**Monitoring Applications in Google Cloud**

Toggle Lab Panel



**Monitoring Applications in Google Cloud**

1 hour 30 minutes Free

Rate Lab

**Overview**

In this lab, you will deploy an application to Google Cloud and then use the tools provided by Google Cloud to monitor it. You will use Cloud Logging, Trace, Profiler, and dashboards and create uptime checks and alerting policies.

**Objectives**

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

* Download a sample app from Github
* Deploy an application to App Engine
* Examine the Cloud logs
* View Profiler information
* Explore Cloud Trace
* Monitor resources using dashboards
* Create uptime checks and alerts

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

1. Make sure you signed into Qwiklabs using an **incognito window**.
2. Note the lab's access time (for example, img/time.pngand make sure you can finish in that time block.

There is no pause feature. You can restart if needed, but you have to start at the beginning.

1. When ready, click img/start_lab.png.
2. Note your lab credentials. You will use them to sign in to Cloud Platform Console. 
3. Click **Open Google Console**.
4. Click **Use another account** and copy/paste credentials for **this** lab into the prompts.

If you use other credentials, you'll get errors or **incur charges**.

1. Accept the terms and skip the recovery resource page.

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

**Task 1: Download a sample app from Github**

Download a sample application from GitHub and preview it in Cloud Shell.

1. In the Cloud Console, click **Activate Cloud Shell** (Cloud Shell).
2. If prompted, click **Continue**.
3. To create a folder called gcp-logging, run the following command:

mkdir gcp-logging

1. Change to the folder you just created:

cd gcp-logging

1. Clone a simple Python Flask app from Github:

git clone https://GitHub.com/GoogleCloudPlatform/training-data-analyst.git

1. Change to the deploying-apps-to-gcp folder:

cd training-data-analyst/courses/design-process/deploying-apps-to-gcp

1. In Cloud Shell, click **Launch code editor** (Cloud Shell Editor).
2. Expand the **gcp-logging/training-data-analyst/courses/design-process/deploying-apps-to-gcp** folder in the navigation pane, and then click **main.py** to open it.
3. Add the following import statement at the top of the file (*line 2*):

import googlecloudprofiler

**Note**: Profiler allows you to monitor the resources your applications use. For more information, see <https://cloud.google.com/profiler/docs/>.

1. After the **main()** function, add the following code snippet to start Profiler (*after line 11*):

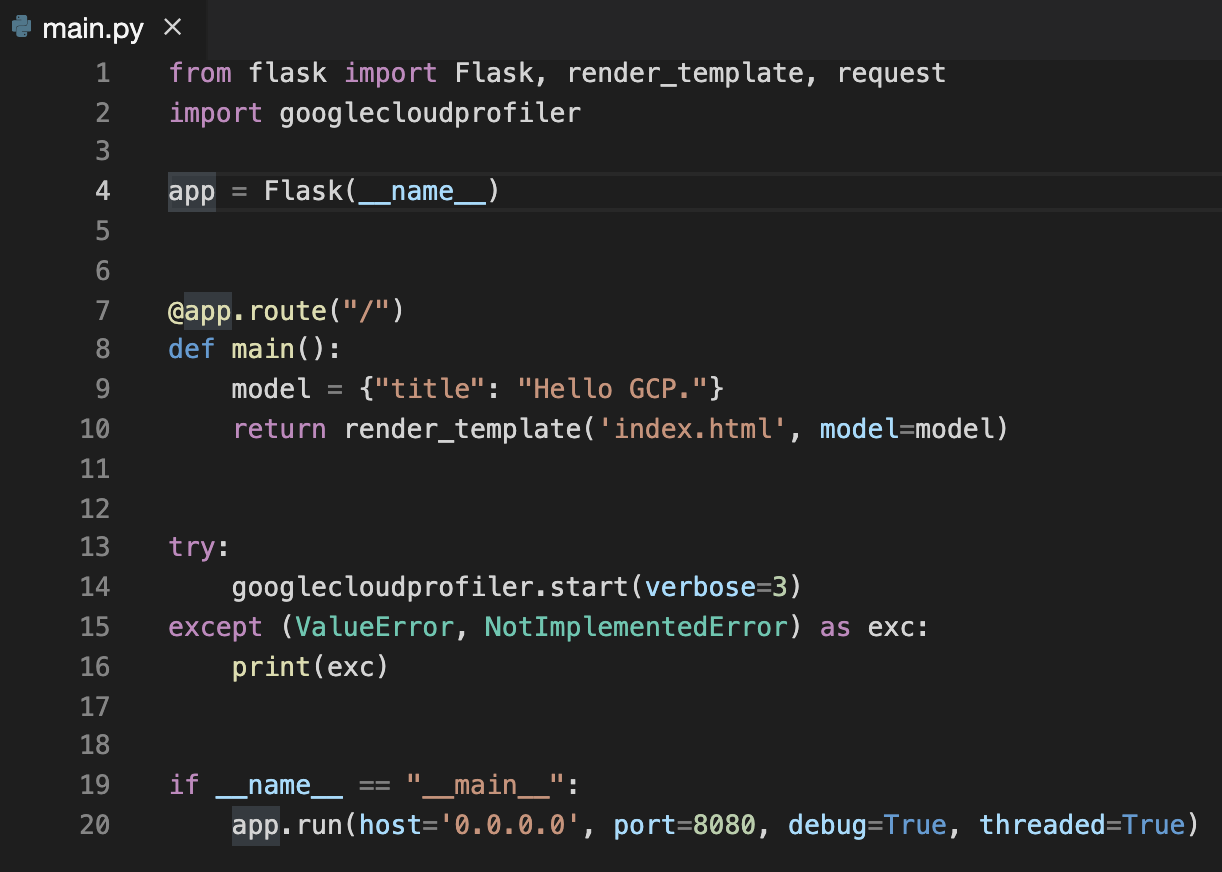
try:

googlecloudprofiler.start(verbose=3)

except (ValueError, NotImplementedError) as exc:

print(exc)

Profiler will continuously report application metrics. Your code should look like this:

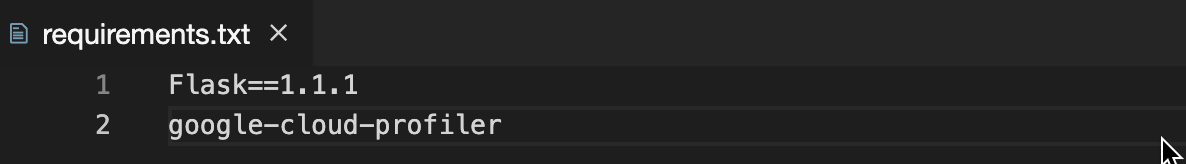


**Note**: This code simply turns Profiler on. Once on, Profiler starts reporting application metrics to Google Cloud.

1. You also have to add the Profiler library to your requirements.txt file. Open that file in the code editor and add the following:

google-cloud-profiler

The file should look like this:



