

GSP060



Overview

In this lab, you launch a service using an infrastructure orchestration tool called Deployment Manager and monitor the service using Cloud Monitoring. In Cloud Monitoring, you 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:

- 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. It collects metrics, events, and metadata from Google Cloud, 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 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

Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click Start Lab, shows how long Google Cloud resources will be made available to you.

This Qwiklabs hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

What you need

To complete this lab, you need:

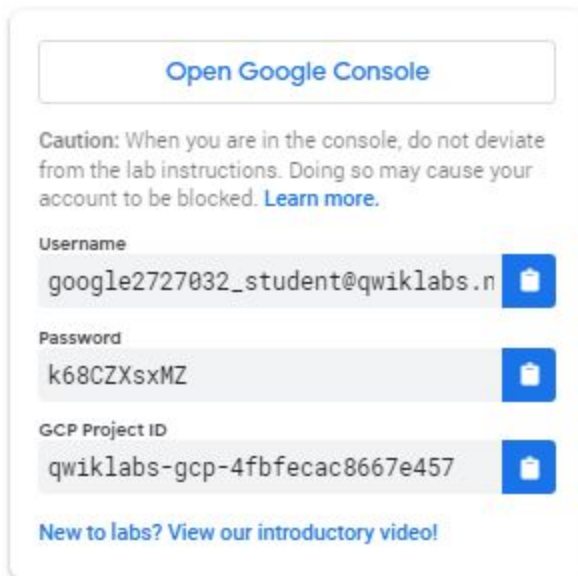
- Access to a standard internet browser (Chrome browser recommended).
- Time to complete the lab.

Note: If you already have your own personal Google Cloud account or project, do not use it for this lab.

Note: If you are using a Pixelbook, open an Incognito window to run this lab.

How to start your lab and sign in to the Google Cloud Console

- 1. Click the Start Lab button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.**



Open Google Console

Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. [Learn more.](#)

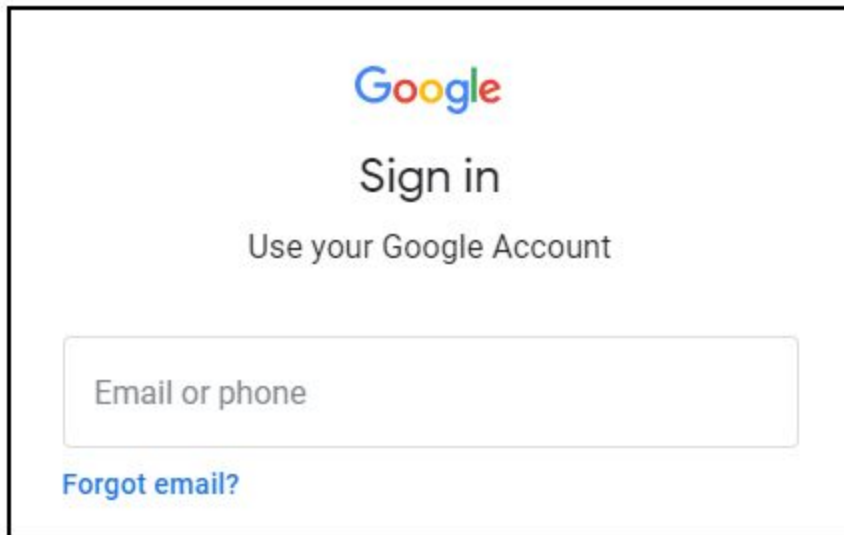
Username
google2727032_student@qwiklabs.n

Password
k68CZXsxMZ

GCP Project ID
qwiklabs-gcp-4fbfecac8667e457

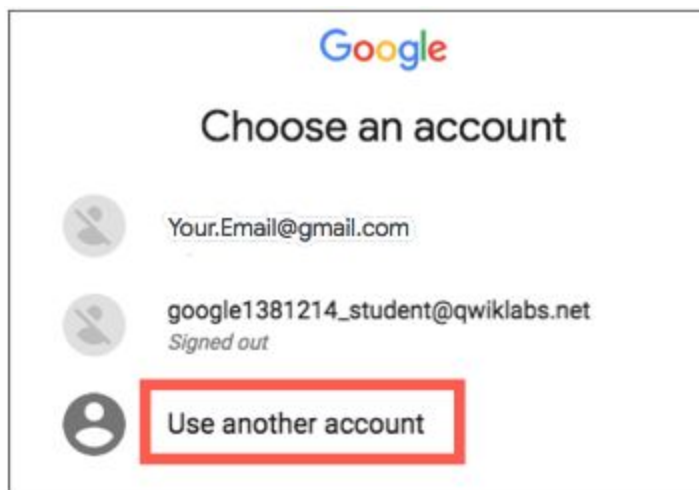
[New to labs? View our introductory video!](#)

- 2. Copy the username, and then click Open Google Console. The lab spins up resources, and then opens another tab that shows the Sign in page.**



Tip: Open the tabs in separate windows, side-by-side.

