

**—/20**

**Checkpoints**

Deploy the application

Check my progress

/ 10

Configure an uptime check and alert policy in Cloud Monitoring

Check my progress

/ 5

Create a test VM with ApacheBench

Check my progress

/ 5

**Deployment Manager - Full Production [ACE]**

1 hour 30 minutes Free

Rate Lab

**Overview**

In this lab, you will launch a service using an infrastructure orchestration tool called Deployment Manager and monitor the service using Cloud Monitoring. In Cloud Monitoring, you will set up basic black box monitoring with a Cloud Monitoring dashboard and establish an Uptime Check (alert notification) to trigger incident response.

More specifically, you will:

1. Install and configure an advanced deployment using Deployment Manager sample templates.
2. Enable Cloud monitoring.
3. Configure Cloud Monitoring Uptime Checks and notifications.
4. Configure a Cloud Monitoring dashboard with two charts, one showing CPU usage and the other ingress traffic.
5. Perform a load test and simulate a service outage.

Cloud Monitoring provides visibility into the performance, uptime, and overall health of cloud-powered applications. Cloud Monitoring collects metrics, events, and metadata from Google Cloud Platform, Amazon Web Services, hosted uptime probes, application instrumentation, and a variety of common application components including Cassandra, Nginx, Apache Web Server, Elasticsearch, and many others. Cloud Monitoring ingests that data and generates insights via dashboards, charts, and alerts. Cloud Monitoring alerting helps you collaborate by integrating with Slack, PagerDuty, HipChat, Campfire, and more.

**Objectives**

In this lab, you will learn to:

* Launch a cloud service from a collection of templates.
* Configure basic black box monitoring of an application.
* Create an uptime check to recognize a loss of service.
* Establish an alerting policy to trigger incident response procedures.
* Create and configure a dashboard with dynamically updated charts.
* Test the monitoring and alerting regimen by applying a load to the service.
* Test the monitoring and alerting regimen by simulating a service outage.

**Setup and Requirements**

**Qwiklabs setup**

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.

**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 GCP resources.

1. In GCP 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 Platform. It comes pre-installed on Cloud Shell and supports tab-completion.

You can list the active account name with this command:

gcloud auth list

Output:

Credentialed accounts:

- <myaccount>@<mydomain>.com (active)

Example output:

Credentialed accounts:

- google1623327\_student@qwiklabs.net

You can list the project ID with this command:

gcloud config list project

Output:

[core]

project = <project\_ID>

Example output:

[core]

project = qwiklabs-gcp-44776a13dea667a6

Full documentation of **gcloud** is available on [Google Cloud gcloud Overview](https://cloud.google.com/sdk/gcloud) .

**Create a virtual environment**

Execute the following command to download and update the packages list.

sudo apt-get update

Python virtual environments are used to isolate package installation from the system.

sudo apt-get install virtualenv

If prompted [Y/n], press Y and then Enter.

virtualenv -p python3 venv

Activate the virtual environment.

source venv/bin/activate

**Clone the Deployment Manager Sample Templates**

Google provides a robust set of sample Deployment Manager templates that you can learn from and build upon.

To clone the repository, enter the following commands in Cloud Shell Command Line to create a directory to hold the Deployment Manager sample templates.

mkdir ~/dmsamples

Go to that directory.

cd ~/dmsamples

Clone the repository to the directory you just made.

git clone https://github.com/GoogleCloudPlatform/deploymentmanager-samples.git

**Example Output:**

remote: Counting objects: 1917, done.

remote: Compressing objects: 100% (31/31), done.

remote: Total 1917 (delta 11), reused 29 (delta 7), pack-reused 1874

Receiving objects: 100% (1917/1917), 426.86 KiB | 0 bytes/s, done.

Resolving deltas: 100% (1060/1060), done.

**Explore the Sample Files**

We just downloaded a collection of sample templates to our directory, let's dive in and explore some of them.

**List the example templates**

Run the following commands to navigate to and list the version2 examples:

cd ~/dmsamples/deploymentmanager-samples/examples/v2

ls

You should see something like this:

access\_context\_manager cloud\_functions common folder\_creation iam\_custom\_role internal\_lb\_haproxy quick\_start ssl vm\_with\_disks

access\_control cloudkms container\_igm gke igm-updater metadata\_from\_file regional\_igm step\_by\_step\_guide vpn\_auto\_subnet

bigtable cloud\_router container\_vm ha-service image\_based\_igm nodejs saltstack template\_modules waiter

build\_configuration cloudsql custom\_machine\_type htcondor instance\_pool nodejs\_l7 single\_vm vlan\_attachment

cloudbuild cloudsql\_import dataproc iam internal\_lb project\_creation sqladmin vm\_startup\_script

Not all of the subdirectories are independent projects. For example, the directory named **common** contains templates that are used by several of the other projects. If you are studying independently later, use the README files as a guides.

The nodejs directory contains everything you'll need to build this lab. Note that there is a nodejs directory and a nodejs\_l7directory. Use nodejs.

**List and examine the Nodejs deployment**

Navigate to and list the version2 examples:

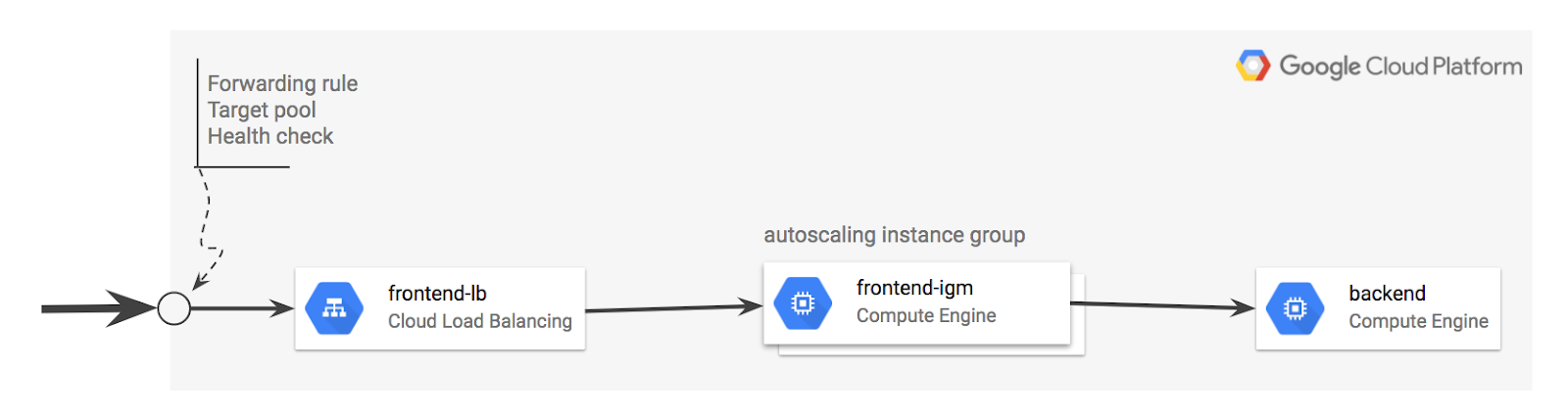
cd nodejs/python

ls

**Example Output:**

frontend.py frontend.py.schema nodejs.py nodejs.py.schema nodejs.yaml

The main deployment manager configuration file is nodejs.yaml. It makes use of templates to generate infrastructure. The rest of the files are templates. Templates use variables defined in the nodejs.yaml configuration file to produce customized results.



**frontend.py**

frontend.py includes frontend.py.schema, which creates an instance template based on container\_instance\_template.py. This template is used to create a managed instance group and an autoscaler. The template also creates a network load balancer that has a forwarding rule with a single public IP address. It will also create:

* A target pool that refers to the managed instance group.
* A health check attached to the target pool.

**nodejs.py**

nodejs.py includes nodejs.py.schema, which brings the frontend and backend templates together.

* Note that the frontend is frontend.py.
* The backend is /common/python/container\_vm.py.
* This is a VM running a Docker container with MySQL, so it doesn't require a custom template.

**Other files**

* /common/python/container\_instance\_template.py
* /common/python/container\_vm.py
* /common/python/container\_helper.py

**Customize the Deployment**

Now that you've downloaded and reviewed the nodejs Deployment Manager template, let's start customizing the deployment.

**Specify the zone**

The nodejs.yaml file requires a zone, and you'll now add one to the file.

1. Enter the following command to open the list of zones:

gcloud compute zones list

Copy the name of a zone for the configuration file to use.

1. Open nodejs.yaml in nano so you can edit the zone value:

nano nodejs.yaml

The nodejs.yaml file contents:

resources:

- name: nodejs

type: nodejs.py

properties:

zone: ZONE\_TO\_RUN

1. Replace ZONE\_TO\_RUN with a zone name that's near you, then exit nano and save the file.

This example shows ZONE\_TO\_RUN set to us-east1-d

resources:

- name: nodejs

type: nodejs.py

properties:

zone: us-east1-d

**Modify the maximum number of instances in the instance group**

Edit the nodejs.py file.

1. Enter this command to open nodejs.py in nano:

nano nodejs.py

1. Verify current scaling limit for frontend in nodejs.py

{

'name': frontend,

'type': 'frontend.py',

'properties': {

'zone': context.properties['zone'],

'dockerImage': 'gcr.io/deployment-manager-examples/nodejsservice',

'port': application\_port,

# Define the variables that are exposed to container as env variables.

'dockerEnv': {

'SEVEN\_SERVICE\_MYSQL\_PORT': mysql\_port,

'SEVEN\_SERVICE\_PROXY\_HOST': '$(ref.' + backend

+ '.networkInterfaces[0].networkIP)'

},

# If left out will default to 1

'size': 2,

# If left out will default to 1

'maxSize': 20

}

},

Current scaling limit is 20 (refer maxSize).

1. Modify the maxSize and set it to 4:

{

'name': frontend,

'type': 'frontend.py',

'properties': {

'zone': context.properties['zone'],

'dockerImage': 'gcr.io/deployment-manager-examples/nodejsservice',

'port': application\_port,

# Define the variables that are exposed to container as env variables.

'dockerEnv': {

'SEVEN\_SERVICE\_MYSQL\_PORT': mysql\_port,

'SEVEN\_SERVICE\_PROXY\_HOST': '$(ref.' + backend

+ '.networkInterfaces[0].networkIP)'

},

# If left out will default to 1

'size': 2,

# If left out will default to 1

'maxSize': 4

}

},

Save the file and exit nano when you're done.

**Run the Application**

Now you'll use Deployment Manager to deploy the application and make it operational. This builds the infrastructure, but it won't allow traffic. After Deployment Manager sets up the infrastructure, you can apply service labels.

**Deploy the application**

Enter this command to name the application advanced-configuration and pass Deployment Manager the configuration file (nodejs.yaml).

gcloud deployment-manager deployments create advanced-configuration --config nodejs.yaml

Output:

The fingerprint of the deployment is PiYc6OsIFkWzQpCDklHvaA==

Waiting for create [operation-1529913842103-56f72d31872d9-90070017-aec5761d]...done.

Create operation operation-1529913842103-56f72d31872d9-90070017-aec5761d completed successfully.

