

LAB COMMANDS LIST

```
# Check the elected project
gcloud config list

# Show any .ssh folder
pwd
ls
ls -a .ssh

# Get our bearings in Cloud Shell
whoami
hostname
curl api.ipify.org

# Check that we have nothing running
gcloud compute instances list

# Don't create a default VM
# Cancel: gcloud compute instances create myhappyvm

# Look at how to set the machine type
gcloud compute instances create myhappyvm -h
gcloud compute instances create myhappyvm --help
gcloud compute machine-types list

# See how to filter
gcloud topic filters

# Show some free-tier-eligible options
gcloud compute machine-types list --filter="NAME:f1-micro"
gcloud compute machine-types list --filter="NAME:f1-micro AND ZONE~us-west"

# Set our defaults to Los Angeles
gcloud config set compute/zone us-west2-b
gcloud config set compute/region us-west2

# Start our instance
gcloud compute instances create --machine-type=f1-micro myhappyvm
ping -c 3 myhappyvm
ping -c 3 internalipaddress
ping -c 3 externalipaddress

# Connect to the VM
ssh externalipaddress
gcloud compute ssh myhappyvm

# Get our bearings -- Skip?
whoami
hostname
curl api.ipify.org

# Get back to Cloud Shell
exit
curl api.ipify.org

# Look at the Cloud Shell .ssh files
cd .ssh
ls
cat google_compute_engine.pub
head -n 10 google_compute_engine
```

```
# Log back onto the VM
gcloud compute ssh myhappyvm

# See that our key is authorized
cd .ssh
ls
cat authorized_keys
cd ..

# Check out the metadata
• curl metadata.google.internal/computeMetadata/v1/
• curl -H "Metadata-Flavor: Google"
  metadata.google.internal/computeMetadata/v1/
• curl -H "Metadata-Flavor: Google"
  metadata.google.internal/computeMetadata/v1/project/
• curl -H "Metadata-Flavor: Google"
  metadata.google.internal/computeMetadata/v1/project/project-id
• curl -H "Metadata-Flavor: Google"
  metadata.google.internal/computeMetadata/v1/project/attributes/
• curl -H "Metadata-Flavor: Google"
  metadata.google.internal/computeMetadata/v1/project/attributes/ssh-keys

# Look at some instance metadata
• curl -H "Metadata-Flavor: Google"
  metadata.google.internal/computeMetadata/v1/instance/
• curl -H "Metadata-Flavor: Google"
  metadata.google.internal/computeMetadata/v1/instance/name
• curl -H "Metadata-Flavor: Google"
  metadata.google.internal/computeMetadata/v1/instance/service-
  accounts/default/
• curl -H "Metadata-Flavor: Google"
  metadata.google.internal/computeMetadata/v1/instance/service-
  accounts/default/email

# See what gcloud knows
gcloud config list

# Look at our buckets
gsutil ls
gsutil ls gs://storage-lab-cli/

# Attempt to delete the VM from within the VM
gcloud compute instances delete myhappyvm

# Exit back to Cloud Shell and actually delete the VM
exit
gcloud compute instances delete myhappyvm
```

GCP LABS

1) GCE

Task 2: Create a virtual machine using the GCP Console

1. In the **Navigation menu** , click **Compute Engine > VM instances**.
2. Click **Create**.
3. On the **Create an Instance** page, for **Name**, type my-vm-1
4. For **Region** and **Zone**, select the region and zone assigned by Qwiklabs.
5. For **Machine type**, accept the default.
6. For **Boot disk**, if the **Image** shown is not **Debian GNU/Linux 9 (stretch)**, click **Change** and select **Debian GNU/Linux 9 (stretch)**.
7. Leave the defaults for **Identity and API access** unmodified.
8. For Firewall, click **Allow HTTP traffic**.
9. Leave all other defaults unmodified.
10. To create and launch the VM, click **Create**.

Note: The VM can take about two minutes to launch and be fully available for use.

Click *Check my progress* to verify the objective.

Create a virtual machine using the GCP Console

Check my progress

Task 3: Create a virtual machine using the gcloud command line

1. In GCP console, on the top right toolbar, click the Open Cloud Shell button.
2. Click **Continue**.
3. To display a list of all the zones in the region to which Qwiklabs assigned you, enter this partial command `gcloud compute zones list | grep` followed by the region that Qwiklabs or your instructor assigned you to.

Your completed command will look like this:

```
gcloud compute zones list | grep us-central1
```

4. Choose a zone from that list other than the zone to which Qwiklabs assigned you. For example, if Qwiklabs assigned you to region us-central1 and zone us-central1-a you might choose zone us-central1-b.
5. To set your default zone to the one you just chose, enter this partial command `gcloud config set compute/zone` followed by the zone you chose.

Your completed command will look like this:

```
gcloud config set compute/zone us-central1-b
```

6. To create a VM instance called **my-vm-2** in that zone, execute this command:

```
7. gcloud compute instances create "my-vm-2" \  
8. --machine-type "n1-standard-1" \  
9. --image-project "debian-cloud" \  
10. --image "debian-9-stretch-v20190213" \  
11. --subnet "default"
```

Note: The VM can take about two minutes to launch and be fully available for use.

7. To close the Cloud Shell, execute the following command:

```
8. exit
```

Click *Check my progress* to verify the objective.

Create a virtual machine using the gcloud command line

Check my progress

Task 4: Connect between VM instances

1. In the **Navigation menu**, click **Compute Engine > VM instances**.

You will see the two VM instances you created, each in a different zone.

Notice that the Internal IP addresses of these two instances share the first three bytes in common. They reside on the same subnet in their Google Cloud VPC even though they are in different zones.

2. To open a command prompt on the **my-vm-2** instance, click **SSH** in its row in the **VM instances** list.
3. Use the ping command to confirm that **my-vm-2** can reach **my-vm-1** over the network:

4. `ping my-vm-1`

Notice that the output of the ping command reveals that the complete hostname of **my-vm-1** is **my-vm-1.c.PROJECT_ID.internal**, where PROJECT_ID is the name of your Google Cloud Platform project. GCP automatically supplies Domain Name Service (DNS) resolution for the internal IP addresses of VM instances.

5. Press **Ctrl+C** to abort the ping command.

6. Use the **ssh** command to open a command prompt on **my-vm-1**:

7. `ssh my-vm-1`

If you are prompted about whether you want to continue connecting to a host with unknown authenticity, enter **yes** to confirm that you do.

8. At the command prompt on **my-vm-1**, install the Nginx web server:

9. `sudo apt-get install nginx-light -y`

10. Use the **nano** text editor to add a custom message to the home page of the web server:

11. `sudo nano /var/www/html/index.nginx-debian.html`

12. Use the arrow keys to move the cursor to the line just below the h1 header. Add text like this, and replace YOUR_NAME with your name:

13. `Hi from YOUR_NAME`

14. Press **Ctrl+O** and then press **Enter** to save your edited file, and then press **Ctrl+X** to exit the nano text editor.

15. Confirm that the web server is serving your new page. At the command prompt on **my-vm-1**, execute this command:

16. `curl http://localhost/`

The response will be the HTML source of the web server's home page, including your line of custom text.

17. To exit the command prompt on **my-vm-1**, execute this command:

18. `exit`

You will return to the command prompt on **my-vm-2**

19. To confirm that **my-vm-2** can reach the web server on **my-vm-1**, at the command prompt on **my-vm-2**, execute this command:

20. `curl http://my-vm-1/`

The response will again be the HTML source of the web server's home page, including your line of custom text.

21. In the **Navigation menu**, click **Compute Engine > VM instances**.

22. Copy the External IP address for **my-vm-1** and paste it into the address bar of a new browser tab. You will see your web server's home page, including your custom text.

If you forgot to click **Allow HTTP traffic** when you created the **my-vm-1** VM instance, your attempt to reach your web server's home page will fail. You can add a [firewall rule](#) to allow inbound traffic to your instances, although this topic is out of scope for this course.

2) GCE and SQL

Task 2: Deploy a web server VM instance

1. In the GCP Console, on the **Navigation menu**, click **Compute Engine > VM instances**.
2. Click **Create**.
3. On the **Create an Instance** page, for **Name**, type `bloghost`
4. For **Region** and **Zone**, select the region and zone assigned by Qwiklabs.
5. For **Machine type**, accept the default.
6. For **Boot disk**, if the **Image** shown is not **Debian GNU/Linux 9 (stretch)**, click **Change** and select **Debian GNU/Linux 9 (stretch)**.
7. Leave the defaults for **Identity and API access** unmodified.
8. For **Firewall**, click **Allow HTTP traffic**.
9. Click **Management, security, disks, networking, sole tenancy** to open that section of the dialog.
10. Enter the following script as the value for **Startup script**:

```
apt-get update

apt-get install apache2 php php-mysql -y

service apache2 restart
```

Be sure to supply that script as the value of the **Startup script** field. If you accidentally put it into another field, it won't be executed when the VM instance starts.

11. Leave the remaining settings as their defaults, and click **Create**.

Instance can take about two minutes to launch and be fully available for use.

12. On the **VM instances** page, copy the **bloghost** VM instance's internal and external IP addresses to a text editor for use later in this lab.

Click *Check my progress* to verify the objective.

Deploy a web server VM instance

Check my progress

Task 3: Create a Cloud Storage bucket using the gsutil command line

All Cloud Storage bucket names must be globally unique. To ensure that your bucket name is unique, these instructions will guide you to give your bucket the same name as your Cloud Platform project ID, which is also globally unique.

Cloud Storage buckets can be associated with either a region or a multi-region location: **US**, **EU**, or **ASIA**. In this activity, you associate your bucket with the multi-region closest to the region and zone that Qwiklabs or your instructor assigned you to.

1. On the **Google Cloud Platform** menu, click **Activate Cloud Shell** . If a dialog box appears, click **Start Cloud Shell**.
2. For convenience, enter your chosen location into an environment variable called **LOCATION**. Enter one of these commands:

```
export LOCATION=US
```

Or

```
export LOCATION=EU
```

Or

```
export LOCATION=ASIA
```

3. In Cloud Shell, the **DEVSHHELL_PROJECT_ID** environment variable contains your project ID. Enter this command to make a bucket named after your project ID:

```
gsutil mb -l $LOCATION gs://$DEVSHHELL_PROJECT_ID
```

4. Retrieve a banner image from a publicly accessible Cloud Storage location:

```
gsutil cp gs://cloud-training/gcpfci/my-excellent-blog.png my-excellent-blog.png
```

5. Copy the banner image to your newly created Cloud Storage bucket:

```
gsutil cp my-excellent-blog.png gs://$DEVSHHELL_PROJECT_ID/my-excellent-blog.png
```

6. Modify the Access Control List of the object you just created so that it is readable by everyone:

```
gsutil acl ch -u allUsers:R gs://$DEVSHHELL_PROJECT_ID/my-excellent-blog.png
```

Click *Check my progress* to verify the objective.

Create a Cloud Storage bucket using the gsutil command line

Check my progress

Task 4: Create the Cloud SQL instance

1. In the GCP Console, on the **Navigation menu**, click **SQL**.
2. Click **Create instance**.
3. For **Choose a database engine**, select **MySQL**.
4. For **Instance ID**, type **blog-db**, and for **Root password** type a password of your choice.

Choose a password that you remember. There's no need to obscure the password because you'll use mechanisms to connect that aren't open access to everyone.

5. Set the region and zone assigned by Qwiklabs.

This is the same region and zone into which you launched the **bloghost** instance. The best performance is achieved by placing the client and the database close to each other.

6. Click **Create**.

Wait for the instance to finish deploying. It will take a few minutes.

7. Click on the name of the instance, **blog-db**, to open its details page.

8. From the SQL instances details page, copy the **Public IP address** for your SQL instance to a text editor for use later in this lab.
9. Click on **Users** menu on the left-hand side, and then click **ADD USER ACCOUNT**.
10. For **User name**, type blogdbuser
11. For **Password**, type a password of your choice. Make a note of it.
12. Click **Create** to create the user account in the database.

Wait for the user to be created.

13. Click the **Connections** tab, and then click **Add network**.

If you are offered the choice between a **Private IP** connection and a **Public IP** connection, choose **Public IP** for purposes of this lab. The **Private IP** feature is in beta at the time this lab was written.

The **Add network** button may be grayed out if the user account creation is not yet complete.

14. For **Name**, type web front end
15. For **Network**, type the external IP address of your **bloghost** VM instance, followed by /32

The result will look like this:

```
35.192.208.2/32
```

Be sure to use the external IP address of your VM instance followed by /32. Do not use the VM instance's internal IP address. Do not use the sample IP address shown here.

16. Click **Done** to finish defining the authorized network.
17. Click **Save** to save the configuration change.

Click *Check my progress* to verify the objective.

Create the Cloud SQL instance

Check my progress

Task 5: Configure an application in a Compute Engine instance to use Cloud SQL

1. On the **Navigation menu**, click **Compute Engine > VM instances**.
2. In the VM instances list, click **SSH** in the row for your VM instance **bloghost**.

3. In your ssh session on **bloghost**, change your working directory to the document root of the web server:

```
cd /var/www/html
```

4. Use the **nano** text editor to edit a file called **index.php**:

```
sudo nano index.php
```

5. Paste the content below into the file:

```
<html>

<head><title>Welcome to my excellent blog</title></head>

<body>

<h1>Welcome to my excellent blog</h1>

<?php

    $dbserver = "CLOUDSQLIP";

    $dbuser = "blogdbuser";

    $dbpassword = "DBPASSWORD";

    // In a production blog, we would not store the MySQL

    // password in the document root. Instead, we would store it in a

    // configuration file elsewhere on the web server VM instance.

    $conn = new mysqli($dbserver, $dbuser, $dbpassword);

    if (mysqli_connect_error()) {

        echo ("Database connection failed: " . mysqli_connect_error());

    } else {

        echo ("Database connection succeeded.");

    }

?>
```

```
</body></html>
```

In a later step, you will insert your Cloud SQL instance's IP address and your database password into this file. For now, leave the file unmodified.

6. Press **Ctrl+O**, and then press **Enter** to save your edited file.
7. Press **Ctrl+X** to exit the nano text editor.
8. Restart the web server:

```
sudo service apache2 restart
```

9. Open a new web browser tab and paste into the address bar your **bloghost** VM instance's external IP address followed by **/index.php**. The URL will look like this:

```
35.192.208.2/index.php
```

Be sure to use the external IP address of your VM instance followed by **/index.php**. Do not use the VM instance's internal IP address. Do not use the sample IP address shown here.

When you load the page, you will see that its content includes an error message beginning with the words:

```
Database connection failed: ...
```

This message occurs because you have not yet configured PHP's connection to your Cloud SQL instance.

10. Return to your ssh session on **bloghost**. Use the **nano** text editor to edit **index.php** again.

```
sudo nano index.php
```

11. In the **nano** text editor, replace **CLOUDSQLIP** with the Cloud SQL instance Public IP address that you noted above. Leave the quotation marks around the value in place.
12. In the **nano** text editor, replace **DBPASSWORD** with the Cloud SQL database password that you defined above. Leave the quotation marks around the value in place.
13. Press **Ctrl+O**, and then press **Enter** to save your edited file.
14. Press **Ctrl+X** to exit the nano text editor.
15. Restart the web server:

```
sudo service apache2 restart
```

16. Return to the web browser tab in which you opened your **bloghost** VM instance's external IP address. When you load the page, the following message appears:

```
Database connection succeeded.
```

In an actual blog, the database connection status would not be visible to blog visitors. Instead, the database connection would be managed solely by the administrator.

Task 6: Configure an application in a Compute Engine instance to use a Cloud Storage object

1. In the GCP Console, click **Storage > Browser**.
2. Click on the bucket that is named after your GCP project.
3. In this bucket, there is an object called **my-excellent-blog.png**. Copy the URL behind the link icon that appears in that object's **Public access** column, or behind the words "Public link" if shown.

If you see neither a link icon nor a "Public link", try refreshing the browser. If you still do not see a link icon, return to Cloud Shell and confirm that your attempt to change the object's Access Control list with the **gsutil acl ch** command was successful.

4. Return to your ssh session on your **bloghost** VM instance.
5. Enter this command to set your working directory to the document root of the web server:

```
cd /var/www/html
```

6. Use the **nano** text editor to edit **index.php**:

```
sudo nano index.php
```

7. Use the arrow keys to move the cursor to the line that contains the **h1** element. Press **Enter** to open up a new, blank screen line, and then paste the URL you copied earlier into the line.
8. Paste this HTML markup immediately before the URL:

```
<img src='
```

9. Place a closing single quotation mark and a closing angle bracket at the end of the URL:

```
'>
```

The resulting line will look like this:

```
<img src='https://storage.googleapis.com/qwiklabs-gcp-0005e186fa559a09/my-excellent-blog.png'>
```

The effect of these steps is to place the line containing `` immediately before the line containing `<h1>...</h1>`

Do not copy the URL shown here. Instead, copy the URL shown by the Storage browser in your own Cloud Platform project.

10. Press **Ctrl+O**, and then press **Enter** to save your edited file.
11. Press **Ctrl+X** to exit the nano text editor.
12. Restart the web server:

```
sudo service apache2 restart
```

13. Return to the web browser tab in which you opened your **bloghost** VM instance's external IP address. When you load the page, its content now includes a banner image.

Task 2: Confirm that needed APIs are enabled

1. Make a note of the name of your GCP project. This value is shown in the top bar of the Google Cloud Platform Console. It will be of the form `qwiklabs-gcp-` followed by hexadecimal numbers.



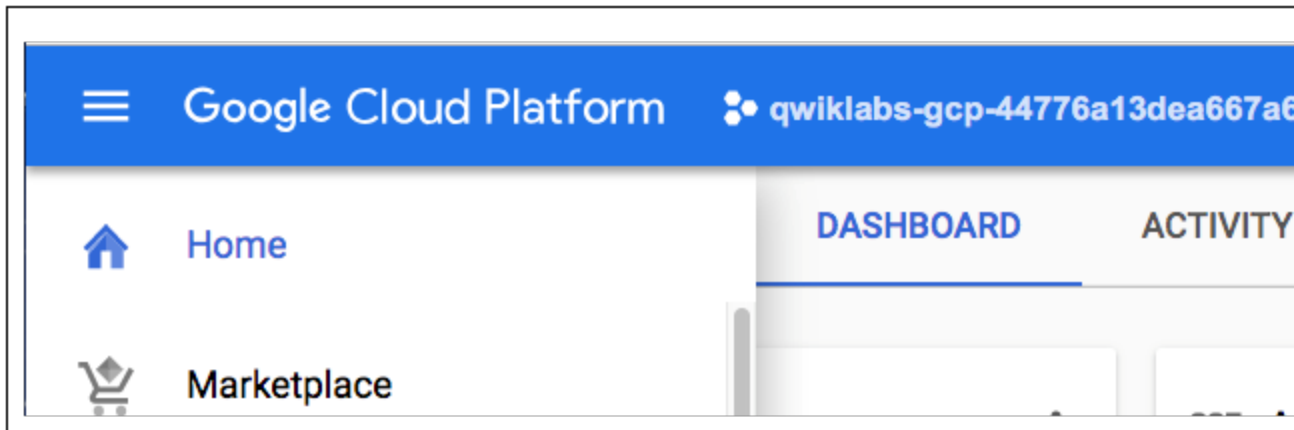
2. In the GCP Console, on the **Navigation menu** (), click **APIs & Services**.
3. Scroll down in the list of enabled APIs, and confirm that both of these APIs are enabled:

- Kubernetes Engine API
- Container Registry API

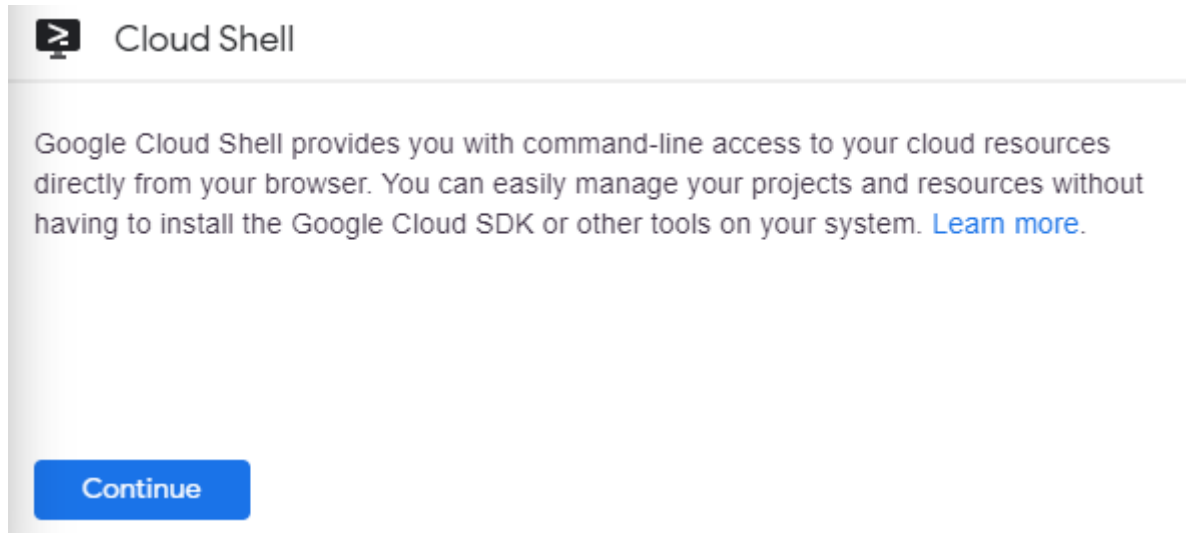
If either API is missing, click **Enable APIs and Services** at the top. Search for the above APIs by name and enable each for your current project. (You noted the name of your GCP project above.)

Task 3: Start a Kubernetes Engine cluster

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



2. Click **Continue**.



3. For convenience, place the zone that Qwiklabs assigned you to into an environment variable called **MY_ZONE**. At the Cloud Shell prompt, type this partial command:

4. `export MY_ZONE=`

followed by the zone that Qwiklabs assigned to you. Your complete command will look similar to this:

```
export MY_ZONE=us-central1-a
```

5. Start a Kubernetes cluster managed by Kubernetes Engine. Name the cluster **webfrontend** and configure it to run 2 nodes:


6. `gcloud container clusters create webfrontend --zone $MY_ZONE --num-nodes 2`

It takes several minutes to create a cluster as Kubernetes Engine provisions virtual machines for you.

7. After the cluster is created, check your installed version of Kubernetes using the `kubectl version` command:

```
8. kubectl version
```

The `gcloud container clusters create` command automatically authenticates `kubectl` for you.

9. View your running nodes in the GCP Console. On the **Navigation menu** (), click **Compute Engine > VM Instances**.

Your Kubernetes cluster is now ready for use.

Click *Check my progress* to verify the objective.

Start a Kubernetes Engine cluster

Check my progress

Task 4: Run and deploy a container

1. From your Cloud Shell prompt, launch a single instance of the `nginx` container. (Nginx is a popular web server.)

```
2. kubectl create deploy nginx --image=nginx:1.17.10
```

In Kubernetes, all containers run in pods. This use of the `kubectl create` command caused Kubernetes to create a deployment consisting of a single pod containing the `nginx` container. A Kubernetes deployment keeps a given number of pods up and running even in the event of failures among the nodes on which they run. In this command, you launched the default number of pods, which is 1.

Note: If you see any deprecation warning about future version you can simply ignore it for now and can proceed further.

2. View the pod running the `nginx` container:

```
3. kubectl get pods
```

4. Expose the nginx container to the Internet:

```
5. kubectl expose deployment nginx --port 80 --type LoadBalancer
```

Kubernetes created a service and an external load balancer with a public IP address attached to it. The IP address remains the same for the life of the service. Any network traffic to that public IP address is routed to pods behind the service: in this case, the nginx pod.

6. View the new service:

```
7. kubectl get services
```

You can use the displayed external IP address to test and contact the nginx container remotely.

It may take a few seconds before the **External-IP** field is populated for your service. This is normal. Just re-run the `kubectl get services` command every few seconds until the field is populated.

8. Open a new web browser tab and paste your cluster's external IP address into the address bar. The default home page of the Nginx browser is displayed.

9. Scale up the number of pods running on your service:

```
10. kubectl scale deployment nginx --replicas 3
```

Scaling up a deployment is useful when you want to increase available resources for an application that is becoming more popular.

11. Confirm that Kubernetes has updated the number of pods:

```
12. kubectl get pods
```

13. Confirm that your external IP address has not changed:

```
14. kubectl get services
```


15. Return to the web browser tab in which you viewed your cluster's external IP address. Refresh the page to confirm that the nginx web server is still responding.

Click *Check my progress* to verify the objective.

3. APP ENGINE

Task 1: Initialize App Engine

1. Initialize your App Engine app with your project and choose its region:

```
2. gcloud app create --project=$DEVSHHELL_PROJECT_ID
```

When prompted, select the [region](#) where you want your App Engine application located.

3. Clone the source code repository for a sample application in the **hello_world** directory:

```
4. git clone https://github.com/GoogleCloudPlatform/python-docs-samples
```

5. Navigate to the source directory:

```
6. cd python-docs-samples/appengine/standard_python3/hello_world
```

Task 2: Run Hello World application locally

In this task, you run the Hello World application in a local, virtual environment in Cloud Shell.

Ensure that you are at the Cloud Shell command prompt.

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

```
2. sudo apt-get update
```

3. Set up a virtual environment in which you will run your application. Python virtual environments are used to isolate package installations from the system.

```
4. sudo apt-get install virtualenv
```

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

```
virtualenv -p python3 venv
```

5. Activate the virtual environment.

```
6. source venv/bin/activate
```

7. Navigate to your project directory and install dependencies.

```
8. pip install -r requirements.txt
```

9. Run the application:

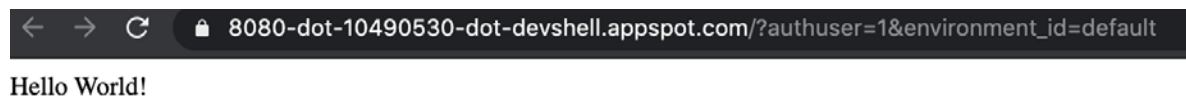
```
10. python main.py
```

Please ignore the warning if any.

6. In **Cloud Shell**, click **Web preview** (🌐) > **Preview on port 8080** to preview the application.

To access the **Web preview** icon, you may need to collapse the **Navigation menu**.

Result:



The screenshot shows a web browser window with the address bar displaying the URL: 8080-dot-10490530-dot-devshell.appspot.com/?authuser=1&environment_id=default. The page content below the address bar says "Hello World!".

7. To end the test, return to Cloud Shell and press **Ctrl+C** to abort the deployed service.

8. Using the Cloud Console, verify that the app is not deployed. In the Cloud Console,

on the **Navigation menu** (), click **App Engine** > **Dashboard**.

Notice that no resources are deployed.

Task 3: Deploy and run Hello World on App Engine

To deploy your application to the App Engine Standard environment:

1. Navigate to the source directory:

```
2. cd ~/python-docs-samples/appengine/standard_python3/hello_world
```

3. Deploy your Hello World application.

```
4. gcloud app deploy
```

If prompted "Do you want to continue (Y/n)?", press Y and then Enter.

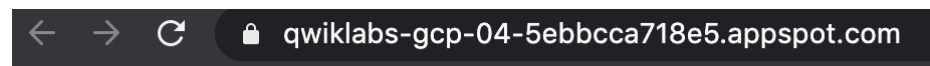
This **app deploy** command uses the *app.yaml* file to identify project configuration.

3. Launch your browser to view the app at http://YOUR_PROJECT_ID.appspot.com

```
4. gcloud app browse
```

Copy and paste the URL into a new browser window.

Result:



Hello World!

Congratulations! You created your first application using App Engine.

Click *Check my progress* to verify the objective.

Deploy the Hello World application to App Engine

Check my progress

Task 4: Disable the application

App Engine offers no option to **Undeploy** an application. After an application is deployed, it remains deployed, although you could instead replace the application with a simple page that says something like "not in service."

However, you can disable the application, which causes it to no longer be accessible to users.

5. DEPLOYMENT MANAGER AND STACKDRIVER

Task 2: Confirm that needed APIs are enabled

1. Make a note of the name of your GCP project. This value is shown in the top bar of the Google Cloud Platform Console. It will be of the form `qwiklabs-gcp-` followed by hexadecimal numbers.
2. In the GCP Console, on the **Navigation menu**, click **APIs & services**.
3. Scroll down in the list of enabled APIs, and confirm that these APIs are enabled:
 - Cloud Deployment Manager v2 API
 - Cloud Runtime Configuration API
 - Cloud Monitoring API
4. If one or more APIs is missing, click the **Enable APIs and Services** button at top. Search for the above APIs by name and enable each for your current project. (You noted the name of your GCP project above.)

Task 3: Create a Deployment Manager deployment

1. In GCP console, on the top right toolbar, click the Open Cloud Shell button. Click **Continue**.
2. For your convenience, place the zone that Qwiklabs assigned you to into an environment variable called `MY_ZONE`. At the Cloud Shell prompt, type this partial command:

```
export MY_ZONE=
```

followed by the zone that Qwiklabs assigned you to. Your complete command will look similar to this:

```
export MY_ZONE=us-central1-a
```

3. At the Cloud Shell prompt, download an editable Deployment Manager template:

```
gsutil cp gs://cloud-training/gcpfc coreinfra/mydeploy.yaml mydeploy.yaml
```

4. In the Cloud Shell, use the sed command to replace the PROJECT_ID placeholder string with your Google Cloud Platform project ID using this command:

```
sed -i -e "s/PROJECT_ID/$DEVSHHELL_PROJECT_ID/" mydeploy.yaml
```

5. In the Cloud Shell, use the sed command to replace the ZONE placeholder string with your Google Cloud Platform zone using this command:

```
sed -i -e "s/ZONE/$MY_ZONE/" mydeploy.yaml
```

6. View the mydeploy.yaml file, with your modifications, with this command:

```
cat mydeploy.yaml
```

The file will look something like this:

```
resources:
- name: my-vm
  type: compute.v1.instance
  properties:
    zone: us-central1-a
    machineType: zones/us-central1-a/machineTypes/n1-standard-1
  metadata:
    items:
      - key: startup-script
        value: "apt-get update"
  disks:
    - deviceName: boot
      type: PERSISTENT
      boot: true
      autoDelete: true
```

```

initializeParams:

  sourceImage: https://www.googleapis.com/compute/v1/projects/debian-
cloud/global/images/debian-9-stretch-v20180806

  networkInterfaces:

    - network: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-
dcdf854d278b50cd/global/networks/default

  accessConfigs:

    - name: External NAT

  type: ONE_TO_ONE_NAT

```

Do not use the above text literally in your own **mydeploy.yaml** file. Be sure that the zone that is named on the **zone:** and **machineType:** lines in your file matches the zone to which Qwiklabs assigned you. Be sure that the GCP project ID on the **network:** line in your file matches the project ID to which Qwiklabs assigned you, not the one in this example.