1. Profiler has to be enabled in the project. In Cloud Shell, enter the following command:

gcloud services enable cloudprofiler.googleapis.com

1. To test the program, enter the following commands to install requirements and then start the program:

sudo pip3 install -r requirements.txt

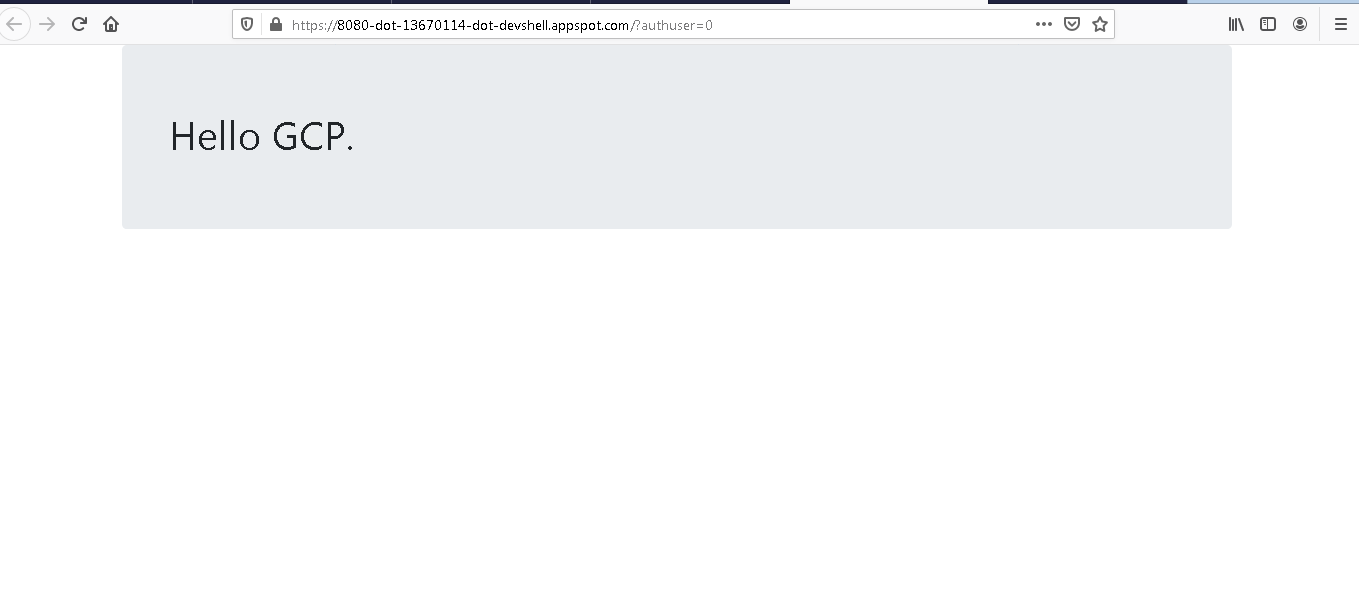
python3 main.py

1. To see the program running, click **Web Preview** (Web Preview) in the Google Cloud Shell toolbar. Then select **Preview on port 8080**.

The program should be displayed in a new browser tab.

1. In Cloud Shell, type Ctrl+C to stop the program.

Click *Check my progress* to verify the objective. Enable the Profiler



**Task 2: Deploy an application to App Engine**

Now you will deploy the program to App Engine and use Google Cloud tools to monitor it.

1. In the Cloud Shell code editor, in the Explorer pane, select the **gcp-logging/training-data-analyst/courses/design-process/deploying-apps-to-gcp** folder.
2. On the **File** menu, click **New File**, and then name the file **app.yaml**.
3. Paste the following into the file you just created:

runtime: python37

1. Save your changes.
2. In a project, an App Engine application has to be created. This is done just once using the gcloud app create command and specifying the region where you want the app to be created. Enter the following command:

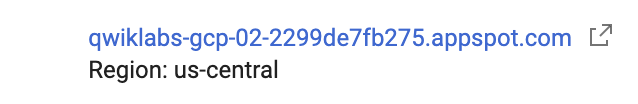
gcloud app create --region=us-central

1. Now deploy your app with the following command:

gcloud app deploy --version=one --quiet

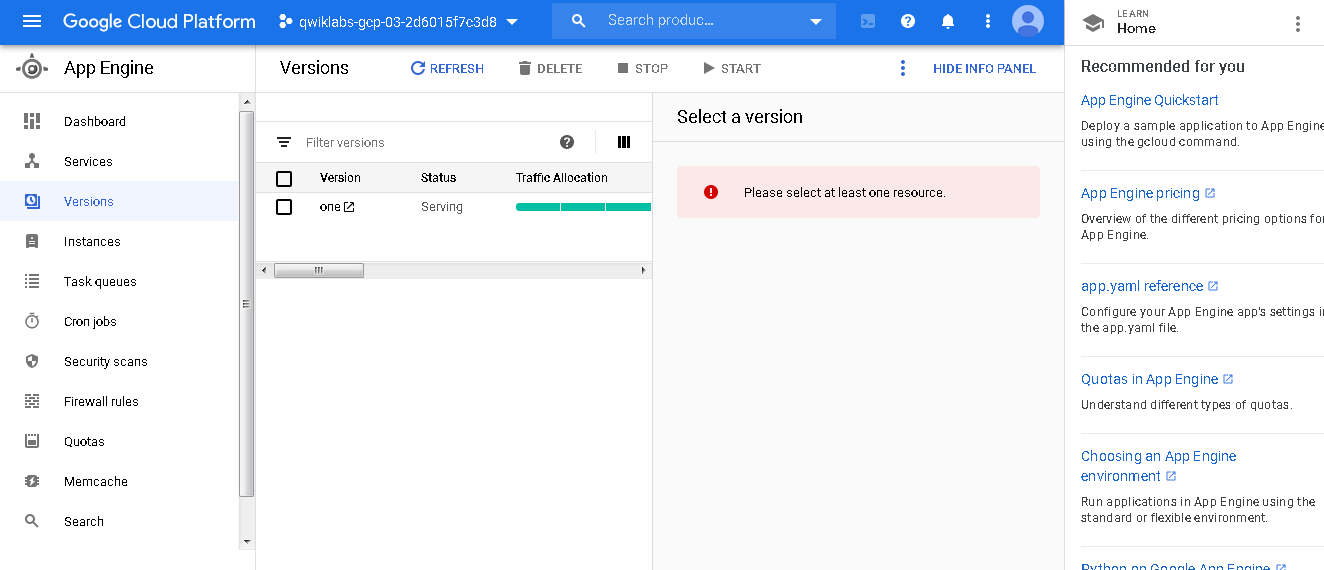
**Note**: This command will take a couple of minutes to complete. Wait for it to complete before continuing.

