**Service Monitoring**

experimentLabschedule1 hour 30 minutesuniversal\_currency\_alt5 Creditsshow\_chartIntermediate

infoThis lab may incorporate AI tools to support your learning.

**Overview**

Google Cloud's Service Monitoring streamlines the creation of microservice Service Level Objectives (SLOs) based on availability, latency, or custom Service Level Indicators (SLIs). In this lab, you use Service Monitoring to create a 99.5% availability SLO and corresponding alert.

Objectives

In this lab, you learn how to perform the following tasks:

* Deploy a test application.
* Use Service Monitoring to create an SLO.
* Tie an alert to the SLO.

**Setup and requirements**

Access Qwiklabs

For each lab, you get a new Google Cloud project and set of resources for a fixed time at no cost.

1. Sign in to Qwiklabs using an **incognito window**.
2. Note the lab's access time (for example, 1:15:00), and make sure you can finish within that time.  
   There is no pause feature. You can restart if needed, but you have to start at the beginning.
3. When ready, click **Start lab**.
4. Note your lab credentials (**Username** and **Password**). You will use them to sign in to the Google Cloud Console.
5. Click **Open Google Console**.
6. Click **Use another account** and copy/paste credentials for **this** lab into the prompts.  
   If you use other credentials, you'll receive errors or **incur charges**.
7. Accept the terms and skip the recovery resource page.

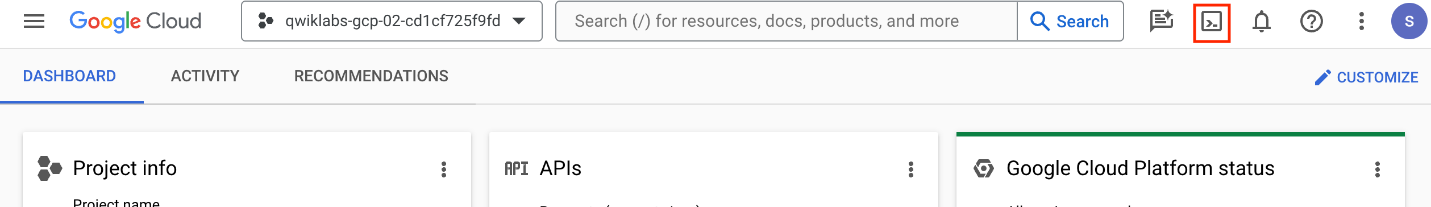
**Note:** Do not click **End Lab** unless you have finished the lab or want to restart it. This clears your work and removes the project.

Activate Google Cloud Shell

Google Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud.

Google Cloud Shell provides command-line access to your Google Cloud resources.

1. In Cloud console, on the top right toolbar, click the Open Cloud Shell button.



1. Click **Continue**.

It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT\_ID*. For example:



**gcloud** is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

* You can list the active account name with this command:

gcloud auth list

Copied!

content\_copy

**Output:**

Credentialed accounts:

- @.com (active)

**Example output:**

Credentialed accounts:

- google1623327\_student@qwiklabs.net

* You can list the project ID with this command:

gcloud config list project

Copied!

content\_copy

**Output:**

[core]

project =

**Example output:**

[core]

project = qwiklabs-gcp-44776a13dea667a6

**Note:**Full documentation of **gcloud** is available in the [gcloud CLI overview guide](https://cloud.google.com/sdk/gcloud" \t "_blank).

**Task 1. Deploy a test application**

In this task, you deploy a test application to App Engine.

Deploy a test application to App Engine

To have something for Service Monitoring to connect to, deploy a basic Node.js application to App Engine standard.

1. In your Cloud Shell terminal, clone https://github.com/haggman/HelloLoggingNodeJS.git repo:

git clone https://github.com/haggman/HelloLoggingNodeJS.git

Copied!

content\_copy

This repository contains a basic Node.js web application used for testing. This is the same application you saw pieces of in the lecture module.

1. Change into the HelloLoggingNodeJS folder and open the index.js in the Cloud Shell code editor:

cd HelloLoggingNodeJS

edit index.js

Copied!

content\_copy

**Note:** If an error indicates that the code editor could not be loaded because third-party cookies are disabled, click **Open in New Window** and switch to the new tab.

1. Take a few minutes to peruse the code.
2. In the cloud shell code editor, look at the app.yaml file. And update the runtime nodejs version.

runtime: nodejs20

Copied!

content\_copy

App Engine standard uses this file to define the runtime required by the application.