7. Build a deployment from the template:

```
gcloud deployment-manager deployments create my-first-depl --config mydeploy.yaml
```

When the deployment operation is complete, the **gcloud** command displays a list of the resources named in the template and their current state.

8. Confirm that the deployment was successful. In the GCP Console, on the **Navigation**



menu (), click **Compute Engine > VM instances**. You will see that a VM instance called **my-vm** has been created, as specified by the template.

9. Click on the VM instance's name to open its VM instance details screen.
10. Scroll down to the **Custom metadata** section. Confirm that the startup script you specified in your Deployment Manager template has been installed.

Click *Check my progress* to verify the objective.

Create a Deployment Manager deployment

Check my progress

Task 4: Update a Deployment Manager deployment

1. Return to your Cloud Shell prompt. Launch the nano text editor to edit the **mydeploy.yaml** file:

```
nano mydeploy.yaml
```

2. Find the line that sets the value of the startup script, value: "apt-get update", and edit it so that it looks like this:

```
value: "apt-get update; apt-get install nginx-light -y"
```


Do not disturb the spaces at the beginning of the line. The YAML templating language relies on indented lines as part of its syntax. As you edit the file, be sure that the **v** in the word **value** in this new line is immediately below the **k** in the word **key** on the line above it.

3. Press **Ctrl+O** and then press **Enter** to save your edited file.
4. Press **Ctrl+X** to exit the **nano** text editor.
5. Return to your Cloud Shell prompt. Enter this command to cause Deployment Manager to update your deployment to install the new startup script:

```
gcloud deployment-manager deployments update my-first-depl --config mydeploy.yaml
```

Wait for the **gcloud** command to display a message confirming that the update operation was completed successfully.



6. In the GCP console, on the **Navigation menu** (), click **Compute Engine > VM instances**.
7. Click on the **my-vm** VM instance's name to open its **VM instance details** pane.
8. Scroll down to the **Custom metadata** section. Confirm that the startup script has been updated to the value you declared in your Deployment Manager template.

Click *Check my progress* to verify the objective.

Update the Deployment Manager deployment

Check my progress

Task 5: View the Load on a VM using Cloud Monitoring

1. In the GCP Console, on the **Navigation menu**, click **Compute Engine > VM instances**.
2. To open a command prompt on the **my-vm** instance, click **SSH** in its row in the **VM instances** list.
3. In the ssh session on **my-vm**, execute this command to create a CPU load:

```
dd if=/dev/urandom | gzip -9 >> /dev/null &
```

This Linux pipeline forces the CPU to work on compressing a continuous stream of random data.

Leave the window containing your SSH session open while you proceed with the lab.

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.

The screenshot shows the Google Cloud Platform Monitoring dashboard. The top navigation bar is purple with the Google Cloud Platform logo and a search bar. The left sidebar contains a navigation menu with icons and labels for Monitoring, Overview, Dashboards, Services, Metrics explorer, Alerting, Uptime checks, Groups, and Settings. The main content area is titled 'Overview' and features a 'Welcome to Monitoring!' message. Below the welcome message, there are links to 'VIEW GCE DASHBOARD' and 'VIEW GKE DASHBOARD'. To the right, there is a 'Logging' section with a 'GO TO LOGGING' link. At the bottom, there is a 'Resource dashboards' table with columns 'Name' and 'Resources'. The table lists 'Disks' (1), 'Firewalls' (5), and 'VM Instances' (1). To the right of the table, there is an 'Incidents' section with a 'No rows to display' message.

Name	Resources
Disks	1
Firewalls	5
VM Instances	1

- Click on **Settings** option from the left panel and confirm that the GCP project which Qwiklabs created for you is shown under the **GCP Projects** section.

GCP Projects

Project name ↑	Project ID
qwiklabs-gcp-03-4df0cc46d983	qwiklabs-gcp-03-4df0cc46d983

[ADD GCP PROJECTS](#)

- Run the commands shown on screen in the SSH window of your VM instance to install both the Monitoring and Logging agents.

```
curl -sSO https://dl.google.com/cloudagents/install-monitoring-agent.sh
sudo bash install-monitoring-agent.sh

curl -sSO https://dl.google.com/cloudagents/install-logging-agent.sh
sudo bash install-logging-agent.sh
```

- Once both of the agents have been installed on your project's VM, click **Metrics Explorer** under the main Cloud Monitoring menu on the far left.
- In the **Metric** pane of **Metrics Explorer**, select the resource type **VM instance** and the metric **CPU usage**.

In the resulting graph, notice that CPU usage increased sharply a few minutes ago.

- Terminate your workload generator. Return to your ssh session on **my-vm** and enter this command:

```
kill %1
```

5. BIGQUERY

Task 2: Load data from Cloud Storage into BigQuery

1. In the Console, on the **Navigation menu** click **BigQuery** then click **Done**.
2. Create a new dataset within your project by selecting your project in the Resources section, then clicking on **CREATE DATASET** on the right.
3. In the **Create Dataset** dialog, for **Dataset ID**, type **logdata**.
4. For **Data location**, select the continent closest to the region your project was created in. click **Create dataset**.
5. Create a new table in the **logdata** to store the data from the CSV file.
6. Click on **Create Table**. On the **Create Table** page, in the **Source** section:
 - For **Create table from**, choose select **Google Cloud Storage**, and in the field, type gs://cloud-training/gcpfci/access_log.csv.
 - Verify **File format** is set to **CSV**.

Note: When you have created a table previously, the Create from Previous Job option allows you to quickly use your settings to create similar tables.

7. In the **Destination** section:
 - For **Dataset name**, leave **logdata** selected.
 - For **Table name**, type **accesslog**.
 - For **Table type**, **Native table** should be selected.
8. Under **Schema** section, for **Auto detect** check the **Schema and input Parameters**.
9. Accept the remaining default values and click **Create Table**.

BigQuery creates a load job to create the table and upload data into the table (this may take a few seconds).
10. (Optional) To track job progress, click **Job History**.
11. When the load job is complete, click **logdata > accesslog**.
12. On the table details page, click **Details** to view the table properties, and then click **Preview** to view the table data.

Each row in this table logs a hit on a web server. The first field, **string_field_0**, is the IP address of the client. The fourth through ninth fields log the day, month, year, hour, minute, and second at which the hit occurred. In this activity, you will learn about the daily pattern of load on this web server.

Click *Check my progress* to verify the objective.

Load data from Cloud Storage into BigQuery

Check my progress

Task 3: Perform a query on the data using the BigQuery web UI

In this section of the lab, you use the BigQuery web UI to query the **accesslog** table you created previously.

1. In the **Query editor** window, type (or copy-and-paste) the following query:
2. Because you told BigQuery to automatically discover the schema when you load the data, the hour of the day during which each web hit arrived is in a field called **int_field_6**.

3. `select int64_field_6 as hour, count(*) as hitcount from logdata.accesslog`
4. `group by hour`

`order by hour`

Notice that the Query Validator tells you that the query syntax is valid (indicated by the green check mark) and indicates how much data the query will process. The amount of data processed allows you to determine the price of the query using the [Cloud Platform Pricing Calculator](#).

5. Click **Run** and examine the results. At what time of day is the website busiest? When is it least busy?

Task 4: Perform a query on the data using the bq command

In this section of the lab, you use the bq command in Cloud Shell to query the **accesslog** table you created previously.

1. On the **Google Cloud Platform** Console, click **Activate Cloud Shell** then click **Continue**.
2. At the Cloud Shell prompt, enter this command:

3. `bq query "select string_field_10 as request, count(*) as requestcount from logdata.accesslog group by request order by requestcount desc"`

The first time you use the bq command, it caches your Google Cloud Platform credentials, and then asks you to choose your default project. Choose the project that Qwiklabs assigned you to. Its name will look like qwiklabs-gcp- followed by a hexadecimal number.

The bq command then performs the action requested on its command line. What URL offered by this web server was most popular? Which was least popular

6. MARKETPLACE – JENKINS

Task 1: Use Marketplace to build a deployment

Navigate to Marketplace

1. In the Cloud Console, on the **Navigation menu**, click **Marketplace**.
2. Locate the Jenkins deployment by searching for **Jenkins Certified by Bitnami**.
3. Click on the deployment and read about the service provided by the software.

Jenkins is an open-source continuous integration environment. You can define jobs in Jenkins that can perform tasks such as running a scheduled build of software and backing up data. Notice the software that is installed as part of Jenkins shown in the left side of the description.

Launch Jenkins

1. Click **Launch**.
2. Verify the deployment, accept the terms and services and click **Deploy**.
3. Click **Close** on the Welcome to Deployment Manager window.

It will take a minute or two for Deployment Manager to set up the deployment. You can watch the status as tasks are being performed. Deployment Manager is acquiring a virtual machine instance and installing and configuring software for you. You will see **jenkins-1 has been deployed** when the process is complete.

Deployment Manager is a Google Cloud service that uses templates written in a combination of YAML, python, and Jinja2 to automate the allocation of Google Cloud

resources and perform setup tasks. Behind the scenes a virtual machine has been created. A startup script was used to install and configure software, and network Firewall Rules were created to allow traffic to the service.

Task 2: Examine the deployment

In this section, you examine what was built in Google Cloud.

View installed software and login to Jenkins

1. In the right pane, click **More about the software** to view additional software details. Look at all the software that was installed.
2. Copy the **Admin user** and **Admin password** values to a text editor.
3. Click **Visit the site** to view the site in another browser tab. If you get an error, you might have to reload the page a couple of times.
4. Log in with the **Admin user** and **Admin password** values.

Note: If you get http 404 error, then remove the **/jenkins/** part from the site address and hit **Enter**. For example: http://35.238.162.236

5. After logging in, if you are asked to Customize Jenkins. Click *Install suggested plugins*, and then click *Restart* after the installation is complete. The restart will take a couple of minutes.

Note: If you are getting an installation error, **retry** the installation and if it fails again, **continue** past the error and **save and finish** before restarting. The code of this solution is managed and supported by Bitnami.

Explore Jenkins

1. In the Jenkins interface, in the left pane, click **Manage Jenkins**. Look at all of the actions available. You are now prepared to manage Jenkins. The focus of this lab is Google Cloud infrastructure, not Jenkins management, so seeing that this menu is available is the purpose of this step.
2. Leave the browser window open to the Jenkins service. You will use it in the next task.

Now you have seen that the software is installed and working properly. In the next task you will open an SSH terminal session to the VM where the service is hosted, and verify that you have administrative control over the service.

Task 3: Administer the service

View the deployment and SSH to the VM

1. In the Cloud Console, on the **Navigation menu**, click **Deployment Manager**.
2. Click **jenkins-1**.
3. Click **SSH** to connect to the Jenkins server.

The Console interface is performing several tasks for you transparently. For example, it has transferred keys to the virtual machine that is hosting the Jenkins software so that you can connect securely to the machine using SSH.

Shut down and restart the services

1. In the SSH window, enter the following command to shut down all the running services:

```
sudo /opt/bitnami/ctlscript.sh stop
```

2. Refresh the browser window for the Jenkins UI. You will no longer see the Jenkins interface because the service was shut down.
3. In the SSH window, enter the following command to restart the services:

```
sudo /opt/bitnami/ctlscript.sh restart
```

4. Return to the browser window for the Jenkins UI and refresh it. You may have to do it a couple of times before the service is reachable.
5. In the SSH window, type `exit` to close the SSH terminal session.


VIRTUAL PRIVATE CLOUD

Task 1. Explore the default network

Each Google Cloud project has a **default** network with subnets, routes, and firewall rules.

View the subnets

The **default** network has a subnet in [each Google Cloud region](#).

- In the Cloud Console, on the **Navigation menu** () , click **VPC network > VPC networks**. Notice the **default** network with its subnets. Each subnet is associated with a Google Cloud region and a private RFC 1918 CIDR block for its internal **IP addresses range** and a **gateway**.

View the routes

Routes tell VM instances and the VPC network how to send traffic from an instance to a destination, either inside the network or outside Google Cloud. Each VPC network comes with some default routes to route traffic among its subnets and send traffic from eligible instances to the internet.

- In the left pane, click **Routes**. Notice that there is a route for each subnet and one for the **Default internet gateway** (0.0.0.0/0). These routes are managed for you, but you can create custom static routes to direct some packets to specific destinations. For example, you can create a route that sends all outbound traffic to an instance configured as a NAT gateway.

View the firewall rules

Each VPC network implements a distributed virtual firewall that you can configure. Firewall rules allow you to control which packets are allowed to travel to which destinations. Every VPC network has two implied firewall rules that block all incoming connections and allow all outgoing connections.

- In the left pane, click **Firewall**. Notice that there are 4 **Ingress** firewall rules for the **default** network:
 - default-allow-icmp
 - default-allow-rdp
 - default-allow-ssh
 - default-allow-internal

These firewall rules allow **ICMP**, **RDP**, and **SSH** ingress traffic from anywhere (0.0.0.0/0) and all **TCP**, **UDP**, and **ICMP** traffic within the network (10.128.0.0/9).

The **Targets**, **Filters**, **Protocols/ports**, and **Action** columns explain these rules.

Delete the Firewall rules

1. In the left pane, click **Firewall**.
2. Select all default network firewall rules.
3. Click **Delete**.
4. Click **Delete** to confirm the deletion of the firewall rules.

Delete the default network

1. In the left pane, click **VPC networks**.
2. Select the **default** network.
3. Click **Delete VPC network**.
4. Click **Delete** to confirm the deletion of the **default** network. Wait for the network to be deleted before continuing.
5. In the left pane, click **Routes**. Notice that there are no routes.
6. In the left pane, click **Firewall**. Notice that there are no firewall rules.

Without a VPC network, there are no routes!

Without a VPC network, you cannot create VM instances, containers, or App Engine applications.




True



False

Try to create a VM instance

Verify that you cannot create a VM instance without a VPC network.

1. On the **Navigation menu** () , click **Compute Engine > VM instances**.
2. Click **Create**.


3. Accept the default values and click **Create**. Notice the error.
4. Click **Management, security, disks, networking, sole tenancy**.
5. Click **Networking**. Notice the **No local network available** error under **Network interfaces**.
6. Click **Cancel**.

As expected, you cannot create a VM instance without a VPC network!

Task 2. Create an auto mode network

You have been tasked to create an auto mode network with two VM instances. Auto mode networks are easy to set up and use because they automatically create subnets in each region. However, you don't have complete control over the subnets created in your VPC network, including regions and IP address ranges used. Feel free to explore more [considerations for choosing an auto mode network](#), but for now, assume that you are using the auto mode network for prototyping purposes.

Create an auto mode VPC network with firewall rules

1. On the **Navigation menu** () , click **VPC network > VPC networks**.
2. Click **Create VPC network**.
3. For **Name**, type **mynetwork**
4. For **Subnet creation mode**, click **Automatic**. Auto mode networks create subnets in each region automatically.
5. For **Firewall rules**, select all available rules.


These are the same standard firewall rules that the default network had. The **deny-all-ingress** and **allow-all-egress** rules are also displayed, but you cannot select or disable them because they are implied. These two rules have a lower **Priority** (higher integers indicate lower priorities) so that the allow ICMP, internal, RDP, and SSH rules are considered first.

6. Click **Create**. When the new network is ready, notice that a subnet was created for each region.
7. Record the IP address range for the subnets in **us-central1** and **eu-west-1**. These will be referred to in the next steps.

Tip: If you ever delete the default network, you can quickly re-create it by creating an auto mode network as you just did.

Create a VM instance in us-central1

Create a VM instance in the us-central1 region. Selecting a region and zone determines the subnet and assigns the internal IP address from the subnet's IP address range.

1. On the **Navigation menu** () , click **Compute Engine > VM instances**.
2. Click **Create**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	mynet-us-vm
Region	us-central1
Zone	us-central1-c
Series	N1
Machine type	n1-standard-1 (1 vCPU, 3.75 GB memory)
Boot disk	Debian GNU/Linux 10 (buster)

4. Click **Create**.
5. Verify that the **Internal IP** for the new instance was assigned from the IP address range for the subnet in **us-central1** (10.128.0.0/20).

The **Internal IP** should be 10.128.0.2 because 10.128.0.1 is reserved for the gateway, and you have not configured any other instances in that subnet.

Create a VM instance in europe-west1

Create a VM instance in the europe-west1 region.

1. Click **Create instance**.
2. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	mynet-eu-vm
Region	europe-west1
Zone	europe-west1-c
Series	N1
Machine type	n1-standard-1 (1 vCPU, 3.75 GB memory)
Boot disk	Debian GNU/Linux 10 (buster)

3. Click **Create**.
4. Verify that the **Internal IP** for the new instance was assigned from the IP address range for the subnet in **europe-west1** (10.132.0.0/20).


The **Internal IP** should be 10.132.0.2 because 10.132.0.1 is reserved for the gateway, and you have not configured any other instances in that subnet.

The **External IP addresses** for both VM instances are ephemeral. If an instance is stopped, any ephemeral external IP addresses assigned to the instance are released back into the general Compute Engine pool and become available for use by other projects. When a stopped instance is started again, a new ephemeral external IP address is assigned to the

instance. Alternatively, you can reserve a static external IP address, which assigns the address to your project indefinitely until you explicitly release it.

Verify connectivity for the VM instances

The firewall rules that you created with **mynetwork** allow ingress SSH and ICMP traffic from within **mynetwork** (internal IP) and outside that network (external IP).

1. On the **Navigation menu** () , click **Compute Engine > VM instances**. Note the external and internal IP addresses for **mynet-eu-vm**.
2. For **mynet-us-vm**, click **SSH** to launch a terminal and connect.

You can SSH because of the **allow-ssh** firewall rule, which allows incoming traffic from anywhere (0.0.0.0/0) for **tcp:22**. The SSH connection works seamlessly because Compute Engine generates an SSH key for you and stores it in one of the following locations:

- By default, Compute Engine adds the generated key to project or instance metadata.
- If your account is configured to use OS Login, Compute Engine stores the generated key with your user account.

Alternatively, you can control access to Linux instances by creating SSH keys and editing public SSH key metadata.

3. To test connectivity to **mynet-eu-vm**'s internal IP, run the following command, replacing **mynet-eu-vm**'s internal IP:

```
ping -c 3 <Enter mynet-eu-vm's internal IP here>
```

You can ping **mynet-eu-vm**'s internal IP because of the **allow-internal** firewall rule.

4. Repeat the same test by running the following:

```
ping -c 3 mynet-eu-vm
```

You can ping **mynet-eu-vm** by its name because VPC networks have an internal DNS service that allows you to address instances by their DNS names instead of their internal IP addresses. This is very useful because the internal IP address can change when you delete and re-create an instance.


5. To test connectivity to **mynet-eu-vm**'s external IP, run the following command, replacing **mynet-eu-vm**'s external IP:

```
ping -c 3 <Enter mynet-eu-vm's external IP here>
```

You can SSH to **mynet-us-vm** and ping **mynet-eu-vm**'s internal and external IP addresses as expected. Alternatively, you can SSH to **mynet-eu-vm** and ping **mynet-us-vm**'s internal and external IP addresses, which also works.

Convert the network to a custom mode network

The auto mode network worked great so far, but you have been asked to convert it to a custom mode network so that new subnets aren't automatically created as new regions become available. This could result in overlap with IP addresses used by manually created subnets or static routes, or could interfere with your overall network planning.

1. On the **Navigation menu** () , click **VPC network > VPC networks**.
2. Click **mynetwork** to open the network details.
3. Click **Edit**.
4. Select **Custom** for the **Subnet creation mode**.
5. Click **Save**.
6. Return to the **VPC networks** page. Wait for the **Mode** of **mynetwork** to change to **Custom**. You can click **Refresh** while you wait.

Click *Check my progress* to verify the objective.

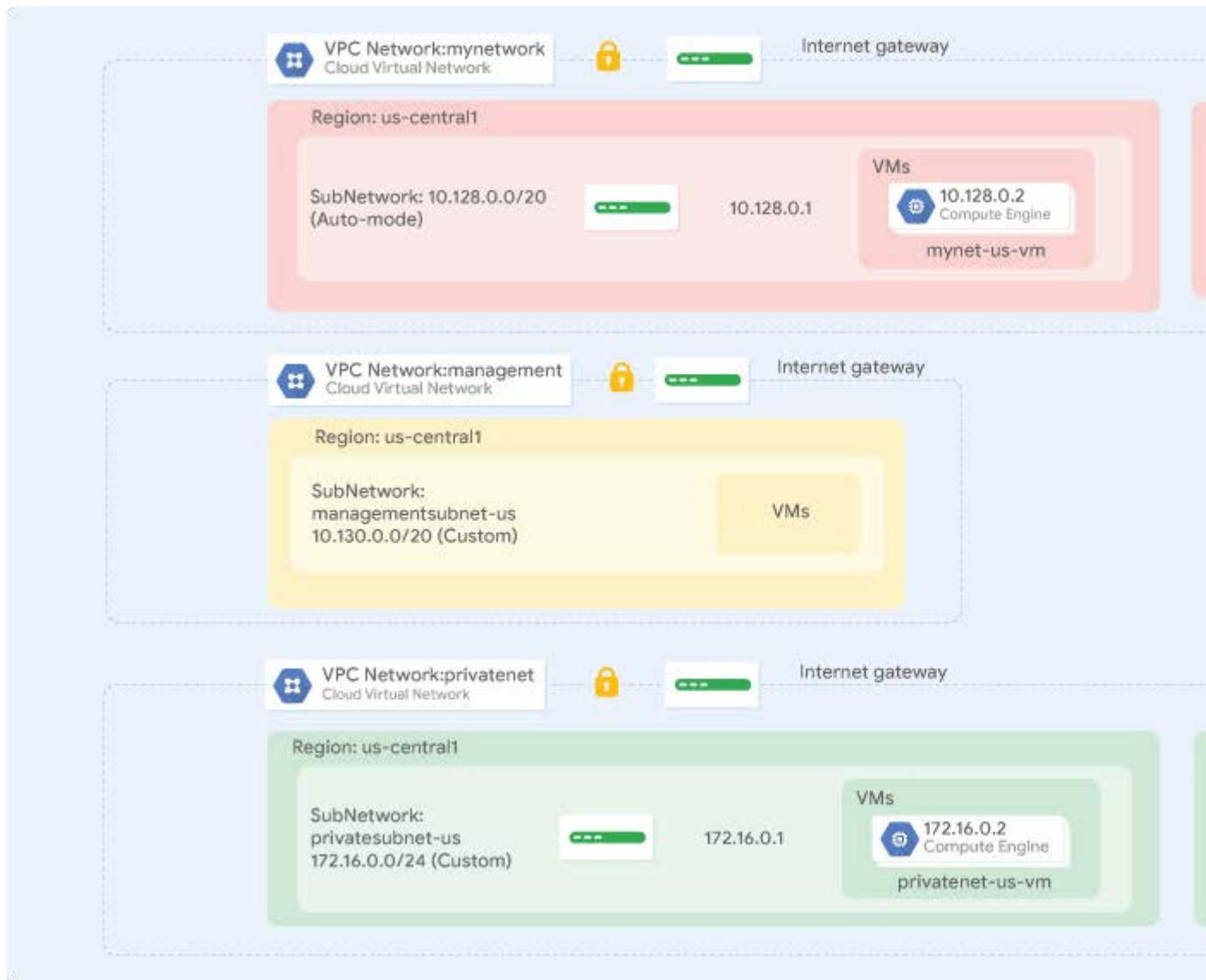
Create a VPC network and VM instances

Check my progress

Converting an auto mode network to a custom mode network is an easy task, and it provides you with more flexibility. We recommend that you use custom mode networks in production.

Task 3. Create custom mode networks


You have been tasked to create two additional custom networks, **managementnet** and **privatenet**, along with firewall rules to allow **SSH**, **ICMP**, and **RDP** ingress traffic and VM instances as shown in this diagram (with the exception of vm-appliance):



Note that the IP CIDR ranges of these networks do not overlap. This allows you to set up mechanisms such as VPC peering between the networks. If you specify IP CIDR ranges that are different from your on-premises network, you could even configure hybrid connectivity using VPN or Cloud Interconnect.

Create the managementnet network

Create the **managementnet** network using the Cloud Console.

1. In the Cloud Console, on the **Navigation menu** () , click **VPC network > VPC networks**.
2. Click **Create VPC Network**.
3. For **Name**, type **managementnet**

4. For **Subnet creation mode**, click **Custom**.
5. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	managementsubnet-us
Region	us-central1
IP address range	10.130.0.0/20


6. Click **Done**.
7. Click **command line**.

These commands illustrate that networks and subnets can be created using the gcloud command line. You will create the **privatenet** network using these commands with similar parameters.

8. Click **Close**.
9. Click **Create**.

Create the privatenet network

Create the **privatenet** network using the gcloud command line.

1. In the Cloud Console, click **Activate Cloud Shell** ().
2. If prompted, click **Continue**.
3. To create the **privatenet** network, run the following command:

```
gcloud compute networks create privatenet --subnet-mode=custom
```

4. To create the **privatesubnet-us** subnet, run the following command:

```
gcloud compute networks subnets create privatesubnet-us --network=privatenet --region=us-central1 --range=172.16.0.0/24
```

5. To create the **privatesubnet-eu** subnet, run the following command:

```
gcloud compute networks subnets create privatesubnet-eu --network=privatenet --region=europe-west1 --range=172.20.0.0/20
```

6. To list the available VPC networks, run the following command:

```
gcloud compute networks list
```

The output should look like this (**do not copy; this is example output**):

NAME	SUBNET_MODE	BGP_ROUTING_MODE	IPV4_RANGE	GATEWAY_IPV4
managementnet	CUSTOM	REGIONAL		
mynetwork	CUSTOM	REGIONAL		
privatenet	CUSTOM	REGIONAL		

7. To list the available VPC subnets (sorted by VPC network), run the following command:


```
gcloud compute networks subnets list --sort-by=NETWORK
```

The output should look like this (**do not copy; this is example output**):

NAME	REGION	NETWORK	RANGE
managementsubnet-us	us-central1	managementnet	10.130.0.0/20
mynetwork	asia-northeast1	mynetwork	10.146.0.0/20
mynetwork	us-west1	mynetwork	10.138.0.0/20

mynetwork	southamerica-east1	mynetwork	10.158.0.0/20
mynetwork	europa-west4	mynetwork	10.164.0.0/20
mynetwork	asia-east1	mynetwork	10.140.0.0/20
mynetwork	europa-north1	mynetwork	10.166.0.0/20
mynetwork	asia-southeast1	mynetwork	10.148.0.0/20
mynetwork	us-east4	mynetwork	10.150.0.0/20
mynetwork	europa-west1	mynetwork	10.132.0.0/20
mynetwork	europa-west2	mynetwork	10.154.0.0/20
mynetwork	europa-west3	mynetwork	10.156.0.0/20
mynetwork	australia-southeast1	mynetwork	10.152.0.0/20
mynetwork	asia-south1	mynetwork	10.160.0.0/20
mynetwork	us-east1	mynetwork	10.142.0.0/20
mynetwork	us-central1	mynetwork	10.128.0.0/20
mynetwork	northamerica-northeast1	mynetwork	10.162.0.0/20
privatesubnet-eu	europa-west1	privatenet	172.20.0.0/20
privatesubnet-us	us-central1	privatenet	172.16.0.0/24

The **managementnet** and **privatenet** networks only have the subnets that you created because they are custom mode networks. **mynetwork** is also a custom mode network, but it started out as an auto mode network, resulting in subnets in each region.

8. In the Cloud Console, on the **Navigation menu** () , click **VPC network > VPC networks**. Verify that the same networks and subnets are listed in the Cloud Console.

Create the firewall rules for managementnet

Create firewall rules to allow **SSH**, **ICMP**, and **RDP** ingress traffic to VM instances on the **managementnet** network.

1. In the Cloud Console, on the **Navigation menu** () , click **VPC network > Firewall**.

2. Click **Create Firewall Rule**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	managementnet-allow-icmp-ssh-rdp
Network	managementnet
Targets	All instances in the network
Source filter	IP Ranges
Source IP ranges	0.0.0.0/0
Protocols and ports	Specified protocols and ports

Make sure to include the **/0** in the **Source IP ranges** to specify all networks.


4. Select **tcp** and specify ports **22** and **3389**.
5. Select **Other protocols** and specify **icmp** protocol.
6. Click **command line**.

These commands illustrate that firewall rules can also be created using the gcloud command line. You will create the **privatenet**'s firewall rules using these commands with similar parameters.

7. Click **Close**.
8. Click **Create**.

Create the firewall rules for privatenet

Create the firewall rules for **privatenet** network using the gcloud command line.

1. Return to **Cloud Shell**. If necessary, click **Activate Cloud Shell** ().
2. To create the **privatenet-allow-icmp-ssh-rdp** firewall rule, run the following command:

```
gcloud compute firewall-rules create privatenet-allow-icmp-ssh-rdp --direction=INGRESS --priority=1000 --network=privatenet --action=ALLOW --rules=icmp,tcp:22,tcp:3389 --source-ranges=0.0.0.0/0
```

The output should look like this (**do not copy; this is example output**):

NAME	NETWORK	DIRECTION	PRIORITY	ALLOW	DENY
privatenet-allow-icmp-ssh-rdp	privatenet	INGRESS	1000	icmp,tcp:22,tcp:3389	


3. To list all the firewall rules (sorted by VPC network), run the following command:

```
gcloud compute firewall-rules list --sort-by=NETWORK
```

The output should look like this (**do not copy; this is example output**):

NAME	NETWORK	DIRECTION	PRIORITY	ALLOW
managementnet-allow-icmp-ssh-rdp	managementnet	INGRESS	1000	icmp,tcp:22,tcp:3389
mynetwork-allow-icmp	mynetwork	INGRESS	1000	icmp
mynetwork-allow-internal	mynetwork	INGRESS	65534	all
mynetwork-allow-rdp	mynetwork	INGRESS	1000	tcp:3389
mynetwork-allow-ssh	mynetwork	INGRESS	1000	tcp:22
privatenet-allow-icmp-ssh-rdp	privatenet	INGRESS	1000	icmp,tcp:22,tcp:3389

The firewall rules for **mynetwork** network have been created for you. You can define multiple protocols and ports in one firewall rule (**privatenet** and **managementnet**) or spread them across multiple rules (**default** and **mynetwork**).