NAME TYPE STATE ERRORS INTENT

advanced-configuration-application-fw compute.v1.firewall COMPLETED []

advanced-configuration-backend compute.v1.instance COMPLETED []

advanced-configuration-frontend-as compute.v1.autoscaler COMPLETED []

advanced-configuration-frontend-hc compute.v1.httpHealthCheck COMPLETED []

advanced-configuration-frontend-igm compute.v1.instanceGroupManager COMPLETED []

advanced-configuration-frontend-it compute.v1.instanceTemplate COMPLETED []

advanced-configuration-frontend-lb compute.v1.forwardingRule COMPLETED []

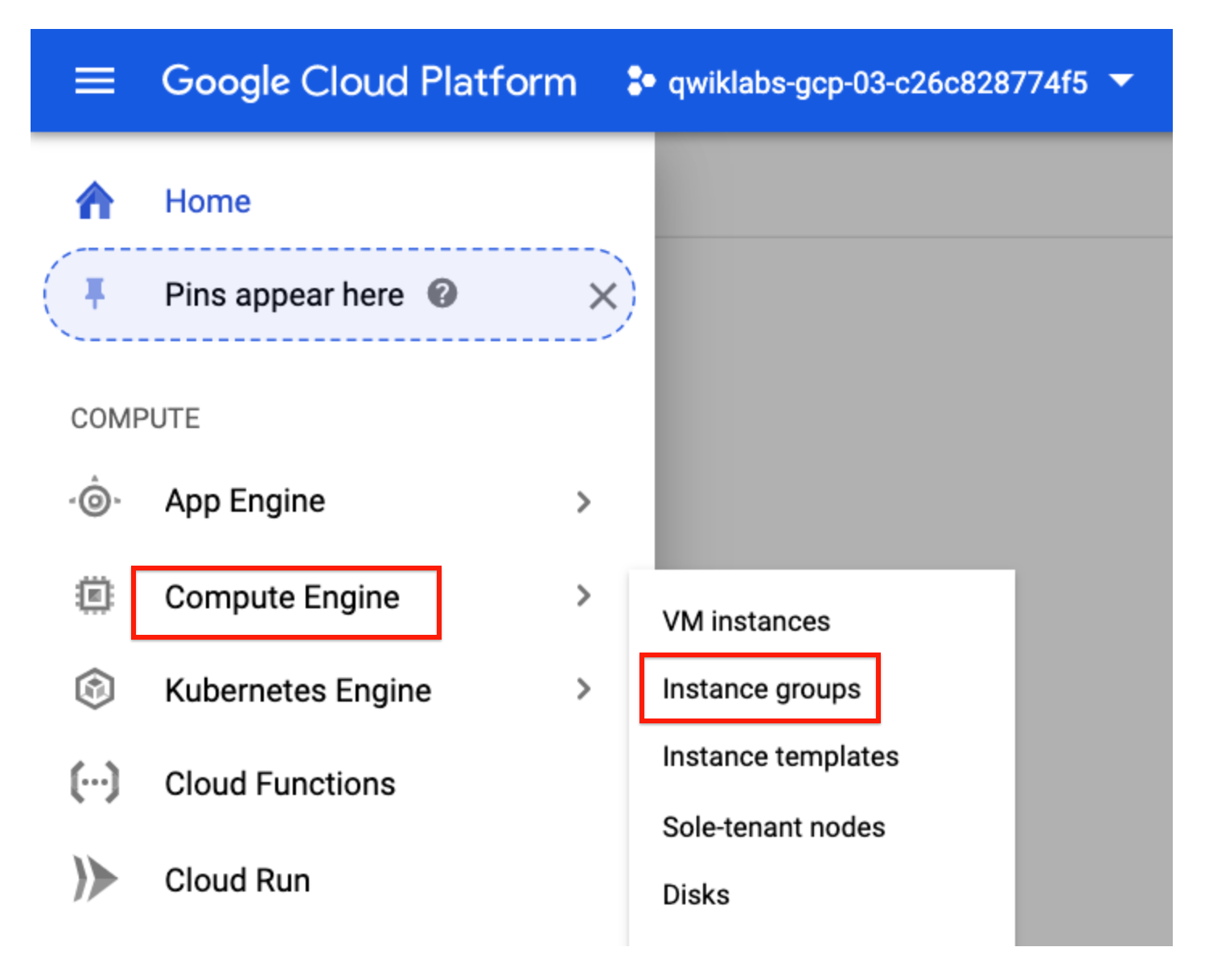
advanced-configuration-frontend-tp compute.v1.targetPool COMPLETED []