If you see the Choose an account page, click Use Another Account.

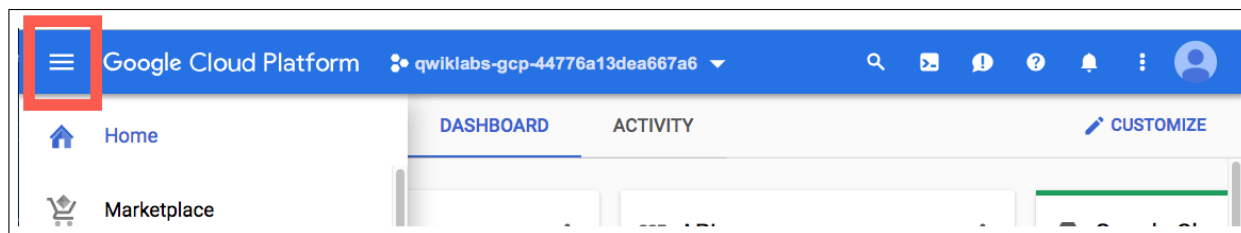


3. In the Sign in page, paste the username that you copied from the Connection Details panel. Then copy and paste the password.
Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).
4. Click through the subsequent pages:
 - Accept the terms and conditions.

- Do not add recovery options or two-factor authentication (because this is a temporary account).
- Do not sign up for free trials.

After a few moments, the Cloud Console opens in this tab.

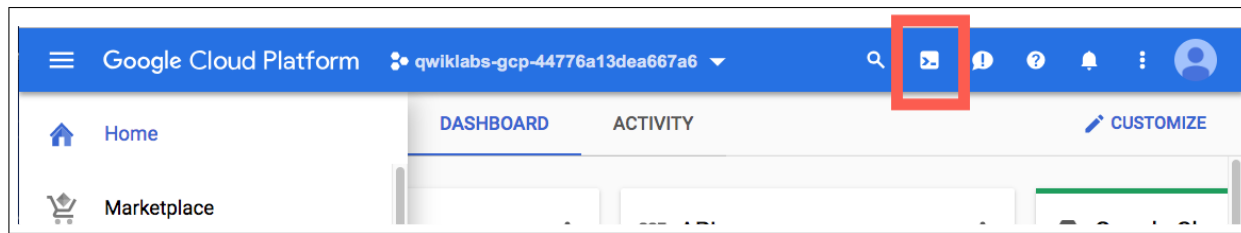
Note: You can view the menu with a list of Google Cloud Products and Services by clicking the Navigation menu at the top-left.