4. In the Cloud Console, on the **Navigation menu** () , click **VPC network > Firewall**. Verify that the same firewall rules are listed in the Cloud Console.

Click *Check my progress* to verify the objective.

Create custom mode VPC networks with firewall rules


Check my progress

Next, create two VM instances:

- **managementnet-us-vm** in **managementsubnet-us**
- **privatenet-us-vm** in **privatesubnet-us**

Create the managementnet-us-vm instance

Create the **managementnet-us-vm** instance using the Cloud Console.

1. In the Cloud Console, on the **Navigation menu** () , click **Compute Engine > VM instances**.
2. Click **Create instance**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	managementnet-us-vm
Region	us-central1
Zone	us-central1-c

Series	N1
Machine type	f1-micro (1 vCPU, 614 MB memory)
Boot disk	Debian GNU/Linux 10 (buster)

- Click **Management, security, disks, networking, sole tenancy**.
- Click **Networking**.
- For **Network interfaces**, click the pencil icon to edit.
- Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Network	managementnet
Subnetwork	managementsubnet-us

The subnets available for selection are restricted to those in the selected region (us-central1).


- Click **Done**.
- Click **command line**.

This illustrates that VM instances can also be created using the gcloud command line. You will create the **privatenet-us-vm** instance using these commands with similar parameters.

- Click **Close**.
- Click **Create**.

Create the privatenet-us-vm instance

Create the **privatenet-us-vm** instance using the gcloud command line.

1. Return to **Cloud Shell**. If necessary, click **Activate Cloud Shell** ().
2. To create the **privatenet-us-vm** instance, run the following command:

```
gcloud compute instances create privatenet-us-vm --zone=us-central1-c --machine-type=f1-micro --subnet=privatesubnet-us --image-family=debian-10 --image-project=debian-cloud --boot-disk-size=10GB --boot-disk-type=pd-standard --boot-disk-device-name=privatenet-us-vm
```

The output should look like this (**do not copy; this is example output**):


NAME	ZONE	MACHINE_TYPE	PREEMPTIBLE	INTERNAL_IP	EXTERNAL_IP	STATUS
privatenet-us-vm	us-central1-c	f1-micro		172.16.0.2	34.66.197.202	RUNNING

3. To list all the VM instances (sorted by zone), run the following command:

```
gcloud compute instances list --sort-by=ZONE
```

The output should look like this (**do not copy; this is example output**):

NAME	ZONE	MACHINE_TYPE	PREEMPTIBLE	INTERNAL_IP	EXTERNAL_IP	STATUS
mynet-eu-vm	europe-west1-c	n1-standard-1		10.132.0.2	34.76.115.41	RUNNING
managementnet-us-vm	us-central1-c	f1-micro		10.130.0.2	35.239.68.123	RUNNING
mynet-us-vm	us-central1-c	n1-standard-1		10.128.0.2	35.202.101.52	RUNNING
privatenet-us-vm	us-central1-c	f1-micro		172.16.0.2	34.66.197.202	RUNNING

4. In the Cloud Console, on the **Navigation menu** () , click **Compute Engine > VM instances**. Verify that the VM instances are listed in the Cloud Console.

5. For **Columns**, select **Network**.

There are three instances in **us-central1-c** and one instance in **europa-west1-c**. However, these instances are spread across three VPC networks (**managementnet**, **mynetwork**, and **privatenet**), with no instance in the same zone and network as another. In the next task, you explore the effect this has on internal connectivity.

Click *Check my progress* to verify the objective.

Create VM instances

Check my progress

You can explore more networking information on each VM instance by clicking the **nic0** link in the **Internal IP** column. The resulting network interface details page shows the subnet along with the IP CIDR range, the firewall rules and routes that apply to the instance, and other network analysis.

Task 4. Explore the connectivity across networks

Explore the connectivity between the VM instances. Specifically, determine the effect of having VM instances in the same zone versus having instances in the same VPC network.

Ping the external IP addresses

Ping the external IP addresses of the VM instances to determine whether you can reach the instances from the public internet.

1. In the Cloud Console, on the **Navigation menu**, click **Compute Engine > VM instances**. Note the external IP addresses for **mynet-eu-vm**, **managementnet-us-vm**, and **privatenet-us-vm**.
2. For **mynet-us-vm**, click **SSH** to launch a terminal and connect.
3. To test connectivity to **mynet-eu-vm**'s external IP, run the following command, replacing **mynet-eu-vm**'s external IP:

```
ping -c 3 <Enter mynet-eu-vm's external IP here>
```

This should work!

4. To test connectivity to **managementnet-us-vm**'s external IP, run the following command, replacing **managementnet-us-vm**'s external IP:

```
ping -c 3 <Enter managementnet-us-vm's external IP here>
```

This should work!

5. To test connectivity to **privatenet-us-vm**'s external IP, run the following command, replacing **privatenet-us-vm**'s external IP:

```
ping -c 3 <Enter privatenet-us-vm's external IP here>
```

This should work!

You can ping the external IP address of all VM instances, even though they are in either a different zone or VPC network. This confirms that public access to those instances is only controlled by the **ICMP** firewall rules that you established earlier.

Ping the internal IP addresses

Ping the internal IP addresses of the VM instances to determine whether you can reach the instances from within a VPC network.

Which instances should you be able to ping from mynet-us-vm using internal IP addresses?

☐

managementnet-us-vm

☐

mynet-eu-vm

☐

privatenet-us-vm

Submit

1. In the Cloud Console, on the **Navigation menu**, click **Compute Engine > VM instances**. Note the internal IP addresses for **mynet-eu-vm**, **managementnet-us-vm**, and **privatenet-us-vm**.

2. Return to the **SSH** terminal for **mynet-us-vm**.
3. To test connectivity to **mynet-eu-vm**'s internal IP, run the following command, replacing **mynet-eu-vm**'s internal IP:

```
ping -c 3 <Enter mynet-eu-vm's internal IP here>
```

You can ping the internal IP address of **mynet-eu-vm** because it is on the same VPC network as the source of the ping (**mynet-us-vm**), even though both VM instances are in separate zones, regions, and continents!

4. To test connectivity to **managementnet-us-vm**'s internal IP, run the following command, replacing **managementnet-us-vm**'s internal IP:

```
ping -c 3 <Enter managementnet-us-vm's internal IP here>
```

This should not work, as indicated by a 100% packet loss!

5. To test connectivity to **privatenet-us-vm**'s internal IP, run the following command, replacing **privatenet-us-vm**'s internal IP:


```
ping -c 3 <Enter privatenet-us-vm's internal IP here>
```

This should not work either, as indicated by a 100% packet loss! You cannot ping the internal IP address of **managementnet-us-vm** and **privatenet-us-vm** because they are in separate VPC networks from the source of the ping (**mynet-us-vm**), even though they are all in the same zone, **us-central1-c**.


6. WORKING WITH VMs

Task 1: Create a utility virtual machine

Create a VM

1. In the Cloud Console, on the **Navigation menu** () , click **Compute Engine > VM instances**.
2. Click **Create**.
3. For **Name**, type a name for your instance. Hover over the question mark icon for advice about what constitutes a properly formed name.
4. For **Region** and **Zone** select **us-central1** and **us-central1-c** respectively.
5. For **Machine configuration**, select **Series** as **N1**.
6. For **Machine type**, examine the options.

Notice that the menu lists the number of vCPUs, the amount of memory, and a symbolic name such as *n1-standard-1*. The symbolic name is the parameter you would use to select the machine type if you were creating a VM using the `gcloud` command. Notice to the right of the zone and machine type that there is a per-month estimated cost.

7. Click **Details** to the right of the **Machine type** list to see the breakdown of estimated costs.
8. For **Machine type**, click **n1-standard-4 (4 vCPUs, 15 GB memory)**. How did the cost change?
9. For **Machine type**, click **n1-standard-1 (1 vCPUs, 3.75 GB memory)**.
10. For **Boot disk**, click **Change**.
11. Click **Version** and select **Debian GNU/Linux 10 (buster)**.
12. Click **Select**.
13. Click **Management, security, disks, networking, sole tenancy**.
14. Click **Networking**.
15. For **Network interfaces**, click the **Edit** icon ().
16. Select **None** for **External IP**.
17. Click **Done**.
18. Leave the remaining settings as their defaults, and click **Create**. Wait until the new VM is created.

External IP addresses that don't fall under the [Free Tier](#) will incur a [small cost](#).

Explore the VM details

1. On the **VM instances** page, click on the name of your VM.
2. Locate **CPU platform** and note the value. Click **Edit**.

Notice that you can't change the machine type, the CPU platform, or the zone.

You can add network tags and allow specific network traffic from the internet through firewalls.

Some properties of a VM are integral to the VM, are established when the VM is created, and cannot be changed. Other properties can be edited. You can add additional disks and you can also determine whether the boot disk is deleted when the instance is deleted. Normally the boot disk defaults to being deleted automatically when the instance is deleted. But sometimes you will want to override this behavior. This feature is very important because you cannot create an image from a boot disk when it is attached to a running instance. So you would need to disable **Delete boot disk when instance is deleted** to enable creating a system image from the boot disk.

3. Examine **Availability policies**.

You can't convert a non-preemptible instance into a preemptible one. This choice must be made at VM creation. A preemptible instance can be interrupted at any time and is available at a lower cost.

If a VM is stopped for any reason, (for example an outage or a hardware failure) the automatic restart feature will start it back up. Is this the behavior you want? Are your applications idempotent (written to handle a second startup properly)?

During host maintenance, the VM is set for live migration. However, you can have the VM terminated instead of migrated.

If you make changes, they can sometimes take several minutes to be implemented, especially if they involve networking changes like adding firewalls or changing the external IP.

4. Click **Cancel**.

Explore the VM logs

1. On the **VM instance details** page for your VM, click **Cloud Logging**.

Notice that you have now navigated to the Cloud Logging page.

This is a structured log view. At the top you can filter by using the pull-down menus, and there is a search box for searching based on labels or text.

2. Click the small triangle to the left of one of the lines to see the kind of information it contains.
3. On the far right, click **View Options** > **Expand All**.

Click *Check my progress* to verify the objective.


Create a utility virtual machine

Check my progress

Task 2: Create a Windows virtual machine

Create a VM



1. On the **Navigation menu** () , click **Compute Engine** > **VM instances**.
2. Click **Create instance**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	Type a name for your VM
Region	europe-west2
Zone	europe-west2-a
Series	N1
Machine type	n1-standard-2(2 vCPUs, 7.5 GB memory)

Boot disk	Change
Public Images > Operating system	Windows Server
Version	Windows Server 2016 Datacenter Core
Boot disk type	SSD persistent disk
Size (GB)	100

4. Click **Select**.
5. For **Firewall**, enable **Allow HTTP traffic** and **Allow HTTPS traffic**.
6. Click **Create**.

When the VM is running, notice that the connection option in the far right column is RDP, not SSH. RDP is the Remote Desktop Protocol. You would need the RDP client installed on your local machine to connect to the Windows desktop.

Note: Installing an RDP client on your local machine is outside the scope of this lab and of the class. For this reason, you will not be connecting to the Windows VM during this lab. However, you will step through the usual procedures up to the point of requiring the RDP client.

Instructions for connecting to Windows VMs are here:

<https://cloud.google.com/compute/docs/instances/windows/connecting-to-windows-instance>

Set the password for the VM

1. Click on the name of your Windows VM to access the **VM instance details**.
2. You don't have a valid password for this Windows VM: you cannot log in to the Windows VM without a password. Click **Set Windows password**.
3. Click **Set**.
4. Copy the provided password, and click **CLOSE**.

You will **not** connect to the Windows VM during this lab. However, the process would look something like the following (depending on the RDP client you installed). The RDP client shown can be installed for Chrome here:

<https://chrome.google.com/webstore/detail/chrome-rdp-for-google-clo/mpbbnannobiobpnfbliimoapbephgifkm?hl=en-US>

On the **VM instances** page, you would click **RDP** for your Windows VM and connect with the password copied earlier.

Click *Check my progress* to verify the objective.


Create a Windows virtual machine

Check my progress

Task 3: Create a custom virtual machine

Create a VM



1. On the **Navigation menu** (), click **Compute Engine > VM instances**.
2. Click **Create instance**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	Type a name for your VM
Region	us-west1
Zone	us-west1-b
Series	N1
Machine type	Custom

Cores	6 vCPU
Memory	32 GB
Boot disk	Debian GNU/Linux 10 (buster)

4. Click **Create**.

Connect via SSH to your custom VM

1. For the custom VM you just created, click **SSH**.
2. To see information about unused and used memory and swap space on your custom VM, run the following command:

```
free
```

3. To see details about the RAM installed on your VM, run the following command:

```
sudo dmidecode -t 17
```

4. To verify the number of processors, run the following command:

```
nproc
```

5. To see details about the CPUs installed on your VM, run the following command:

```
lscpu
```

6. To exit the SSH terminal, run the following command:

```
exit
```

Click *Check my progress* to verify the objective.

Create a custom virtual machine

Check my progress

Task 4: Review

In this lab, you created several virtual machine instances of different types with different characteristics. One was a small utility VM for administration purposes. You also created a standard VM and a custom VM. You launched both Windows and Linux VMs and deleted VMs.

IAM

Task 1: Setup for two users

Sign in to the Cloud Console as the first user

1. For this lab, Qwiklabs has provisioned you with two user names available in the **Connection Details** dialog. Sign in to the Cloud Console in an Incognito window as usual with the **Username 1** provided in Qwiklabs. Note that both user names use the same single password.

Sign in to the Cloud Console as the second user


1. Open another tab in your incognito window.
2. Browse to console.cloud.google.com.
3. Click on the user icon in the top-right corner of the screen, and then click **Add account**.
4. Sign in to the Cloud Console with the **Username 2** provided in Qwiklabs.


Note: At some points in this lab, if you sign out of the **Username 1** account, the **Username 2** account is deleted by Qwiklabs. So remain signed in to **Username 1** until you are done using **Username 2**.

Task 2: Explore the IAM console

Make sure you are on the **Username 1** Cloud Console tab.

Navigate to the IAM console and explore roles

1. On the **Navigation menu** () , click **IAM & admin > IAM**.
2. Click **Add** and explore the roles in the drop-down menu. Note the various roles associated with each resource by navigating the **Roles** menu.


3. Click **Cancel**.
4. Switch to the **Username 2** Cloud Console tab.
5. On the **Navigation menu** () , click **IAM & admin > IAM**. Browse the list for the lines with the names associated with **Username 1** and **Username 2** in the Qwiklabs **Connection Details** dialog.

Username 2 currently has access to the project, but does not have the Project Owner role, so it cannot edit any of the roles. Hover over the pencil icon for **Username 2** to verify this.

6. Switch back to the **Username 1** Cloud Console tab.
7. In the IAM console, for **Username 2**, click on the pencil icon. **Username 2** currently has the **Project Viewer** role. Do not change the Project Role.
8. Click **Cancel**.

Task 3: Prepare a resource for access testing

Create a bucket and upload a sample file

1. Switch to the **Username 1** Cloud Console tab if you aren't already there.
2. On the **Navigation menu** () , click **Storage > Browser**.
3. Click **Create bucket**.
4. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	Enter a globally unique name
Location type	Multi-Regional

Note the bucket name: it will be used in a later step and referred to as [YOUR_BUCKET_NAME]

5. Click **Create**.

6. Click **Upload files**.
7. Upload any sample file from your local machine.
8. After the upload completes, click **Close** on the upload window.
9. When the file has been uploaded, click on the three dots at the end of the line containing the file, and click **Rename**.
10. Rename the file to **sample.txt**, and click **Rename**.

Click *Check my progress* to verify the objective.

Create a bucket and upload a sample file


Check my progress

Verify project viewer access

1. Switch to the **Username 2** Cloud Console tab.
2. In the Console, navigate to **Navigation menu > Storage > Browser**.
3. Verify that **Username 2** can see the bucket.

Task 4: Remove project access

Remove Project Viewer role for Username 2

1. Switch to the **Username 1** Cloud Console tab.
2. On the **Navigation menu** () , click **IAM & admin > IAM**.
3. Select **Username 2** and click the **Remove** icon.

Verify that you're removing access for **Username 2**. If you accidentally remove access for **Username 1** you will have to restart this lab!

4. Confirm by clicking **Confirm** button.

Notice that the user has disappeared from the list! The user has no access now.

Click *Check my progress* to verify the objective.


Remove project access

Check my progress

Verify that Username 2 has lost access


1. Switch to the **Username 2** Cloud Console tab.

2. On the **Navigation menu** () , click **Home**.

3. On the **Navigation menu** () , click **Storage > Browser**. An error will be displayed. If not, refresh the page. **Username 2** still has a Google Cloud account, but has no access to the project.

Task 5: Add storage access

Add storage permissions

1. Copy the value of **Username 2** from the Qwiklabs **Connection Details** dialog.
2. Switch to the **Username 1** Cloud Console tab.
3. On the **Navigation menu** () , click **IAM & admin > IAM**.
4. Click **Add** to add the user.
5. For **New members**, paste the **Username 2** value you copied from the Qwiklabs **Connection Details** dialog.
6. For **Select a role**, select **Cloud Storage > Storage Object Viewer**.
7. Click **Save**.

Click *Check my progress* to verify the objective.

Add storage permissions


Check my progress

Verify that Username 2 has storage access

1. Switch to the **Username 2** Cloud Console tab.

Username 2 doesn't have Project Viewer roles, so that user can't see the project or any of its resources in the Console. However, the user has specific access to Cloud Storage.



2. To start Cloud Shell, click **Activate Cloud Shell** (). If prompted, click **Continue**.
3. To view the contents of the bucket you created earlier, run the following command, replacing [YOUR_BUCKET_NAME] with the unique name of the Cloud Storage bucket you created:

```
gsutil ls gs://[YOUR_BUCKET_NAME]
```

As you can see, **Username 2** has limited access to Cloud Storage.


4. Close the **Username 2** Cloud Console tab. The rest of the lab is performed on the **Username 1** Cloud Console tab.
5. Switch to the **Username 1** Cloud Console tab.

Task 6: Set up the Service Account User

In this part of the lab, you assign narrow permissions to service accounts and learn how to use the Service Account User role.

Create a service account



1. On the **Navigation menu** (), click **IAM & admin > Service accounts**.
2. Click **Create service account**.
3. Specify the **Service account name** as **read-bucket-objects** .
4. Click **Create**.
5. Specify the **Role** as **Cloud Storage > Storage Object Viewer** .
6. Click **Continue**.

7. Click **Done**.

Add the user to the service account

1. Select the **read-bucket-objects** service account.
2. Click **Add member** in the **Permissions** panel. If you do not see the Permission panel, click on **Show Info panel**.

You will grant the user the role of Service Account User, which allows that person to use a service account on a VM, if they have access to the VM.

You could perform this activity for a specific user, group, or domain.

For training purposes, you will grant the Service Account User role to everyone at a company called Altostrat.com. Altostrat.com is a fake company used for demonstration and training.

3. Specify the following, and leave the remaining settings as their defaults:


Property	Value (type value or select option as specified)
New members	altostrat.com
Select a role	Service Accounts > Service Account User

4. Click **Save**.

Grant Compute Engine access

You now give the entire organization at Altostrat the Compute Engine Admin role.



1. On the **Navigation menu** () , click **IAM & admin > IAM**.
2. Click **Add**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
----------	--------------------------------------------------

New members	altostrat.com
Select a role	Compute Engine > Compute Instance Admin (v1)


4. Click **Save**.

This step is a rehearsal of the activity you would perform for a specific user.

This action gives the user limited abilities with a VM instance. The user will be able to connect via SSH to a VM and perform some administration tasks.

Create a VM with the Service Account User



1. On the **Navigation menu** () , click **Compute Engine > VM instances**.
2. Click **Create**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	demoiam
Region	us-central1
Zone	us-centra1-c
Series	N1
Machine Type	f1-micro
Boot disk	Debian GNU/Linux 10 (buster)

Service account	read-bucket-objects
-----------------	---------------------

4. Click **Create**.

Click *Check my progress* to verify the objective.

Set up the Service Account User and create a VM

Check my progress

Task 7: Explore the Service Account User role

At this point, you might have the user test access by connecting via SSH to the VM and performing the next actions. As the owner of the project, you already possess the Service Account User role. So you can simulate what the user would experience by just using SSH to access the VM from the Cloud Console.

The actions you perform and results will be the same as if you were the target user.

Use the Service Account User

1. For **demoiam**, click **SSH** to launch a terminal and connect.
2. Run the following command:

```
gcloud compute instances list
```

Result (**do not copy; this is example output**):

```
ERROR: (gcloud.compute.instances.list) Some requests did not succeed:
- Required 'compute.zones.list' permission for 'projects/qwiklabs-gcp'
```

What happened? Why?

3. Copy the sample.txt file from the bucket you created earlier. Note that the trailing period is part of the command below. It means copy to "this location":

```
gsutil cp gs://[YOUR_BUCKET_NAME]/sample.txt .
```

Result (**do not copy; this is example output**):

```
Copying gs://train-test-iam/sample.txt...  
/ [1 files][ 28.0 B/ 28.0 B]  
Operation completed over 1 objects/28.0 B.
```

4. To rename the file you copied, run the following command:

```
mv sample.txt sample2.txt
```

5. To copy the renamed file back to the bucket, run the following command:

```
gsutil cp sample2.txt gs://[YOUR_BUCKET_NAME]
```

Result (**do not copy; this is example output**):

```
AccessDeniedException: 403 Caller does not have storage.objects.create access to bucket  
train-test-iam.
```

What happened?

Because you connected via SSH to the instance, you can "act as the service account," essentially assuming the same permissions. The service account the instance was started with had the Storage Viewer role, which permits downloading objects from GCS buckets in the project. To list instances in a project, you need to grant the compute.instance.list permission. Because the service account did not have this permission, you could not list instances running in the project. Because the service account *did* have permission to download objects, it could download an object from the bucket. It did not have permission to write objects, so you got a "403 access denied" message.


Task 8: Review

In this lab you exercised granting and revoking Cloud IAM roles, first to a user, **Username 2**, and then to a Service Account User. You could allocate Service Account User credentials and "bake" them into a VM to create specific-purpose authorized bastion hosts.

GAME ENGINE USING VMs

Task 1: Create the VM

Define a VM using advanced options

1. In the Cloud Console, on the **Navigation menu** () , click **Compute Engine > VM instances**.
2. Click **Create**.
3. Specify the following and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	mc-server
Region	us-central1
Zone	us-central1-a
Boot disk	Debian GNU/Linux 9 (stretch)
Identity and API access > Access scopes	Set access for each API
Storage	Read Write

4. Click **Management, security, disks, networking, sole tenancy**.
5. Click **Disks**. You will add a disk to be used for game storage.
6. Click **Add new disk**.
7. Specify the following and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	minecraft-disk
Disk type	SSD Persistent Disk
Source type	None (blank disk)
Size (GB)	50
Encryption	Google-managed key

8. Click **Done**. This creates the disk and automatically attaches it to the VM when the VM is created.

9. Click **Networking**.

10. Specify the following and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Network tags	minecraft-server
Network interfaces	Click default to edit the interface
External IP	Create IP Address
Name	mc-server-ip

11. Click **Reserve**.

12. Click **Done**.

13. Click **Create**.

Task 2: Prepare the data disk

Create a directory and format and mount the disk

The disk is attached to the instance, but it is not yet mounted or formatted.

1. For **mc-server**, click **SSH** to open a terminal and connect.
2. To create a directory that serves as the mount point for the data disk, run the following command:

```
sudo mkdir -p /home/minecraft
```

3. To format the disk, run the following command:

```
sudo mkfs.ext4 -F -E lazy_itable_init=0,\  
lazy_journal_init=0,discard \  
/dev/disk/by-id/google-minecraft-disk
```

Result (**do not copy; this is example output**):

```
mke2fs 1.42.12 (29-Aug-2014)  
Discarding device blocks: done  
Creating filesystem with 13107200 4k blocks and 3276800 inodes  
Filesystem UUID: 3d5b0563-f29e-4107-ad1a-ba7bf11dcf7c  
Superblock backups stored on blocks:  
  
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,  
    4096000, 7962624, 11239424  
Allocating group tables: done  
Writing inode tables: done  
Creating journal (32768 blocks): done  
Writing superblocks and filesystem accounting information: done
```

4. To mount the disk, run the following command:

```
sudo mount -o discard,defaults /dev/disk/by-id/google-minecraft-disk /home/minecraft
```

No output is displayed after the disk is mounted.

Click *Check my progress* to verify the objective.

Create the VM and prepare the data disk

Check my progress

Task 3: Install and run the application

The Minecraft server runs on top of the Java Virtual Machine (JVM), so it requires the Java Runtime Environment (JRE) to run. Because the server doesn't need a graphical user interface, you use the headless version of the JRE. This reduces the JRE's resource usage on the machine, which helps ensure that the Minecraft server has enough room to expand its own resource usage if needed.

Install the Java Runtime Environment (JRE) and the Minecraft server

1. In the SSH terminal for **mc-server**, to update the Debian repositories on the VM, run the following command:

```
sudo apt-get update
```

2. After the repositories are updated, to install the headless JRE, run the following command:

```
sudo apt-get install -y default-jre-headless
```

3. To navigate to the directory where the persistent disk is mounted, run the following command:

```
cd /home/minecraft
```

4. To install **wget**, run the following command:

```
sudo apt-get install wget
```

5. If prompted to continue, type **Y**
6. To download the current Minecraft server JAR file (1.11.2 JAR), run the following command:

```
sudo wget
https://launcher.mojang.com/v1/objects/d0d0fe2b1dc6ab4c65554cb734270872b72dadd6/
server.jar
```

Initialize the Minecraft server

1. To initialize the Minecraft server, run the following command:

```
sudo java -Xmx1024M -Xms1024M -jar server.jar nogui
```

Result (**do not copy; this is example output**):

```
[21:01:54] [main/ERROR]: Failed to load properties from file: server.properties
[21:01:54] [main/WARN]: Failed to load eula.txt
[21:01:54] [main/INFO]: You need to agree to the EULA in order to run the server. Go to
eula.txt for more info.
```

The Minecraft server won't run unless you accept the terms of the End User Licensing Agreement (EULA).

Click *Check my progress* to verify the objective.

Install the Java Runtime Environment (JRE) and the Minecraft server

Check my progress

2. To see the files that were created in the first initialization of the Minecraft server, run the following command:

```
sudo ls -l
```

You could edit the server.properties file to change the default behavior of the Minecraft server.

3. To edit the EULA, run the following command:

```
sudo nano eula.txt
```

4. Change the last line of the file from `eula=false` to `eula=true`
5. Press **Ctrl+O**, **ENTER** to save the file and then press **Ctrl+X** to exit nano.

Don't try to restart the Minecraft server yet. You use a different technique in the next procedure.

Create a virtual terminal screen to start the Minecraft server

If you start the Minecraft server again now, it is tied to the life of your SSH session: that is, if you close your SSH terminal, the server is also terminated. To avoid this issue, you can use `screen`, an application that allows you to create a virtual terminal that can be "detached," becoming a background process, or "reattached," becoming a foreground process. When a virtual terminal is detached to the background, it will run whether you are logged in or not.

1. To install `screen`, run the following command:

```
sudo apt-get install -y screen
```

2. To start your Minecraft server in a `screen` virtual terminal, run the following command: (Use the `-S` flag to name your terminal `mcs`)

```
sudo screen -S mcs java -Xmx1024M -Xms1024M -jar server.jar nogui
```

Result (**do not copy; this is example output**):

```
...
[21:06:06] [Server-Worker-1/INFO]: Preparing spawn area: 83%
[21:06:07] [Server-Worker-1/INFO]: Preparing spawn area: 85%
[21:06:07] [Server-Worker-1/INFO]: Preparing spawn area: 86%
[21:06:08] [Server-Worker-1/INFO]: Preparing spawn area: 88%
[21:06:08] [Server-Worker-1/INFO]: Preparing spawn area: 89%
[21:06:09] [Server-Worker-1/INFO]: Preparing spawn area: 91%
[21:06:09] [Server-Worker-1/INFO]: Preparing spawn area: 93%
[21:06:10] [Server-Worker-1/INFO]: Preparing spawn area: 95%
[21:06:10] [Server-Worker-1/INFO]: Preparing spawn area: 98%
[21:06:11] [Server-Worker-1/INFO]: Preparing spawn area: 99%
[21:06:11] [Server thread/INFO]: Time elapsed: 55512 ms
[21:06:11] [Server thread/INFO]: Done (102.484s)! For help, type "help"
```

Detach from the screen and close your SSH session

1. To detach the screen terminal, press **Ctrl+A, Ctrl+D**. The terminal continues to run in the background. To reattach the terminal, run the following command:

```
sudo screen -r mcs
```

2. If necessary, exit the screen terminal by pressing **Ctrl+A, Ctrl+D**.
3. To exit the SSH terminal, run the following command:


```
exit
```

Congratulations! You set up and customized a VM and installed and configured application software—a Minecraft server!

Task 4: Allow client traffic

Up to this point, the server has an external static IP address, but it cannot receive traffic because there is no firewall rule in place. Minecraft server uses TCP port 25565 by default. So you need to configure a firewall rule to allow these connections.

Create a firewall rule

1. In the Cloud Console, on the **Navigation menu** () , click **VPC network > Firewall**.
2. Click **Create firewall rule**.
3. Specify the following and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	minecraft-rule
Target	Specified target tags
Target tags	minecraft-server

Source filter	IP ranges
Source IP ranges	0.0.0.0/0
Protocols and ports	Specified protocols and ports

4. For **tcp**, specify port **25565**.
5. Click **Create**. Users can now access your server from their Minecraft clients.