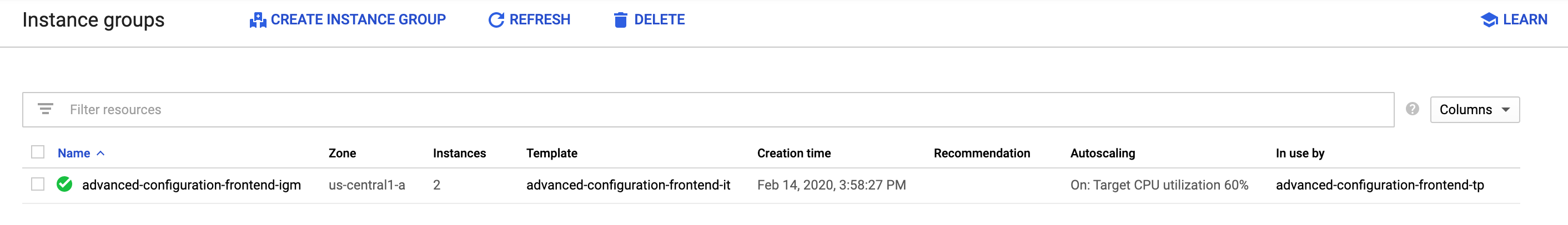
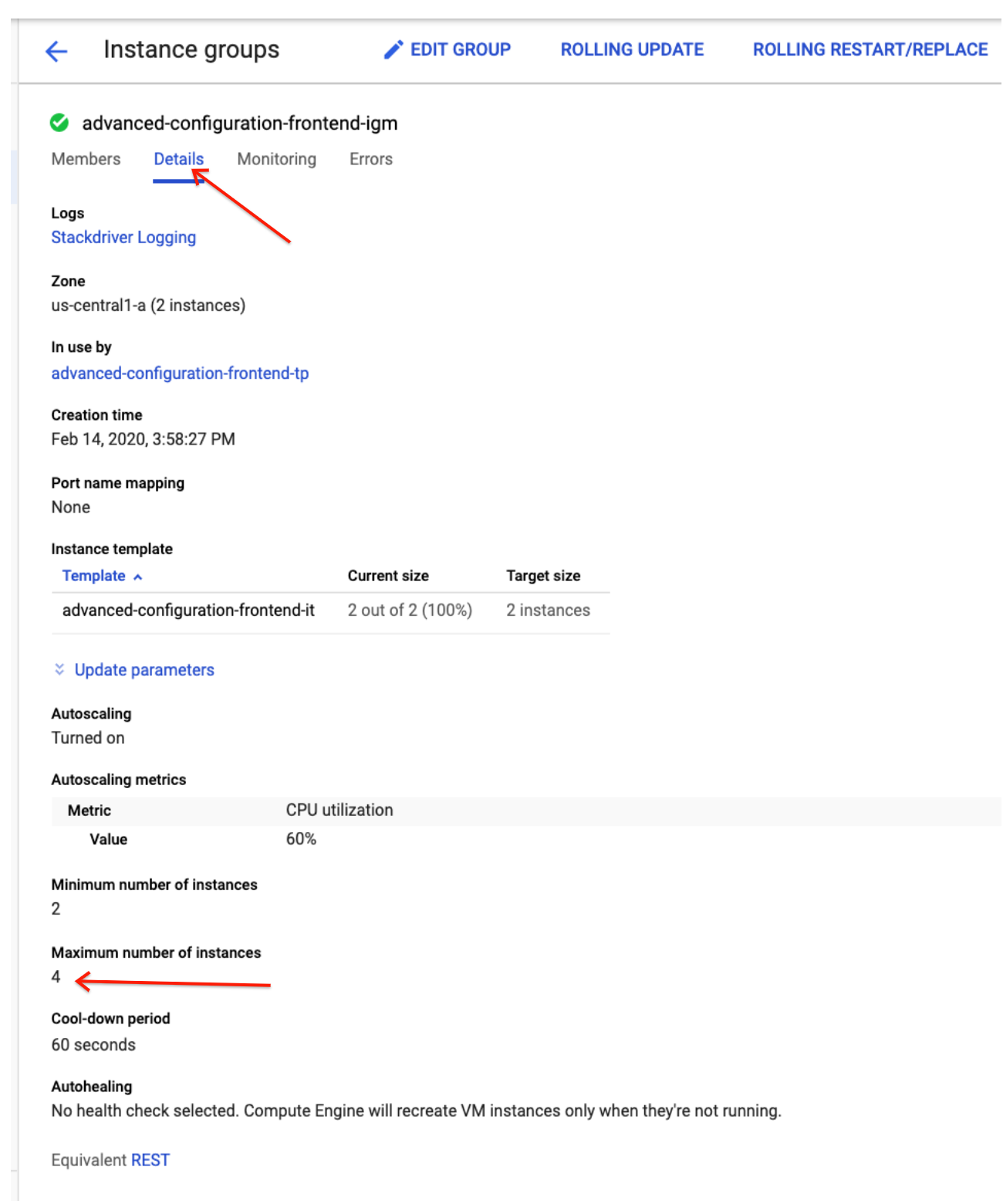
Click **Check my progress** below to check your lab progress.

Deploy the application

Confirm the limit for the maximum number of instances:

1. Go to **Compute Engine** > **Instance groups**.



1. Click on **advanced-configuration-frontend-igm**. 
2. Click on the **Details** tab, then verify the maximum number of instances. 

You'll see it has been set to 4.

**Verify that the application is operational**

The application takes a few minutes to start. You can view it in the Deployment Manager part of Cloud Console (**Navigation menu** > **Deployment Manager**), or you can see the instances in the Compute Engine part of Console (**Navigation menu** > **Compute Engine** > **VM**).

To verify that the application is running, you will open a browser to access port 8080 and view the service. Since the IP address was established dynamically when the Deployment Manager implemented the global forwarding rule (specified in the template), you'll need to find that address to test the application.

**Find the global load balancer forwarding rule IP address**

1. Enter the following command in the Cloud Command Line to find your forwarding IP address.

gcloud compute forwarding-rules list

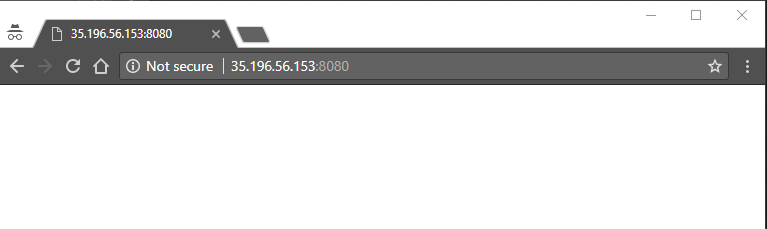
**Example output:**

369c66095dd87ce8.pngYour forwarding IP address is the IP\_ADDRESS listed in your output. Be sure to record it; you'll use it a few times in this lab.

1. Open a new window in your browser. In the address field, enter the following URL, replacing <your IP address> with your forwarding IP address:

http://<your IP address>:8080

You will see a blank page, similar to below, with your own IP address:



It may take several minutes for the service to become operational. If you get an error, such as a 404, wait about two minutes and try again.

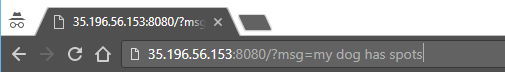
1. Now enter log information by entering a log message in your browser address line.

http://<your forwarding IP address>:8080/?msg=<enter\_a\_message>

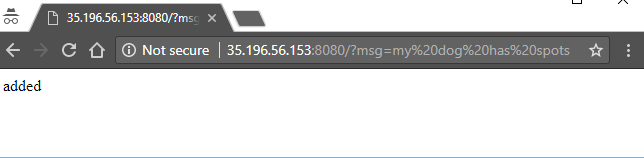
Replace enter\_a\_message with any kind of message.