1. In the Cloud Console, on the **Navigation menu** (Navigation menu), click **App Engine** > **Dashboard**. The upper-right corner of the dashboard should display a link to your application similar to this:



**Note**: By default, the URL to an App Engine instance is in the form of https://project-id/appspot.com.

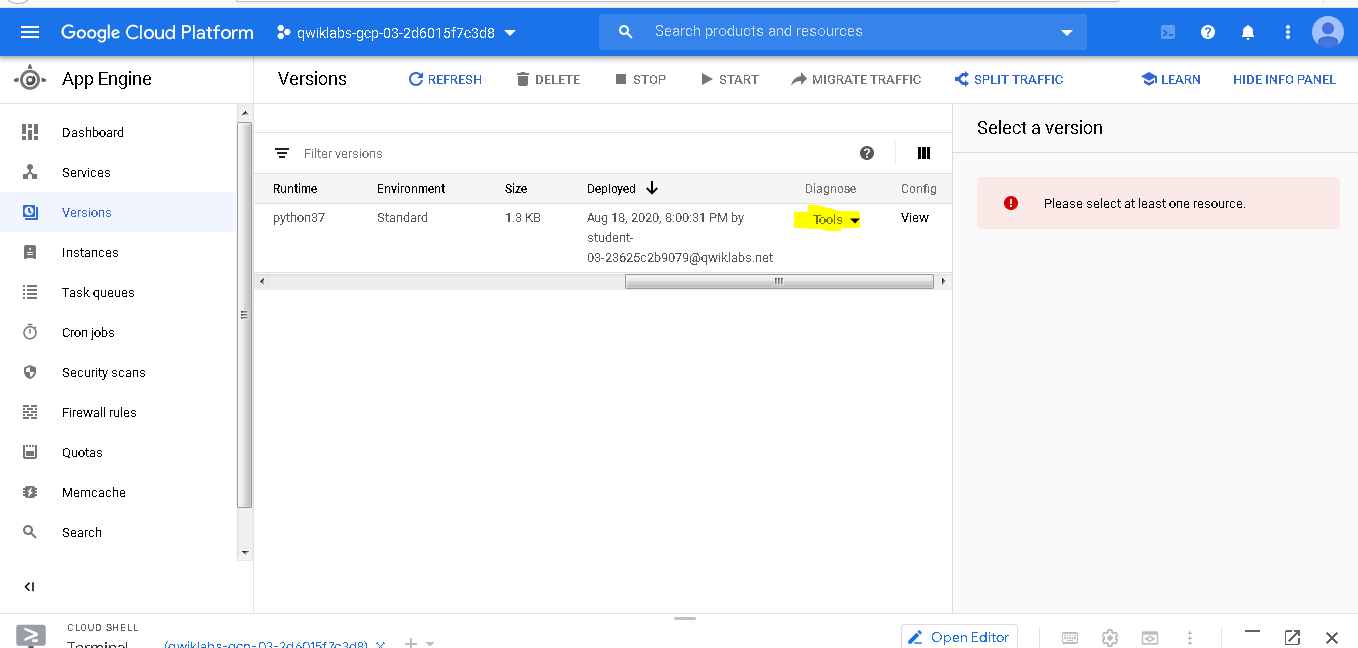
1. Click on the link to test your program.
2. Refresh your browser a few times to make some requests.



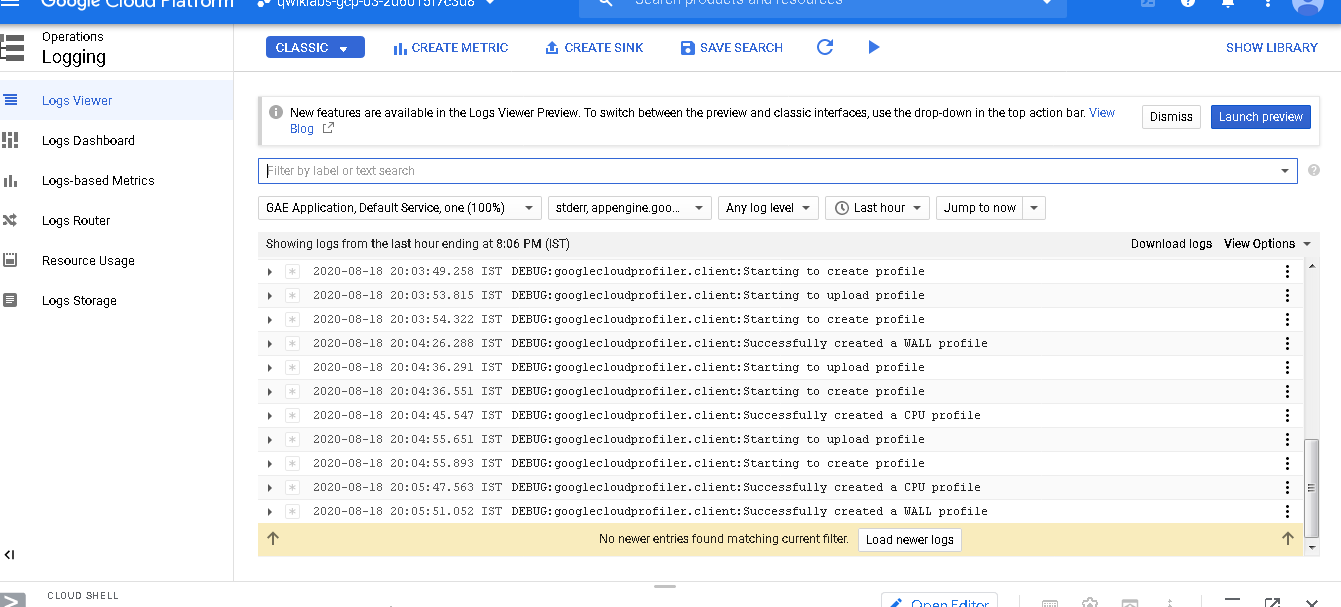
Click *Check my progress* to verify the objective. Deploy an application to App Engine