Verify server availability

1. In the left pane, click **External IP addresses**.
2. Locate and copy the **External IP address** for the **mc-server** VM.
3. Use the following website to test your Minecraft server: <https://mcsrvstat.us/>

If the above website is not working, you can use a different site or the Chrome extension:

- <https://dinnerbone.com/minecraft/tools/status/>

Click *Check my progress* to verify the objective.


Allow client traffic

Check my progress

Task 5: Schedule regular backups

Backing up your application data is a common activity. In this case, you configure the system to back up Minecraft world data to Cloud Storage.

Create a Cloud Storage bucket

1. On the **Navigation menu** () , click **Compute Engine > VM instances**.
2. For **mc-server**, click **SSH**.

3. Create a globally unique bucket name, and store it in the environment variable `YOUR_BUCKET_NAME`. To make it unique, you can use your Project ID. Run the following command:

```
export YOUR_BUCKET_NAME=<Enter your bucket name here>
```

4. Verify it with echo:

```
echo $YOUR_BUCKET_NAME
```

5. To create the bucket using the `gsutil` tool, part of the Cloud SDK, run the following command:

```
gsutil mb gs://$YOUR_BUCKET_NAME-minecraft-backup
```

If this command failed, you might not have created a unique bucket name. If so, choose another bucket name, update your environment variable, and try to create the bucket again.

To make this environment variable permanent, you can add it to the root's **.profile** by running this command: `echo YOUR_BUCKET_NAME=$YOUR_BUCKET_NAME >> ~/.profile`

Create a backup script

1. In the mc-server SSH terminal, navigate to your home directory:

```
cd /home/minecraft
```

2. To create the script, run the following command:

```
sudo nano /home/minecraft/backup.sh
```

3. Copy and paste the following script into the file:

```
#!/bin/bash

screen -r mcs -X stuff '/save-all\n/save-off\n'

/usr/bin/gsutil cp -R ${BASH_SOURCE%/*}/world gs://${YOUR_BUCKET_NAME}-minecraft-backup/${date "+%Y%m%d-%H%M%S"}-world

screen -r mcs -X stuff '/save-on\n'
```

4. Press **Ctrl+O**, **ENTER** to save the file, and press **Ctrl+X** to exit nano.

The script saves the current state of the server's world data and pauses the server's auto-save functionality. Next, it backs up the server's world data directory (world) and places its contents in a timestamped directory (<timestamp>-world) in the Cloud Storage bucket. After the script finishes backing up the data, it resumes auto-saving on the Minecraft server.

5. To make the script executable, run the following command:


```
sudo chmod 755 /home/minecraft/backup.sh
```

Test the backup script and schedule a cron job

1. In the mc-server SSH terminal, run the backup script:

```
./home/minecraft/backup.sh
```

2. After the script finishes, return to the Cloud Console.

3. To verify that the backup file was written, on the **Navigation menu** (), click **Storage > Browser**.

4. Click on the backup bucket name. You should see a folder with a date-time stamp name. Now that you've verified that the backups are working, you can schedule a cron job to automate the task.

5. In the mc-server SSH terminal, open the cron table for editing:

```
sudo crontab -e
```

6. When you are prompted to select an editor, type the number corresponding to **nano**, and press **ENTER**.

7. At the bottom of the cron table, paste the following line:

```
0 */4 * * * /home/minecraft/backup.sh
```

That line instructs cron to run backups every 4 hours.

8. Press **Ctrl+O**, **ENTER** to save the cron table, and press **Ctrl+X** to exit nano.

This creates about 300 backups a month in Cloud Storage, so you will want to regularly delete them to avoid charges. Cloud Storage offers the Object Lifecycle Management feature to set a Time to Live (TTL) for objects, archive older versions of objects, or "downgrade" storage classes of objects to help manage costs.

Click *Check my progress* to verify the objective.

Schedule regular backups

Check my progress


Task 6: Server maintenance

To perform server maintenance, you need to shut down the server.

Connect via SSH to the server, stop it and shut down the VM

1. In the mc-server SSH terminal, run the following command:

```
sudo screen -r -X stuff '/stop\n'
```

2. In the Cloud Console, on the **Navigation menu** () , click **Compute Engine > VM instances**.
3. Click **mc-server**.
4. Click **Stop**.
5. In the confirmation dialog, click **Stop** to confirm. You will be logged out of your SSH session.

To start up your instance again, visit the instance page and then click **Start**. To start the Minecraft server again, you can establish an SSH connection with the instance, remount your persistent disk, and start your Minecraft server in a new screen terminal, just as you did previously.

Automate server maintenance with startup and shutdown scripts

Instead of following the manual process to mount the persistent disk and launch the server application in a screen, you can use metadata scripts to create a startup script and a shutdown script to do this for you.

1. Click **mc-server**.
2. Click **Edit**.
3. For **Custom metadata**, specify the following:

Key	Value
-----	-------

startup-script-url	https://storage.googleapis.com/cloud-training/archinfra/mcserver/startup.sh
shutdown-script-url	https://storage.googleapis.com/cloud-training/archinfra/mcserver/shutdown.sh

You'll have to click **Add item** to add the shutdown-script-url. When you restart your instance, the startup script automatically mounts the Minecraft disk to the appropriate directory, starts your Minecraft server in a screen session, and detaches the session. When you stop the instance, the shutdown script shuts down your Minecraft server before the instance shuts down. It's a best practice to store these scripts in Cloud Storage.

4. Click **Save**.

Click *Check my progress* to verify the objective.

Server maintenance

Check my progress

Task 7: Review

In this lab, you created a customized virtual machine instance by installing base software (a headless JRE) and application software (a Minecraft game server). You customized the VM by attaching and preparing a high-speed SSD data disk, and you reserved a static external IP so the address would remain consistent. Then you verified availability of the gaming server online. You set up a backup system to back up the server's data to a Cloud Storage bucket, and you tested the backup system. Then you automated backups using cron. Finally, you set up maintenance scripts using metadata for graceful startup and shutdown of the server.

End your lab

CLOUD IAM LAB

Task 1: Setup for two users

Sign in to the Cloud Console as the first user

1. For this lab, Qwiklabs has provisioned you with two user names available in the **Connection Details** dialog. Sign in to the Cloud Console in an Incognito window

as usual with the **Username 1** provided in Qwiklabs. Note that both user names use the same single password.

Sign in to the Cloud Console as the second user

1. Open another tab in your incognito window.
2. Browse to console.cloud.google.com.
3. Click on the user icon in the top-right corner of the screen, and then click **Add account**.
4. Sign in to the Cloud Console with the **Username 2** provided in Qwiklabs.

Note: At some points in this lab, if you sign out of the **Username 1** account, the **Username 2** account is deleted by Qwiklabs. So remain signed in to **Username 1** until you are done using **Username 2**.

Task 2: Explore the IAM console

Make sure you are on the **Username 1** Cloud Console tab.

Navigate to the IAM console and explore roles

1. On the **Navigation menu**, click **IAM & admin > IAM**.
2. Click **Add** and explore the roles in the drop-down menu. Note the various roles associated with each resource by navigating the **Roles** menu.
3. Click **Cancel**.
4. Switch to the **Username 2** Cloud Console tab.
5. On the **Navigation menu**, click **IAM & admin > IAM**. Browse the list for the lines with the names associated with **Username 1** and **Username 2** in the Qwiklabs **Connection Details** dialog.

Username 2 currently has access to the project, but does not have the Project Owner role, so it cannot edit any of the roles. Hover over the pencil icon for **Username 2** to verify this.

6. Switch back to the **Username 1** Cloud Console tab.
7. In the IAM console, for **Username 2**, click on the pencil icon. **Username 2** currently has the **Project Viewer** role. Do not change the Project Role.
8. Click **Cancel**.

Task 3: Prepare a resource for access testing

Create a bucket and upload a sample file

1. Switch to the **Username 1** Cloud Console tab if you aren't already there.
2. On the **Navigation menu**, click **Storage > Browser**.
3. Click **Create bucket**.
4. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	Enter a globally unique name
Location type	Multi-Regional

Note the bucket name: it will be used in a later step and referred to as [YOUR_BUCKET_NAME]

5. Click **Create**.
6. Click **Upload files**.
7. Upload any sample file from your local machine.
8. After the upload completes, click **Close** on the upload window.
9. When the file has been uploaded, click on the three dots at the end of the line containing the file, and click **Rename**.
10. Rename the file to **sample.txt**, and click **Rename**.

Click *Check my progress* to verify the objective.

Create a bucket and upload a sample file

Check my progress

Verify project viewer access

1. Switch to the **Username 2** Cloud Console tab.

2. In the Console, navigate to **Navigation menu** > **Storage** > **Browser**.
3. Verify that **Username 2** can see the bucket.

Task 4: Remove project access

Remove Project Viewer role for Username 2

1. Switch to the **Username 1** Cloud Console tab.
2. On the **Navigation menu**, click **IAM & admin** > **IAM**.
3. Select **Username 2** and click the **Remove** icon.

Verify that you're removing access for **Username 2**. If you accidentally remove access for **Username 1** you will have to restart this lab!

4. Confirm by clicking **Confirm** button.

Notice that the user has disappeared from the list! The user has no access now.

Click *Check my progress* to verify the objective.

Remove project access

Check my progress

Verify that Username 2 has lost access

1. Switch to the **Username 2** Cloud Console tab.
2. On the **Navigation menu**, click **Home**.
3. On the **Navigation menu**, click **Storage** > **Browser**. An error will be displayed. If not, refresh the page. **Username 2** still has a Google Cloud account, but has no access to the project.

Task 5: Add storage access

Add storage permissions

1. Copy the value of **Username 2** from the Qwiklabs **Connection Details** dialog.

2. Switch to the **Username 1** Cloud Console tab.
3. On the **Navigation menu**, click **IAM & admin > IAM**.
4. Click **Add** to add the user.
5. For **New members**, paste the **Username 2** value you copied from the Qwiklabs **Connection Details** dialog.
6. For **Select a role**, select **Cloud Storage > Storage Object Viewer**.
7. Click **Save**.

Click *Check my progress* to verify the objective.

Add storage permissions

Check my progress

Verify that Username 2 has storage access

1. Switch to the **Username 2** Cloud Console tab.

Username 2 doesn't have Project Viewer roles, so that user can't see the project or any of its resources in the Console. However, the user has specific access to Cloud Storage.

2. To start Cloud Shell, click **Activate Cloud Shell**. If prompted, click **Continue**.
3. To view the contents of the bucket you created earlier, run the following command, replacing [YOUR_BUCKET_NAME] with the unique name of the Cloud Storage bucket you created:

```
gsutil ls gs://[YOUR_BUCKET_NAME]
```

As you can see, **Username 2** has limited access to Cloud Storage.

4. Close the **Username 2** Cloud Console tab. The rest of the lab is performed on the **Username 1** Cloud Console tab.
5. Switch to the **Username 1** Cloud Console tab.

Task 6: Set up the Service Account User

In this part of the lab, you assign narrow permissions to service accounts and learn how to use the Service Account User role.

Create a service account

1. On the **Navigation menu**, click **IAM & admin > Service accounts**.
2. Click **Create service account**.
3. Specify the **Service account name** as **read-bucket-objects** .
4. Click **Create**.
5. Specify the **Role** as **Cloud Storage > Storage Object Viewer** .
6. Click **Continue**.
7. Click **Done**.

Add the user to the service account

1. Select the **read-bucket-objects** service account.
2. Click **Add member** in the **Permissions** panel. If you do not see the Permission panel, click on **Show Info panel**.

You will grant the user the role of Service Account User, which allows that person to use a service account on a VM, if they have access to the VM.

You could perform this activity for a specific user, group, or domain.

For training purposes, you will grant the Service Account User role to everyone at a company called Altostrat.com. Altostrat.com is a fake company used for demonstration and training.

3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
New members	altostrat.com
Select a role	Service Accounts > Service Account User

4. Click **Save**.

Grant Compute Engine access

You now give the entire organization at Altostrat the Compute Engine Admin role.

1. On the **Navigation menu**, click **IAM & admin > IAM**.
2. Click **Add**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
New members	altostrat.com
Select a role	Compute Engine > Compute Instance Admin (v1)

4. Click **Save**.

This step is a rehearsal of the activity you would perform for a specific user.

This action gives the user limited abilities with a VM instance. The user will be able to connect via SSH to a VM and perform some administration tasks.

Create a VM with the Service Account User

1. On the **Navigation menu**, click **Compute Engine > VM instances**.
2. Click **Create**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	demoiam
Region	us-central1

Zone	us-centra1-c
Series	N1
Machine Type	f1-micro
Boot disk	Debian GNU/Linux 10 (buster)
Service account	read-bucket-objects

4. Click **Create**.

Click *Check my progress* to verify the objective.

Set up the Service Account User and create a VM

Check my progress

Task 7: Explore the Service Account User role

At this point, you might have the user test access by connecting via SSH to the VM and performing the next actions. As the owner of the project, you already possess the Service Account User role. So you can simulate what the user would experience by just using SSH to access the VM from the Cloud Console.

The actions you perform and results will be the same as if you were the target user.

Use the Service Account User

1. For **demoiam**, click **SSH** to launch a terminal and connect.
2. Run the following command:

```
gcloud compute instances list
```

Result (**do not copy; this is example output**):

```
ERROR: (gcloud.compute.instances.list) Some requests did not succeed:
```

```
- Required 'compute.zones.list' permission for 'projects/qwiklabs-gcp'
```

What happened? Why?

3. Copy the sample.txt file from the bucket you created earlier. Note that the trailing period is part of the command below. It means copy to "this location":

```
gsutil cp gs://[YOUR_BUCKET_NAME]/sample.txt .
```

Result **(do not copy; this is example output)**:

```
Copying gs://train-test-iam/sample.txt...
```

```
/ [1 files][ 28.0 B/ 28.0 B]
```

```
Operation completed over 1 objects/28.0 B.
```

4. To rename the file you copied, run the following command:

```
mv sample.txt sample2.txt
```

5. To copy the renamed file back to the bucket, run the following command:

```
gsutil cp sample2.txt gs://[YOUR_BUCKET_NAME]
```

Result **(do not copy; this is example output)**:

```
AccessDeniedException: 403 Caller does not have storage.objects.create access to bucket train-test-iam.
```

What happened?

Because you connected via SSH to the instance, you can "act as the service account," essentially assuming the same permissions. The service account the instance was started

with had the Storage Viewer role, which permits downloading objects from GCS buckets in the project. To list instances in a project, you need to grant the `compute.instance.list` permission. Because the service account did not have this permission, you could not list instances running in the project. Because the service account *did* have permission to download objects, it could download an object from the bucket. It did not have permission to write objects, so you got a "403 access denied" message.

Task 8: Review

In this lab you exercised granting and revoking Cloud IAM roles, first to a user, **Username 2**, and then to a Service Account User. You could allocate Service Account User credentials and "bake" them into a VM to create specific-purpose authorized bastion hosts.

STORAGE

Task 1: Preparation

Create a Cloud Storage bucket

1. On the **Navigation menu**, click **Storage > Browser**.

A bucket must have a globally unique name. You could use part of your `PROJECT_ID_1` in the name to help make it unique. For example, if the `PROJECT_ID_1` is "myproj-154920," your bucket name might be "storecore154920."

2. Click **Create bucket**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	Enter a globally unique name
Location type	Multi-region
Access control	Fine-grained (<i>object-level permission in addition to your bucket-level permissions</i>)

4. Make a note of the bucket name. It will be used later in this lab and referred to as `[BUCKET_NAME_1]`.


5. Click **Create**.

Click *Check my progress* to verify the objective.

Create a Cloud Storage bucket

Check my progress

Download a sample file using CURL and make two copies

1. In the Cloud Console, click **Activate Cloud Shell** ().
2. If prompted, click **Continue**.
3. Store [BUCKET_NAME_1] in an environment variable:

```
export BUCKET_NAME_1=<enter bucket name 1 here>
```

4. Verify it with echo:

```
echo $BUCKET_NAME_1
```

5. Run the following command to download a sample file (this sample file is a publicly available Hadoop documentation HTML file):

```
curl \
http://hadoop.apache.org/docs/current/\
hadoop-project-dist/hadoop-common/\
ClusterSetup.html > setup.html
```

6. To make copies of the file, run the following commands:

```
cp setup.html setup2.html
```

```
cp setup.html setup3.html
```

Task 2: Access control lists (ACLs)

Copy the file to the bucket and configure the access control list

1. Run the following command to copy the first file to the bucket:

```
gsutil cp setup.html gs://$BUCKET_NAME_1/
```

2. To get the default access list that's been assigned to setup.html, run the following command:

```
gsutil acl get gs://$BUCKET_NAME_1/setup.html > acl.txt  
cat acl.txt
```

3. To set the access list to private and verify the results, run the following commands:

```
gsutil acl set private gs://$BUCKET_NAME_1/setup.html  
gsutil acl get gs://$BUCKET_NAME_1/setup.html > acl2.txt  
cat acl2.txt
```

4. To update the access list to make the file publicly readable, run the following commands:

```
gsutil acl ch -u AllUsers:R gs://$BUCKET_NAME_1/setup.html  
gsutil acl get gs://$BUCKET_NAME_1/setup.html > acl3.txt  
cat acl3.txt
```

Click *Check my progress* to verify the objective.

Make file publicly readable

Check my progress

Examine the file in the Cloud Console

1. In the Cloud Console, on the **Navigation menu**, click **Storage > Browser**.
2. Click [BUCKET_NAME_1].
3. Verify that for file setup.html, **Public access** has a **Public link** available.

Delete the local file and copy back from Cloud Storage

1. Return to **Cloud Shell**. If necessary, click **Activate Cloud Shell** (▶).
2. Run the following command to delete the setup file:

```
rm setup.html
```

3. To verify that the file has been deleted, run the following command:

```
ls
```

4. To copy the file from the bucket again, run the following command:

```
gsutil cp gs://$BUCKET_NAME_1/setup.html setup.html
```

Task 3: Customer-supplied encryption keys (CSEK)

Generate a CSEK key

For the next step, you need an AES-256 base-64 key.

1. Run the following command to create a key:

```
python3 -c 'import base64; import os; print(base64.encodebytes(os.urandom(32)))'
```

Result (**do not copy; this is example output**):


```
b'tmxElCaabWvJqR7uXEWQF39DhWTcDvChzuCmpHe6sb0=\n'
```

2. Copy the value of the generated key excluding b' and \n' from the command output. Key should be in form of tmxElCaabWvJqR7uXEWQF39DhWTcDvChzuCmpHe6sb0=.

Modify the boto file

The encryption controls are contained in a gsutil configuration file named .boto.

1. To view and open the boto file, run the following commands:

```
ls -al  
  
nano .boto
```

If the .boto file is empty, close the nano editor with **Ctrl+X** and generate a new .boto file using the gsutil config -n command. Then, try opening the file again with the above commands.

If the .boto file is still empty, you might have to locate it using the gsutil version -l command.

2. Locate the line with "#encryption_key="

The bottom of the nano editor provides you with shortcuts to quickly navigate files. Use the **Where Is** shortcut to quickly locate the line with the *#encryption_key=*.

3. Uncomment the line by removing the # character, and paste the key you generated earlier at the end.

Example (**do not copy; this is an example**):

```
Before:  
  
# encryption_key=  
  
After:  
  
encryption_key=tmxElCaabWvJqR7uXEWQF39DhWTcDvChzuCmpHe6sb0=
```

4. Press **Ctrl+O**, **ENTER** to save the boto file, and then press **Ctrl+X** to exit nano.

Upload the remaining setup files (encrypted) and verify in the Cloud Console

1. To upload the remaining setup.html files, run the following commands:

```
gsutil cp setup2.html gs://$BUCKET_NAME_1/  
gsutil cp setup3.html gs://$BUCKET_NAME_1/
```

2. Return to the Cloud Console.
3. Click [BUCKET_NAME_1]. Both setup2.html and setup3.html files show that they are customer-encrypted.

Click *Check my progress* to verify the objective.

Customer-supplied encryption keys (CSEK)

Check my progress

Delete local files, copy new files, and verify encryption

1. To delete your local files, run the following command in Cloud Shell:

```
rm setup*
```

2. To copy the files from the bucket again, run the following command:

```
gsutil cp gs://$BUCKET_NAME_1/setup* ./
```

3. To cat the encrypted files to see whether they made it back, run the following commands:

```
cat setup.html  
cat setup2.html  
cat setup3.html
```

Task 4: Rotate CSEK keys

Move the current CSEK encrypt key to decrypt key

1. Run the following command to open the .boto file:

```
nano .boto
```

2. Comment out the current encryption_key line by adding the # character to the beginning of the line.

The bottom of the nano editor provides you with shortcuts to quickly navigate files. Use the **Where Is** shortcut to quickly locate the line with the *#encryption_key*.

3. Uncomment decryption_key1 by removing the # character, and copy the current key from the encryption_key line to the decryption_key1 line.

Result (**do not copy; this is example output**):

Before:

```
encryption_key=2dFWQGnKhjOcz4h0CudPdVHLG2g+OoxP8FQOIKKTzsg=
```

```
# decryption_key1=
```

After:

```
# encryption_key=2dFWQGnKhjOcz4h0CudPdVHLG2g+OoxP8FQOIKKTzsg=
```

```
decryption_key1=2dFWQGnKhjOcz4h0CudPdVHLG2g+OoxP8FQOIKKTzsg=
```

4. Press **Ctrl+O**, **ENTER** to save the boto file, and then press **Ctrl+X** to exit nano.

Note: In practice, you would delete the old CSEK key from the encryption_key line.

Generate another CSEK key and add to the boto file

1. Run the following command to generate a new key:

```
python3 -c 'import base64; import os; print(base64.encodebytes(os.urandom(32)))'
```

2. Copy the value of the generated key excluding b' and \n' from the command output. Key should be in form of tmxEIcaabWvJqR7uXEWQF39DhWTcDvChzuCmpHe6sb0=.
3. To open the boto file, run the following command:

```
nano .boto
```

4. Uncomment encryption and paste the new key value for encryption_key=.

Result (**do not copy; this is example output**):

Before:

```
# encryption_key=2dFWQGnKhjOcz4h0CudPdVHLG2g+OoxP8FQOIKKTzsg=
```

After:

```
encryption_key=HbFK4I8CaStcvKKIx6aNpdTse0kTsfZNUjFpM+YUEjY=
```

5. Press **Ctrl+O**, **ENTER** to save the boto file, and then press **Ctrl+X** to exit nano.

Rewrite the key for file 1 and comment out the old decrypt key

When a file is encrypted, rewriting the file decrypts it using the decryption_key1 that you previously set, and encrypts the file with the new encryption_key.

You are rewriting the key for setup2.html, but not for setup3.html, so that you can see what happens if you don't rotate the keys properly.

1. Run the following command:

```
gsutil rewrite -k gs://$BUCKET_NAME_1/setup2.html
```

2. To open the boto file, run the following command:

```
nano .boto
```

3. Comment out the current `decryption_key1` line by adding the `#` character back in.

Result (**do not copy; this is example output**):

Before:

```
decryption_key1=2dFWQGnKhjOcz4h0CudPdVHLG2g+OoxP8FQOIKKTzsg=
```

After:

```
# decryption_key1=2dFWQGnKhjOcz4h0CudPdVHLG2g+OoxP8FQOIKKTzsg=
```

4. Press **Ctrl+O**, **ENTER** to save the `boto` file, and then press **Ctrl+X** to exit nano.

Note: In practice, you would delete the old CSEK key from the `decryption_key1` line.

Download setup 2 and setup3

1. To download `setup2.html`, run the following command:

```
gsutil cp gs://$BUCKET_NAME_1/setup2.html recover2.html
```

2. To download `setup3.html`, run the following command:

```
gsutil cp gs://$BUCKET_NAME_1/setup3.html recover3.html
```

What happened? `setup3.html` was not rewritten with the new key, so it can no longer be decrypted, and the copy will fail.

You have successfully rotated the CSEK keys.

Task 5: Enable lifecycle management

View the current lifecycle policy for the bucket

1. Run the following command to view the current lifecycle policy:

```
gsutil lifecycle get gs://$BUCKET_NAME_1
```

There is no lifecycle configuration. You create one in the next steps.

Create a JSON lifecycle policy file

1. To create a file named *life.json*, run the following command:

```
nano life.json
```

2. Paste the following value into the life.json file:

```
{  
  "rule":  
  [  
    {  
      "action": {"type": "Delete"},  
      "condition": {"age": 31}  
    }  
  ]  
}
```

These instructions tell Cloud Storage to delete the object after 31 days.

3. Press **Ctrl+O**, **ENTER** to save the file, and then press **Ctrl+X** to exit nano.

Set the policy and verify

1. To set the policy, run the following command:

```
gsutil lifecycle set life.json gs://$BUCKET_NAME_1
```

2. To verify the policy, run the following command:

```
gsutil lifecycle get gs://$BUCKET_NAME_1
```

Click *Check my progress* to verify the objective.

Enable lifecycle management

Check my progress

Task 6: Enable versioning

View the versioning status for the bucket and enable versioning

1. Run the following command to view the current versioning status for the bucket:

```
gsutil versioning get gs://$BUCKET_NAME_1
```

The Suspended policy means that it is not enabled.

2. To enable versioning, run the following command:

```
gsutil versioning set on gs://$BUCKET_NAME_1
```

3. To verify that versioning was enabled, run the following command:

```
gsutil versioning get gs://$BUCKET_NAME_1
```

Click *Check my progress* to verify the objective.

Enable versioning

Check my progress

Create several versions of the sample file in the bucket

1. Check the size of the sample file:

```
ls -al setup.html
```

2. Open the setup.html file:

```
nano setup.html
```

3. Delete any 5 lines from setup.html to change the size of the file.
4. Press **Ctrl+O**, **ENTER** to save the file, and then press **Ctrl+X** to exit nano.
5. Copy the file to the bucket with the -v versioning option:

```
gsutil cp -v setup.html gs://$BUCKET_NAME_1
```

6. Open the setup.html file:

```
nano setup.html
```

7. Delete another 5 lines from setup.html to change the size of the file.
8. Press **Ctrl+O**, **ENTER** to save the file, and then press **Ctrl+X** to exit nano.
9. Copy the file to the bucket with the -v versioning option:

```
gsutil cp -v setup.html gs://$BUCKET_NAME_1
```

List all versions of the file

1. To list all versions of the file, run the following command:

```
gsutil ls -a gs://$BUCKET_NAME_1/setup.html
```


2. Highlight and copy the name of the oldest version of the file (the first listed), referred to as [VERSION_NAME] in the next step.

Make sure to copy the full path of the file, starting with **gs://**

3. Store the version value in the environment variable [VERSION_NAME].

```
export VERSION_NAME=<Enter VERSION name here>
```

4. Verify it with echo:

```
echo $VERSION_NAME
```

Result (**do not copy; this is example output**):

```
gs://BUCKET_NAME_1/setup.html#1584457872853517
```

Download the oldest, original version of the file and verify recovery

1. Download the original version of the file:

```
gsutil cp $VERSION_NAME recovered.txt
```

2. To verify recovery, run the following commands:

```
ls -al setup.html
```

```
ls -al recovered.txt
```

You have recovered the original file from the backup version. Notice that the original is bigger than the current version because you deleted lines.

Task 7: Synchronize a directory to a bucket

Make a nested directory and sync with a bucket

Make a nested directory structure so that you can examine what happens when it is recursively copied to a bucket.

1. Run the following commands:

```
mkdir firstlevel  
mkdir ./firstlevel/secondlevel  
cp setup.html firstlevel  
cp setup.html firstlevel/secondlevel
```

2. To sync the firstlevel directory on the VM with your bucket, run the following command:

```
gsutil rsync -r ./firstlevel gs://$BUCKET_NAME_1/firstlevel
```

Examine the results

1. In the Cloud Console, on the **Navigation menu**, click **Storage > Browser**.
2. Click [BUCKET_NAME_1]. Notice the subfolders in the bucket.
3. Click on **/firstlevel** and then on **/secondlevel**.
4. Compare what you see in the Cloud Console with the results of the following command:

```
gsutil ls -r gs://$BUCKET_NAME_1/firstlevel
```

5. Exit Cloud Shell:

```
exit
```

Task 8: Cross-project sharing

Switch to the second project

1. Open a new incognito tab.
2. Navigate to console.cloud.google.com.
3. Click the project selector dropdown in the title bar.
4. Click **All**, and then click the second project provided for you in the Qwiklabs Connection Details dialog. Remember that the Project ID is a unique name across all Google Cloud projects. The second project ID will be referred to as [PROJECT_ID_2].

Prepare the bucket

1. In the Cloud Console, on the **Navigation menu**, click **Storage > Browser**.
2. Click **Create bucket**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	Enter a globally unique name
Location type	Multi-region
Access control	Fine-grained (<i>object-level permission in addition to your bucket-level permissions</i>)

4. Note the bucket name. It will be referred to as [BUCKET_NAME_2] in the following steps.
5. Click **Create**.

Upload a text file to the bucket

1. Upload a file to [BUCKET_NAME_2]. Any small example file or text file will do.
2. Note the file name (referred to as [FILE_NAME]); you will use it later.

Create an IAM Service Account

1. In the Cloud Console, on the **Navigation menu**, click **IAM & admin > Service accounts**.
2. Click **Create service account**.
3. On Service account details page, specify the **Service account name** as **cross-project-storage**.
4. Click **Create**.
5. On the Service account permissions page, specify the role as **Cloud Storage > Storage Object Viewer**.
6. Click **Continue** and then **Done**.
7. Click the **cross-project-storage** service account to add the JSON key.
8. Click **Add Key** dropdown and select **Create new key**.
9. Select **JSON** as the key type and click **Create**. A JSON key file will be downloaded. You will need to find this key file and upload it in into the VM in a later step.
10. Click **Close**.
11. On your hard drive, rename the JSON key file to **credentials.json**.
12. In the upper pane, switch back to [PROJECT_ID_1].

Click *Check my progress* to verify the objective.