These are log message examples - yours will have a different IP address:

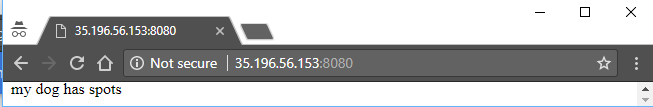
http://35.196.56.153:8080/?msg=my dog has spots



After you enter the log, the browser returns added.



1. View the log by going to http://<your IP address>:8080. For example:



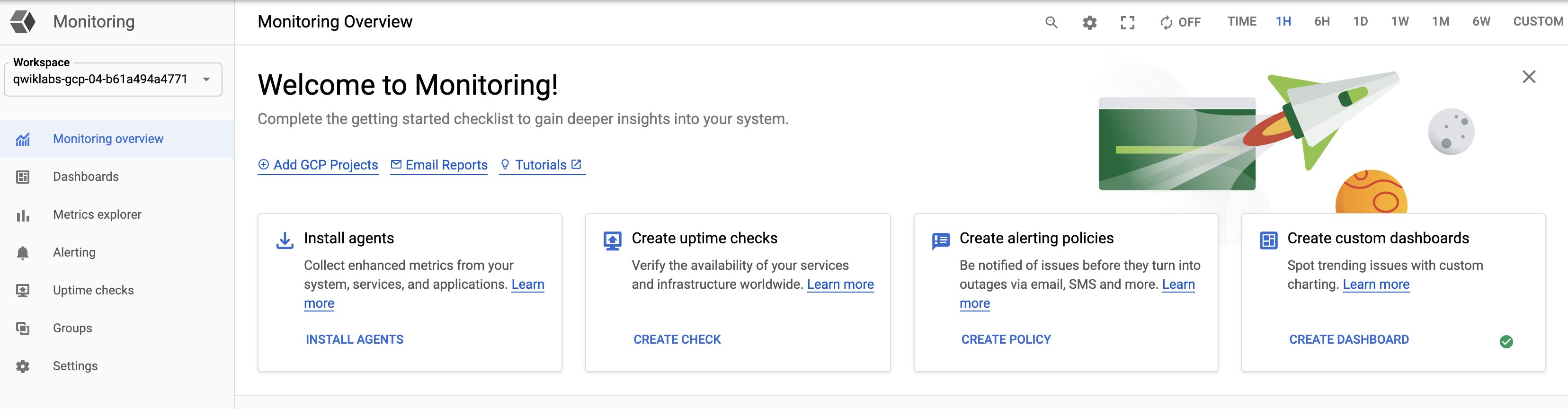
Go ahead and create more logs and see them at http://<your IP address>:8080.

**Create a Monitoring workspace**

You will now setup a Monitoring workspace that's tied to your Qwiklabs GCP Project. The following steps create a new account that has a free trial of Monitoring.

1. In the Google Cloud Platform Console, click on **Navigation menu** > **Monitoring**.
2. Wait for your workspace to be provisioned.

When the Monitoring dashboard opens, your workspace is ready.



**Configure an uptime check and alert policy in Cloud Monitoring**

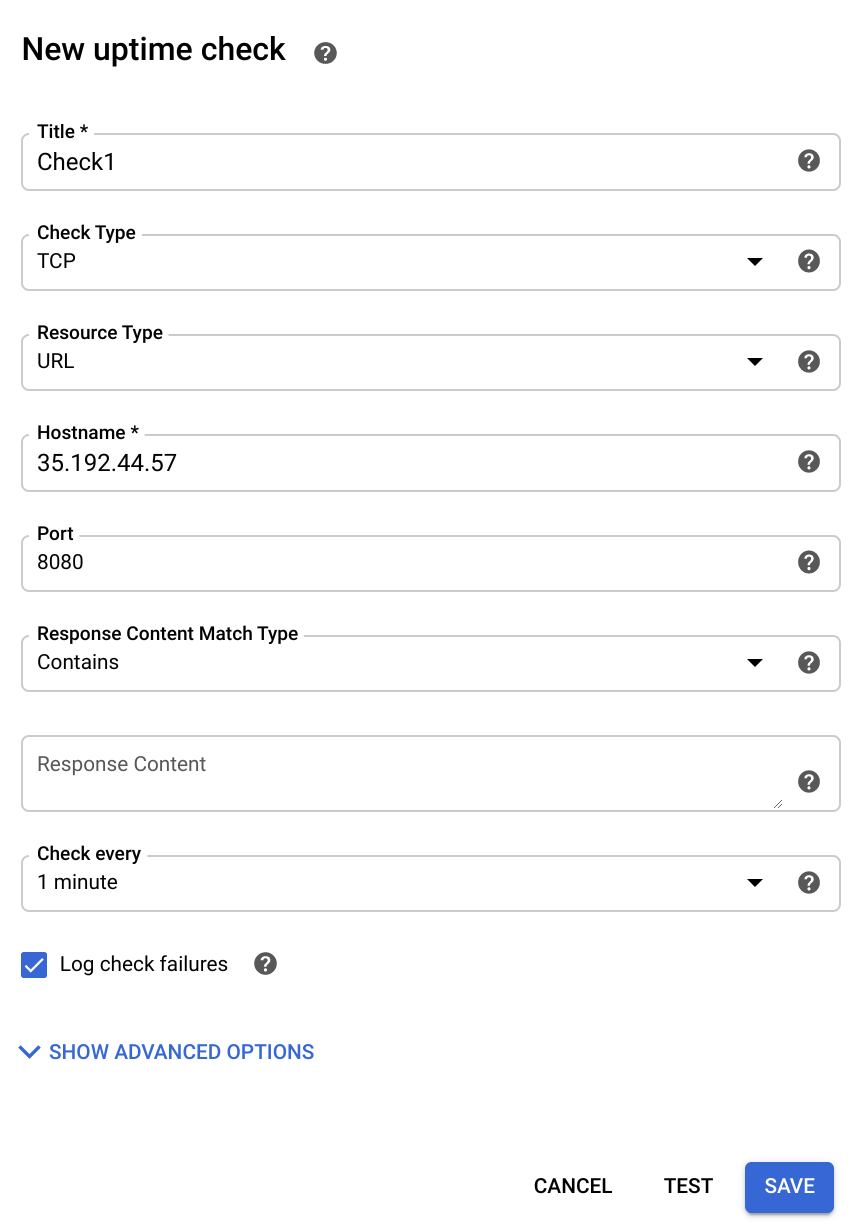
Now that Cloud Monitoring is running, you'll set up alerts and a dashboard.

