.
  - We decide that customers will tolerate 99.5% availability (SLO).
  - That gives us an error budget of 5%.
- What portion of our error budget are we using currently?

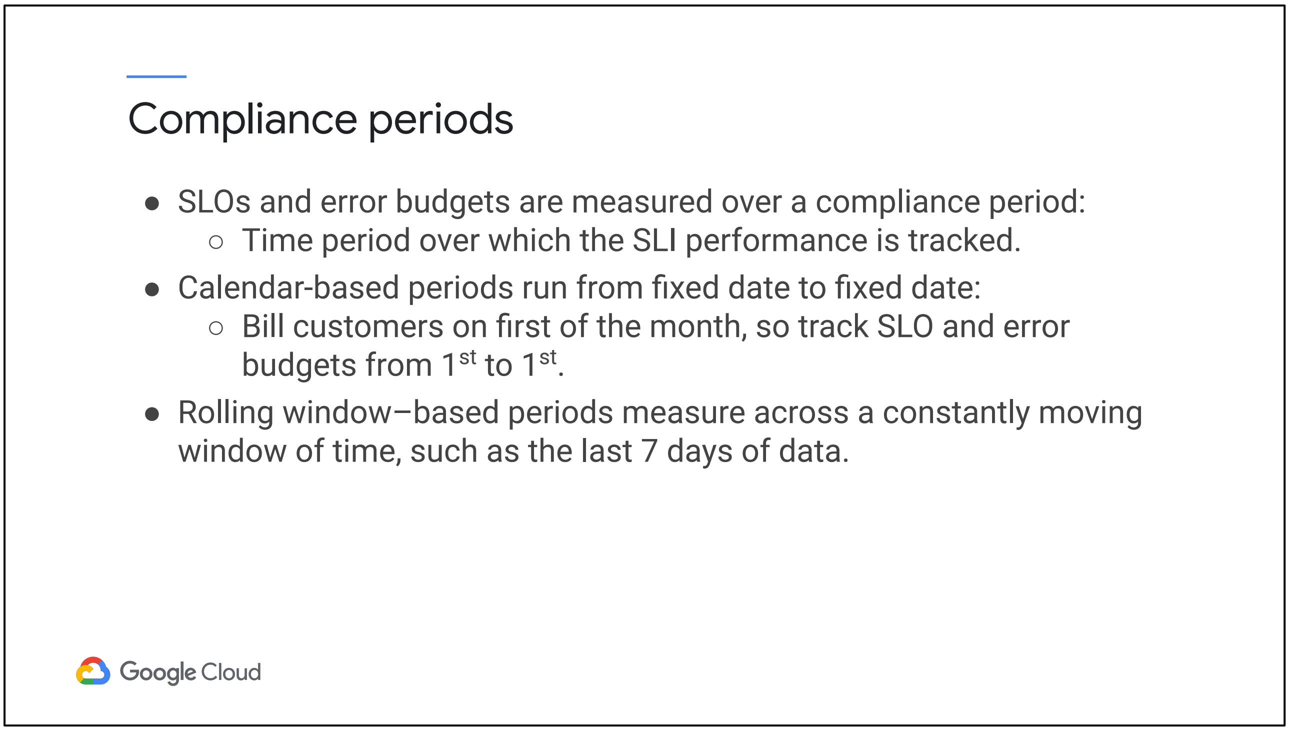
Google Cloud

As we discussed earlier in the module, error budgets are 100% - SLO%.

For example, if a service returns one error about every 1000 requests, then it's operating at 99.9% availability. If we determine that customers will tolerate 99.5% availability, and that's where we've set our SLO, that gives us an error budget of 5%.

In this example, what portion of our error budget are we using currently?

20%. The error budget is 5%. We are currently getting errors about 1% of the time. That means we've used 1 out of 5, or 20% of our error budget.