Create the resources in the second project

Check my progress

Create a VM

1. On the **Navigation menu**, click **Compute Engine > VM instances**.
2. Click **Create**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
----------	--------------------------------------------------

Name	crossproject
Region	europe-west1
Zone	europe-west1-d
Series	N1
Machine type	n1-standard-1
Boot disk	Debian GNU/Linux 10 (buster)

4. Click **Create**.

SSH to the VM

1. For **crossproject**, click **SSH** to launch a terminal and connect.

If the message appears like **Connection via Cloud Identity-Aware Proxy Failed** then click **Connect without Identity-Aware Proxy**.

2. Store [BUCKET_NAME_2] in an environment variable:

```
export BUCKET_NAME_2=<enter bucket name 2 here>
```

3. Verify it with echo:

```
echo $BUCKET_NAME_2
```

4. Store [FILE_NAME] in an environment variable:

```
export FILE_NAME=<enter FILE_NAME here>
```

5. Verify it with echo:

```
echo $FILE_NAME
```

6. List the files in [PROJECT_ID_2]:

```
gsutil ls gs://$BUCKET_NAME_2/
```

Result (**do not copy; this is example output**):

```
AccessDeniedException: 403 Caller does not have storage.objects.list access to bucket [BUCKET_NAME_2].
```

Authorize the VM

1. To upload credentials.json through the SSH VM terminal, click on the gear icon in the upper-right corner, and then click **Upload file**.
2. Select credentials.json and upload it.
3. Click **Close** in the File Transfer window.
4. Verify that the JSON file has been uploaded to the VM:

```
ls
```

Result (**do not copy; this is example output**):

```
credentials.json
```

5. Enter the following command in the terminal to authorize the VM to use the Google Cloud API:

```
gcloud auth activate-service-account --key-file credentials.json
```

The image you are using has the Google Cloud SDK pre-installed; therefore, you don't need to initialize the Google Cloud SDK. If you are attempting this lab in a different environment, make sure you have followed these procedures regarding installing the Google Cloud SDK:

<https://cloud.google.com/sdk/downloads>

Verify access

1. Retry this command:

```
gsutil ls gs://$BUCKET_NAME_2/
```

2. Retry this command:

```
gsutil cat gs://$BUCKET_NAME_2/$FILE_NAME
```

3. Try to copy the credentials file to the bucket:

```
gsutil cp credentials.json gs://$BUCKET_NAME_2/
```

Result **(do not copy; this is example output)**:

```
Copying file://credentials.json [Content-Type=application/json]...
```

```
AccessDeniedException: 403 Caller does not have storage.objects.create access to bucket [BUCKET_NAME_2].
```

Modify role

1. In the upper pane, switch back to [PROJECT_ID_2].
2. In the Cloud Console, on the **Navigation menu**, click **IAM & admin > IAM**.
3. Click the pencil icon for the **cross-project-storage** service account (You might have to scroll to the right to see this icon).
4. Click on the **Storage Object Viewer** role, and then click **Cloud Storage > Storage Object Admin**.

5. Click **Save**. If you don't click **Save**, the change will not be made.

Click *Check my progress* to verify the objective.

Create and verify the resources in the first project

Check my progress

It doesn't look like you've completed this step yet. Try again.

Verify changed access

1. Return to the SSH terminal for **crossproject**.
2. Copy the credentials file to the bucket:

```
gsutil cp credentials.json gs://$BUCKET_NAME_2/
```

Result (**do not copy; this is example output**):

```
Copying file:///credentials.json [Content-Type=application/json]...
- [1 files][ 2.3 KiB/ 2.3 KiB]
Operation completed over 1 objects/2.3 KiB.
```

In this example the VM in PROJECT_ID_1 can now upload files to Cloud Storage in a bucket that was created in another project.

Note that the project where the bucket was created is the billing project for this activity. That means if the VM uploads a ton of files, it will not be billed to PROJECT_ID_1, but instead to PROJECT_ID_2.

Task 9: Review

In this lab you learned to create and work with buckets and objects, and you learned about the following features for Cloud Storage:


- CSEK: Customer-supplied encryption key
- Use your own encryption keys
- Rotate keys

- ACL: Access control list
- Set an ACL for private, and modify to public
- Lifecycle management
- Set policy to delete objects after 31 days
- Versioning
- Create a version and restore a previous version
- Directory synchronization
- Recursively synchronize a VM directory with a bucket
- Cross-project resource sharing using IAM
- Use IAM to enable access to resources across projects

CLOUD SQL CONNECTIONS

Task 1: Create a Cloud SQL database

In this task, you configure a SQL server according to Google Cloud best practices and create a Private IP connection.

1. On the **Navigation menu** () , click **SQL**.
2. Click **Create instance**.
3. Click **Choose MySQL**.
4. Specify the following, and leave the remaining settings as their defaults:

Property	Value
Instance ID	wordpress-db
Root password	type a password

Region	us-central1
Zone	Any
Database Version	MySQL 5.7

Note the root password; it will be used in a later step and referred to as [ROOT_PASSWORD].

5. Expand **Show configuration options**.
6. Expand the **Connectivity** section.
7. Select **Private IP**.
8. In the dialog box, click **Enable API**, click **Allocate and connect**, and then click **Close**. This enables [Private Services Access](#) and attaches a Private IP address to your Cloud SQL server.

Private IP is an internal connection, unlike external IP, which egresses to the internet.

9. Expand the **Machine type and storage** section.
10. Provision the right amount of vCPU and memory. To choose a **Machine Type**, click **Change**, and then explore your options.

A few points to consider:

- ☐ Shared-core machines are good for prototyping, and are not covered by [Cloud SLA](#).
- ☐ Each vCPU is subject to a 250 MB/s network throughput cap for peak performance. Each additional core increases the network cap, up to a theoretical maximum of 2000 MB/s.
- ☐ For performance-sensitive workloads such as online transaction processing (OLTP), a general guideline is to ensure that your instance has enough memory to contain the entire working set and accommodate the number of active connections.

11. For this lab, select **db-n1-standard-1**, and then click **Select**.
12. Next, choose **Storage type** and **Storage capacity**.

A few points to consider:

☐ SSD (solid-state drive) is the best choice for most use cases. HDD (hard-disk drive) offers lower performance, but [storage costs](#) are significantly reduced, so HDD may be preferable for storing data that is infrequently accessed and does not require very low latency.

☐ There is a direct relationship between the storage capacity and its throughput.

13. Add a few zeros to the storage capacity to see how it affects the throughput. Reset the slider to 10GB.

Setting your storage capacity too low without enabling an automatic storage increase can cause your instance to lose its SLA.

14. Click **Close**.

15. Click **Create** at the bottom of the page to create the database instance.

You might have to wait for the Private IP changes to propagate before the **Create** button becomes clickable.

Click *Check my progress* to verify the objective.

Create a Cloud SQL instance


Check my progress

Task 2: Configure a proxy on a virtual machine

When your application does not reside in the same VPC connected network and region as your Cloud SQL instance, use a proxy to secure its external connection.


In order to configure the proxy, you need the Cloud SQL instance connection name.

The lab comes with 2 virtual machines preconfigured with Wordpress and its dependencies. You can view the startup script and service account access by clicking on a virtual machine name. Notice that we used the principle of least privilege and only allow SQL access for that VM. There's also a network tag and a firewall preconfigured to allow port 80 from any host.

1. On the **Navigation menu** () click **Compute Engine**.
2. Click **SSH** next to **wordpress-europe-proxy**.
3. Download the Cloud SQL Proxy and make it executable:

```
wget https://dl.google.com/cloudsql/cloud_sql_proxy.linux.amd64 -O cloud_sql_proxy &&  
chmod +x cloud_sql_proxy
```

In order to start the proxy, you need the connection name of the Cloud SQL instance. Keep your SSH window open and return to the Cloud Console.

4. On the **Navigation menu** () , click **SQL**.
5. Click on the **wordpress-db** instance and wait for a green checkmark next to its name, which indicates that it is operational (this could take a couple of minutes).
6. Note the **Instance connection name**; it will be used later and referred to as [SQL_CONNECTION_NAME].
7. In addition, for the application to work, you need to create a table. Click **Databases**.
8. Click **Create database**, type **wordpress**, which is the name the application expects, and then click **Create**.
9. Return to the SSH window and save the connection name in an environment variable, replacing [SQL_CONNECTION_NAME] with the unique name you copied in a previous step.

```
export SQL_CONNECTION=[SQL_CONNECTION_NAME]
```

10. To verify that the environment variable is set, run:

```
echo $SQL_CONNECTION
```

The connection name should be printed out.

11. To activate the proxy connection to your Cloud SQL database and send the process to the background, run the following command:

```
./cloud_sql_proxy -instances=$SQL_CONNECTION=tcp:3306 &
```

The expected output is

```
Listening on 127.0.0.1:3306 for [SQL_CONNECTION_NAME]  
Ready for new connections
```

12. Press ENTER.

The proxy will listen on 127.0.0.1:3306 (localhost) and proxy that connects securely to your Cloud SQL over a secure tunnel using the machine's external IP address.

Click *Check my progress* to verify the objective.

Create a database and configure a proxy on a Virtual Machine

Check my progress

Task 3: Connect an application to the Cloud SQL instance

In this task, you will connect a sample application to the Cloud SQL instance.

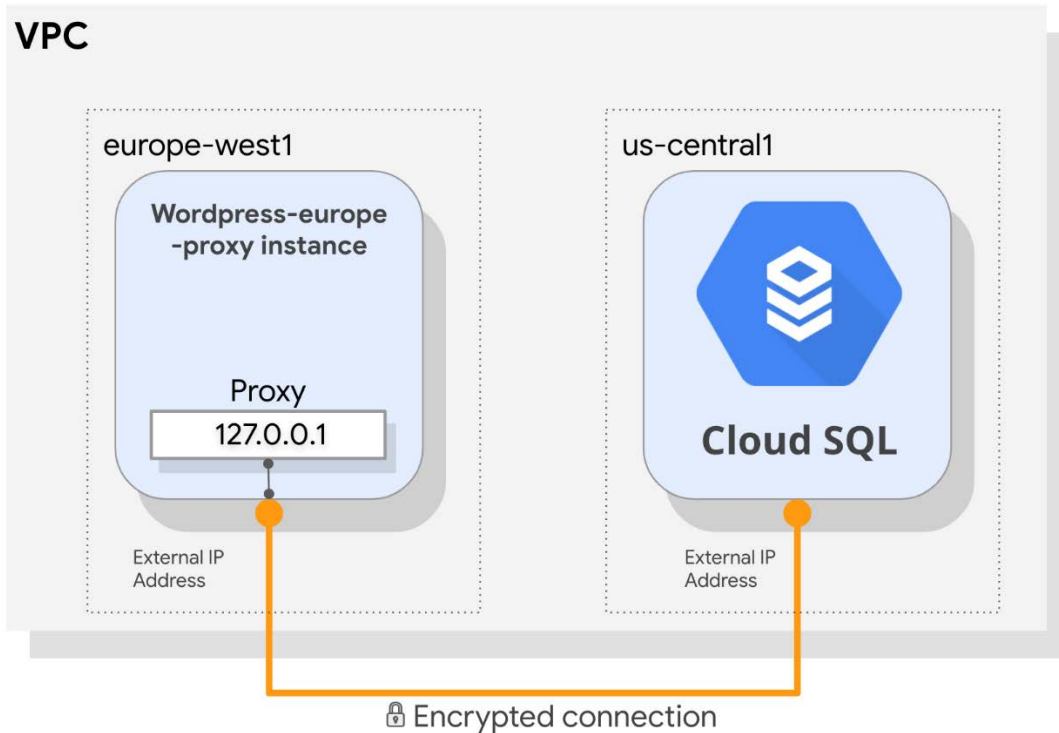
1. Configure the Wordpress application. To find the external IP address of your virtual machine, query its metadata:

```
curl -H "Metadata-Flavor: Google"
http://169.254.169.254/computeMetadata/v1/instance/network-interfaces/0/access-
configs/0/external-ip && echo
```

2. Go to the **wordpress-europe-proxy** external IP address in your browser and configure the Wordpress application.
3. Click **Let's Go**.
4. Specify the following, replacing [ROOT_PASSWORD] with the password you configured upon machine creation, and leave the remaining settings as their defaults:

Property	Value
Username	root
Password	[ROOT_PASSWORD]
Database Host	127.0.0.1

You are using 127.0.0.1, localhost as the Database IP because the proxy you initiated listens on this address and redirects that traffic to your SQL server securely.



5. Click **Submit**.
6. When a connection has been made, click **Run the installation** to instantiate Wordpress and its database in your Cloud SQL. This might take a few moments to complete.
7. Populate your demo site's information with random information and click **Install Wordpress**. You won't have to remember or use these details.

Installing Wordpress might take up to 3 minutes, because it propagates all its data to your SQL Server.

8. When a 'Success!' window appears, remove the text after the IP address in your web browser's address bar and press ENTER. You'll be presented with a working Wordpress Blog!


Hello world!

Welcome to WordPress. This is your first post. Edit or delete it, then start writing!

Task 4: Connect to Cloud SQL via internal IP

If you can host your application in the same region and VPC connected network as your Cloud SQL, you can leverage a more secure and performant configuration using Private IP.

By using Private IP, you will increase performance by reducing latency and minimize the attack surface of your Cloud SQL instance because you can communicate with it exclusively over internal IPs.

1. In the Cloud Console, on the **Navigation menu** () , click **SQL**.
2. Click **wordpress-db**.
3. Note the Private IP address of the Cloud SQL server; it will be referred to as [SQL_PRIVATE_IP].
4. On the **Navigation menu**, click **Compute Engine**.

Notice that **wordpress-us-private-ip** is located at us-central1, where your Cloud SQL is located, which enables you to leverage a more secure connection.

5. Copy the external IP address of **wordpress-us-private-ip**, paste it in a browser window, and press ENTER.
6. Click **Let's Go**.
7. Specify the following, and leave the remaining settings as their defaults:

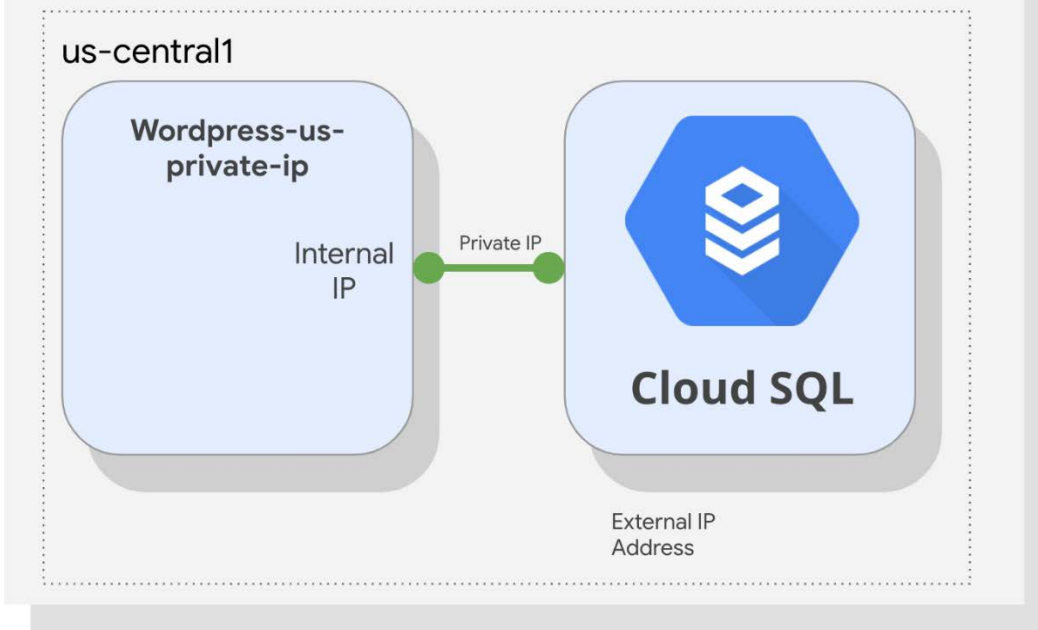
Property	Value
Username	root
Password	type the [ROOT_PASSWORD] configured when the Cloud SQL instance was created
Database Host	[SQL_PRIVATE_IP]

8. Click **Submit**.

Notice that this time you are creating a direct connection to a Private IP, instead of configuring a proxy. That connection is private, which means that it doesn't egress to the internet and therefore benefits from better performance and security.

9. Click **Run the installation**. An 'Already Installed!' window is displayed, which means that your application is connected to the Cloud SQL server over private IP.
10. In your web browser's address bar, remove the text after the IP address and press ENTER. You'll be presented with a working Wordpress Blog!

VPC

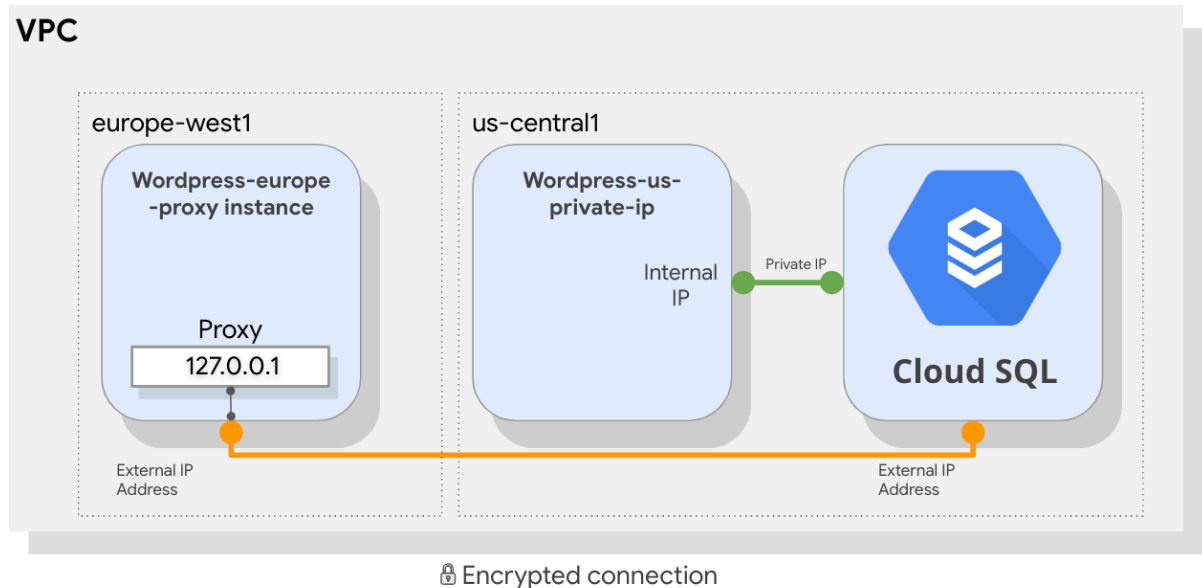


Task 5: Review

In this lab, you created a Cloud SQL database and configured it to use both an external connection over a secure proxy and a Private IP address, which is more secure and performant. Remember that you can only connect via Private IP if the application and the Cloud SQL server are collocated in the same region and are part of the same VPC network. If your application is hosted in another region, VPC, or even project, use a proxy to secure its connection over the external

connection.

Project



BIQQQUERY

Task 1: Use BigQuery to import data

Sign in to BigQuery and create a dataset

1. In the Cloud Console, on the **Navigation menu**, click **BigQuery**.
2. If prompted, click **Done**.
3. click on to your **Project ID** (starts with qwiklabs-gcp) and click **Create Dataset**.

You can export billing data directly to BigQuery as outlined [here](#). However, for the purposes of this lab, a sample CSV billing file has been prepared for you. It is located in a Cloud Storage bucket where it is accessible to your student account. You will import this billing information into a BigQuery table and examine it.

4. Specify the following:

Property	Value (type value or select option as specified)
Dataset ID:	imported_billing_data

Data location:	US
Default table expiration > Number of days after table creation:	In 1 days

5. Click **Create Dataset**. You should see **imported_billing_data** in the left pane.

Create a table and import

1. Point to **imported_billing_data**, and then click **Create Table** to create a new table.
2. For **Source**, specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Create table from:	Google Cloud Storage
Select file from GCS bucket	gs://cloud-training/archinfra/export-billing-example.csv
File format	CSV

3. For **Destination**, specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Table name	sampleinfotable
Table type	Native table

4. Under **Schema** for **Auto detect** click **Schema and input parameters**.
5. Open **Advanced options**

6. Under **Header rows to skip** specify 1
7. Click **Create Table**. After the job is completed, the table appears below the dataset in the left pane.

Task 2: Examine the table

1. Click **sampleinfotable**.

This displays the schema that BigQuery automatically created based on the data it found in the imported CSV file. Notice that there are strings, integers, timestamps, and floating values.

2. Click **Details**. As you can see in **Number of Rows**, this is a relatively small table with 44 rows.
3. Click **Preview**.

Task 3: Compose a simple query

When you reference a table in a query, both the dataset ID and table ID must be specified; the project ID is optional.

If the project ID is not specified, BigQuery will default to the current project.

All the information you need is available in the BigQuery interface. In the column on the left, you see the dataset ID (imported_billing_data) and table ID (sampleinfotable).

Recall that clicking on the table name brings up the **Schema** with all of the field names.

Now construct a simple query based on the **Cost** field.

1. click **Compose New Query**.
2. Paste the following in Query Editor:

```
SELECT * FROM `imported_billing_data.sampleinfotable`  
WHERE Cost > 0
```

3. Click **Run**.

Task 4: Analyze a large billing dataset with SQL

In the next activity, you use BigQuery to analyze a sample dataset with 22,537 lines of billing data.

The **cloud-training-prod-bucket.arch_infra.billing_data** dataset used in this task is shared with the public. For more information on public datasets and how to share datasets with the public, refer to the [documentation](#).

1. For New Query, paste the following in Query Editor:

```
SELECT

product,

resource_type,

start_time,

end_time,

cost,

project_id,

project_name,

project_labels_key,

currency,

currency_conversion_rate,

usage_amount,

usage_unit

FROM

`cloud-training-prod-bucket.arch_infra.billing_data`
```

2. Click **Run**. Verify that the resulting table has 22,537 lines of billing data.
3. To find the latest 100 records where there were charges ($\text{cost} > 0$), for New Query, paste the following in Query Editor:

```
SELECT

product,

resource_type,

start_time,

end_time,
```

```

cost,

project_id,

project_name,

project_labels_key,

currency,

currency_conversion_rate,

usage_amount,

usage_unit

FROM

`cloud-training-prod-bucket.arch_infra.billing_data`

WHERE

Cost > 0

ORDER BY end_time DESC

LIMIT

100

```

4. Click **Run**.

5. To find all charges that were more than 3 dollars, for Compose New Query, paste the following in Query Editor:

```

SELECT

product,

resource_type,

start_time,

end_time,

cost,

project_id,

project_name,

```

```
project_labels_key,  
  
currency,  
  
currency_conversion_rate,  
  
usage_amount,  
  
usage_unit  
  
FROM  
  
`cloud-training-prod-bucket.arch_infra.billing_data`  
  
WHERE  
  
cost > 3
```

6. Click **Run**.
7. To find the product with the most records in the billing data, for New Query, paste the following in Query Editor:

```
SELECT  
  
product,  
  
COUNT(*) AS billing_records  
  
FROM  
  
`cloud-training-prod-bucket.arch_infra.billing_data`  
  
GROUP BY  
  
product  
  
ORDER BY billing_records DESC
```

8. Click **Run**.

MONITORING AND METRICS

Task 2: Custom dashboards

Create a dashboard

1. In the left pane, click **Dashboards > Create Dashboard**.
2. For **New Dashboard Name**, type **My Dashboard**, and press **Confirm**.

Add a chart

1. Click **Add Chart**.
2. For **Title**, give your chart a name (you can revise this before you save based on the selections you make).
3. For **Find resource type and metric**, select **VM Instance**.
4. For **Metrics**, select a metric to chart for the Instance resource, such as **CPU utilization** or **Network traffic**.

Note: If you are getting a 'loading failed' error message, you might have to refresh the page.

5. Click **Filter** and explore the various options.
6. Click **View Options** and explore adding a Threshold or changing the **Chart mode**.
7. Click **Save** to add the chart to your dashboard.

Metrics Explorer

The **Metrics Explorer** allows you to examine resources and metrics without having to create a chart on a dashboard. Try to recreate the chart you just created using the **Metrics Explorer**.

1. In the left pane, click **Metrics Explorer**.
2. For **Find resource type and metric**, type a metric or resource name.
3. Explore the various options and try to recreate the chart you created earlier.

Not all metrics are currently available on the Metrics Explorer, so you might not be able to find the exact metric you used on the previous step.

Task 3: Alerting policies

Create an alert and add the first condition

1. In the left pane, click **Alerting > Create Policy**.
2. Click **Add Condition**.
3. For **Find resource type and metric**, select **VM Instance**.

If you cannot locate the **VM Instance** resource type, you might have to refresh the page.

4. Select a metric you are interested in evaluating, such as **CPU usage** or **CPU Utilization**.
5. For **Condition**, select **is above**.
6. 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**.
7. Click **ADD**.

Add a second condition

1. Click **Add Condition**.
2. Repeat the steps above to specify the second condition for this policy. For example, repeat the condition for a different instance. Click **ADD**.
3. In **Policy Triggers**, for **Trigger when**, click **All conditions are met**.
4. Click **Next**.

Configure notifications and finish the alerting policy

1. Click on dropdown arrow next to **Notification Channels**, then click on **Manage Notification Channels**.

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

2. Scroll down the page and click on **ADD NEW** for **Email**.
3. Enter your personal email in the **Email Address** field and a **Display name**.
4. Click **Save**.

5. Go back to the previous **Create alerting policy** tab.
6. Click on **Notification Channels** again, then click on the **Refresh icon** to get the display name you mentioned in the previous step.
7. Now, select your **Display name** and click **OK**.
8. Click **Next**.
9. Enter a name of your choice in **Alert name** field.
10. Click **Save**.

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

Create alerting policies

Check my progress

Task 4: Resource groups

1. In the left pane, click **Groups > Create Group**.
2. Enter a name for the group. For example: **VM instances**
3. In the **Criteria** section, type **nginx** in the value field below **Contains**.
4. Click **DONE**.
5. Click **CREATE**.
6. Review the dashboard Cloud Monitoring created for your group.

Task 5: Uptime monitoring

1. In the Monitoring tab, click on **Uptime Checks**.
2. Click **Create Uptime Check**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
----------	--------------------------------------------------

Title	<i>Enter a title then click Next</i>
Protocol	HTTP
Resource Type	Instance
Applies To	Group
Group	<i>Select your group</i>
Check Frequency	1 minute

4. Click on **Next** to leave the other details to default and click **Test** to verify that your uptime check can connect to the resource.
5. When you see a green check mark everything can connect. Click **Create**.

The uptime check you configured takes a while for it to become active.

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

Create uptime monitoring

Check my progress

Task 6: Review

In this lab, you learned how to:

- Monitor your projects
- Create a Cloud Monitoring workspace
- Create alerts with multiple conditions
- Add charts to dashboards
- Create resource groups

- Create uptime checks for your services

DEBUGGING

Task 1: Create an application

Get and test the application

1. In the Cloud Console, launch Cloud Shell by clicking **Activate Cloud Shell**. If prompted, click **Continue**.
2. To create a local folder and get the App Engine **Hello world** application, run the following commands:

```
mkdir appengine-hello  
cd appengine-hello  
gsutil cp gs://cloud-training/archinfra/gae-hello/* .
```

3. To run the application using the local development server in Cloud Shell, run the following command:

```
dev_appserver.py $(pwd)
```

4. In Cloud Shell, click **Web Preview** > **Preview on port 8080** to view the application. You may have to collapse the **Navigation menu** pane to access the **Web Preview** icon.

A new browser window opens to the localhost and displays the message **Hello, World!**

5. In Cloud Shell, press **Ctrl+C** to exit the development server.

Deploy the application to App Engine

1. To deploy the application to App Engine, run the following command:

```
gcloud app deploy app.yaml
```

2. If prompted for a region, enter the number corresponding to a region.
3. When prompted, type **Y** to continue.
4. When the process is done, verify that the application is working by running the following command:

```
gcloud app browse
```

If Cloud Shell does not detect your browser, click the link in the Cloud Shell output to view your app. You might have to refresh the page for the application to load.

5. If needed, press **Ctrl+C** to exit the development mode.

Click *Check my progress* to verify the objective.

Create an application

Check my progress

Introduce an error to break the application

1. To examine the main.py file, run the following command:

```
cat main.py
```

Notice that the application imports webapp2.

You will break the configuration by replacing the import library with one that doesn't exist.

2. To use the sed stream editor to change the import library to the nonexistent webapp22, run the following command:

```
sed -i -e 's/webapp2/webapp22/' main.py
```

3. To verify the change you made in the main.py file, run the following command:

```
cat main.py
```

Notice that the application now tries to import webapp22.

Re-deploy the application to App Engine

1. To re-deploy the application to App Engine, run the following command:

```
gcloud app deploy app.yaml --quiet
```

The `--quiet` flag disables all interactive prompts when running `gcloud` commands. If input is required, defaults will be used. In this case, it avoids the need for you to type `Y` when prompted to continue the deployment.

2. When the process is done, verify that the application is broken by running the following command:

```
gcloud app browse
```

If Cloud Shell does not detect your browser, click the link in the Cloud Shell output to view your app.

3. If needed, press **Ctrl+C** to exit development mode.
4. Leave Cloud Shell open.

Task 2: Explore Cloud Error Reporting

View Error Reporting and trigger additional errors

1. In the Cloud Console, on the **Navigation menu**, click **Error Reporting**.

You should see an error regarding the failed import of webapp22.

2. Click **Auto reload**.
3. In Cloud Shell, run the following command:

```
gcloud app browse
```

If Cloud Shell does not detect your browser, click the link in the Cloud Shell output to view your app.

4. Click the resulting link several times to generate more errors.

The number of errors is displayed in the **Occurrences** column. The graph shows the frequency of errors over time, and the number represents the sum of errors. This is a very handy visual indicator of the state of the error.

The **First seen** and **Last seen** columns show when the error was first seen and when it was last seen, respectively. This can help identify changes that might have triggered the error. In this case, it was the upload of the new version of app engine code.

Which service(s) are currently supported by Cloud Error Reporting?