**Configure an uptime check**

1. In the Cloud Monitoring tab, click on **Uptime Checks** > **Create Uptime Check**.
2. Specify the following:

|  |  |
| --- | --- |
| **Property** | **Value** |
| Title | Check1 |
| Check Type | TCP |
| Resource Type | URL |
| Hostname | <your forwarding address> |
| Port | 8080 |
| Response content contains the text | <leave blank> |
| Check every | 1 minute |

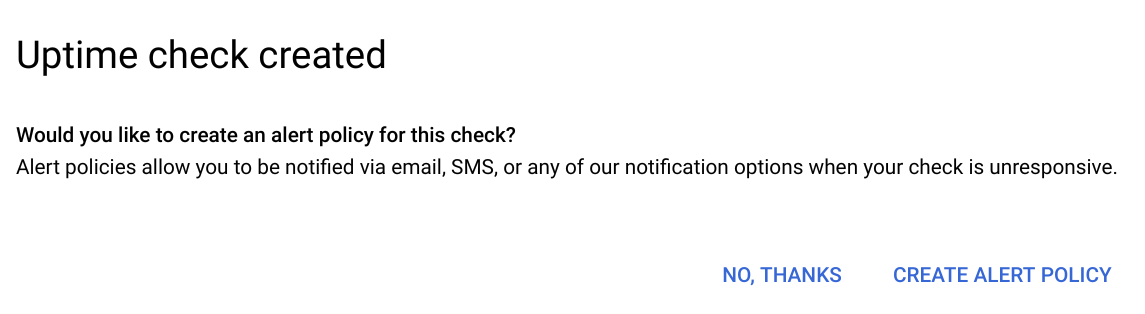
1. Click **Test** to test the check:



If the test fails, make sure that the service is still working. Also check to see that the firewall rule exists and is correct.

1. After the test succeeds, click **Save**.

After the Uptime Check is saved, Cloud Monitoring offers to create an alerting policy. Click on **No, Thanks**.



**Configure an alerting policy and notification**

1. Click on **Alerting** > **Create Policy**.
2. Name the policy as **Test**.
3. Click on **Add Condition**.
4. For **Find resource type and metric**, select **GCE VM Instance**.
5. For **Metrics**, select a metric you are interested in evaluating, such as **CPU usage** or **CPU utilization**.
6. For **Condition**, select **is above**.
7. Specify the threshold value and for how long the metric must cross this set value before the alert is triggered. For example, for **THRESHOLD**, type **20** and set **FOR** to **1 minute**.
8. Click **Add**.
9. In **Notifications**, click on **Add Notification Channel** select Notification Channel type as "Email" and add your personal email. Then Click **Add**.

In a later step, you will trigger an event that will notify you via email.

1. Click **Save**.

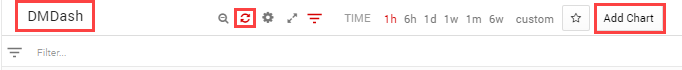
Click **Check my progress** below to check your lab progress.

Configure an Uptime Check and Alert Policy in Cloud Monitoring

**Configure a Dashboard with a Couple of Useful Charts**

**Configure a dashboard**

1. On the Cloud Monitoring window or tab, click on **Dashboards** > **Create Dashboard**
2. Give the New Dashboard Name as **DMDash**. Then Click **Confirm**.
3. Click on **Add Chart**.



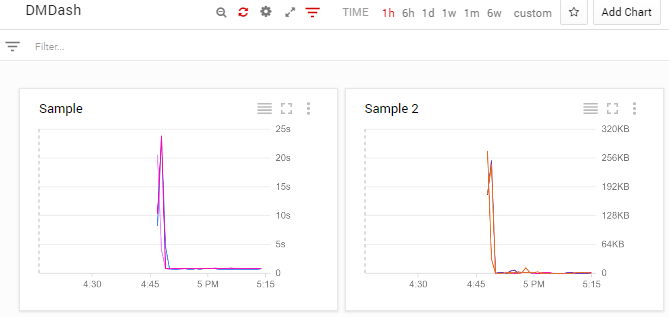
1. Configure the chart as follows:

|  |  |
| --- | --- |
| **Property** | **Value** |
| Title | Sample |
| Resource type | GCE VM Instance |
| Metric Type | CPU Usage |

1. Click **Save**.
2. Click on **Add Chart** to add another chart to the dashboard with the following properties:

|  |  |
| --- | --- |
| **Property** | **Value** |
| Title | Sample2 |
| Resource type | GCE VM Instance |
| Metric Type | Sent Bytes |