Compliance periods

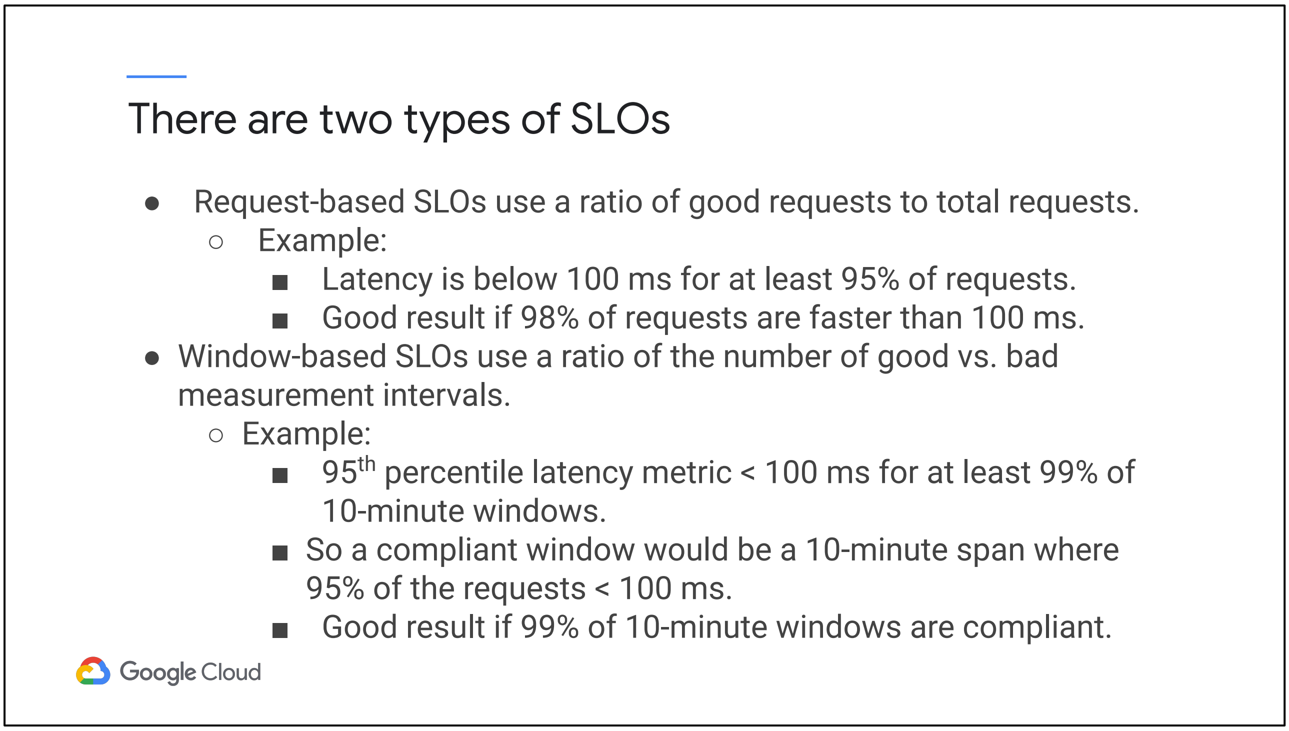
- SLOs and error budgets are measured over a compliance period:
  - Time period over which the SLI performance is tracked.
- Calendar-based periods run from fixed date to fixed date:
  - Bill customers on first of the month, so track SLO and error budgets from 1st to 1st.
- Rolling window–based periods measure across a constantly moving window of time, such as the last 7 days of data.

Google Cloud

SLOs and error budgets are measured over a period of compliance; that is, a period over which the SLI performance is tracked.

There are two fundamental types of compliance periods:

- Calendar-based periods run *from* a fixed date *to* a fixed date. Perhaps you bill customers on the first of the month, so you track SLOs and error budgets from the 1st of the month to the 1st of the next month.
- Rolling window–based periods measure across a constantly moving window of time. Perhaps we always want to be in compliance over a rolling window containing the last seven days of data.



There are two types of SLOs

- Request-based SLOs use a ratio of good requests to total requests.
    - Example:
        - Latency is below 100 ms for at least 95% of requests.
        - Good result if 98% of requests are faster than 100 ms.
- Window-based SLOs use a ratio of the number of good vs. bad measurement intervals.
    - Example:
        - 95th percentile latency metric < 100 ms for at least 99% of 10-minute windows.
        - So a compliant window would be a 10-minute span where 95% of the requests < 100 ms.
        - Good result if 99% of 10-minute windows are compliant.