1. In the cloud shell code editor, look at the package.json file.

Not only does this define the Node.js application dependencies, but it also defines the start script App Engine uses to serve requests.

1. Return to the Cloud Shell window. If the Cloud Shell is not visible, click **Open Terminal**.
2. In the Cloud Shell terminal, create a new App Engine app:

gcloud app create --region=us-east1

Copied!

content\_copy

This must be done once in each new project that is running App Engine applications. App Engine is a regional technology, thus the *region* switch.

1. Deploy the Hello Logging app to App Engine:

gcloud app deploy

Copied!

content\_copy

Wait until the deploy completes before moving on.

1. When prompted, type **y** and press **Enter**.
2. Copy the URL to your newly deployed app from the console (https://qwiklabs-gcp-\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*.appspot.com) and open it in a new browser tab.
3. Verify a Hello World! response.

Click *Check my progress* to verify the objective.

Assessment Completed!

Deploy an application to App Engine.

Check my progress

*Assessment Completed!*

**Task 2. Use Service Monitoring to create an availability SLO**

In this task, you:

* Use Service Monitoring to create an availability SLO.
* Create an alert tied to your SLO.
* Trigger the alert.

Place some load on the application

1. At the top of the Cloud Shell interface, press the **Add** icon to Open a new tab.
2. In the new tab, use a simple bash *while* loop to generate load on your application:

while true; \

do curl -s https://$DEVSHELL\_PROJECT\_ID.appspot.com/random-error \

-w '\n' ;sleep .1s;done

Copied!

content\_copy

The loop generates ten requests per second. The URL is to the /random-error route, which generates an error about every 1000 requests, so you should see approximately 1 error every 100s.

1. Leave the loop running in its Cloud Shell tab and move on to the next step.

Use Service Monitoring to create an availability SLO

We have a working App Engine application that is currently throwing an error approximately every 1000 requests. Imagine we want to create an availability SLO with a target of 99.5%, and an alert that will notify us if our SLO is in danger. That's exactly what Service Monitoring makes easy.

1. In the Google Cloud Console, use the **Navigation menu** (The navigation menu icon.) to navigate to **App Engine | Dashboard**. You can already see information on your running service and the load you are placing on it.
2. Scroll down to the Server Errors section. Have any errors been generated yet? If not, wait a couple of minutes and refresh the page. You should see one every few minutes.
3. Use the **Navigation menu** to navigate to **Error Reporting**.

Notice the error is also being caught here. We will discuss Error Reporting in a later module.

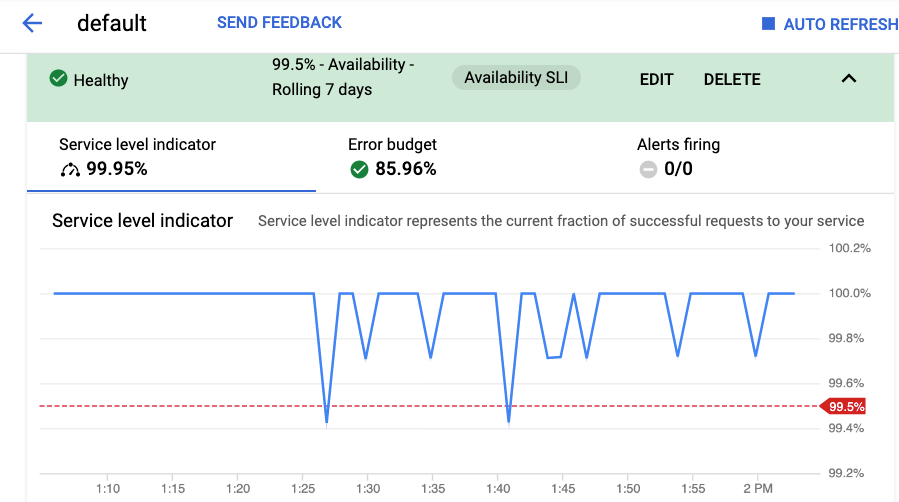
1. Use the **Navigation menu** to navigate to **Monitoring**.

It takes a moment for the monitoring workspace to create.