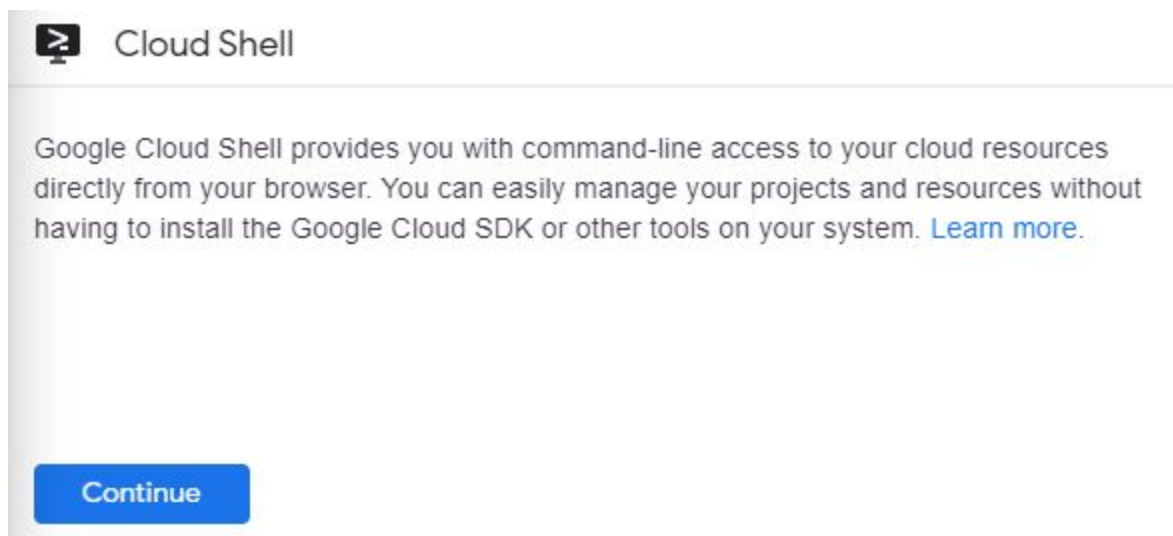
Activate Cloud Shell

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

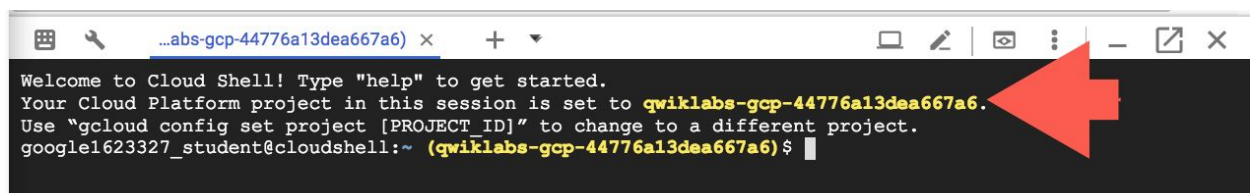
In the Cloud Console, in the top right toolbar, click the Activate Cloud Shell button.



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
```

(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
```

For full documentation of gcloud see the [gcloud command-line tool overview](#).

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_17         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 guide.

The `nodejs` directory contains everything you need to build this lab. Note that there is a `nodejs` directory and a `nodejs_17` directory. Use `nodejs`.

List and examine the Nodejs deployment

Navigate to and list the version2 examples:

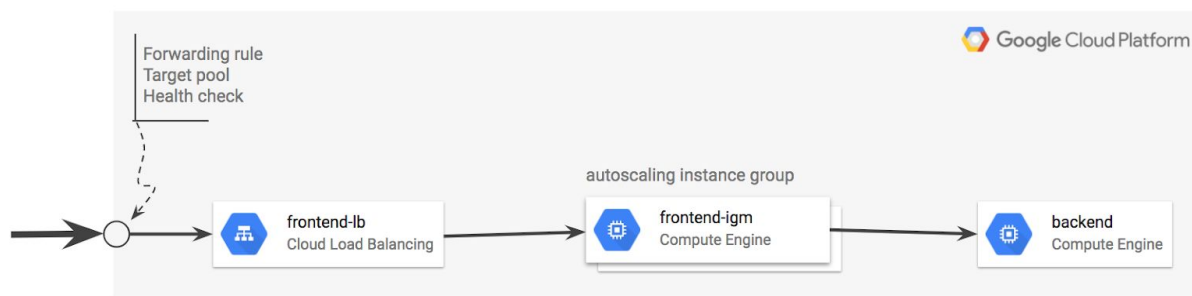
```
cd nodejs/python
```

```
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 also creates:

- 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.

2. 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
```

3. 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
```

2. 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).

3. 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 PiYc60sIFkWzQpCDklHvaA==
Waiting for create
[operation-1529913842103-56f72d31872d9-90070017-aec5761d]...done.
Create operation operation-1529913842103-56f72d31872d9-90070017-aec5761d
completed successfully.
```

NAME	TYPE	STATE
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 []

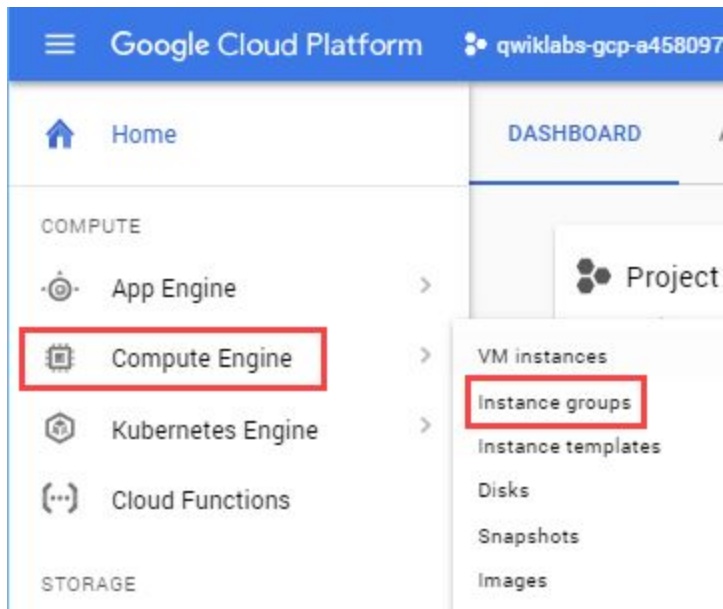
Click Check my progress below to check your lab progress.