Google Cloud

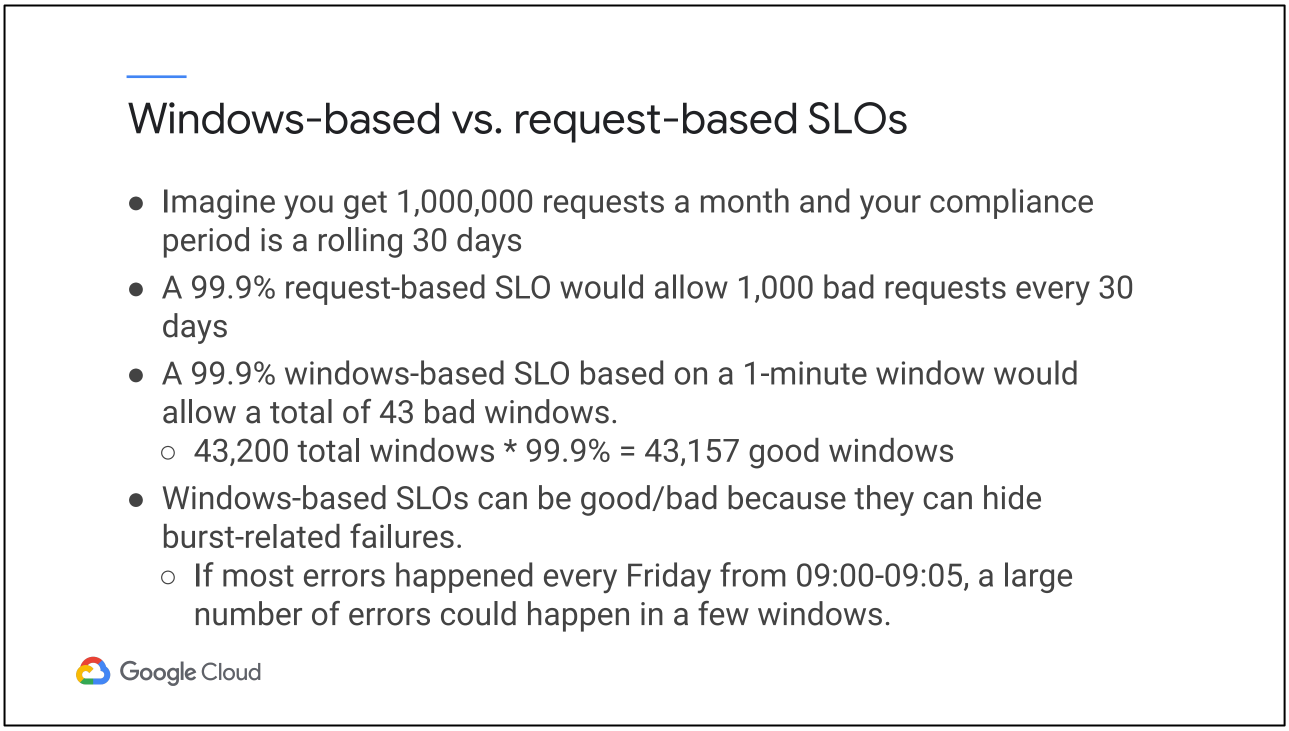
There are two fundamental ways Service Monitoring can approach SLO compliance calculations.

Request-based SLOs use a ratio of good requests to total requests.

For example, we want a request-based SLO with a latency below 100 ms for at least 95% of requests. So we'd be happy if 98% of requests were faster than 100 ms.

Window-based SLOs use a ratio of the number of good vs. bad measurement intervals, or windows. So each window represents a data point, rather than all the data points that comprise the window.

For example, take a 95th percentile latency SLO that needs to be less than 100 ms for at least 99% of 10-minute windows. Here, a compliant window would be a 10-min period, over which 95% of the requests were less than 100 ms. We'd be happy if 99% of 10-minute windows were compliant.



Windows-based vs. request-based SLOs

- Imagine you get 1,000,000 requests a month and your compliance period is a rolling 30 days
- A 99.9% request-based SLO would allow 1,000 bad requests every 30 days
- A 99.9% windows-based SLO based on a 1-minute window would allow a total of 43 bad windows.
  - 43,200 total windows * 99.9% = 43,157 good windows
- Windows-based SLOs can be good/bad because they can hide burst-related failures.
  - If most errors happened every Friday from 09:00-09:05, a large number of errors could happen in a few windows.

Google Cloud

Let's look at another pair of window-based vs. request-based SLO examples.

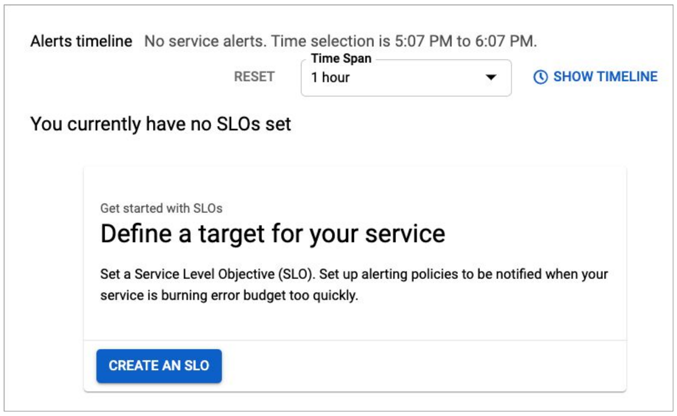
Imagine you get 1,000,000 requests a month, and your compliance period is a rolling 30 days.

If you were looking for a 99.9% request-based SLO, that would translate to 1,000 total bad requests every 30 days.

On the other hand, a 99.9% windows-based SLO, averaged across one-minute windows, would allow a total of 43 bad windows, or
43,200 total windows * 99.9% = 43,157 good windows.