☐

Kubernetes

☐

App Engine Flexible

☐

App Engine Standard

☐

Compute Engine

Submit

View details and identify the cause

1. Click the Error name: **ImportError: No module named webapp22**.

Now you can see a detailed graph of the errors. The **Response Code** field shows the explicit error: a **500 Internal Server Error**.

2. For **Stack trace sample**, click **Parsed**. This opens the Cloud Debugger, showing the error in the code!

View the logs and fix the error

1. At the bottom of the Debug page, just below the code, find and open **View logs** in a new window or tab. Here you can find more detailed historical information about the error.
2. Introduce more errors by refreshing the page of your application. If you closed your application, use gcloud app browse and click the link to view the broken app.

3. On the **Cloud Error Reporting** page, ensure that **Auto Reload** is enabled to watch the addition of new errors.
4. In Cloud Shell, fix the error by running the following command:

```
sed -i -e 's/webapp22/webapp2/' main.py
```

5. To re-deploy the application to App Engine, run the following command:

```
gcloud app deploy app.yaml --quiet
```

6. When the process is done, to verify that the application is working again, run the following command:

```
gcloud app browse
```

If Cloud Shell does not detect your browser, click the link in the Cloud Shell output to view your app.

7. On the **Cloud Error Reporting** page, ensure that **Auto Reload** is enabled to and see that no new errors are added.

Task 3: Review

In this lab you deployed an application to App Engine. Then you introduced a code bug and broke the application. You used Cloud Error Reporting to identify the issue, and then analyzed the problem, finding the root cause using Cloud Debugger. You modified the code to fix the problem, and then saw the results in Cloud Operations.

VPN

HTTPS LOAD BALANCING

Task 1. Configure a health check firewall rule

Health checks determine which instances of a load balancer can receive new connections. For HTTP load balancing, the health check probes to your load-balanced instances come

from addresses in the ranges **130.211.0.0/22** and **35.191.0.0/16**. Your firewall rules must allow these connections.

Create the health check rule

Create a firewall rule to allow health checks.

1. In the Cloud Console, on the **Navigation menu** () , click **VPC network > Firewall**. Notice the existing **ICMP**, **internal**, **RDP**, and **SSH** firewall rules.

Each Google Cloud project starts with the **default** network and these firewall rules.

2. Click **Create Firewall Rule**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	fw-allow-health-checks
Network	default
Targets	Specified target tags
Target tags	allow-health-checks
Source filter	IP Ranges
Source IP ranges	130.211.0.0/22 and 35.191.0.0/16
Protocols and ports	Specified protocols and ports

Make sure to include the **/22** and **/16** in the **Source IP ranges**.

4. Select **tcp** and specify port **80**.
5. Click **Create**.

Click *Check my progress* to verify the objective.

Configure health check firewall rule


Check my progress

Task 2: Create a NAT configuration using Cloud Router

The Google Cloud VM backend instances that you setup in Task 3 will not be configured with external IP addresses.

Instead, you will setup the Cloud NAT service to allow these VM instances to send outbound traffic only through the Cloud NAT, and receive inbound traffic through the load balancer.

Create the Cloud Router instance

1. In the Cloud Console, on the **Navigation menu** () , click **Network services > Cloud NAT**.
2. Click **Get started**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Gateway name	nat-config
Region	us-central1

4. Click **Cloud Router**, and select **Create new router**.
5. For **Name**, type **nat-router-us-central1**.
6. Click **Create**.
7. In Create a NAT gateway, click **Create**.

Wait until the NAT Gateway Status changes to Running before moving onto the next task.

Click *Check my progress* to verify the objective.


Create a NAT configuration using Cloud Router

Check my progress

Task 3: Create a custom image for a web server

Create a custom web server image for the backend of the load balancer.

Create a VM

1. In the Cloud Console, on the **Navigation menu** () , click **Compute Engine > VM instances**.
2. Click **Create**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	webserver
Region	us-central1
Zone	us-central1-a
Series	N1
Machine type	f1-micro (1 vCPU)

4. Click **Management, security, disks, networking, sole tenancy**.
5. Click **Disks**, and clear **Delete boot disk when instance is deleted**.

6. Click **Networking**.
 - For **Network tags**, type **allow-health-checks**.
 - Under **Network interfaces** , click **default**.
 - Under **External IP** dropdown, select **None**.
7. Click **Done**.
8. Click **Create**.

Customize the VM

1. For **webserver**, click **SSH** to launch a terminal and connect.
2. If you see the **Connection via Cloud Identity-Aware Proxy Failed** popup, click **Retry**.
3. To install Apache2, run the following commands:

```
sudo apt-get update  
sudo apt-get install -y apache2
```

4. To start the Apache server, run the following command:

```
sudo service apache2 start
```

5. To test the default page for the Apache2 server, run the following command:

```
curl localhost
```

The default page for the Apache2 server should be displayed.

Set the Apache service to start at boot

The software installation was successful. However, when a new VM is created using this image, the freshly booted VM does not have the Apache web server running. Use the

following command to set the Apache service to automatically start on boot. Then test it to make sure it works.

1. In the webserver SSH terminal, set the service to start on boot:

```
sudo update-rc.d apache2 enable
```

2. In the Cloud Console, select **webserver**, and then click **Reset**.
3. In the confirmation dialog, click **Reset**.

Reset will stop and reboot the machine. It keeps the same IPs and the same persistent boot disk, but memory is wiped. Therefore, if the Apache service is available after the reset, the **update-rc** command was successful.

4. Check the server by connecting via SSH to the VM and entering the following command:

```
sudo service apache2 status
```

NOTE: If you see the **Connection via Cloud Identity-Aware Proxy Failed** popup, click **Retry**.

5. The result should show **Started The Apache HTTP Server**.

Prepare the disk to create a custom image

Verify that the boot disk will not be deleted when the instance is deleted.

1. On the VM instances page, click **webserver** to view the VM instance details.
2. Under **Boot disk**, verify that **When deleting instance** is set to **Keep disk**.
3. Return to the VM instances page, click **webserver**, and click **Delete**.
4. In the confirmation dialog, click **Delete**.
5. In the left pane, click **Disks** and verify that the **webserver** disk exists.

Create the custom image

1. In the left pane, click **Images**.

2. Click **Create image**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	mywebserver
Source	Disk
Source disk	webserver

4. Click **Create**.

You have created a custom image that multiple identical web servers can be started from. At this point, you could delete the **webserver** disk.

The next step is to use that image to define an instance template that can be used in the managed instance groups.

Click *Check my progress* to verify the objective.

Create a custom image for a web server


Check my progress

Task 4. Configure an instance template and create instance groups

A managed instance group uses an instance template to create a group of identical instances. Use these to create the backends of the HTTP load balancer.

Configure the instance template

An instance template is an API resource that you can use to create VM instances and managed instance groups. Instance templates define the machine type, boot disk image, subnet, labels, and other instance properties.

1. In the Cloud Console, on the **Navigation menu** () , click **Compute Engine > Instance templates**.

2. Click **Create instance template**.
3. For **Name**, type **mywebserver-template**.
4. For **Series**, select **N1**.
5. For **Machine type**, select **f1-micro (1 vCPU)**.
6. For **Boot disk**, click **Change**.
7. Click **Custom images**.
8. For **Image**, Select **mywebserver**.
9. Click **Select**.
10. Click **Management, security, disks, networking, sole tenancy**.
11. Click **Networking**.
 - For **Network tags**, type **allow-health-checks**.
 - Under **External IP** dropdown, select **None**.
12. Click **Create**.

Create the managed instance groups

Create a managed instance group in **us-central1** and one in **europa-west1**.

1. On the **Navigation menu**, click **Compute Engine > Instance groups**.
2. Click **Create Instance group**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	us-central1-mig

Location	Multiple zones
Region	us-central1
Instance template	mywebserver-template

4. Under **Autoscaling metrics**, click on the edit pencil icon.
5. Under **Metric type**, select **HTTP load balancing utilization**.
6. Enter Target HTTP load balancing utilization to 80.
7. Click **Done**.
8. Set Cool down period to 60 seconds.
9. Enter Minimum number of instances 1 and Maximum number of instances 2.

Managed instance groups offer **autoscaling** capabilities that allow you to automatically add or remove instances from a managed instance group based on increases or decreases in load. Autoscaling helps your applications gracefully handle increases in traffic and reduces cost when the need for resources is lower. You just define the autoscaling policy, and the autoscaler performs automatic scaling based on the measured load.

10. For **Health check**, select **Create a health check**.
11. Specify the following, and leave the remaining settings as their defaults:

Property	Value (select option as specified)
Name	http-health-check
Protocol	TCP

Port	80
------	----

Managed instance group health checks proactively signal to delete and recreate instances that become unhealthy.

12. Click **Save and continue**.

13. For **Initial delay**, type 60. This is how long the Instance Group waits after initializing the boot-up of a VM before it tries a health check. You don't want to wait 5 minutes for this during the lab, so you set it to 1 minute.

14. Click **Create**.

15. Click **OK**.

NOTE: If a warning window will appear stating that **There is no backend service attached to the instance group**. Ignore this; you will configure the load balancer with a backend service in the next section of the lab.

Repeat the same procedure for **europe-west1-mig** in **europe-west1**:

16. Click **Create Instance group**.

17. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	europe-west1-mig
Location	Multiple zones
Region	europe-west1
Instance template	mywebserver-template

Autoscaling metrics > Metric Type	HTTP load balancing utilization
Target HTTP load balancing utilization	80
Minimum number of instances	1
Maximum number of instances	2
Cool down period	60

18. For **Health check**, select **http-health-check (TCP)**.

19. For **Initial delay**, type 60.

20. Click **Create**.

21. Click **OK** in the dialog window.

Click *Check my progress* to verify the objective.

Configure an instance template and create instance groups

Check my progress

It doesn't look like you've completed this step yet. Try again.

Verify the backends

Verify that VM instances are being created in both regions.

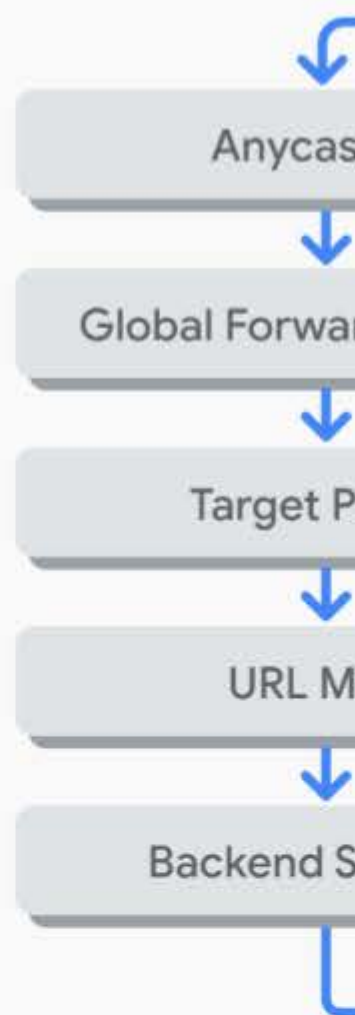
1. On the **Navigation menu**, click **Compute Engine > VM instances**. Notice the instances that start with *us-central1-mig* and *europa-west1-mig*. These instances are part of the managed instance groups.

Task 5. Configure the HTTP load balancer

Configure the HTTP load balancer to balance traffic between the two backends (**us-central1-mig** in us-central1 and **europa-west1-mig** in europa-west1) as illustrated in the network diagram:



Cloud Load
Balancing



Network: Default



Virtual Private
Cloud

Region: us-ce

SubNetwork: us

Backend/Instan
us-central1



C
En

Start the configuration

1. On the **Navigation menu**, click **Network Services > Load balancing**.
2. Click **Create load balancer**.
3. Under **HTTP(S) Load Balancing**, click **Start configuration**.
4. Select **From Internet to my VMs**, then click **Continue**.
5. For **Name**, type **http-lb**.

Configure the backend

Backend services direct incoming traffic to one or more attached backends. Each backend is composed of an instance group and additional serving capacity metadata.

1. Click **Backend configuration**.
2. For **Backend services & backend buckets**, click **Create or select backend services & backend buckets > Backend services > Create a backend service**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (select option as specified)
Name	http-backend
Backend type	Instance group
Instance group	us-central1-mig
Port numbers	80
Balancing mode	Rate

Maximum RPS	50
Capacity	100

This configuration means that the load balancer attempts to keep each instance of **us-central1-mig** at or below 50 requests per second (RPS).

- Click **Done**.
- Click **Add backend**.
- Specify the following, and leave the remaining settings as their defaults:

Property	Value (select option as specified)
Instance group	europe-west1-mig
Port numbers	80
Balancing mode	Utilization
Maximum backend utilization	80
Capacity	100

This configuration means that the load balancer attempts to keep each instance of **europe-west1-mig** at or below 80% CPU utilization.

- Click **Done**.
- For **Health Check**, select **http-health-check (TCP)**.
- Expand **Advanced configurations (Session affinity, connection draining timeout, security policies)** option and check the **Enable logging** checkbox.

10. Specify **Sample rate** as 1.

11. Click **Create**.

Configure the frontend

The host and path rules determine how your traffic will be directed. For example, you could direct video traffic to one backend and direct static traffic to another backend. However, you are not configuring the host and path rules in this lab.

1. Click **Frontend configuration**.
2. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Protocol	HTTP
IP version	IPv4
IP address	Ephemeral
Port	80

3. Click **Done**.
4. Click **Add Frontend IP and port**.
5. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Protocol	HTTP

IP version	IPv6
IP address	Ephemeral
Port	80

6. Click **Done**.

HTTP(S) load balancing supports both IPv4 and IPv6 addresses for client traffic. Client IPv6 requests are terminated at the global load balancing layer and then proxied over IPv4 to your backends.

Review and create the HTTP load balancer

1. Click **Review and finalize**.
2. Review the **Backend services** and **Frontend**.
3. Click **Create**. Wait for the load balancer to be created.
4. Click on the name of the load balancer (**http-lb**).
5. Note the IPv4 and IPv6 addresses of the load balancer for the next task. They will be referred to as [LB_IP_v4] and [LB_IP_v6], respectively.

The IPv6 address is the one in hexadecimal format.

Click *Check my progress* to verify the objective.

Configure the HTTP load balancer

Check my progress

Task 6. Stress test the HTTP load balancer

Now that you have created the HTTP load balancer for your backends, it is time to verify that traffic is forwarded to the backend service.

The HTTP load balancer should forward traffic to the region that is closest to you.

checkTrue



False


Access the HTTP load balancer

1. Open a new tab in your browser and navigate to `http://[LB_IP_v4]`. Make sure to replace `[LB_IP_v4]` with the IPv4 address of the load balancer.

Accessing the HTTP load balancer might take a couple of minutes. In the meantime, you might get a 404 or 502 error. Keep trying until you see the page of one of the backends.

Stress test the HTTP load balancer

Create a new VM to simulate a load on the HTTP load balancer. Then determine whether traffic is balanced across both backends when the load is high.

1. In the Cloud Console, on the **Navigation menu** () , click **Compute Engine > VM instances**.
2. Click **Create instance**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	stress-test
Region	us-west1
Zone	us-west1-c
Series	N1
Machine type	f1-micro (1 vCPU)

Because **us-west1** is closer to **us-central1** than to **eu-west-1**, traffic should be forwarded only to **us-central1-mig** (unless the load is too high).

4. For **Boot Disk**, click **Change**.
5. Click **Custom images**.
6. For **Image**, select **mywebserver**.
7. Click **Select**.
8. Click **Create**. Wait for the **stress-test** instance to be created.
9. For **stress-test**, click **SSH** to launch a terminal and connect.
10. To create an environment variable for your load balancer IP address, run the following command:

```
export LB_IP=<Enter your [LB_IP_v4] here>
```

11. Verify it with echo:

```
echo $LB_IP
```

12. To place a load on the load balancer, run the following command:


```
ab -n 500000 -c 1000 http://$LB_IP/
```

Click *Check my progress* to verify the objective.

Stress test the HTTP load balancer

Check my progress


It doesn't look like you've completed this step yet. Try again.

13. In the Cloud Console, on the **Navigation menu** () , click **Network Services > Load balancing**.
14. Click **Backends**.

15. Click **http-backend**.

16. Monitor the **Frontend Location (Total inbound traffic)** between North America and the two backends for a couple of minutes.

At first, traffic should just be directed to **us-central1-mig**, but as the RPS increases, traffic is also directed to **europa-west1-mig**. This demonstrates that by default traffic is forwarded to the closest backend, but if the load is very high, traffic can be distributed across the backends.

17. In the Cloud Console, on the **Navigation menu** () , click **Compute Engine > Instance groups**.

18. Click on **us-central1-mig** to open the instance group page.

19. Click **Monitoring** to monitor the number of instances and LB capacity.

20. Repeat the same for the **europa-west1-mig** instance group.

Depending on the load, you might see the backends scale to accommodate the load.

Task 7. Review

In this lab, you configured an HTTP load balancer with backends in us-central1 and europa-west1. Then you stress-tested the load balancer with a VM to demonstrate global load balancing and autoscaling.

CONFIGURING AN INTERNAL LOAD BALANCER

Task 1. Configure internal traffic and health check firewall rules.

Configure firewall rules to allow internal traffic connectivity from sources in the 10.10.0.0/16 range. This rule allows incoming traffic from any client located in the subnet.

Health checks determine which instances of a load balancer can receive new connections. For HTTP load balancing, the health check probes to your load-balanced instances come from addresses in the ranges **130.211.0.0/22** and **35.191.0.0/16**. Your firewall rules must allow these connections.

Explore the my-internal-app network

The network **my-internal-app** with **subnet-a** and **subnet-b** and firewall rules for **RDP**, **SSH**, and **ICMP** traffic have been configured for you.

- In the Cloud Console, on the **Navigation menu** (≡), click **VPC network > VPC networks**. Notice the **my-internal-app** network with its subnets: **subnet-a** and **subnet-b**.

Each Google Cloud project starts with the **default** network. In addition, the **my-internal-app** network has been created for you as part of your network diagram.

You will create the managed instance groups in **subnet-a** and **subnet-b**. Both subnets are in the **us-central1** region because an internal load balancer is a regional service. The managed instance groups will be in different zones, making your service immune to zonal failures.

Create the firewall rule to allow traffic from any sources in the 10.10.0.0/16 range

Create a firewall rule to allow traffic in the 10.10.0.0/16 subnet.

1. On the **Navigation menu** (≡), click **VPC network > Firewall**. Notice the **app-allow-icmp** and **app-allow-ssh-rdp** firewall rules.

These firewall rules have been created for you.

2. Click **Create Firewall Rule**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	fw-allow-lb-access
Network	my-internal-app
Targets	Specified target tags
Target tags	backend-service
Source filter	IP Ranges


Source IP ranges	10.10.0.0/16
Protocols and ports	Allow all

Make sure to include the **/16** in the **Source IP ranges**.

4. Click **Create**.

Create the health check rule

Create a firewall rule to allow health checks.

1. On the **Navigation menu** () , click **VPC network > Firewall**.
2. Click **Create Firewall Rule**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	fw-allow-health-checks
Network	my-internal-app
Targets	Specified target tags
Target tags	backend-service
Source filter	IP Ranges
Source IP ranges	130.211.0.0/22 35.191.0.0/16

Protocols and ports	Specified protocols and ports
---------------------	-------------------------------

Make sure to include the **/22** and **/16** in the **Source IP ranges**.

4. For **tcp**, specify port **80**.
5. Click **Create**.

Click *Check my progress* to verify the objective.

Configure internal traffic and health check firewall rules


Check my progress

Task 2: Create a NAT configuration using Cloud Router

The Google Cloud VM backend instances that you setup in Task 3 will not be configured with external IP addresses.

Instead, you will setup the Cloud NAT service to allow these VM instances to send outbound traffic only through the Cloud NAT, and receive inbound traffic through the load balancer.

Create the Cloud Router instance

1. In the Cloud Console, on the **Navigation menu** () , click **Network services > Cloud NAT**.
2. Click **Get started**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Gateway name	nat-config
VPC network	my-internal-app

Region	us-central1
--------	-------------

4. Click **Cloud Router**, and select **Create new router**.
5. For **Name**, type **nat-router-us-central1**.
6. Click **Create**.
7. In Create a NAT gateway, click **Create**.

Wait until the NAT Gateway Status changes to Running before moving onto the next task.

Click *Check my progress* to verify the objective.

Create a NAT configuration using Cloud Router


Check my progress

Task 3. Configure instance templates and create instance groups

A managed instance group uses an instance template to create a group of identical instances. Use these to create the backends of the internal load balancer.

Configure the instance templates

An instance template is an API resource that you can use to create VM instances and managed instance groups. Instance templates define the machine type, boot disk image, subnet, labels, and other instance properties. Create an instance template for both subnets of the **my-internal-app** network.

1. On the **Navigation menu** () , click **Compute Engine > Instance templates**.
2. Click **Create instance template**.
3. For **Name**, type **instance-template-1**
4. Click **Management, security, disks, networking, sole tenancy**.
5. Click **Management**.
6. Under **Metadata**, specify the following:

Key	Value
startup-script-url	gs://cloud-training/gcpnet/ilb/startup.sh

The **startup-script-url** specifies a script that is executed when instances are started. This script installs Apache and changes the welcome page to include the client IP and the name, region, and zone of the VM instance. You can explore this script [here](#).

- Click **Networking**.
- For **Network interfaces**, specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Network	my-internal-app
Subnetwork	subnet-a
Network tags	backend-service
External IP	None

The network tag **backend-service** ensures that the firewall rule to allow traffic from any sources in the 10.10.0.0/16 subnet and the Health Check firewall rule applies to these instances.

- Click **Create**. Wait for the instance template to be created.

Create another instance template for **subnet-b** by copying **instance-template-1**:

- Select the **instance-template-1** and click **Copy**.
- Click **Management, security, disks, networking, sole tenancy**.


12. Click **Networking**.

13. For **Network interfaces**, select **subnet-b** as the **Subnetwork**.

14. Click **Create**.

Create the managed instance groups

Create a managed instance group in **subnet-a** (us-central1-a) and **subnet-b** (us-central1-b).

1. On the **Navigation menu** () , click **Compute Engine > Instance groups**.
2. Click **Create Instance group**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	instance-group-1
Location	Single-zone
Region	us-central1
Zone	us-central1-a
Instance template	instance-template-1
Autoscaling metrics > metrics type (Click the pencil edit icon)	CPU utilization
Target CPU utilization	80

Minimum number of instances	1
Maximum number of instances	5
Cool-down period	45

Managed instance groups offer **autoscaling** capabilities that allow you to automatically add or remove instances from a managed instance group based on increases or decreases in load. Autoscaling helps your applications gracefully handle increases in traffic and reduces cost when the need for resources is lower. Just define the autoscaling policy, and the autoscaler performs automatic scaling based on the measured load.

- Click **Create**.

Repeat the same procedure for **instance-group-2** in **us-central1-b**:

- Click **Create Instance group**.
- Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	instance-group-2
Location	Single-zone
Region	us-central1
Zone	us-central1-b
Instance template	instance-template-2

Autoscaling metrics > metric type (Click the pencil edit icon)	CPU utilization
Target CPU utilization	80
Minimum number of instances	1
Maximum number of instances	5
Cool-down period	45

7. Click **Create**.

Verify the backends

Verify that VM instances are being created in both subnets and create a utility VM to access the backends' HTTP sites.

1. On the **Navigation menu**, click **Compute Engine > VM instances**. Notice two instances that start with *instance-group-1* and *instance-group-2*.

These instances are in separate zones, and their internal IP addresses are part of the **subnet-a** and **subnet-b** CIDR blocks.

2. Click **Create Instance**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	utility-vm
Region	us-central1

Zone	us-central1-f
Series	N1
Machine type	f1-micro (1 vCPU)
Boot disk	Debian GNU/Linux 10 (buster)

- Click **Management, security, disks, networking, sole tenancy**.
- Click **Networking**.
- For **Network interfaces**, click the pencil icon to edit.
- Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Network	my-internal-app
Subnetwork	subnet-a
Primary internal IP	Ephemeral (Custom)
Custom ephemeral IP address	10.10.20.50
External IP	None

- Click **Done**.
- Click **Create**.

10. Note that the internal IP addresses for the backends are *10.10.20.2* and *10.10.30.2*.

If these IP addresses are different, replace them in the two **curl** commands below.

Click *Check my progress* to verify the objective.

Configure instance templates and create instance groups

Check my progress

11. For **utility-vm**, click **SSH** to launch a terminal and connect. If you see the **Connection via Cloud Identity-Aware Proxy Failed** popup, click **Retry**.

12. To verify the welcome page for *instance-group-1-xxxx*, run the following command:

```
curl 10.10.20.2
```

The output should look like this (**do not copy; this is example output**):

```
<h1>Internal Load Balancing Lab</h1><h2>Client IP</h2>Your IP address :  
10.10.20.50<h2>Hostname</h2>Server Hostname:  
  
instance-group-1-1zn8<h2>Server Location</h2>Region and Zone: us-central1-a
```

13. To verify the welcome page for *instance-group-2-xxxx*, run the following command:

```
curl 10.10.30.2
```

The output should look like this (**do not copy; this is example output**):

```
<h1>Internal Load Balancing Lab</h1><h2>Client IP</h2>Your IP address :  
10.10.20.50<h2>Hostname</h2>Server Hostname:  
  
instance-group-2-q5wp<h2>Server Location</h2>Region and Zone: us-central1-b
```

Which of these fields identify the location of the backend?

☐

Client IP

☐

Server Location



Server Hostname

Submit

This will be useful when verifying that the internal load balancer sends traffic to both backends.

14. Close the SSH terminal to **utility-vm**:

```
exit
```

Task 4. Configure the internal load balancer

Configure the internal load balancer to balance traffic between the two backends (**instance-group-1** in us-central1-a and **instance-group-2** in us-central1-b), as illustrated in the network diagram:

Subnet A 10.10.20.0/24

10.10.20.50



Client
Instance



Zone: us-central1-a

Backend 1



Instance Group 1

instance IP:10.10.20.2



Instance 1


instance IP:10.10.20.3



Instance 2

Instance group IG1

Start the configuration

1. In the Cloud Console, on the **Navigation menu** () , click **Network Services > Load balancing**.
2. Click **Create load balancer**.
3. Under **TCP Load Balancing**, click **Start configuration**.
4. For **Internet facing or internal only**, select **Only between my VMs**.

Choosing **Only between my VMs** makes this load balancer internal. This choice requires the backends to be in a single region (us-central1) and does not allow offloading TCP processing to the load balancer.

5. Click **Continue**.
6. For **Name**, type **my-ilb**.

Configure the regional backend service

The backend service monitors instance groups and prevents them from exceeding configured usage.

1. Click **Backend configuration**.
2. Specify the following, and leave the remaining settings as their defaults:

Property	Value (select option as specified)
Region	us-central1
Network	my-internal-app
Instance group	instance-group-1 (us-central1-a)

3. Click **Done**.
4. Click **Add backend**.

5. For **Instance group**, select **instance-group-2 (us-central1-b)**.
6. Click **Done**.
7. For **Health Check**, select **Create a health check**.
8. Specify the following, and leave the remaining settings as their defaults:

Property	Value (select option as specified)
Name	my-ilb-health-check
Protocol	TCP
Port	80

Health checks determine which instances can receive new connections. This HTTP health check polls instances every 5 seconds, waits up to 5 seconds for a response, and treats 2 successful or 2 failed attempts as healthy or unhealthy, respectively.

9. Click **Save and Continue**.
10. Verify that there is a blue check mark next to **Backend configuration** in the Cloud Console. If there isn't, double-check that you have completed all the steps above.

Configure the frontend

The frontend forwards traffic to the backend.

1. Click **Frontend configuration**.
2. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
----------	--------------------------------------------------

Subnetwork	subnet-b
Internal IP	Reserve a static internal IP address

- Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	my-ilb-ip
Static IP address	Let me choose
Custom IP address	10.10.30.5

- Click **Reserve**.
- For **Ports**, type **80**.
- Click **Done**.

Review and create the internal load balancer

- Click **Review and finalize**.
- Review the **Backend** and **Frontend**.
- Click **Create**. Wait for the load balancer to be created before moving to the next task.

Click *Check my progress* to verify the objective.

Configure the Internal Load Balancer

Check my progress

Task 5. Test the internal load balancer

Verify that the *my-ibb* IP address forwards traffic to **instance-group-1** in us-central1-a and **instance-group-2** in us-central1-b.

Access the internal load balancer

1. On the **Navigation** menu, click **Compute Engine > VM instances**.
2. For **utility-vm**, click **SSH** to launch a terminal and connect.
3. To verify that the internal load balancer forwards traffic, run the following command:

```
curl 10.10.30.5
```

The output should look like this (**do not copy; this is example output**):

```
<h1>Internal Load Balancing Lab</h1><h2>Client IP</h2>Your IP address :
10.10.20.50<h2>Hostname</h2>Server Hostname:
instance-group-1-1zn8<h2>Server Location</h2>Region and Zone: us-central1-a
```

As expected, traffic is forwarded from the internal load balancer (10.10.30.5) to the backend.

4. Run the same command a couple of times:

```
curl 10.10.30.5
```

```
curl 10.10.30.5
```

```
curl 10.10.30.5
```

```
curl 10.10.30.5
```

```
curl 10.10.30.5
```

```
curl 10.10.30.5
```

```
curl 10.10.30.5
```

```
curl 10.10.30.5
```

```
curl 10.10.30.5
```

```
curl 10.10.30.5
```

You should be able to see responses from **instance-group-1** in us-central1-a and **instance-group-2** in us-central1-b. If not, run the command again.

Task 1. Explore the Google Cloud Console

In this task, you explore the Google Cloud Console and create resources.

Verify that your project is selected

1. In the **Select a project** drop-down list in the title bar select the project ID that Qwiklabs provided with your authentication credentials.

The project ID will resemble **qwiklabs-Google Cloud-** followed by a long hexadecimal number.


2. Click **Cancel** to close the dialog.


Your title bar should indicate the project ID as shown in the screenshot. Each lab in the Qwiklabs environment has a unique project ID, as well as unique authentication credentials.

Navigate to Google Cloud Storage and create a bucket

Cloud Storage allows world-wide storage and retrieval of any amount of data at any time. You can use Cloud Storage for a range of scenarios including serving website content, storing data for archival and disaster recovery, or distributing large data objects to users via direct download.