Deploy the application

Check my progress

Confirm the limit for the maximum number of instances:

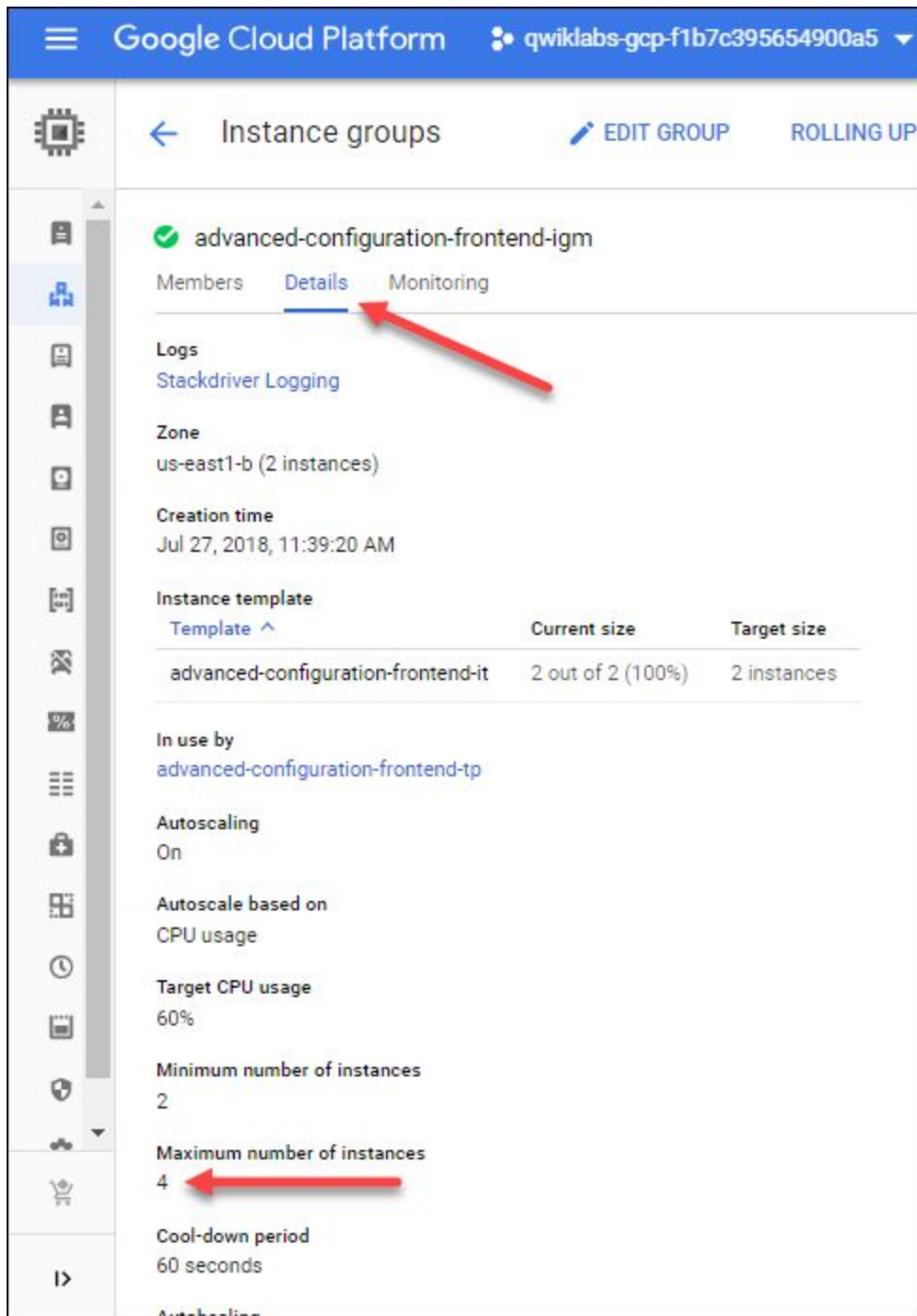
1. Go to Compute Engine > Instance groups.