1. Click **Save** DMDash should look like this:

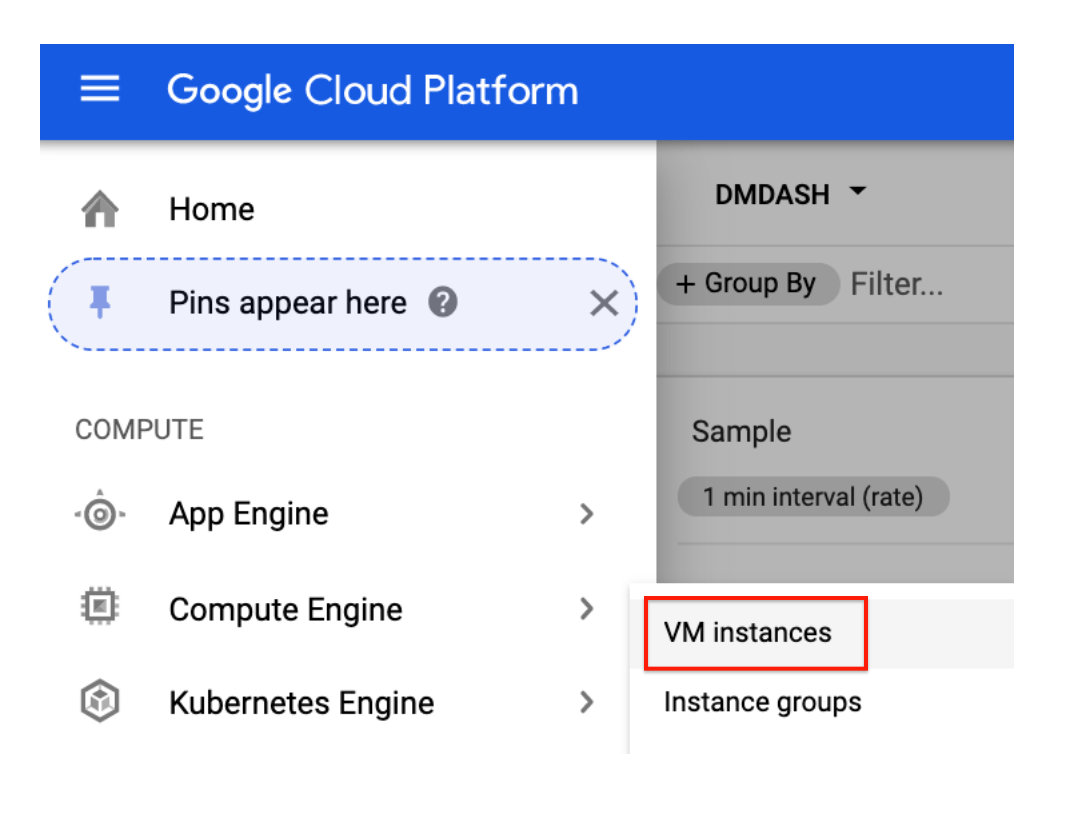


**Create a test VM with ApacheBench**

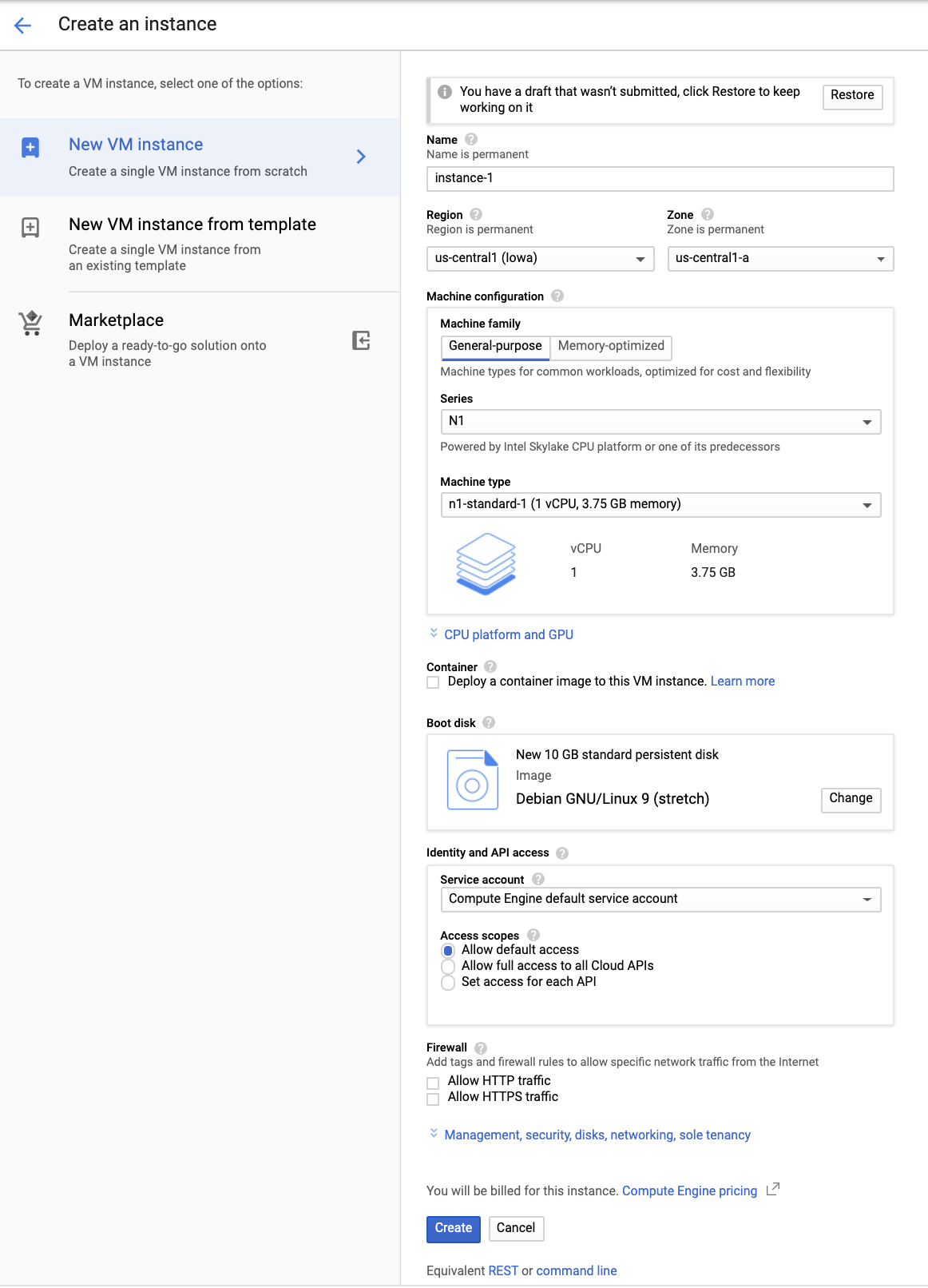
Now that you've configured monitoring for traffic in a specified region, see if it works. You'll install and use ApacheBench to apply 3 levels of load to the service and then view the Cloud Monitoring Dashboard you've set up.