Cloud Storage buckets must have a globally unique name. In your organization, you should follow Google Cloud's [recommended best practices for naming buckets](#). For this lab, we can easily get a unique name for our bucket by using the ID of the Google Cloud project that Qwiklabs created for us, because Google Cloud project IDs are also globally unique.

1. In the Google Cloud Console, on the **Navigation menu** () , click **Home** .
2. In the **Dashboard** tab of the resulting screen, the **Project info** section shows your Google Cloud project ID. Select and copy the project ID. Because this project ID was created for you by Qwiklabs, it will resemble **qwiklabs-Google Cloud-** followed by a long hexadecimal number.


3. In the Google Cloud Console, on the **Navigation menu** () , click **Storage > Browser**.
4. Click **Create bucket**.

5. For **Name**, paste in the Google Cloud project ID string you copied in an earlier step. Leave all other values as their defaults.

These lab instructions will later refer to the name that you typed as [BUCKET_NAME].

6. Click **Create**.



The Google Cloud Console has a **Notifications** () icon. Feedback from the underlying commands is sometimes provided there. You can click the icon to check the notifications for additional information and history.

Create a virtual machine (VM) instance

Google Compute Engine offers virtual machines running in Google's datacenters and on its network as a service. Google Kubernetes Engine makes use of Compute Engine as a component of its architecture. For this reason, it's helpful to learn a bit about Compute Engine before learning about Kubernetes Engine.

1. On the **Navigation menu**, click **Compute Engine > VM instances**.
2. Click **Create Instance**.
3. For **Name**, type first-vm as the name for your instance.
4. For **Region**, select **us-central1**.
5. For **Zone**, select **us-central1-c**.
6. For **Machine type**, examine the options.

The **Machine type**: menu lists the number of virtual CPUs, the amount of memory, and a symbolic name such as *e1-standard-1*. The symbolic name is the parameter you use to select the machine type when using the `gcloud` command to create a VM. To the right of the region, zone, and machine type is a per-month estimated cost.

7. To see the breakdown of estimated costs, click **Details** to the right of the **Machine type** list underneath the estimated costs.
8. For **Machine type**, click **2 vCPUs (e1-standard-2)**.

How did the cost change?

9. For **Machine type**, click **e1-micro (2 shared vCPU)**.

The micro type is a shared-core VM that is inexpensive.

10. For **Firewall**, click **Allow HTTP traffic**.
11. Leave the remaining settings as their defaults, and click **Create**.

Wait until the new VM is created.

Explore the VM details

1. On the **VM instances** page, click the name of your VM, first-vm.
2. Locate **CPU platform**, notice the value, and click **Edit**.

You can't change the machine type, the CPU platform, or the zone of a running Google Cloud VM. You can add network tags and allow specific network traffic from the internet through firewalls.

Some properties of a VM are integral to the VM and are established when the VM is created. They cannot be changed. Other properties can be edited. For example, you can add disks, and you can determine whether the boot disk is deleted when the instance is deleted.

3. Scroll down and examine **Availability policies**.

Compute Engine offers preemptible VM instances, which cost less per hour but can be terminated by Google Cloud at any time. These preemptible instances can save you a lot of money, but you must make sure that your workloads are suitable to be interrupted. You can't convert a non-preemptible instance into a preemptible one. This choice must be made at VM creation.

If a VM is stopped for any reason (for example, an outage or a hardware failure), the automatic restart feature starts it back up. Is this the behavior you want? Are your applications idempotent (written to handle a second startup properly)?

During host maintenance, the VM is set for live migration. However, you can have the VM terminated instead of migrated.

If you make changes, they can sometimes take several minutes to be implemented, especially if they involve networking changes, like adding firewalls or changing the external IP.

4. Click **Cancel**.

Create an IAM service account

An IAM service account is a special type of Google account that belongs to an application or a virtual machine, instead of to an individual end user.

1. On the **Navigation menu**, click **IAM & admin > Service accounts**.
2. Click **+ Create service account**.
3. On the **Service account details** page, specify the **Service account name** as test-service-account.
4. Click **Create**.
5. On the **Service account permissions** page, specify the role as **Project > Editor**.
6. Click **Continue**.
7. Click **Done**.
8. On the **Service accounts** page, move to the extreme right of the test-service-account and click on the three dots.
9. Click **Create Key**.
10. Select **JSON** as the key type.
11. Click **Create**.

A JSON key file is downloaded. In a later step, you find this key file and upload it to the VM.

12. Click **Close**.

Click *Check my progress* to verify the objective.

Create a bucket, VM instance with necessary firewall rule and an IAM service account.

Check my progress

Task 2. Explore Cloud Shell


Cloud Shell provides you with command-line access to your cloud resources directly from your browser. With Cloud Shell, Cloud SDK command-line tools such as gcloud are always available, up to date, and fully authenticated.

Cloud Shell provides the following features and capabilities:

- Temporary Compute Engine VM
 - Command-line access to the instance through a browser
 - 5 GB of persistent disk storage (\$HOME dir)
 - Preinstalled Cloud SDK and other tools
 - gcloud: for working with Compute Engine, Google Kubernetes Engine (GKE) and many Google Cloud services
 - gsutil: for working with Cloud Storage
 - kubectl: for working with GKE and Kubernetes
 - bq: for working with BigQuery
 - Language support for Java, Go, Python, Node.js, PHP, and Ruby
 - Web preview functionality
 - Built-in authorization for access to resources and instances
- After 1 hour of inactivity, the Cloud Shell instance is recycled. Only the /home directory persists. Any changes made to the system configuration, including environment variables, are lost between sessions.

In this task, you use Cloud Shell to create and examine some resources.

Open Cloud Shell and explore its features

1. On the Google Cloud Console title bar, click **Activate Cloud Shell** (.
2. When prompted, click **Continue**.

Cloud Shell opens at the bottom of the Google Cloud Console window.

The following icons are on the far right of Cloud Shell toolbar:

- **Hide/Restore:** This icon hides and restores the window, giving you full access to the Google Cloud Console without closing Cloud Shell.

- **Open in new window:** Having Cloud Shell at the bottom of the Google Cloud Console is useful when you are issuing individual commands. But when you edit files or want to see the full output of a command, clicking this icon displays Cloud Shell in a full-sized terminal window.
- **Close all tabs:** This icon closes Cloud Shell. Everytime you close Cloud Shell, the virtual machine is recycled and all machine context is lost. However, data that you stored in your home directory is still available to you the next time you start Cloud Shell.

Use Cloud Shell to set up the environment variables for this task

In Cloud Shell, use the following commands to define the environment variables used in this task.

1. Replace [BUCKET_NAME] with the name of the first bucket from task 1.
2. Replace [BUCKET_NAME_2] with a globally unique name.

You can append a 2 to the globally unique bucket name that you used previously.

```
MY_BUCKET_NAME_1=[BUCKET_NAME]
MY_BUCKET_NAME_2=[BUCKET_NAME_2]
MY_REGION=us-central1
```

When you are working in the Cloud Shell or writing scripts, creating environment variables is a good practice. You can easily and consistently re-use these environment variables, which makes your work less error-prone.


Make sure you replace the full placeholder string, such as [BUCKET_NAME] with the unique name that you choose, for example MY_BUCKET_NAME_1=unique_bucket_name.

Move the credentials file you created earlier into Cloud Shell

You downloaded a JSON-encoded credentials file in an earlier task when you created your first Cloud IAM service account.

1. On your local workstation, locate the JSON key that you just downloaded and rename the file to credentials.json.



2. In Cloud Shell, click the three dots () icon in the Cloud Shell toolbar to display further options.
3. Click **Upload file** and upload the credentials.json file from your local machine to the Cloud Shell VM.
4. Click the **X** icon to close the file upload pop-up window.


5. In Cloud Shell, type **ls** and press ENTER to confirm that the file was uploaded.

Create a second Cloud Storage bucket and verify it in the Google Cloud Console

The **gsutil** command, which is supplied by the Cloud SDK, lets you work with Cloud Storage from the command line. In this task, you use the **gsutil** command in Cloud Shell.

1. In Cloud Shell, use the **gsutil** command to create a bucket.

```
gsutil mb gs://$MY_BUCKET_NAME_2
```

2. In the Google Cloud Console, on the **Navigation menu** ()
click **Storage > Browser**, or click **Refresh** if you are already in the Storage Browser.
The second bucket should appear in the **Buckets** list.

Use the **gcloud** command line to create a second virtual machine

1. In Cloud Shell, execute the following command to list all the zones in a given region:

```
gcloud compute zones list | grep $MY_REGION
```

2. Select a zone from the first column of the list. Notice that Google Cloud zones' names consist of their region name, followed by a hyphen and a letter.
You may choose a zone that is the same as or different from the zone that you used for the first VM in task 1.

3. Execute the following command to store your chosen zone in an environment variable.
You replace [ZONE] with your selected zone.

```
MY_ZONE=[ZONE]
```

4. Set this zone to be your default zone by executing the following command.

```
gcloud config set compute/zone $MY_ZONE
```

5. Execute the following command to store a name in an environment variable you will use to create a VM. You will call your second VM **second-vm**.

```
MY_VMNAME=second-vm
```


6. Create a VM in the default zone that you set earlier in this task using the new environment variable to assign the VM name.

```
gcloud compute instances create $MY_VMNAME \
--machine-type "e2-standard-2" \
--image-project "debian-cloud" \
--image-family "debian-9" \
--subnet "default"
```

7. List the virtual machine instances in your project.

```
gcloud compute instances list
```

You will see both your newly created and your first virtual machine in the list.

8. In the Google Cloud Console, on the **Navigation menu** () , click **Compute Engine > VM Instances**. Just as in the output of `gcloud compute instances list`, you will see both of the virtual machines you created.
9. Look at the External IP column. Notice that the external IP address of the first VM you created is shown as a link. (If necessary, click the HIDE INFO PANEL button to reveal the External IP column.) The Google Cloud Console offers the link because you configured this VM's firewall to allow HTTP traffic.
10. Click the link you found in your VM's External IP column. Your browser will present a Connection refused message in a new browser tab. This message occurs because, although there is a firewall port open for HTTP traffic to your VM, no Web server is running there. Close the browser tab you just created.

Use the `gcloud` command line to create a second service account


1. In Cloud Shell, execute the following command to create a new service account:

```
gcloud iam service-accounts create test-service-account2 --display-name "test-service-account2"
```

If you see the following output, type **y** and press **ENTER**:

Output (do not copy)

```
API [iam.googleapis.com] not enabled on project [560255523887]. Would
you like to enable and retry (this will take a few minutes)? (y/N)?
```

2. In the Google Cloud Console, on the **Navigation menu** () , click **IAM & admin > Service accounts**.

Refresh the page till you see **test-service-account2**.

Click *Check my progress* to verify the objective.


Create a second bucket, VM instance and an IAM service account.

Check my progress

3. In Cloud Shell, execute the following command to grant the second service account the Project viewer role:

```
gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT --member
serviceAccount:test-service-
account2@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com --role roles/viewer
```

GOOGLE_CLOUD_PROJECT is an environment variable that is automatically populated in Cloud Shell and is set to the project ID of the current context.

4. In the Google Cloud Console, on the **Navigation menu** () , click **IAM & admin > IAM**. Select the new service account called **test-service-account2**.
5. On the right hand side of the page, click on pencil icon and expand the Viewer role. You will see **test-service-account2** listed as a member of the Viewer role.

Task 3. Work with Cloud Storage in Cloud Shell

Download a file to Cloud Shell and copy it to Cloud Storage

1. Copy a picture of a cat from a Google-provided Cloud Storage bucket to your Cloud Shell.


```
gsutil cp gs://cloud-training/ak8s/cat.jpg cat.jpg
```

2. Copy the file into one of the buckets that you created earlier.

```
gsutil cp cat.jpg gs://$MY_BUCKET_NAME_1
```

3. Copy the file from the first bucket into the second bucket:

```
gsutil cp gs://$MY_BUCKET_NAME_1/cat.jpg gs://$MY_BUCKET_NAME_2/cat.jpg
```

4. In the Google Cloud Console, on the **Navigation menu** () , click **Storage > Browser**, select the buckets that you created, and verify that both contain the cat.jpg file.

Set the access control list for a Cloud Storage object

1. To get the default access list that's been assigned to cat.jpg (when you uploaded it to your Cloud Storage bucket), execute the following two commands:

```
gsutil acl get gs://$MY_BUCKET_NAME_1/cat.jpg > acl.txt
cat acl.txt
```

The output should look like the following example, but with different numbers. This output shows that anyone with a Project Owner, Editor, or Viewer role for the project has access (Owner access for Owners/Editors and Reader access for Viewers).

Output (do not copy)

```
[
  {
    "entity": "project-owners-560255523887",
    "projectTeam": {
      "projectNumber": "560255523887",
      "team": "owners"
    },
    "role": "OWNER"
  },
  {
    "entity": "project-editors-560255523887",
    "projectTeam": {
      "projectNumber": "560255523887",
      "team": "editors"
    },
    "role": "OWNER"
  },
  {
    "entity": "project-viewers-560255523887",
    "projectTeam": {
      "projectNumber": "560255523887",
      "team": "viewers"
    },
    "role": "READER"
  },
  {
    "email": "google12345678_student@qwiklabs.net",
    "entity": "user-google12345678_student@qwiklabs.net",
    "role": "OWNER"
  }
]
```

2. To change the object to have private access, execute the following command:

```
gsutil acl set private gs://$MY_BUCKET_NAME_1/cat.jpg
```

3. To verify the new ACL that's been assigned to cat.jpg, execute the following two commands:

```
gsutil acl get gs://$MY_BUCKET_NAME_1/cat.jpg > acl-2.txt  
cat acl-2.txt
```

The output should look similar to the following example. Now only the original creator of the object (your lab account) has Owner access.

Output (do not copy)

```
[  
  {  
    "email": "google12345678_student@qwiklabs.net",  
    "entity": "user-google12345678_student@qwiklabs.net",  
    "role": "OWNER"  
  }  
]
```

Authenticate as a service account in Cloud Shell

1. In Cloud Shell, execute the following command to view the current configuration:

```
gcloud config list
```

You should see output that looks like the following example. In your output, the zone should be equal to the zone that you set when you created your second VM in task 2. The account and project should match your Qwiklabs lab credentials.

Output (do not copy)

```
[component_manager]  
disable_update_check = True  
[compute]  
gce_metadata_read_timeout_sec = 5  
zone = us-central1-a  
[core]  
account = google12345678_student@qwiklabs.net  
disable_usage_reporting = False  
project = qwiklabs-Google Cloud-1aeffb5d0acb416  
[metrics]  
environment = devshell
```

Your active configuration is: [cloudshell-16441]

2. In Cloud Shell, execute the following command to change the authenticated user to the first service account (which you created in an earlier task) through the credentials that you downloaded to your local machine and then uploaded into Cloud Shell (credentials.json).

```
gcloud auth activate-service-account --key-file credentials.json
```

Cloud Shell is now authenticated as test-service-account.

3. To verify the active account, execute the following command:

```
gcloud config list
```

You should see output that looks like the following example. The account is now set to the test-service-account service account.

Output (do not copy)

```
[component_manager]
disable_update_check = True
[compute]
gce_metadata_read_timeout_sec = 5
zone = us-central1-a
[core]
account = test-service-account@qwiklabs-Google Cloud-
1aeffbc5d0acb416.iam.gserviceaccount.com
disable_usage_reporting = False
project = qwiklabs-Google Cloud-1aeffbc5d0acb416
[metrics]
environment = devshell
```

Your active configuration is: [cloudshell-16441]

4. To verify the list of authorized accounts in Cloud Shell, execute the following command:

```
gcloud auth list
```

You should see output that looks like the following example.

Output (do not copy)

```
Credentialed Accounts
```

ACTIVE ACCOUNT

```
google12345678_student@qwiklabs.net
* test-service-account@qwiklabs-Google Cloud-
1aeffb5d0acb416.iam.gserviceaccount.com
```

To set the active account, run:

```
$ gcloud config set account `ACCOUNT`
```

5. To verify that the current account (test-service-account) cannot access the cat.jpg file in the first bucket that you created, execute the following command:

```
gsutil cp gs://$MY_BUCKET_NAME_1/cat.jpg ./cat-copy.jpg
```

Because you restricted access to this file to the owner earlier in this task you should see output that looks like the following example.

Output (do not copy)

```
Copying gs://test-bucket-123/cat.jpg...
AccessDeniedException: 403 KiB]
```

6. Verify that the current account (test-service-account) can access the cat.jpg file in the second bucket that you created:

```
gsutil cp gs://$MY_BUCKET_NAME_2/cat.jpg ./cat-copy.jpg
```

Because access has not been restricted to this file you should see output that looks like the following example.

Output (do not copy)

```
Copying gs://test-bucket-123/cat.jpg...
- [1 files][ 81.7 KiB/ 81.7 KiB]
Operation completed over 1 objects/81.7 KiB.
```

7. To switch to the lab account, execute the following command.
You replace [USERNAME] with the username provided in the Qwiklabs Connection Details pane on the left of the lab instructions page. .

```
gcloud config set account [USERNAME]
```

8. To verify that you can access the cat.jpg file in the [BUCKET_NAME] bucket (the first bucket that you created), execute the following command.

```
gsutil cp gs://$MY_BUCKET_NAME_1/cat.jpg ./copy2-of-cat.jpg
```

You should see output that looks like the following example. The lab account created the bucket and object and remained an Owner when the object access control list (ACL) was converted to private, so the lab account can still access the object.


Output (do not copy)

```
Copying gs://test-bucket-123/cat.jpg...  
- [1 files][ 81.7 KiB/ 81.7 KiB]  
Operation completed over 1 objects/81.7 KiB.
```

9. Make the first Cloud Storage bucket readable by everyone, including unauthenticated users.

```
gsutil iam ch allUsers:objectViewer gs://$MY_BUCKET_NAME_1
```

This is an appropriate setting for hosting public website content in Cloud Storage.

10. In the Google Cloud Console, on the **Navigation menu** ()
click **Storage > Browser**, select the first storage bucket that you created. Notice that the cat.jpg file has a Public link. Copy this link.
11. Open an incognito browser tab and paste the link into its address bar. You will see a picture of a cat. Leave this browser tab open.

Click *Check my progress* to verify the objective.

Work with the Cloud Storage in Cloud Shell.

Check my progress

It doesn't look like you've completed this step yet. Try again.

Task 4. Explore the Cloud Shell code editor

In this task, you explore using the Cloud Shell code editor.

Open the Cloud Shell code editor

1. In Cloud Shell, click the Open in new window icon on the top right. Then click the pencil icon to open the Cloud Shell code editor.



Open in new window

A new tab opens with the Cloud Shell Code editor and the Cloud Shell. The Google Cloud console remains on the original tab. You can switch between the Google Cloud Console and Cloud Shell by clicking the tab.

2. In Cloud Shell, execute the following command to clone a git repository:

```
git clone https://github.com/googlecode-labs/orchestrate-with-kubernetes.git
```

The orchestrate-with-kubernetes folder appears in the left pane of the Cloud Shell code editor window.

3. In Cloud Shell, execute the following command to create a test directory:

```
mkdir test
```

The test folder now appears in the left pane of the Cloud Shell code editor window.

4. In the Cloud Shell code editor, click the arrow to the left of orchestrate-with-kubernetes to expand the folder.
5. Click the cleanup.sh file to open it in the right pane of the Cloud Shell code editor window.
6. Add the following text as the last line of the cleanup.sh file:

```
echo Finished cleanup!
```

No action is necessary to save your work.

7. In Cloud Shell, execute the following commands to change directory and display the contents of cleanup.sh:

```
cd orchestrate-with-kubernetes  
cat cleanup.sh
```

8. Verify that the output of cat cleanup.sh includes the line of text that you added.
9. In the Cloud Shell code editor, click to open the File menu and choose New File. Name the file index.html.
10. In the right hand pane, paste in this HTML text:

```
<html><head><title>Cat</title></head>
```

```
<body>
<h1>Cat</h1>


</body></html>
```

Use your local computer's keyboard shortcut to paste: `Cmd-V` for a Mac, `Ctrl-V` for a Windows or Linux machine.

11. Replace the string REPLACE_WITH_CAT_URL with the URL of the cat image from an earlier task. The URL will look like this:

Example (do not copy)

```
https://storage.googleapis.com/qwiklabs-Google Cloud-1aeffb5d0acb416/cat.jpg
```

12. On the **Navigation menu** () , click **Compute Engine > VM instances**.

13. In the row for your first VM, click the SSH button.

14. In the SSH login window that opens on your VM, install the nginx Web server:

```
sudo apt-get update
sudo apt-get install nginx
```

It may take few minutes to complete the process.

15. In your Cloud Shell window, copy the HTML file you created using the Code Editor to your virtual machine:

```
gcloud compute scp index.html first-vm:index.nginx-debian.html --zone=us-central1-c
```

If you are prompted whether to add a host key to your list of known hosts, answer **y**.
If you are prompted to enter a passphrase, press the **Enter** key to respond with an empty passphrase. Press the **Enter** key again when prompted to confirm the empty passphrase.


16. In the **SSH** login window for your VM, copy the HTML file from your home directory to the document root of the nginx Web server:

```
sudo cp index.nginx-debian.html /var/www/html
```

Click *Check my progress* to verify the objective.

Install the nginx Web server and customize the welcome page.


Check my progress

17. On the **Navigation menu** () , click **Compute Engine > VM instances**. Click the link in the External IP column for your first VM. A new browser tab opens, containing a Web page that contains the cat image.

End your lab

CONTAINERS AND CLOUD BUILD

Task 1: Confirm that needed APIs are enabled

1. Make a note of the name of your Google Cloud project. This value is shown in the top bar of the Google Cloud Console. It will be of the form qwiklabs-gcp- followed by hexadecimal numbers.
2. In the Google Cloud Console, on the **Navigation menu** () , click **APIs & Services**.
3. Click **Enable APIs and Services**.
4. In the **Search for APIs & Services** box, enter Cloud Build.
5. In the resulting card for the Cloud Build API, if you do not see a message confirming that the Cloud Build API is enabled, click the ENABLE button.
6. Use the Back button to return to the previous screen with a search box. In the search box, enter Container Registry.
7. In the resulting card for the Container Registry API, if you do not see a message confirming that the Container Registry API is enabled, click the ENABLE button.

Task 2. Building Containers with DockerFile and Cloud Build

You can write build configuration files to provide instructions to Cloud Build as to which tasks to perform when building a container. These build files can fetch dependencies, run unit tests, analyses and more. In this task, you'll create a DockerFile and use it as a build configuration script with Cloud Build. You will also create a simple shell script (quickstart.sh) which will represent an application inside the container.

1. On the Google Cloud Console title bar, click **Activate Cloud Shell**.
2. When prompted, click **Continue**.

Cloud Shell opens at the bottom of the Google Cloud Console window.

3. Create an empty quickstart.sh file using the nano text editor.

```
nano quickstart.sh
```

4. Add the following lines in to the quickstart.sh file:

```
#!/bin/sh  
echo "Hello, world! The time is $(date)."
```

5. Save the file and close nano by pressing the **CTRL+X** key, then press **Y** and **Enter**.
6. Create an empty Dockerfile file using the nano text editor.

```
nano Dockerfile
```

7. Add the following Dockerfile command:

```
FROM alpine
```

This instructs the build to use the Alpine Linux base image.

8. Add the following Dockerfile command to the end of the Dockerfile:

```
COPY quickstart.sh /
```

This adds the quickstart.sh script to the / directory in the image.

9. Add the following Dockerfile command to the end of the Dockerfile:

```
CMD ["/quickstart.sh"]
```

This configures the image to execute the /quickstart.sh script when the associated container is created and run.

The Dockerfile should now look like:

```
FROM alpine  
COPY quickstart.sh /  
CMD ["/quickstart.sh"]
```

10. Save the file and close nano by pressing the **CTRL+X** key, then press **Y** and **Enter**.
11. In Cloud Shell, run the following command to make the quickstart.sh script executable.

```
chmod +x quickstart.sh
```


12. In Cloud Shell, run the following command to build the Docker container image in Cloud Build.

```
gcloud builds submit --tag gcr.io/${GOOGLE_CLOUD_PROJECT}/quickstart-image .
```

Important

Don't miss the dot (".") at the end of the command. The dot specifies that the source code is in the current working directory at build time.

When the build completes, your Docker image is built and pushed to Container Registry.

13. In the Google Cloud Console, on the **Navigation menu** () , click **Container Registry > Images**.

The quickstart-image Docker image appears in the list

Task 3. Building Containers with a build configuration file and Cloud Build

Cloud Build also supports custom build configuration files. In this task you will incorporate an existing Docker container using a custom YAML-formatted build file with Cloud Build.

1. In Cloud Shell enter the following command to clone the repository to the lab Cloud Shell.

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

2. Create a soft link as a shortcut to the working directory.

```
ln -s ~/training-data-analyst/courses/ak8s/v1.1 ~/ak8s
```

3. Change to the directory that contains the sample files for this lab.

```
cd ~/ak8s/Cloud_Build/a
```

A sample custom cloud build configuration file called cloudbuild.yaml has been provided for you in this directory as well as copies of the Dockerfile and the quickstart.sh script you created in the first task.

4. In Cloud Shell, execute the following command to view the contents of cloudbuild.yaml.

```
cat cloudbuild.yaml
```

You will see the following:


```
steps:
- name: 'gcr.io/cloud-builders/docker'
  args: [ 'build', '-t', 'gcr.io/$PROJECT_ID/quickstart-image', '.' ]
images:
- 'gcr.io/$PROJECT_ID/quickstart-image'
```

This file instructs Cloud Build to use Docker to build an image using the Dockerfile specification in the current local directory, tag it with gcr.io/\$PROJECT_ID/quickstart-image (\$PROJECT_ID is a substitution variable automatically populated by Cloud Build with the project ID of the associated project) and then push that image to Container Registry.

5. In Cloud Shell, execute the following command to start a Cloud Build using cloudbuild.yaml as the build configuration file:

```
gcloud builds submit --config cloudbuild.yaml .
```

The build output to Cloud Shell should be the same as before. When the build completes, a new version of the same image is pushed to Container Registry.


6. In the Google Cloud Console, on the **Navigation menu** () , click **Container Registry > Images** and then click quickstart-image.

Two versions of quickstart-image are now in the list.

Click *Check my progress* to verify the objective.

Build two Container images in Cloud Build.

Check my progress

7. In the Google Cloud Console, on the **Navigation menu** () , click **Cloud Build > History**.

Two builds appear in the list.

8. Click the build ID for the build at the top of the list.

The details of the build, including the build log, are displayed.

Task 4. Building and Testing Containers with a build configuration file and Cloud Build

The true power of custom build configuration files is their ability to perform other actions, in parallel or in sequence, in addition to simply building containers: running tests on your newly built containers, pushing them to various destinations, and even deploying them to Kubernetes Engine. In this lab, we will see a simple example: a build configuration file that tests the container it built and reports the result to its calling environment.

1. In Cloud Shell, change to the directory that contains the sample files for this lab.

```
cd ~/ak8s/Cloud_Build/b
```

As before, the quickstart.sh script and the a sample custom cloud build configuration file called cloudbuild.yaml has been provided for you in this directory. These have been slightly modified to demonstrate Cloud Build's ability to test the containers it has build. There is also a Dockerfile present, which is identical to the one used for the previous task.

2. In Cloud Shell, execute the following command to view the contents of cloudbuild.yaml.

```
cat cloudbuild.yaml
```

You will see the following:

```
steps:
- name: 'gcr.io/cloud-builders/docker'
  args: [ 'build', '-t', 'gcr.io/$PROJECT_ID/quickstart-image', '.' ]
- name: 'gcr.io/$PROJECT_ID/quickstart-image'
  args: ['fail']
images:
- 'gcr.io/$PROJECT_ID/quickstart-image'
```

In addition to its previous actions, this build configuration file runs the quickstart-image it has created. In this task, the quickstart.sh script has been modified so that it simulates a test failure when an argument ['fail'] is passed to it.

3. In Cloud Shell, execute the following command to start a Cloud Build using cloudbuild.yaml as the build configuration file:

```
gcloud builds submit --config cloudbuild.yaml .
```

You will see output from the command that ends with text like this:

Output (do not copy)

```
Finished Step #1
ERROR
ERROR: build step 1 "gcr.io/ivil-charmer-227922klabs-gcp-49ab2930eea05/quickstart-
image" failed: exit status 127
-----
ERROR: (gcloud.builds.submit) build f3e94c28-fba4-4012-a419-48e90fca7491 completed
with status "FAILURE"
```

4. Confirm that your command shell knows that the build failed:

```
echo $?
```


The command will reply with a non-zero value. If you had embedded this build in a script, your script would be able to act up on the build's failure.

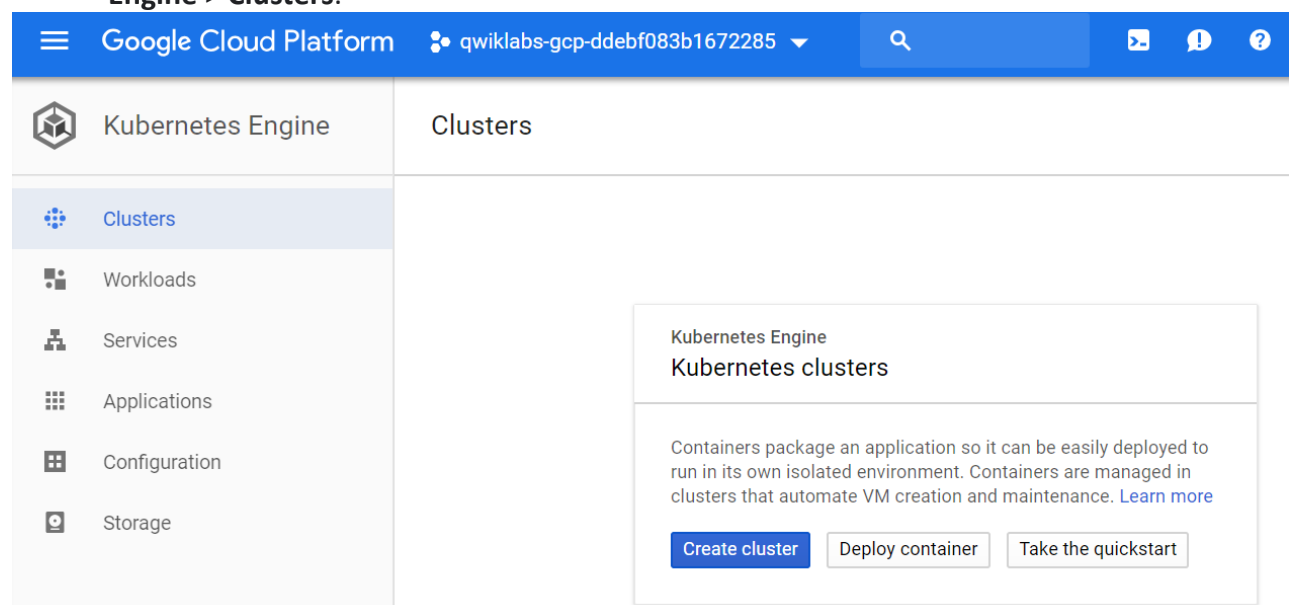
MANAGING GKE CLUSTERS

Task 1. Deploy GKE clusters

In this task, you use the Google Cloud Console and Cloud Shell to deploy GKE clusters.