2. Click on advanced-configuration-frontend-igm.

Instance groups								
CREATE INSTANCE GROUP REFRESH EDIT DELETE								
<input type="text" value="Filter instance groups"/> Columns ▾								
<input type="checkbox"/>	Name ▾	Zone	Instances	Template	Creation time	Recommendation	Autoscaling	In use by
<input type="checkbox"/>	✓ advanced-configuration-frontend-igm	us-east1-d	2	advanced-configuration-frontend-ig	Mar 20, 2018, 11:52:05 PM		Target CPU usage 60%	advanced-configur

3. Click on the Details tab, then verify the maximum number of instances.



The screenshot shows the Google Cloud Platform console interface. The top navigation bar includes the Google Cloud Platform logo and the account ID 'qwiklabs-gcp-f1b7c395654900a5'. The main heading is 'Instance groups'. Below this, there are three tabs: 'Members', 'Details', and 'Monitoring'. The 'Details' tab is selected, and a red arrow points to it. The details section shows the following information:

- Logs:** Stackdriver Logging
- Zone:** us-east1-b (2 instances)
- Creation time:** Jul 27, 2018, 11:39:20 AM
- Instance template:** advanced-configuration-frontend-it
- Current size:** 2 out of 2 (100%)
- Target size:** 2 instances
- In use by:** advanced-configuration-frontend-tp
- Autoscaling:** On
- Autoscale based on:** CPU usage
- Target CPU usage:** 60%
- Minimum number of instances:** 2
- Maximum number of instances:** 4 (indicated by a red arrow)
- Cool-down period:** 60 seconds

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, 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:

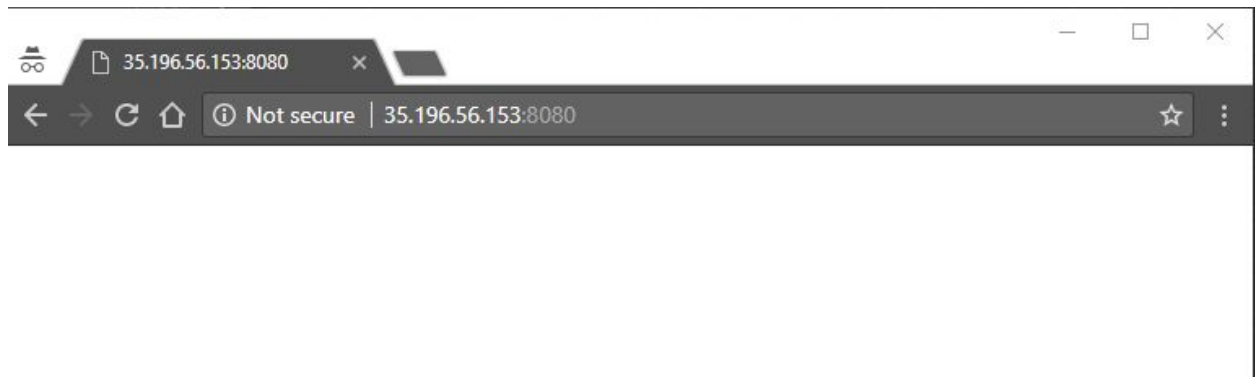
NAME	REGION	IP_ADDRESS	IP_PROTOCOL	TARGET
advanced-configuration-frontend-lb	us-east1	35.196.56.153	TCP	us-east1/targetPoo

Your 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.

2. 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 see a blank page, similar to below, with your own IP address:



It may take up to 10 minutes for the service to become operational. If you get an error, such as a 404, or a time out, wait about two minutes and try again.

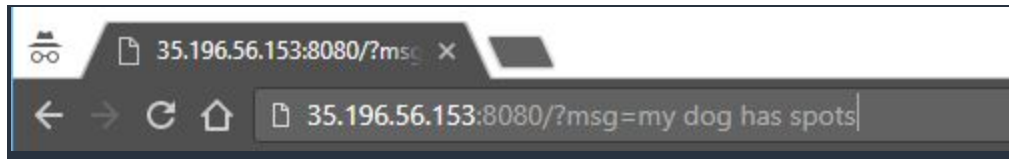
3. Now enter log information by entering a log message in your browser address line.

```
http://<your forwarding IP address>:8080/?msg=<enter_a_message>
```

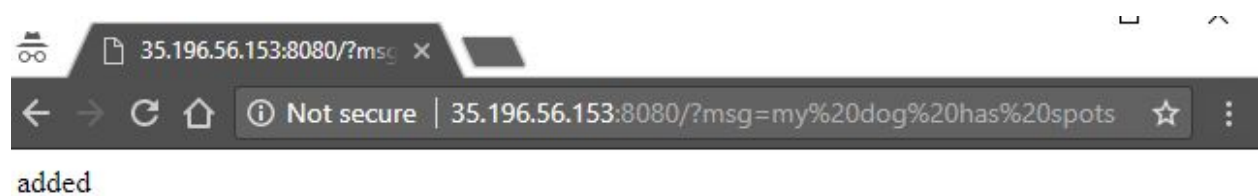
Replace enter_a_message with any kind of message.