1. Once it loads, click **SLOs**.
2. Notice that Service Monitoring already sees your default App Engine application. "If it doesn't, wait a minute, refresh the page and click +Define a service, select default, and submit it."
3. Click the default App Engine application to drill into it.
4. Click **+Create SLO** to start the new SLO dialog.
5. Select the **Availability** metric, leave the evaluation method set to **Request-based**, and then click **Continue**.
6. Take a moment to investigate the details the SLI details displayed, then click **Continue**.
7. To define the SLO, set the Period type to **Rolling** and the Period length to **7** days to calculate the SLO on a constantly moving 7-day window of time.
8. Set the Goal to **99.5**% and the charts fill in, though it's typically difficult to see that 99.5 to 99.9 difference.
9. Click the red dashed line, and the chart will zoom in to make things easier to see.
10. Click **Continue**, notice the default name, and submit the new SLO by clicking **Create SLO**.

Investigate the new SLO and create an alert for it

* Under the Current status of 1 SLO section, expand the new SLO and investigate the information it displays. Move between the three tabs, **Service level indicator**, **Error budget**, and **Alerts firing**, investigating each.

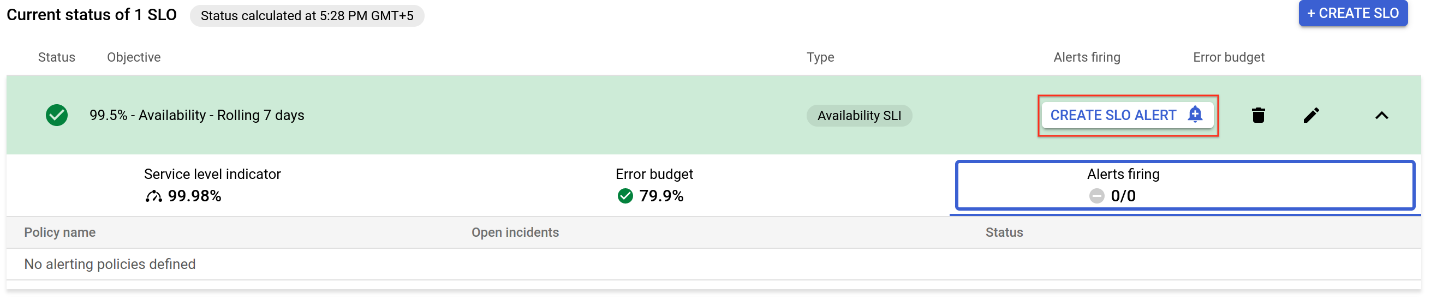


Create an alert tied to the availability SLO

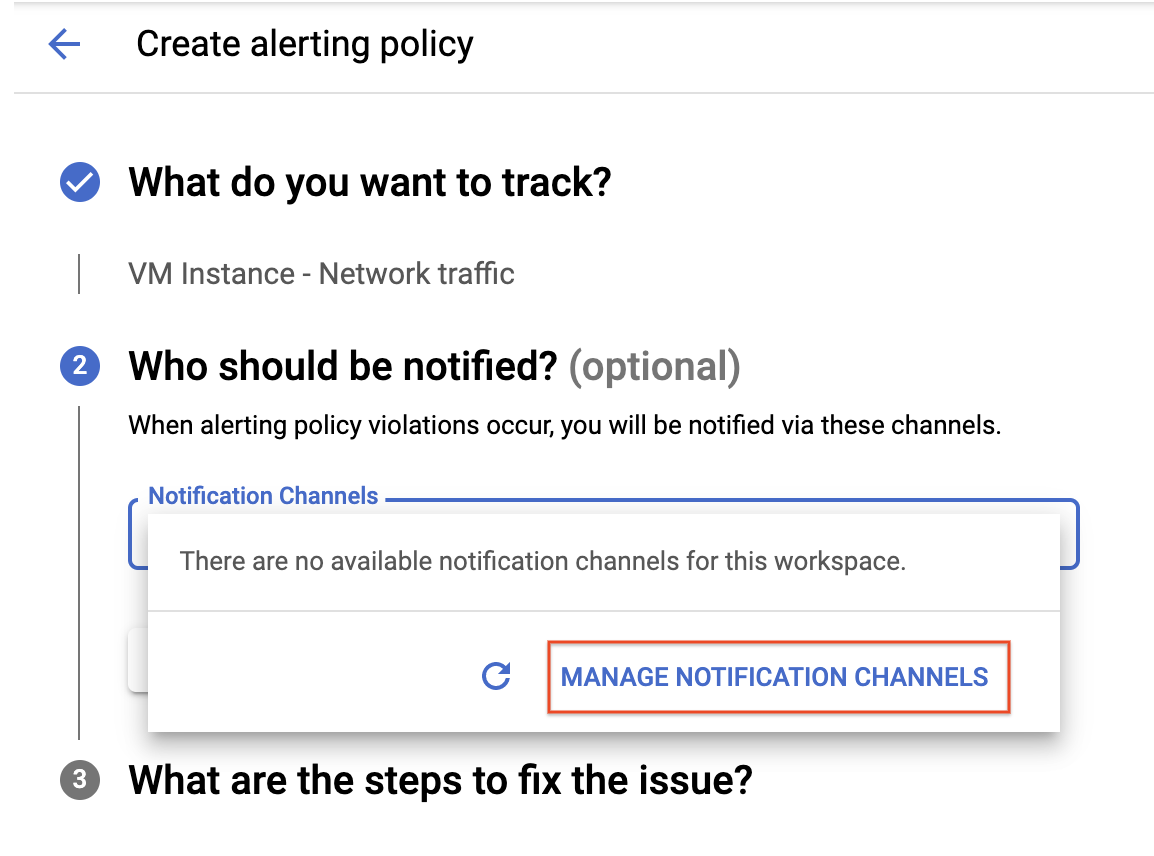
The SLO has been created and so far, you are well within your objective. Since the SLO target is 99.5%, and the SLI should be showing a current measurement level of about 99.9%, that means that your application is using approximately 1/5 of its error budget, so the error budget should be displaying about 80%. If you start to burn through your error budget at an unexpectedly fast rate, it would be nice for an alert to fire to let you know.