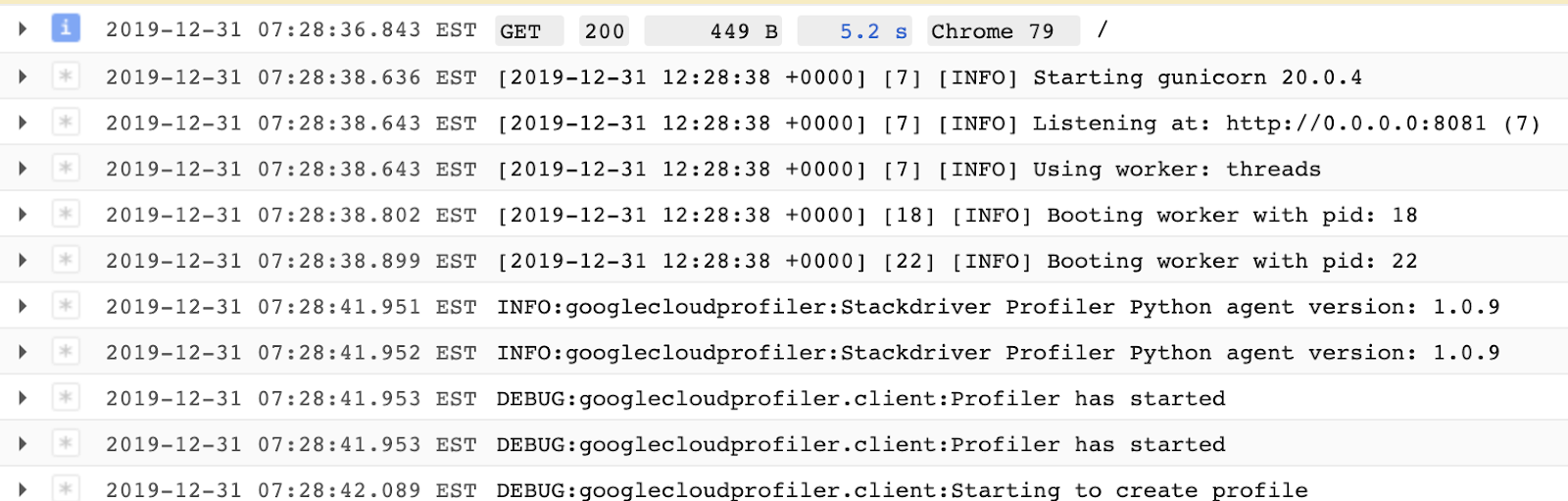
**Task 3: Examine the Cloud logs**

1. Return to the Console and click the **App Engine** > **Versions** link on the left.
2. Click **Tools** in the **Diagnose** column of the table, and then click **Logs**.



1. The logs should indicate that Profiler has started and profiles are being generated. If you get to this point too quickly, wait a minute and click **Refresh**

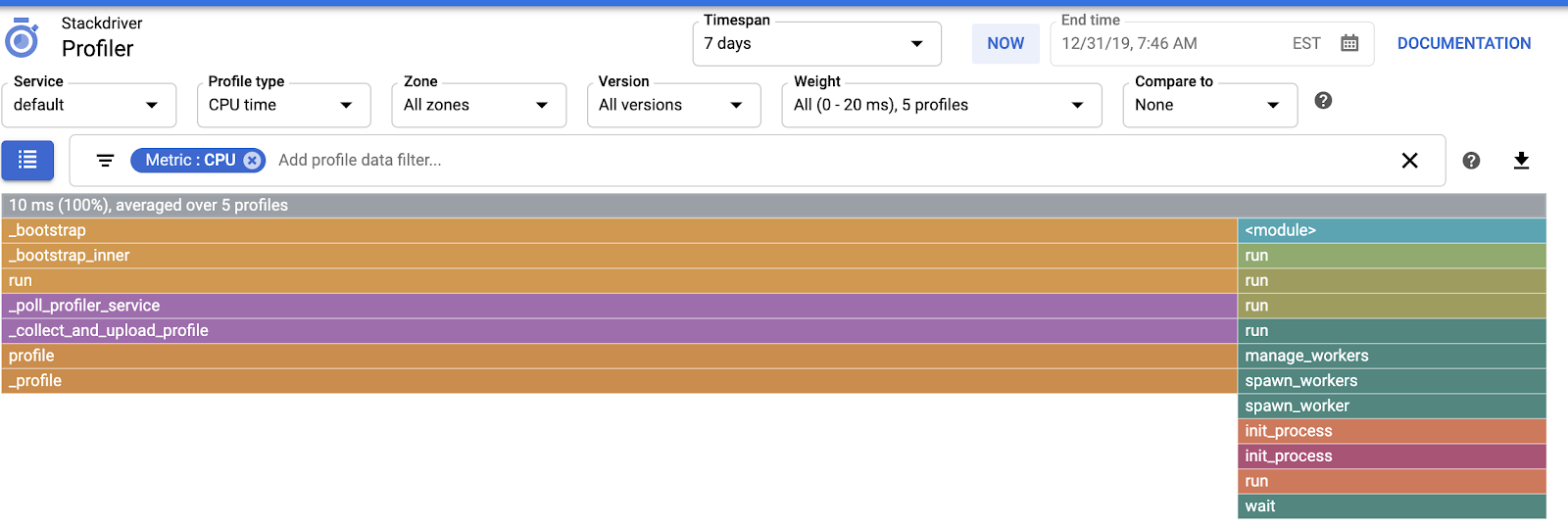
.



Click *Check my progress* to verify the objective. Examine the Cloud logs

**Task 4: View Profiler information**

1. In the Cloud Console, on the **Navigation menu** (Navigation menu), click **Profiler**. The screen should look similar to this:



**Note**: The gray bar at the top represents the total amount of CPU time used by the program. The bars below that represent the amount of CPU time used by the program's functions relative to the total. At this point, there is no traffic, so the chart is not very interesting. Throw some load at the application.

1. On the **Navigation menu**, click **Compute Engine**.
2. Click **Create** to create a virtual machine.
3. Change the region to someplace other than us-central1 (*the App Engine app is in us-central1*). Accept all the rest of the defaults and click **Create**.
4. When the VM is ready, click **SSH** to log in to it.
5. You will generate some traffic to your App Engine app using the web testing tool called Apache Bench. Enter the following commands to install it:

sudo apt update

sudo apt install apache2-utils -y

1. Enter the following command to generate some traffic to your App Engine application:

ab -n 1000 -c 10 https://<your-project-id>.appspot.com/

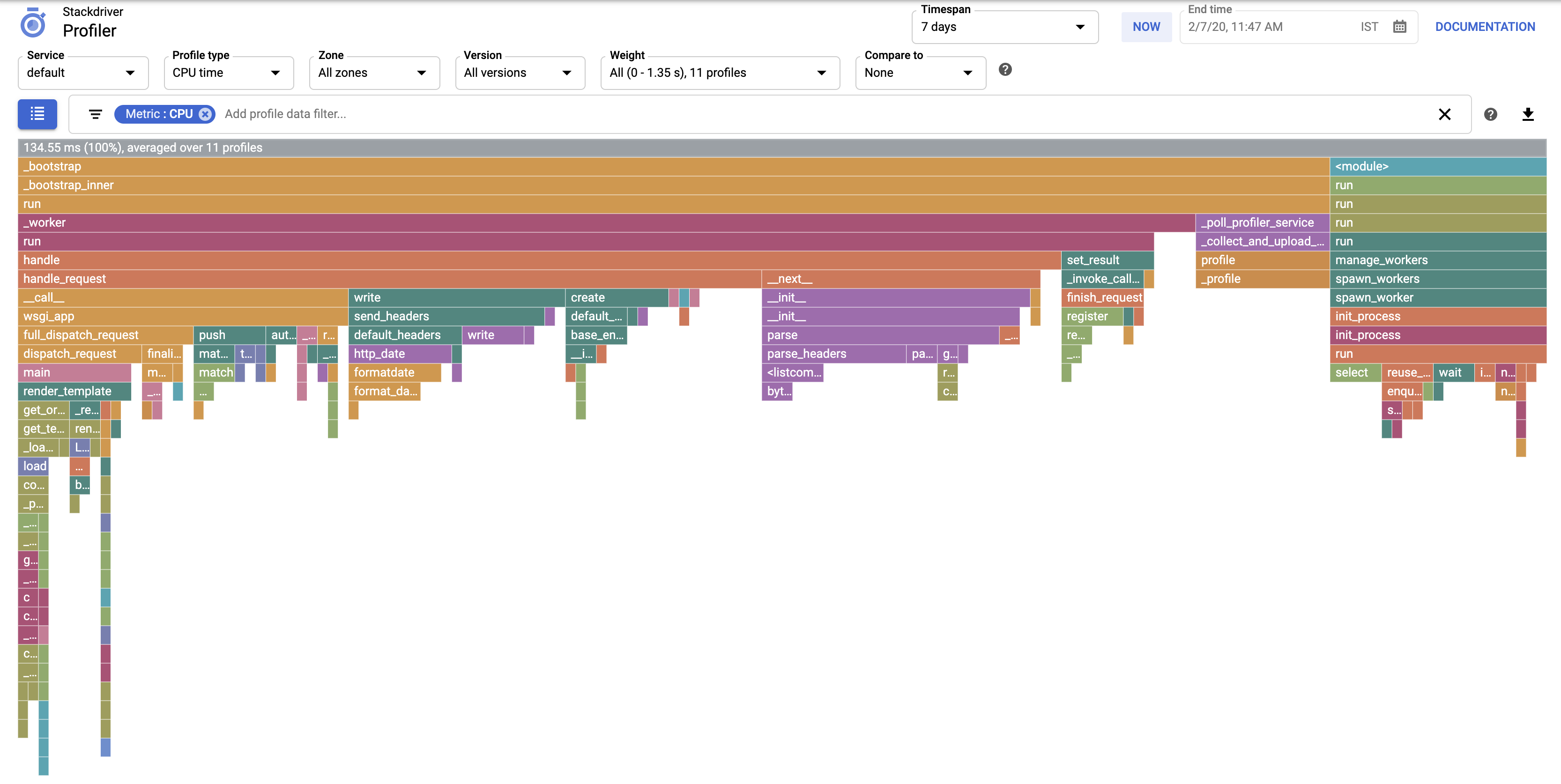
The command will make a thousand requests, 10 at a time, to your application.

**Note**: You have to change the URL to point to your application. Recall that you can find the URL in the App Engine Dashboard. It is also on the browser tab you used to test your app, if you haven't closed it. Also, make sure you insert a slash (/) at the end of the URL.

1. When the requests are finished, on the **Navigation menu**, click **Profiler**.

Now there is a more interesting chart. Each bar represents a function. The width of the bars represents how much CPU time each function consumed.

The Profiler is a way developers can track down parts of a program that are consuming too many resources.



**Task 5: Explore Cloud Trace**

1. Every request to your application is added to the **Trace** list. On the **Navigation menu** (Navigation menu), click **Trace**.

The overview screen shows recent requests and allows you to create reports to analyze traffic. Because your program is new and has only one page, it's not very interesting, but in a real app there would be lots of useful information.

1. Click **Trace list**.

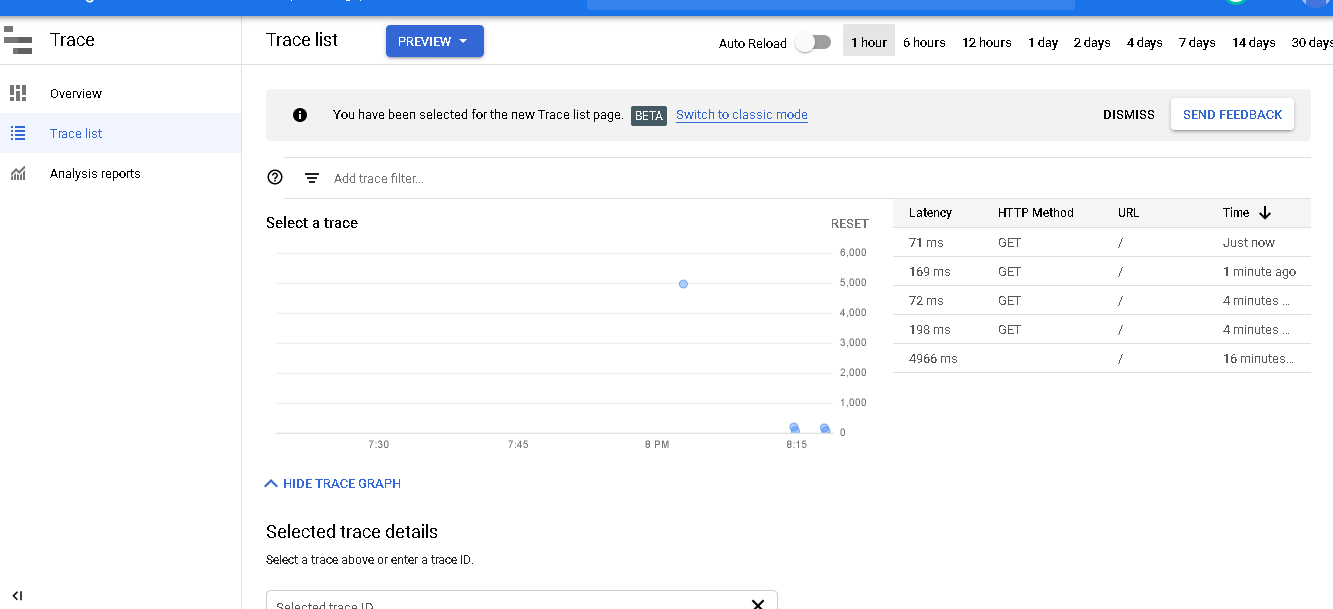
This shows a history of requests and their latency. Again, it's not very exciting because the application hasn't been running for very long. The chart in the upper-left plots requests and how long they took. The table to the right shows a list of requests. If you select a request, more detail will be displayed at the bottom of the screen.