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

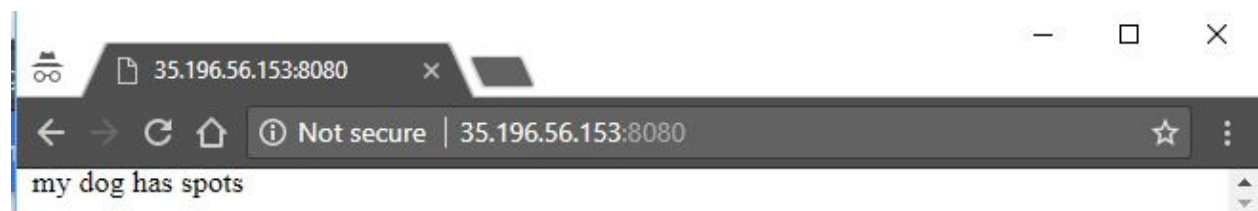
```
http://35.196.56.153:8080/?msg=my dog has spots
```



After you enter the log, the browser returns added.



4. 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>.

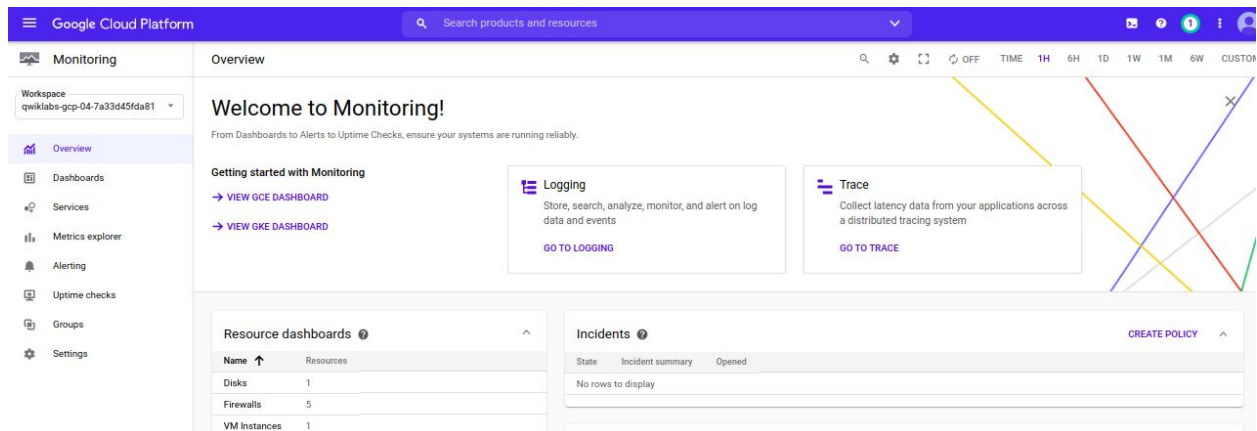
Configure an uptime check and alert policy in Cloud Monitoring

Create a Monitoring workspace

Now set up a Monitoring workspace that's tied to your Google Cloud Project. The following steps create a new account that has a free trial of Monitoring.

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

When the Monitoring dashboard opens, your workspace is ready.



Configure an uptime check

The Cloud Monitoring window opens.

1. In the Cloud Monitoring tab, click on Uptime Checks
2. Click Create Uptime Check.
3. In the New uptime check page Specify the following:

Property	Value
Title	Check1
Check Type	TCP
Resource Type	URL
Hostname	<your forwarding address> (you previously recorded)
Port	8080
Response content contains the text	<leave blank>
Check every	1 minute

4. Click Test to test the check:

New uptime check ?

Title *

Check1



Check Type

TCP



Resource Type

URL



Hostname *

35.192.44.57



Port

8080



Response Content Match Type

Contains



Response Content



Check every

1 minute



☒ Log check failures ?

✓ SHOW ADVANCED OPTIONS

CANCEL

TEST

SAVE

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

5. After the test succeeds, click Save.

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

Uptime check created

Would you like to create an alert policy for this check?