Windows-based SLOs are good and bad because they can hide burst related failures. If the system returns nothing but errors, but only every Friday morning from 9:00-9:05, then you'd never violate your SLO, but no one would want to use the system the first thing Friday mornings :-)

Service Monitoring makes SLO creation easy


Alerts timeline No service alerts. Time selection is 5:07 PM to 6:07 PM.

RESET | Time Span: 1 hour | SHOW TIMELINE

You currently have no SLOs set

Get started with SLOs

Define a target for your service

Set a Service Level Objective (SLO). Set up alerting policies to be notified when your service is burning error budget too quickly.

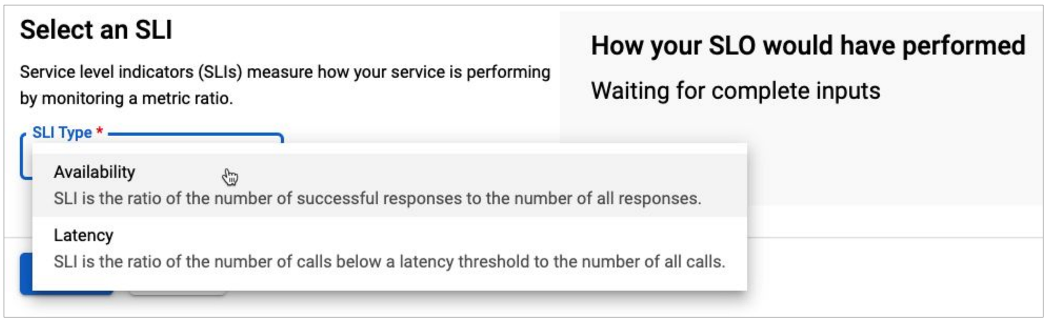
CREATE AN SLO

Google Cloud

Service Monitoring makes SLO creation easy. On the Services overview page, select one of the listed services. If a service is built on a Google Cloud compute technology that supports Service Monitoring, it will automatically be listed.

Next, click **Create an SLO**.

SLIs based on availability or latency



Select an SLI

Service level indicators (SLIs) measure how your service is performing by monitoring a metric ratio.

SLI Type *

Availability
SLI is the ratio of the number of successful responses to the number of all responses.

Latency
SLI is the ratio of the number of calls below a latency threshold to the number of all calls.

How your SLO would have performed

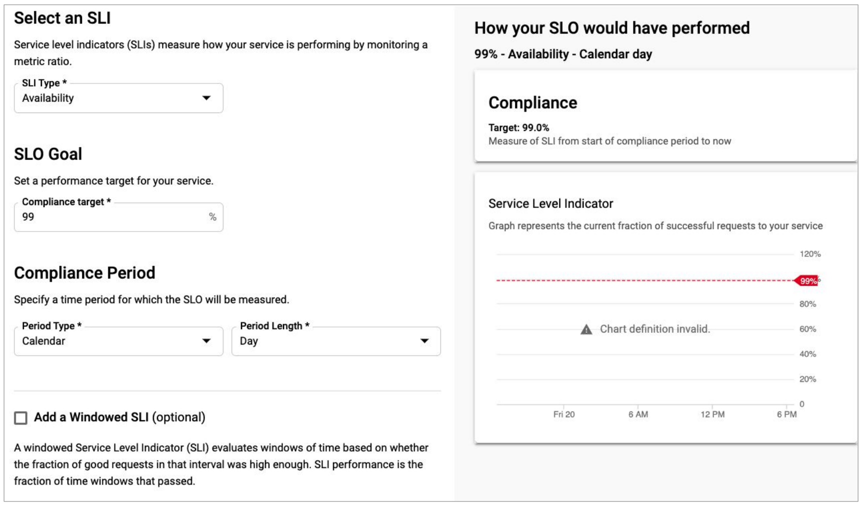
Waiting for complete inputs

Google Cloud

Select an option from the SLI Type list. At the time of this writing, Service Monitoring supports availability- and latency-based SLIs.

- **Availability** is a ratio of the number of successful responses to the number of all responses.
- **Latency** is the ratio of the number of calls that are below the specified **Latency Threshold** to the number of all calls.

Set compliance periods, type, and goal



Select an SLI

Service level indicators (SLIs) measure how your service is performing by monitoring a metric ratio.

SLI Type *: Availability

SLO Goal

Set a performance target for your service.

Compliance target *: 99 %

Compliance Period

Specify a time period for which the SLO will be measured.

Period Type *: Calendar

Period Length *: Day

☐ **Add a Windowed SLI** (optional)

A windowed Service Level Indicator (SLI) evaluates windows of time based on whether the fraction of good requests in that interval was high enough. SLI performance is the fraction of time windows that passed.