**Create a VM**

1. In the Cloud Console, click **Compute Engine** > **VM instances**.



1. Click **Create instance**, use all the default settings in the **Create an instance** dialog.



1. Click **Create**.

**Install ApacheBench**

1. Still in the VM Instances window, click the instance-1 **SSH** button to SSH into the VM you just created.
2. Enter the following commands to install the latest ApacheBench:

sudo apt-get update

sudo apt-get -y install apache2-utils

Click **Check my progress** below to check your lab progress.

Create a Test VM with ApacheBench.

**Apply and monitor load**

Now you'll use ApacheBench to apply load to the service. Watch the DMDash dashboard in Cloud Monitoring to monitor the CPU usage and the Network Inbound Traffic. You'll also be able to track the number of instances in Cloud Monitoring by mousing over the lines, or by viewing the instances in the Cloud Console (**Navigation menu** > **Compute Engine** > **VM**).

1. In the **SSH** window, enter this command for ApacheBench to apply load to the service. Replace your forwarding IP for Your\_IP. Run the following command two or three times to create traffic.

ab -n 1000 -c 100 http://<Your\_IP>:8080/

**Sample output:**

This is ApacheBench, Version 2.3 <$Revision: 1757674 $>

Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/

Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 35.196.195.26 (be patient)

Completed 100 requests

Completed 200 requests

Completed 300 requests

Completed 400 requests

Completed 500 requests

Completed 600 requests

Completed 700 requests

Completed 800 requests

Completed 900 requests

Completed 1000 requests

Finished 1000 requests

Server Software:

Server Hostname: 35.196.195.26

Server Port: 8080

Document Path: /

Document Length: 40 bytes

Concurrency Level: 100

Time taken for tests: 0.824 seconds

Complete requests: 1000

Failed requests: 0

Total transferred: 140000 bytes

HTML transferred: 40000 bytes

Requests per second: 1213.57 [#/sec] (mean)

Time per request: 82.402 [ms] (mean)

Time per request: 0.824 [ms] (mean, across all concurrent requests)

Transfer rate: 165.92 [Kbytes/sec] received

Connection Times (ms)

min mean[+/-sd] median max

Connect: 36 37 0.5 37 40

Processing: 36 43 8.7 38 73

Waiting: 36 43 8.7 38 73

Total: 73 80 9.1 75 112

Percentage of the requests served within a certain time (ms)

50% 75

66% 78

75% 81

80% 84

90% 95

95% 101

98% 106

99% 110

100% 112 (longest request)

Wait a few minutes, then increase the load to 5000.

1. Run this command two or three times to create traffic:

ab -n 5000 -c 100 http://<Your\_IP>:8080/

Wait a few minutes, then increase the load to 10000.

1. Run this command two or three times to create traffic:

ab -n 10000 -c 100 http://<Your\_IP>:8080/

Now see what happens when you lower the CPU usage per instance.

1. In the cloud console, click **Navigation menu** > **Compute Engine** > **Instance groups**.
2. Click on the name of your instance group, then **Edit Group**.
3. Click on pencil icon of CPU utilization. Change the **Target CPU utilization** to **20** then click **Done**.
4. Click **Save**.

The target CPU usage is the total value of the CPU usage for all VMs in the instance group. This controls when autoscaling occurs. In production you would usually have this set to at least 60%. For this exercise you will set it temporarily to 20% to make it quicker to examine autoscaling.

Run this command two or three times to create traffic:

ab -n 10000 -c 100 http://<Your\_IP>:8080/

**Expected behavior:** The load consumed more than 20% of the cumulative CPU in the group, triggering autoscaling. A new instance was started.

Now see what happens when you turn autoscaling off.

1. Go to **Compute Engine** > **Instance groups**.
2. Click the name of your instance group, then **Edit Group**.
3. Change **Autoscaling mode** to **Don't autoscale**.
4. Click **Save**.

Wait a few minutes, then run this command two or three times to create traffic:

ab -n 10000 -c 100 http://<Your\_IP>:8080/

**Expected behavior:** With autoscaling off, no new instances are created, cumulative CPU usage increases.

**Results**

There will be a lag time of around 5 minutes before you see changes in the Cloud Monitoring Dashboard.

**Simulate a Service Outage**

To simulate an outage, remove the firewall.

1. Click **Navigation menu** > **VPC Networks** > **Firewall rules**.
2. Check the box next to **tcp:8080** protocols / ports, then click **Delete** at the top of the page. Click **Delete** again to confirm.
3. You will receive a notification email in 15 to 30 minutes.

**Test your knowledge**

Test your knowledge about Google Cloud Platform by taking our quiz.

**End your lab**

When you have completed your lab, click **End Lab**. Qwiklabs removes the resources you’ve used and cleans the account for you.

You will be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**.

The number of stars indicates the following:

* 1 star = Very dissatisfied
* 2 stars = Dissatisfied
* 3 stars = Neutral
* 4 stars = Satisfied
* 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, please use the **Support** tab.

Copyright 2020 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.