There are several ways to create an alert for an SLO in Service Monitoring.

1. Because you are looking at the expanded SLO interface, click the **Alerts firing** tab and select **CREATE SLO ALERT**.



1. Set the Display name to **Really short window test**. Because you are doing a test and not setting values, that would make sense in production.
2. Set the Lookback duration to **10** minutes and the burn rate threshold to **1.5**.
3. Click **Next**.
4. Click on drop down arrow next to **Notification Channels**, then click on **Manage Notification Channels**.



A **Notification channels** page will open in new tab.

1. Scroll down the page and click on **ADD NEW** for **Email**.
2. In **Create Email Channel** dialog box, enter your personal email address in the **Email Address** field and a **Display name**.
3. Click on **Save**.
4. For Who should be notified, use the Manage notification channels link to add your email address as a notification channel and select that. Remember, this link opens a new tab so close it once your email address has been added, and then **Save** the new alert.
5. Click on **Notification Channels** again, then click on the **Refresh icon** to get the display name you mentioned in the previous step.
6. Now, select your **Display name** and click **OK**.
7. Click **Next**.
8. Skip **What are the steps to fix the issue? (optional)** and click **Save**.
9. On the SLO page, switch back to the **Service level indicator** tab. It should not display our alert as a red dotted line.
10. Once again, clicking the line will zoom in the view. In the upper-right corner of the page, click **Auto Refresh** so the charts update automatically.

Trigger the alert

Modify our application and trigger the alert.

1. Switch back to your Cloud Shell view and **Open Editor**, if it's not already displayed, and re-open index.js.
2. Scroll to the /random-error route found at approximately line 126 and modify the value next to Math.random from 1000 to 20.

So instead of generating an error every 1000 requests, we are not going to get an error every 20 requests. That will drop our availability from 99.9$ to about 95%, which should trigger the alert.

1. Close the Cloud Shell code editor and switch to the terminal window.

You have two tabs, one that's running the test loop and one that's standard.

1. In the standard (non-busy) tab, redeploy the change to App Engine:

gcloud app deploy

Copied!

content\_copy

1. When prompted, type **y** and press **Enter**.
2. Once the redeploy completes, switch to the tab running the test loop and verify the uptick in errors.
3. Switch back to the your Service Monitoring page and in the upper-right corner, verify a green check next to auto refresh.
4. Verify that your SLO is expanded and that you can see the Service level indicator.

After a few minutes, the SLI value and chart should show clearly the decrease in performance down to about the 95% level. Within a few minutes, you should also receive the alert notification email.

**Note:**You may see your error budget quickly drop disproportionately. The error budget calculation is made over the whole SLO window, which should be a rolling period of 7 days, but because you just started the application, your total dataset is very small, thus causing the SLO interface to display a much larger decrease in your error budget than is really happening.

If you fixed the problem, the error budget would rapidly fill back up and you would see you actually have budget remaining, though it might take a couple of days to show that.

Click *Check my progress* to verify the objective.

Assessment Completed!

Create an SLO and tie an alert to the SLO

Check my progress

*Assessment Completed!*

Congratulations! You used Service Monitoring to create an availability related SLO and corresponding alert. Nice job.

**End your lab**