1. Return to the SSH window where you entered the Apache Bench command previously.
2. Enter the ab command again:

ab -n 1000 -c 10 https://<your-project-id>.appspot.com/

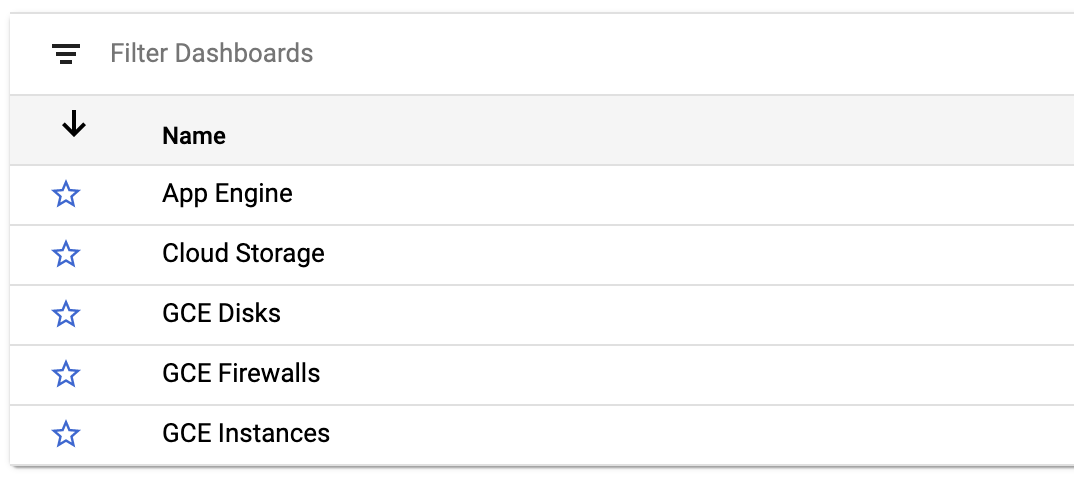
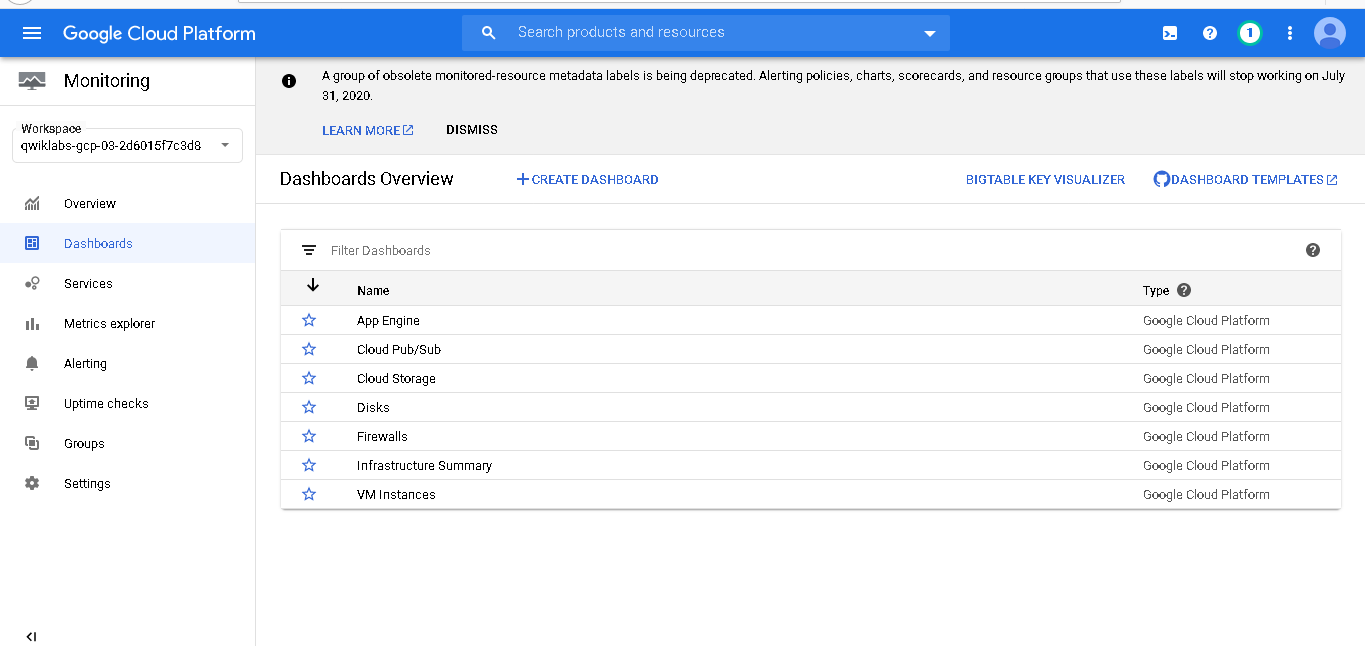
You can also experiment with different values for the -n and -c parameters.

1. Repeat this a couple of times, and then return to the Trace list page.



**Task 6: Monitor resources using Dashboards**

1. In the Cloud Console, on the **Navigation menu** (Navigation menu), click **Monitoring**.
2. In the left pane, click **Dashboards**. Cloud Monitoring analyzes the resources used in your projects and generates some default dashboards for you. In this exercise you have used App Engine and Compute Engine virtual machines, so a table similar to the one shown below should be displayed:

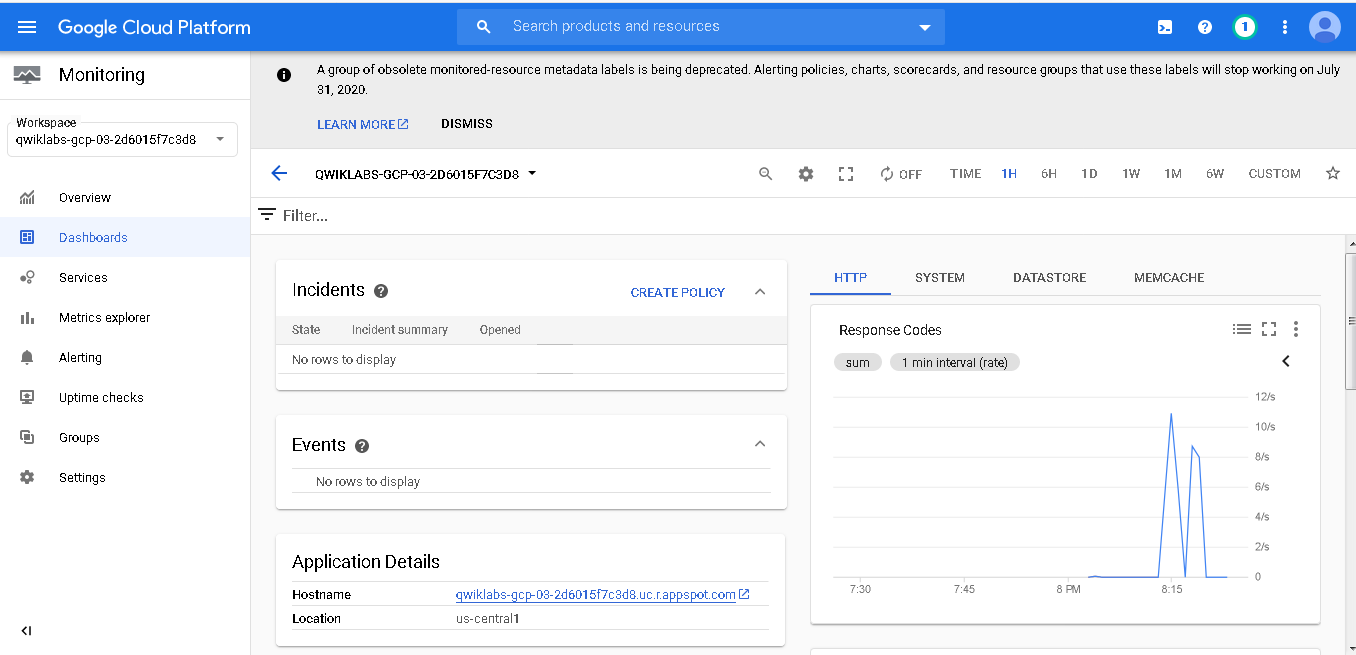


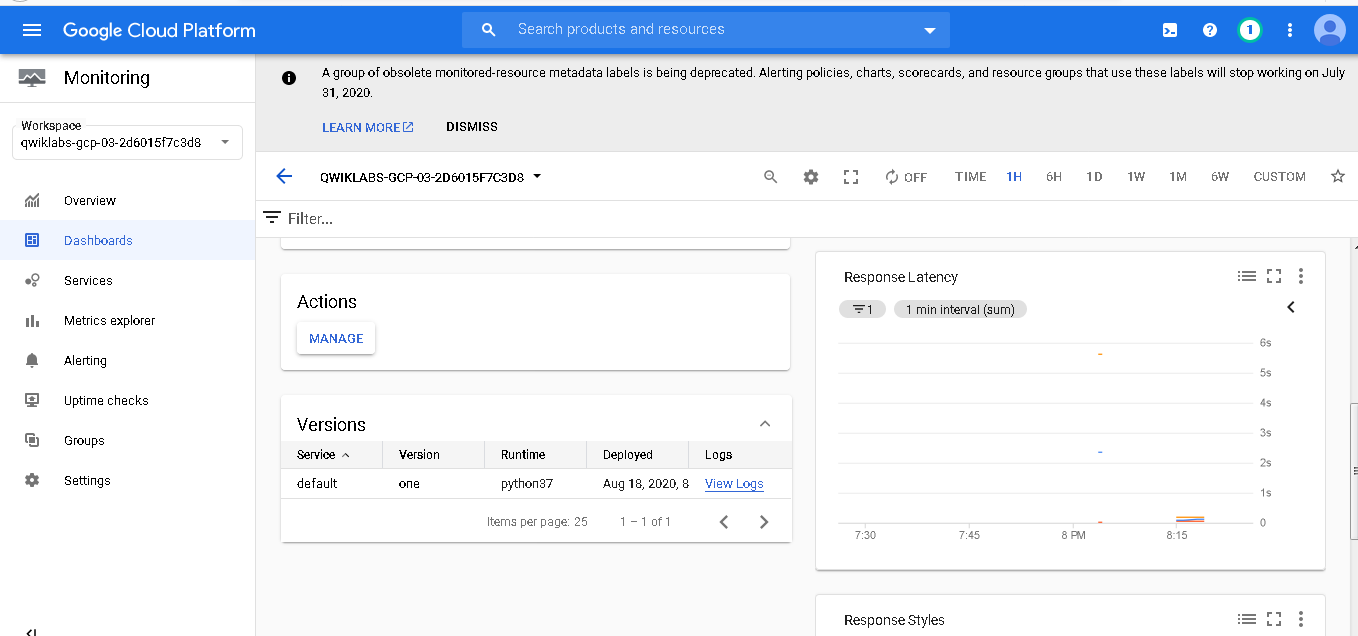
3.Click on the **App Engine** dashboard, and then select your project name. A dashboard of pertinent information for your App Engine application will appear.

1. In the left pane, click **Dashboards**.
2. Click on the **GCE Instances** dashboard, and then select your instance. A dashboard for your VM will appear.

If you don't see GCE Instances right away, wait a minute and refresh your browser.