How your SLO would have performed

99% - Availability - Calendar day

Compliance

Target: 99.0%

Measure of SLI from start of compliance period to now

Service Level Indicator

Graph represents the current fraction of successful requests to your service

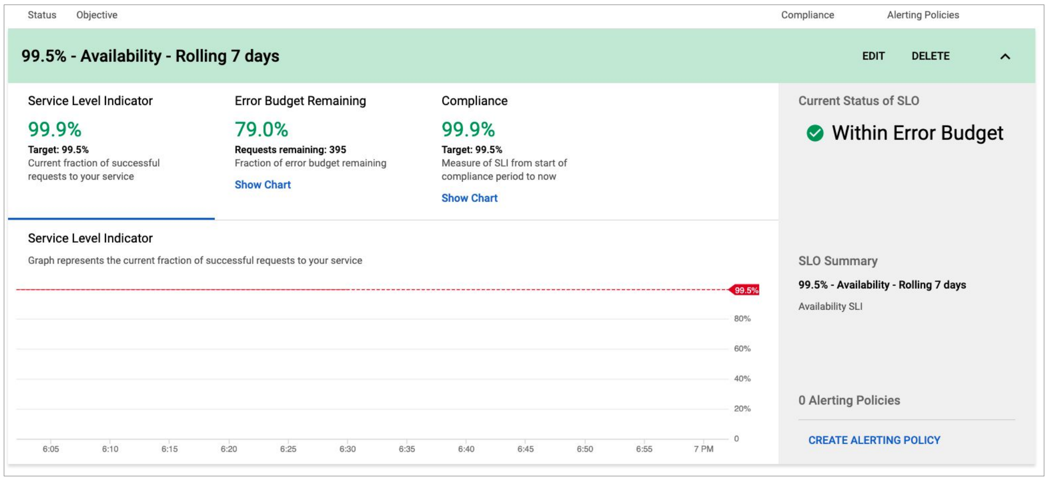
Google Cloud

In the SLO Goal section, enter a percentage in the Compliance target field to set the performance target for the SLI. Service Monitoring uses this value to calculate the error budget you have for this SLO.

In the Compliance Period section, select the Period Type and the Period Length. You will recall from earlier that the two compliance period types are calendar-based and rolling.

Optionally, select Add a Windowed SLI. As discussed, a windowed SLI can help you catch (or ignore) periods of time when the service won't meet the SLO Compliance target.

SLO status is easy to monitor


Status | Objective | Compliance | Alerting Policies

99.5% - Availability - Rolling 7 days

EDIT DELETE

Service Level Indicator
99.9%
Target: 99.5%
Current fraction of successful requests to your service

Error Budget Remaining
79.0%
Requests remaining: 395
Fraction of error budget remaining
Show Chart

Compliance
99.9%
Target: 99.5%
Measure of SLI from start of compliance period to now
Show Chart

Current Status of SLO
Within Error Budget

Service Level Indicator
Graph represents the current fraction of successful requests to your service

SLO Summary
99.5% - Availability - Rolling 7 days
Availability SLI

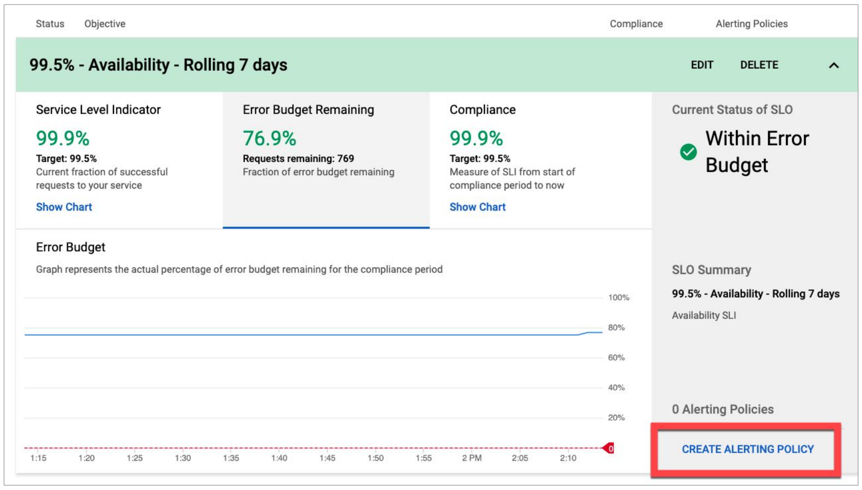
0 Alerting Policies

CREATE ALERTING POLICY

Google Cloud

After the SLO has been created, it’s easy to monitor the SLI current status, error budget, compliance, and alert status.