Alert policies allow you to be notified via email, SMS, or any of our notification options when your check is unresponsive.

[NO, THANKS](#)

[CREATE ALERT POLICY](#)

Configure an alerting policy and notification

1. Click on Alerting > Create Policy.
2. Name the policy Test.
3. Click Add Condition.
4. For Find resource type and metric, select Compute Engine 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 trigger an event that notify you via email.
10. Click Save.

Click Check my progress below to check your lab progress.

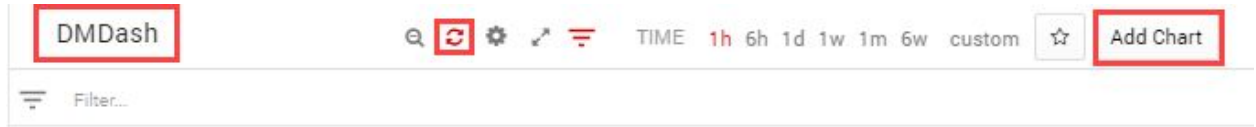
Configure an Uptime Check and Alert Policy in Cloud Monitoring

Check my progress

Configure a Dashboard with a Couple of Useful Charts

Configure a dashboard

1. In the left menu, click Dashboards > Create Dashboard
2. Name the New Dashboard DMDash. and then Click Confirm.
3. Click Add Chart in the top right.



4. Set the following properties:

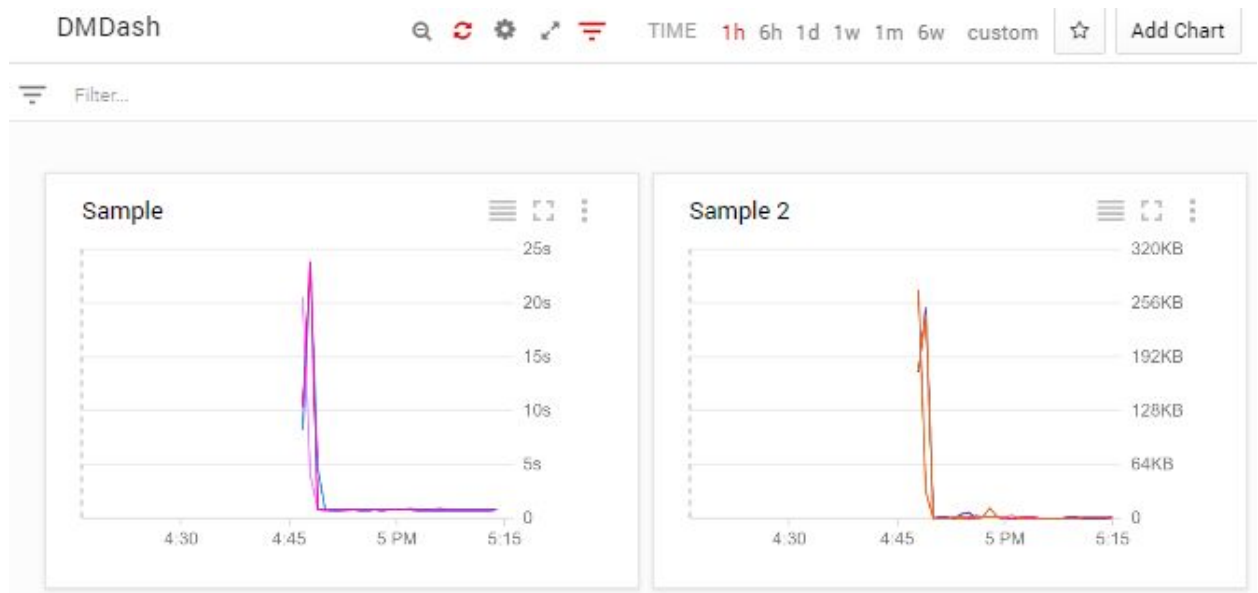
Property	Value
Title	Sample
Resource type	Compute Engine VM Instance
Metric Type	CPU Usage

5. Click Save.
6. Click on Add Chart to add another chart to the dashboard with the following properties:

Property	Value
Title	Sample 2
Resource type	Compute Engine VM Instance
Metric Type	Sent bytes

7.