1. Optionally, return to the Dashboards page and click the **Create Dashboard** link. Try to create a custom dashboard.



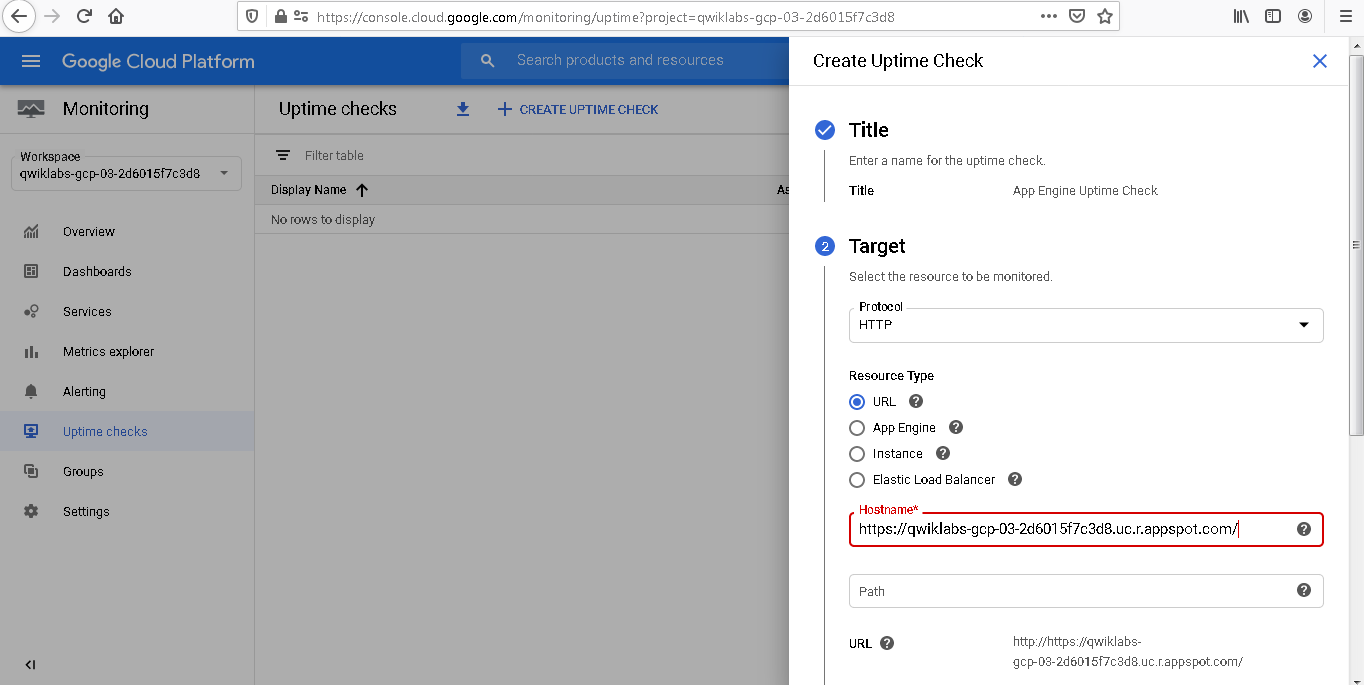


**Task 7: Create uptime checks and alerts**

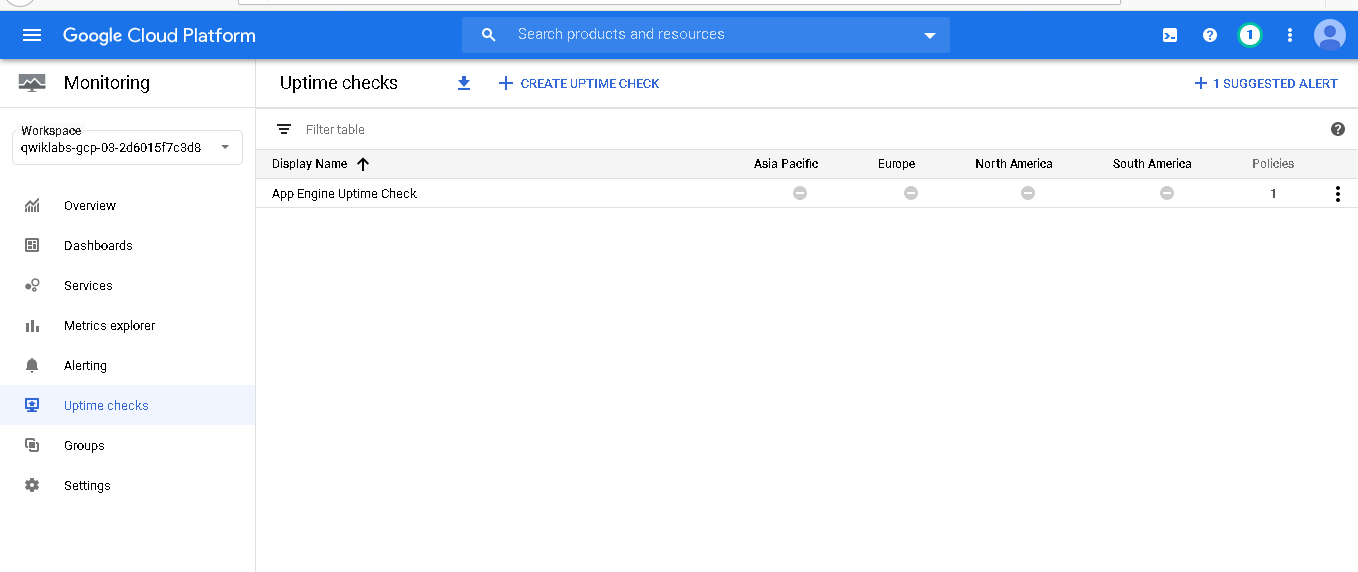
1. In the left pane, click **Uptime checks**, and then click the **Create Uptime Check** link at the top. Fill out the form as follows:

|  |  |
| --- | --- |
| **Property** | **Value** |
| **Title** | App Engine Uptime Check |
| **Check Type** | HTTPS |
| **Resource Type** | URL |
| **Hostname** | <your-project-id>.appspot.com |
| **Path** | / |
| **Check every** | 1 minute |

1. Click **Test**, and then click **Save**.

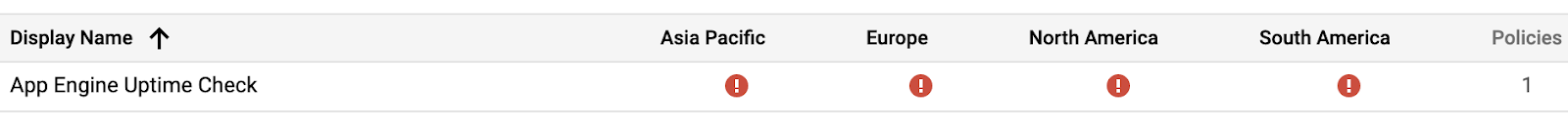


1. When you are asked whether you want to create an Alerting Policy, do so.
2. Name the policy **Uptime Check Policy**, and then click **Save**.
3. On the resulting Alerts screen, name your alerting policy **Uptime Check Alert**.
4. Click **Add Notification channel**. Send yourself an email when the alert happens.
5. Optionally, enter a message in the Documentation box, and then click **Save**.

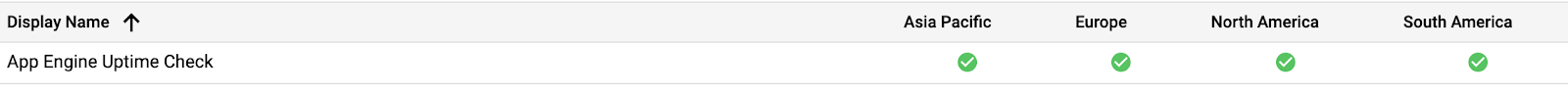


Click *Check my progress* to verify the objective. Create uptime checks and alerts

1. Disable the application to see whether your uptime check and alerting policy work. In the Cloud Console, on the **Navigation menu**, click **App Engine**.
2. Click **Settings**.
3. Click **Disable application**. Follow the instructions to disable the application.
4. Return to the **App Engine Dashboard** and test the URL. It shouldn't work anymore.
5. Return to **Monitoring** and then click **Uptime checks**. Your uptime check should be failing. *If you get there too fast, wait a minute and click refresh*.



1. Click **Alerting**. An incident should have been fired.
2. Check your email. You should get a message from Cloud Monitoring.
3. Return to **App Engine Settings** and re-enable your application. Then return to the **Uptime checks** page. The uptime check should be working again. *If not, wait a minute and then click refresh*.



1. Return to the **Alerting** page. Your incident should be resolved. *As before, you might have to wait a minute and then click refresh*.
2. Check your email again. You should get a second email indicating that the alert recovered.
3. To make sure you don't get any emails after the project is deleted, delete your notification channel. At the top of the **Alerting** page, click **Edit Notification Channels**.
4. Find your email address and click the trash can icon to delete it.
5. Now click **Uptime checks** and delete your App Engine Uptime check.

**Congratulations!**

In this lab, you deployed an application to Google Cloud and then used the tools provided by Google Cloud to monitor it. You used Cloud Logging, Trace, Profiler, and dashboards and created uptime checks and alerting policies.

**End your lab**