SLO linked alerts are easy to create



Status Objective Compliance Alerting Policies

99.5% - Availability - Rolling 7 days

EDIT DELETE

Service Level Indicator	Error Budget Remaining	Compliance
99.9%	76.9%	99.9%
Target: 99.5%	Requests remaining: 769	Target: 99.5%
Current fraction of successful requests to your service	Fraction of error budget remaining	Measure of SLI from start of compliance period to now
Show Chart		Show Chart

Current Status of SLO

Within Error Budget

Error Budget

Graph represents the actual percentage of error budget remaining for the compliance period

SLO Summary

99.5% - Availability - Rolling 7 days

Availability SLI

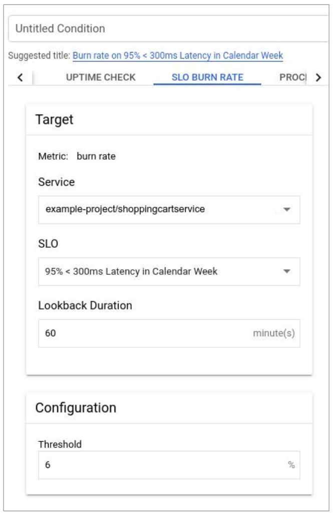
0 Alerting Policies

CREATE ALERTING POLICY

Google Cloud

Creating an alert is as easy as clicking **Create Alerting Policy**.

Configure an alert condition for SLO error budget rate


Untitled Condition

Suggested title: Burn rate on 95% < 300ms Latency in Calendar Week

Target

Metric: burn rate

Service: example-project/shoppingcartservice

SLO: 95% < 300ms Latency in Calendar Week

Lookback Duration: 60 minute(s)

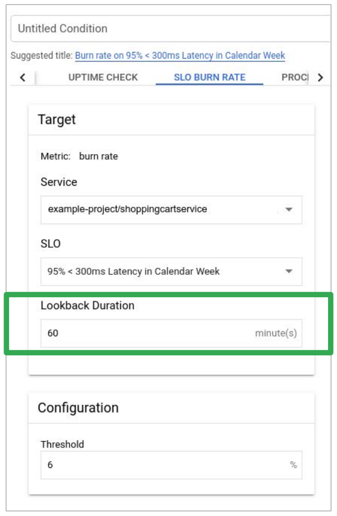
Configuration

Threshold: 6 %

Google Cloud

Service Monitoring can trigger an alert when a service is on track to violate an SLO. The alerting policy can be based on the rate of consumption of your error budget. You specify a lookback period, for example the last 60 minutes of time, and an error budget consumption percentage over that period. Service Monitoring will set the rest of the alert policy settings automatically.

Configure an alert condition for SLO error budget rate


Untitled Condition
Suggested title: Burn rate on 95% < 300ms Latency in Calendar Week
UPTIME CHECK
SLO BURN RATE
PROC

Target

Metric: burn rate

Service: example-project/shoppingcartservice

SLO: 95% < 300ms Latency in Calendar Week

Lookback Duration: 60 minute(s)

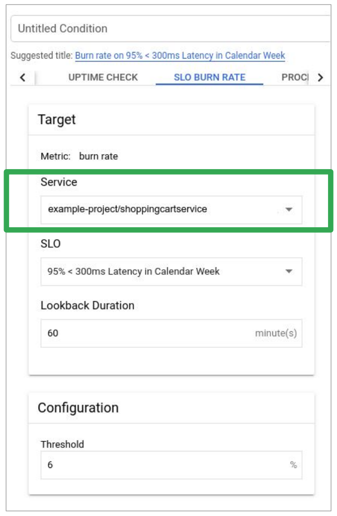
Configuration

Threshold: 6 %

Google Cloud

Determining what values you should set for the lookback period and consumption percentage might take some trial and error. You could use the default lookback period of 60 minutes as a starting point. To determine the consumption percentage, monitor the service behavior to see what percentage of the total error budget (over the compliance period) was consumed in the previous 60 minutes. You want to set the consumption percentage so that you don't expend more error budget in the lookback period than you can afford, but you don't want to set off an alert unnecessarily.

Configure an alert condition for SLO error budget rate


Untitled Condition

Suggested title: Burn rate on 95% < 300ms Latency in Calendar Week

Target

Metric: burn rate

Service: example-project/shoppingcartservice

SLO: 95% < 300ms Latency in Calendar Week

Lookback Duration: 60 minute(s)

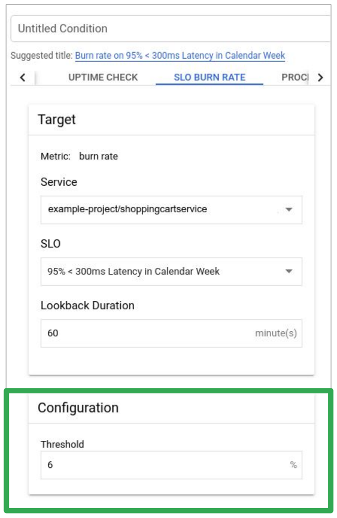
Configuration

Threshold: 6 %

Google Cloud

For example, suppose you created an SLO with the following name: 95% < 300ms Latency in Calendar Week