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

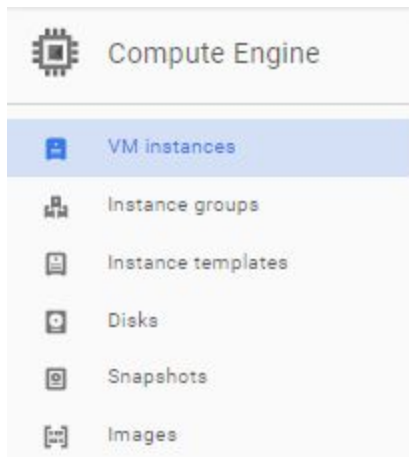


Create a test VM with ApacheBench

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**.



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

Create an instance

To create a VM instance, select one of the options:



New VM instance

Create a single VM instance from scratch



New VM instance from template

Create a single VM instance from an existing template



Marketplace

Deploy a ready-to-go solution onto a VM instance



? You have a draft that wasn't submitted, click Restore to keep working on it

Restore

Name

Name is permanent

instance-1

Region

Region is permanent

us-central1 (Iowa)

Zone

Zone is permanent

us-central1-a

Machine configuration

Machine family

General-purpose

Memory-optimized

Machine types for common workloads, optimized for cost and flexibility

Series

N1

Powered by Intel Skylake CPU platform or one of its predecessors

Machine type

n1-standard-1 (1 vCPU, 3.75 GB memory)



vCPU

1

Memory

3.75 GB

⌵ CPU platform and GPU

Container

☐ Deploy a container image to this VM instance. [Learn more](#)

Boot disk



New 10 GB standard persistent disk

Image

Debian GNU/Linux 9 (stretch)

Change

Identity and API access

Service account

Compute Engine default service account

Access scopes

☒ Allow default access

☐ Allow full access to all Cloud APIs

☐ Set access for each API

Firewall

Add tags and firewall rules to allow specific network traffic from the Internet

☐ Allow HTTP traffic

☐ Allow HTTPS traffic

⌵ Management, security, disks, networking, sole tenancy

You will be billed for this instance. [Compute Engine pricing](#)

Create

Cancel

Equivalent [REST](#) or [command line](#)

3. 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.

Check my progress

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 (you previously recorded). 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.

2. 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.

3. 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, click on the Details tab, and then click Edit Group.
3. Change the CPU utilization. Click CPU utilization, change Target CPU utilization to 20, and 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 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 is 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 Network > Firewall.
2. Check the box next to the firewall rule which contains tcp:8080, 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

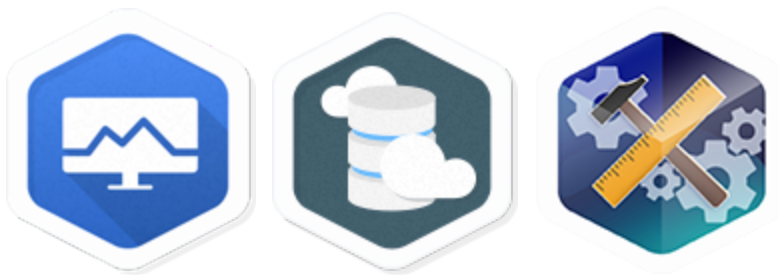
Test your knowledge about Google cloud Platform by taking our quiz.

Cloud Monitoring collects metrics, events, and metadata from Google Cloud, 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.

checkTrue

False

Congratulations!



Finish your Quest

This self-paced lab is part of the [Deployment Manager](#), [Cloud Architecture](#), and [Cloud Engineering](#) Quests. A Quest is a series of related labs that form a learning path. Completing this Quest earns you the badge above, to recognize your achievement. You can make your badge public and link to them in your online resume or social media account. Enroll in a Quest and get immediate completion credit if you've taken this lab. [See other available Qwiklabs Quests](#).

Take your next lab

Continue your Quest with [Create a Custom Network Using Deployment Manager](#), or check out these suggestions:

- [Create a Network Load-Balanced Logbook Application](#)
- [Deployment Manager for Appserver](#)

Next steps / learn more

- Check out how Cloud Monitoring can be used, see [Cloud Monitoring Documentation](#).
- Learn more about Deployment Manager, see [Deployment Manager Fundamentals](#).
- How about our tutorials? There are [several step-by-step tutorials](#) in the documentation. This lab is based on one of those tutorials, but includes additional content. The Deployment Manager template used in this lab generates an advanced HTTP(S) Load Balanced deployment for a

Logbook sample application. [Sample templates](#) are available in both Jinja and Python. This lab used the Python version.

Google Cloud Training & Certification

...helps you make the most of Google Cloud technologies. [Our classes](#) include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. [Certifications](#) help you validate and prove your skill and expertise in Google Cloud technologies.

Manual last updated June 19, 2020

Lab last tested June 19, 2020

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.

Continue questing