Configure an alert condition for SLO burn rate



Untitled Condition

Suggested title: Burn rate on 95% < 300ms Latency in Calendar Week

Target

Metric: burn rate

Service: example-project/shoppingcartservice

SLO: 95% < 300ms Latency in Calendar Week

Lookback Duration: 60 minute(s)

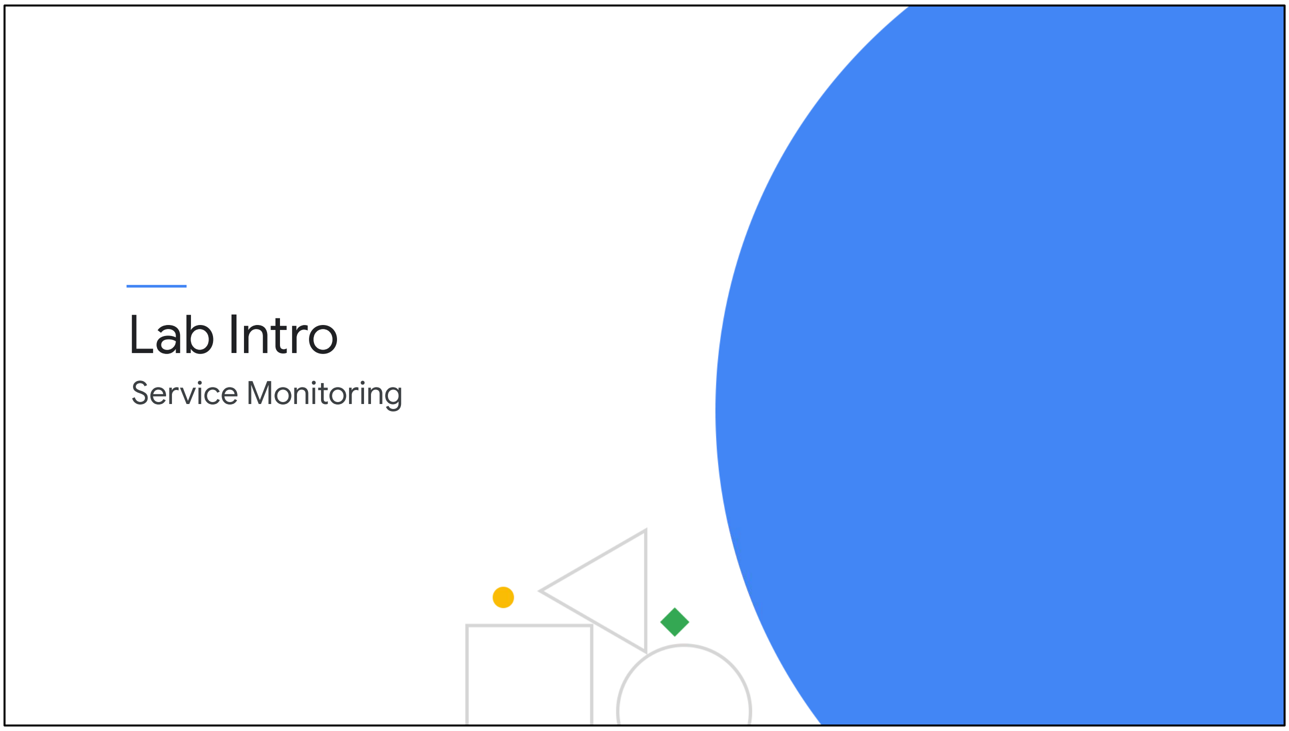
Configuration

Threshold: 6 %

Google Cloud

With this SLO, only 5% of the total number of requests in a week can have a latency > 300ms. Hitting or exceeding 5% consumes your total error budget. If you set the lookback period to one hour, each lookback period is 1/168 of your compliance period (there are 168 hours in a week). To calculate the hourly consumption percentage that doesn't exceed the total error budget for the week: 5% ÷ 168 ≈ 0.3%

Because latency for your service can fluctuate depending on load or other conditions, setting 0.3% as the consumption percentage might trigger unnecessary alerts. You could start with a value twice that, or 0.6%, and then monitor your service and adjust the value as needed.



Lab Intro

Service Monitoring

Google Cloud Service Monitoring streamlines the creation of Service Level Objectives based on latency- and availability-based Service Level Indicators. In this lab you use Service Monitoring to create a 99.5% availability SLO and corresponding alert.

Quiz

In evaluating your alerting policies, which below best describes precision.

A. The proportion of events detected that were significant

B. The proportion of significant events detected

C. How long it takes to send notifications in various conditions

D. How long alerts fire after an issue is resolved

Quiz

In evaluating your alerting policies, which below best describes precision.

[✓ A. The proportion of events detected that were significant]

B. The proportion of significant events detected

C. How long it takes to send notifications in various conditions

D. How long alerts fire after an issue is resolved

Quiz

In evaluating your alerting policies, which below best describes recall.

A. The proportion of events detected that were significant

B. The proportion of significant events detected

C. How long it takes to send notifications in various conditions

D. How long alerts fire after an issue is resolved

Google Cloud

Quiz

In evaluating your alerting policies, which below best describes recall.

A. The proportion of events detected that were significant

[✓ B. The proportion of significant events detected]

C. How long it takes to send notifications in various conditions

D. How long alerts fire after an issue is resolved

Google Cloud

Quiz

In GCP alerting, which below is a valid notification channel?

A. SMS and Email

B. Webhooks

C. Slack

D. All of the above

Google Cloud

Quiz

In GCP alerting, which below is a valid notification channel?

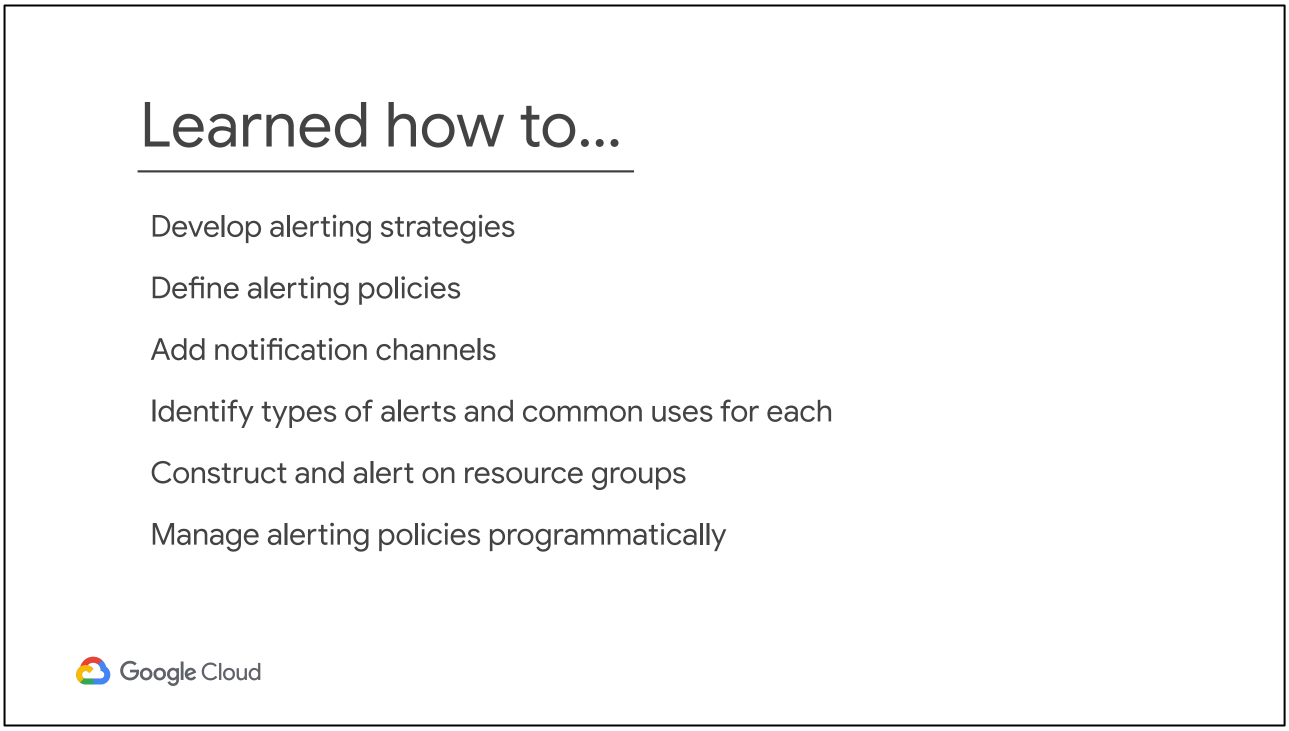
A. SMS and Email

B. Webhooks

C. Slack

[✓ D. All of the above]

Google Cloud



Learned how to...

Develop alerting strategies

Define alerting policies

Add notification channels

Identify types of alerts and common uses for each

Construct and alert on resource groups

Manage alerting policies programmatically

Google Cloud

Fantastic job. In this module, you have learned how to:

Develop alerting strategies

Define alerting policies

Add notification channels

Identify types of alerts and common uses for each

Construct and alert on resource groups

And manage alerting policies programmatically

Great work.