Use the Google Cloud Console to deploy a GKE cluster

1. In the Google Cloud Console, on the **Navigation menu** () , click **Kubernetes Engine > Clusters**.



The screenshot shows the Google Cloud Platform console interface. At the top, the navigation bar includes the Google Cloud Platform logo, the project name 'qwiklabs-gcp-ddeb083b1672285', a search bar, and notification icons. The left sidebar contains the navigation menu with options: Kubernetes Engine, Clusters (selected), Workloads, Services, Applications, Configuration, and Storage. The main content area displays the 'Kubernetes Engine Clusters' page. It features a heading 'Kubernetes Engine Kubernetes clusters' and a descriptive paragraph: 'Containers package an application so it can be easily deployed to run in its own isolated environment. Containers are managed in clusters that automate VM creation and maintenance. [Learn more](#)'. Below the text are three buttons: 'Create cluster' (highlighted in blue), 'Deploy container', and 'Take the quickstart'.

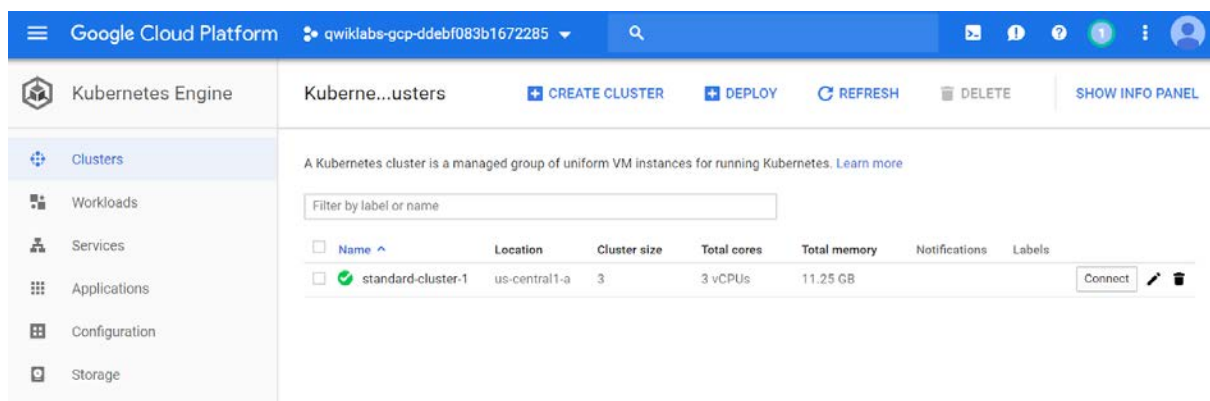
2. Click **Create cluster** to begin creating a GKE cluster.
3. Examine the console UI and the controls to change the cluster name, the cluster location, Kubernetes version, the number of nodes, and the node resources such as the machine type in the default node pool.

Clusters can be created across a region or in a single zone. A single zone is the default. When you deploy across a region the nodes are deployed to three separate zones and the total number of nodes deployed will be three times higher.

4. Change the cluster name to **standard-cluster-1** and zone to **us-central1-a**. Leave all the values at their defaults and click **Create**.

The cluster begins provisioning.

Note: You need to wait a few minutes for the cluster deployment to complete. When provisioning is complete, the **Kubernetes Engine > Clusters** page looks like the screenshot:



Click *Check my progress* to verify the objective.

Deploy GKE cluster

Check my progress

5. Click the cluster name **standard-cluster-1** to view the cluster details

Kubernetes Engine

← Clusters

EDIT

DELETE

ADD NODE POOL

DEPLOY

CONNECT

Clusters

Workloads

Services & Ingress

Applications

Configuration

Storage

Object Browser

standard-cluster-1

Details Storage Nodes

Cluster

Master version	1.14.10-gke.17	Upgrade available
Endpoint	34.67.14.111	Show cluster certificate
Client certificate	Disabled	
Binary Authorization	Disabled	
Kubernetes alpha features	Disabled	
Total size	3	
Master zone	us-central1-a	
Node zones	us-central1-a	
Network	default	
Subnet	default	
VPC-native (alias IP)	Enabled	
Pod address range	10.12.0.0/14	
Default maximum pods per node	110	
Service address range	10.16.0.0/20	
Intranode visibility	Disabled	
Kubernetes Engine Monitoring	System and workload logging and monitoring	
Private cluster	Disabled	
Master authorized networks	Disabled	
Network policy	Disabled	
NodeLocal DNSCache	Disabled	
Legacy authorization	Disabled	
Maintenance window	Any time	

- You can scroll down the page to view more details.
- Click the **Storage** and **Nodes** tabs under the cluster name (standard-cluster-1) at the top to view more of the cluster details.

Task 2. Modify GKE clusters

It is easy to modify many of the parameters of existing clusters using either the Google Cloud Console or Cloud Shell. In this task, you use the Google Cloud Console to modify the size of GKE clusters.

- In the Google Cloud Console, click **Edit** at the top of the details page for **standard-cluster-1**.
- Scroll down to the **Node Pools** section and click **default pool**.
- In the Google Cloud Console, click **Edit** at the top of the details page.
- In the **Size** section, change the number of nodes from 3 to 4.

Kubernetes Engine

[←](#)
Edit node pool
DELETE

Clusters

Workloads

Services & Ingress

Applications

Configuration

Storage

Object Browser

Edit default-pool

Node version
1.14.10-gke.17
[CHANGE](#)

Size
Number of nodes *

☐ Enable autoscaling [?](#)

Nodes
Image type
Container-Optimized OS (cos)
[CHANGE](#)

Management
☒ Enable auto-upgrade [?](#)
☒ Enable auto-repair [?](#)
☐ Enable surge upgrade [?](#)
Max surge Max unavailable

Security
☐ Enable GKE Metadata Server [?](#)

SAVE CANCEL

5. Scroll to the bottom and click **Save**.

6. In the Google Cloud Console, on the **Navigation menu** () , click **Kubernetes Engine > Clusters**.

When the operation completes, the **Kubernetes Engine > Clusters** page should show that standard-cluster-1 now has four nodes.

Google Cloud Platform
qwiklabs-gcp-ddeb083b1672285

Kubernetes Engine

Kubernetes clusters
CREATE CLUSTER
DEPLOY
REFRESH
SHOW INFO PANEL

Clusters

Workloads

Services

Applications

Configuration

Storage

A Kubernetes cluster is a managed group of uniform VM instances for running Kubernetes. [Learn more](#)

Filter by label or name

<input type="checkbox"/> Name ^	Location	Cluster size	Total cores	Total memory	Notifications	Labels
<input checked="" type="checkbox"/> standard-cluster-1	us-central1-a	4	4 vCPUs	15.00 GB		Connect


Click *Check my progress* to verify the objective.

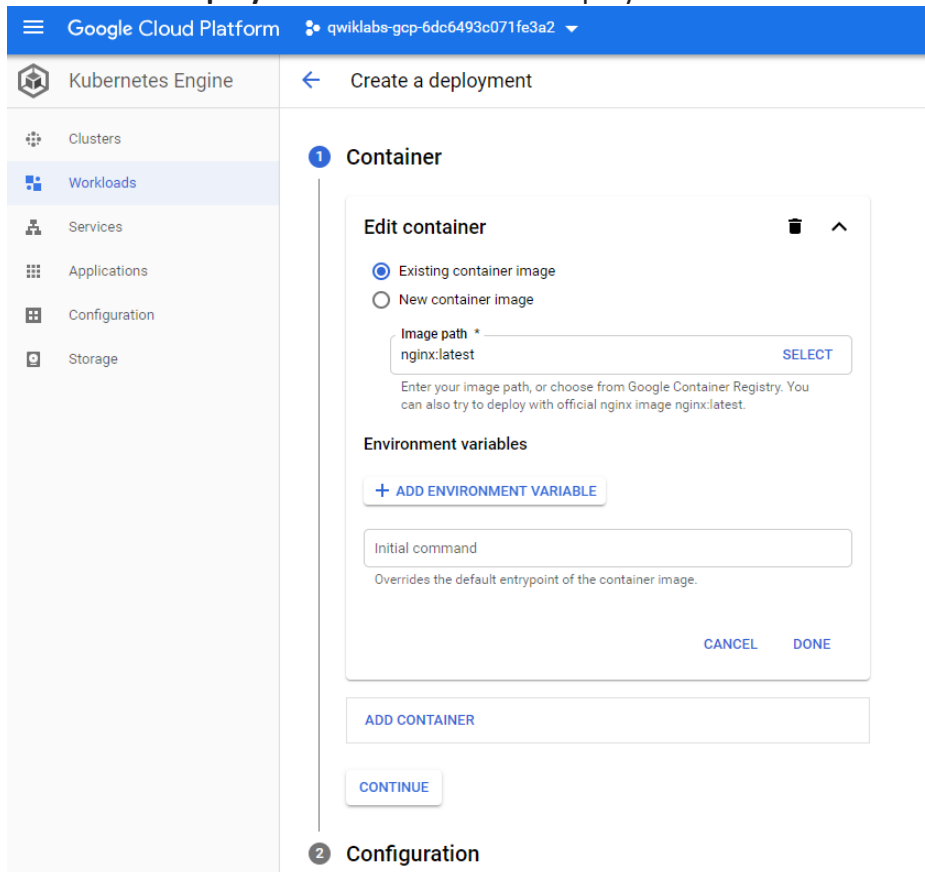
Modify GKE clusters

Check my progress

Task 3. Deploy a sample workload

In this task, using the Google Cloud console you will deploy a Pod running the nginx web server as a sample workload.

1. In the Google Cloud Console, on the **Navigation menu** () , click **Kubernetes Engine > Workloads**.
2. Click **Deploy** to show the Create a deployment wizard.



Google Cloud Platform qwiklabs-gcp-6dc6493c071fe3a2

Kubernetes Engine Create a deployment

1 Container

Edit container

☒ Existing container image
☐ New container image

Image path *
nginx:latest [SELECT](#)

Enter your image path, or choose from Google Container Registry. You can also try to deploy with official nginx image nginx:latest.

Environment variables

[+ ADD ENVIRONMENT VARIABLE](#)

Initial command
Overrides the default endpoint of the container image.

[CANCEL](#) [DONE](#)

[ADD CONTAINER](#)

[CONTINUE](#)

2 Configuration

3. Click **Continue** to accept the default container image, nginx.latest, which deploys a Pod with a single container running the latest version of nginx.

Google Cloud Platform

qwiklabs-gcp-6dc6493c071fe3a2

Kubernetes Engine

Create a deployment

Clusters

Workloads

Services

Applications

Configuration

Storage

✓ Container

2 Configuration

A deployment is a configuration which defines how Kubernetes deploys, manages, and scales your container image. Kubernetes will ensure your system matches this configuration.

Application name *
nginx-1

Namespace *
default

Labels

Key	Value
app	nginx-1

+ ADD KUBERNETES LABEL

Configuration YAML

Kubernetes deployments are defined declaratively using YAML files. The best practice is to store these files in version control, so you can track changes to your deployment configuration over time.

VIEW YAML

Cluster

Kubernetes Cluster
standard-cluster-1 (us-central1-a)

Cluster in which the deployment will be created.

CREATE NEW CLUSTER

DEPLOY

4. Scroll to the bottom of the window and click the **Deploy** button leaving the **Configuration** details at the defaults.
5. When the deployment completes your screen will refresh to show the details of your new nginx deployment.

The screenshot displays the Google Cloud Platform console for a Kubernetes deployment named 'nginx-1'. The left navigation pane shows 'Workloads' selected. The main panel shows the deployment is in a healthy state with 3 replicas. The right sidebar offers options to expose the deployment and links to relevant documentation. The 'Active revisions' table at the bottom provides a detailed view of the deployment's history.

Revision	Name	Status	Summary	Created on	Pods running/Pods total
1	nginx-1-68bc45996d	OK	nginx: nginx:latest	Dec 16, 2018, 1:35:15 PM	3/3


Click *Check my progress* to verify the objective.

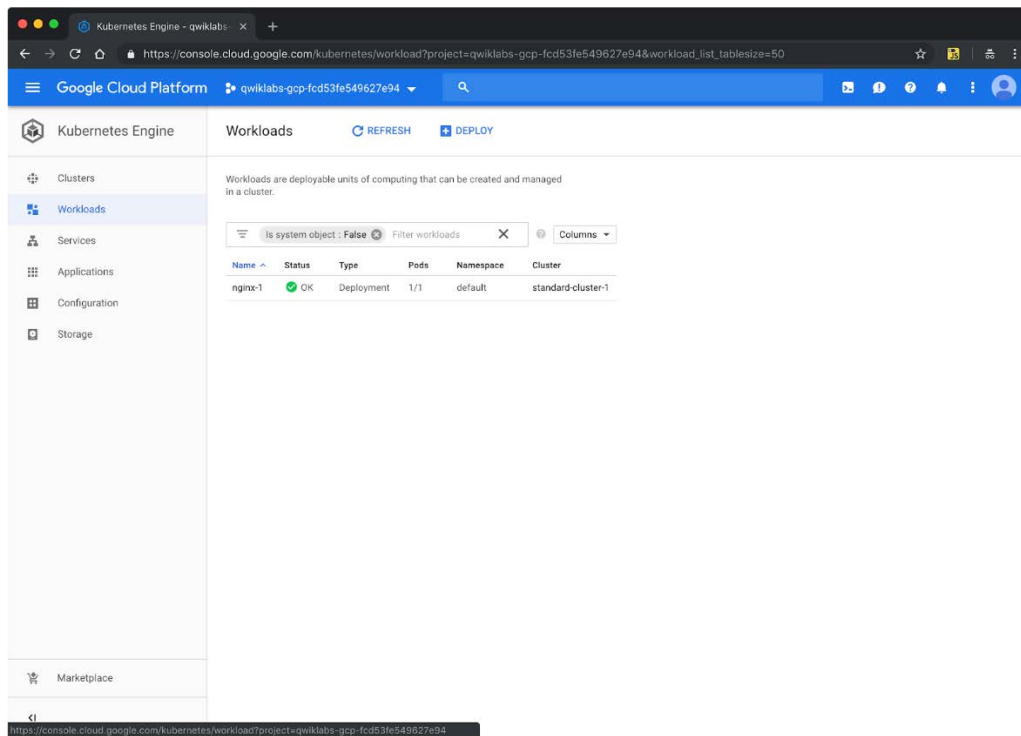
Deploy a sample nginx workload

Check my progress

Task 4. View details about workloads in the Google Cloud Console

In this task, you view details of your GKE workloads directly in the Google Cloud Console.

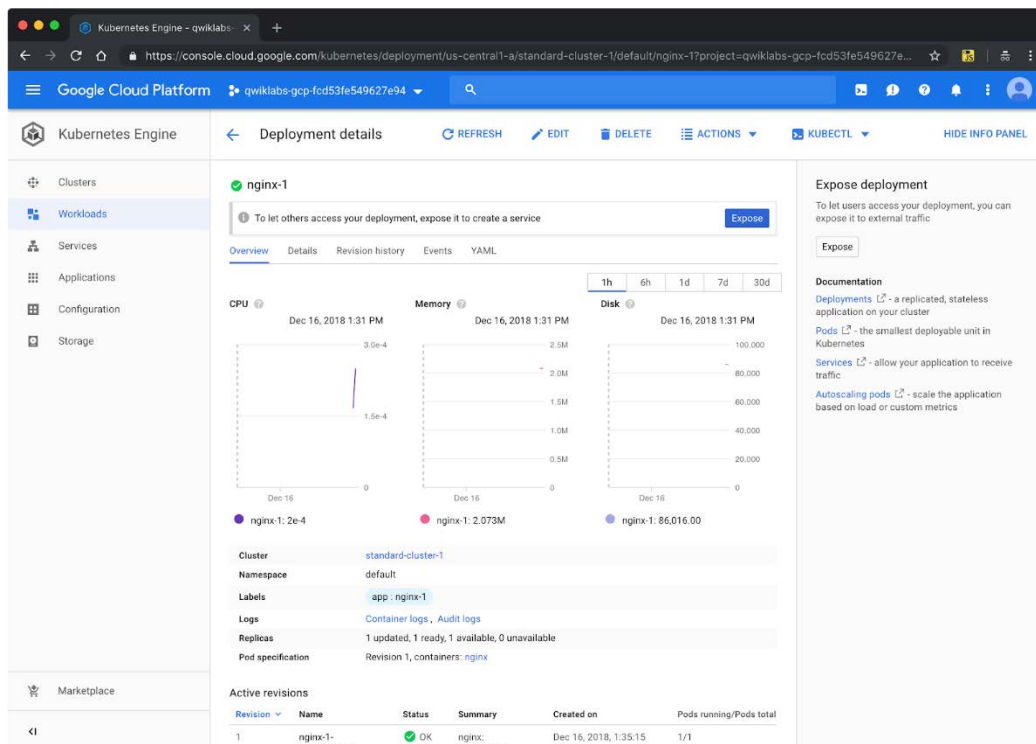
1. In the Google Cloud Console, on the **Navigation menu** () , click **Kubernetes Engine > Workloads**.



2. In the Google Cloud Console, on the **Kubernetes Engine > Workloads** page, click **nginx-1**.

You may see Pods (3/3) as the default deployment will start with three pods but will scale back to 1 after a few minutes. You can continue with the lab.

This displays the overview information for the workload showing details like resource utilization charts, links to logs, and details of the Pods associated with this workload.



3. In the Google Cloud Console, click the **Details** tab for the **nginx-1** workload. The Details tab shows more details about the workload including the Pod specification, number and status of Pod replicas and details about the horizontal Pod autoscaler.

The screenshot shows the Google Cloud Platform console interface. On the left, the 'Kubernetes Engine' sidebar is visible with 'Workloads' selected. The main content area displays the 'Deployment details' for 'nginx-1'. The 'Details' tab is active, showing a table of deployment metadata and a 'Pod specification' section. The deployment was created on Dec 16, 2018, at 1:35:15 PM. It has 1 updated, 1 ready, 1 available, and 0 unavailable replicas. The pod specification shows a single revision with labels 'app: nginx-1', restart policy 'Always', and container 'nginx'. A horizontal pod autoscaler is also configured with a name 'nginx-1-hpa', 1 min replica, and 5 max replicas, with a metric of 'CPU Utilization'. On the right, there is an 'Expose deployment' section with an 'Expose' button and a 'Documentation' section with links to 'Deployments', 'Pods', 'Services', and 'Autoscaling pods'.

Cluster	standard cluster-1
Namespace	default
Created	Dec 16, 2018, 1:35:15 PM
Labels	app: nginx-1
Annotations	deployment.kubernetes.io/revision: 1
Replicas	1 updated, 1 ready, 1 available, 0 unavailable
Label selector	app = nginx-1
Update strategy	Rolling update, Max unavailable: 1, Max surge: 1
Min time ready before available	0 s
Progress deadline	600 s
Revision history limit	10

Revision	1
Labels	app: nginx-1
Restart policy	Always
Containers	nginx

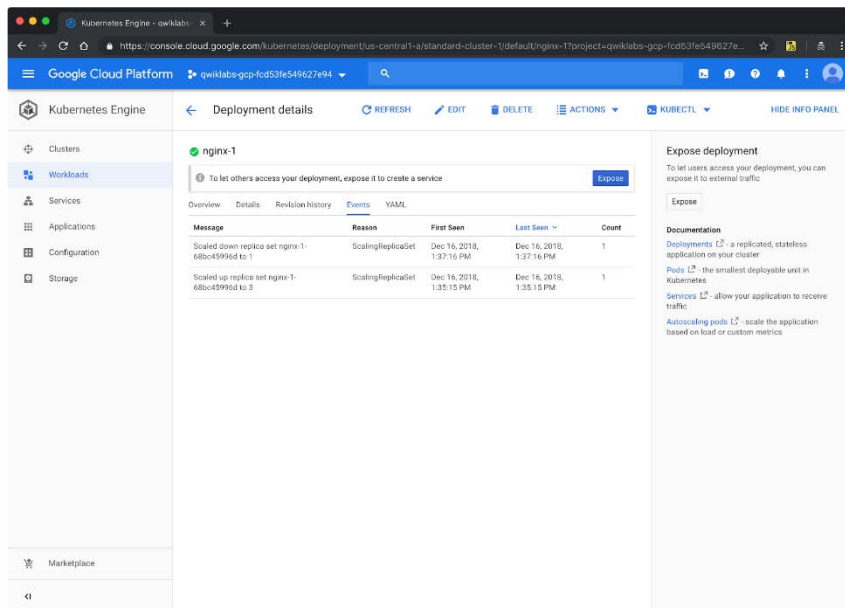
Name	nginx-1-hpa
Min replicas	1
Max replicas	5
Metric	CPU Utilization

4. Click the **Revision History** tab. This displays a list of the revisions that have been made to this workload.

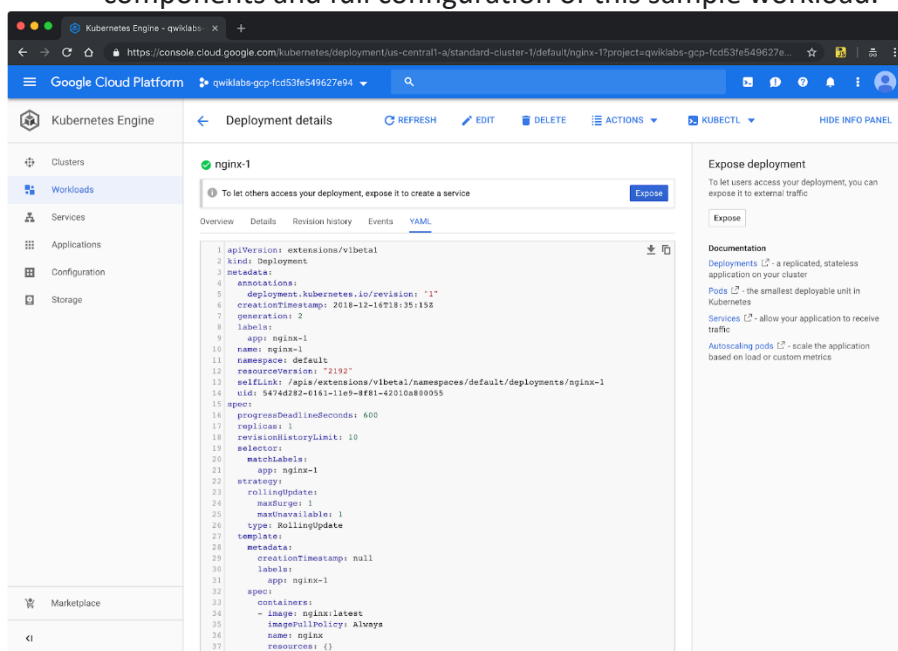
The screenshot shows the Google Cloud Platform console interface with the 'Revision History' tab selected for the 'nginx-1' deployment. The 'Revision History' tab displays a table with one revision. The revision has a name 'nginx-1-68bc48936d', a summary 'nginx: nginx latest', and was created on Dec 16, 2018, at 1:35:15 PM. The 'Expose deployment' and 'Documentation' sections on the right are also visible.

Revision	Name	Summary	Created on
1	nginx-1-68bc48936d	nginx: nginx latest	Dec 16, 2018, 1:35:15 PM

5. Click the **Events** tab. This tab lists events associated with this workload.



6. And then the **YAML** tab. This tab provides the complete YAML file that defines this components and full configuration of this sample workload.



7. Still in the Google Cloud Console's **Details** tab for the **nginx-1** workload, click the **Overview** tab, scroll down to the **Managed Pods** section and click the name of one of the Pods to view the details page for that Pod.

The screenshot shows the 'Deployment details' page for a Kubernetes deployment named 'nginx-1'. The page is divided into several sections:

- Cluster:** standard-cluster-1
- Namespace:** default
- Labels:** app: nginx-1
- Logs:** Container logs, Audit logs
- Replicas:** 1 updated, 1 ready, 1 available, 0 unavailable
- Pod specification:** Revision 1, containers: nginx
- Active revisions:** A table showing the deployment history. The current revision is 1, with a status of 'OK' and a summary of 'nginx: nginx latest'. It was created on Dec 16, 2018, at 1:35:15 PM, and has 1/1 pods running.
- Managed pods:** A table showing the pods managed by the deployment. The current pod is 'nginx-1-68bc45996d-mng5x', which is 'Running' and has 0 restarts. It was created on Dec 16, 2018, at 1:35:15 PM.
- Services:** No matching services.
- Autoscaler:** Min/max replicas: 1/5, Metric: CPU Utilization, Current/Target value: 0%/80%.

On the right side, there is an 'Expose deployment' section with an 'Expose' button, and a 'Documentation' section with links to 'Deployments', 'Pods', and 'Services'.

Note:

The default deployment will start with three pods but will scale back to 1 after a few minutes so you may need to refresh the Overview page to make sure you have a valid Pod to inspect.

- The Pod Details page provides information on the Pod configuration and resource utilization and the node where the Pod is running.

The screenshot shows the 'Pod details' page for a specific pod named 'nginx-1-68bc45996d-mng5x'. The page is divided into several sections:

- Cluster:** standard-cluster-1
- Namespace:** default
- Created:** Dec 16, 2018, 1:35:15 PM
- Labels:** app: nginx-1, pod-template-hash: 2467015528
- Annotations:** kubernetes.io/init-ranger: LimitRanger plugin set: cpu request for container nginx
- Stackdriver logs:** Container logs, Audit logs
- Node:** gke-standard-cluster-1-default-pool-7cfba57-m4nf
- Termination grace period:** 30 sec
- Restart policy:** Always
- Phase:** Running
- Start time:** Dec 16, 2018, 1:35:15 PM
- Conditions:** Initialized: True, Ready: True, PodScheduled: True
- Controllers:** Replica Set: nginx-1-68bc45996d

At the top, there are three graphs showing resource utilization over time (1 PM to 1:45 PM):

- CPU:** Shows a sharp spike to approximately 3.0e-4 at 1:35 PM.
- Memory:** Shows a steady increase from 0 to approximately 2.5M.
- Disk:** Shows a steady increase from 0 to approximately 100,000.

- In the **Pod details** page, you can click the Events and Logs tabs to view event details and links to container logs in Cloud Operations.

Google Cloud Platform | qwiklabs-gcp-03-cbbd6d8658f1 | Search products and resources

Kubernetes Engine | Pod details | REFRESH | EDIT | DELETE | EXPOSE | KUBECTL

Clusters | Workloads | Services & Ingress | Applications | Configuration | Storage | Object Browser | Migrate to containers

nginx-1-76949974bb-r6q52

DETAILS | EVENTS | LOGS | YAML

Message	Reason	First Seen	Last Seen ↓	Count
Started container nginx-1	Started	Sep 8, 2020, 6:40:09 PM	Sep 8, 2020, 6:40:09 PM	1
Created container nginx-1	Created	Sep 8, 2020, 6:40:09 PM	Sep 8, 2020, 6:40:09 PM	1
Successfully pulled image "nginx:latest"	Pulled	Sep 8, 2020, 6:40:07 PM	Sep 8, 2020, 6:40:07 PM	1
Pulling image "nginx:latest"	Pulling	Sep 8, 2020, 6:40:03 PM	Sep 8, 2020, 6:40:03 PM	1
Successfully assigned default/nginx-1-76949974bb-r6q52 to gke-standard-cluster-1-default-pool-ac6681a4-k2k1	Scheduled	Sep 8, 2020, 6:40:03 PM	Sep 8, 2020, 6:40:03 PM	1

Google Cloud Platform | qwiklabs-gcp-03-cbbd6d8658f1 | Search products and resources

Kubernetes Engine | Pod details | REFRESH | EDIT | DELETE | EXPOSE | KUBECTL

Clusters | Workloads | Services & Ingress | Applications | Configuration | Storage | Object Browser | Migrate to containers

nginx-1-76949974bb-r6q52

DETAILS | EVENTS | LOGS | YAML

You can view this cluster's logs in Cloud Logging.

[View all logs in Cloud Logging](#) | [Container logs](#) | [Audit logs](#)

10. Click the **YAML** tab to view the detailed YAML file for the Pod configuration.

Google Cloud Platform | qwiklabs-gcp-fcd53fe549627e94 | Search products and resources

Kubernetes Engine | Pod details | REFRESH | EDIT | DELETE | EXPOSE | KUBECTL

Clusters | Workloads | Services | Applications | Configuration | Storage | Marketplace

nginx-1-68bc45996d-mng5x

Details | Events | Logs | **YAML**

```

1 apiVersion: v1
2 kind: Pod
3 metadata:
4   annotations:
5     kubernetes.io/limit-ranger: 'LimitRanger plugin set: cpu request for container
6     nginx'
7   creationTimestamp: 2018-12-16T18:35:15Z
8   generateName: nginx-1-68bc45996d-
9   labels:
10    app: nginx-1
11    pod-template-hash: '2467015528'
12   name: nginx-1-68bc45996d-mng5x
13   namespace: default
14   ownerReferences:
15    - apiVersion: extensions/v1beta1
16      blockOwnerDeletion: true
17      controller: true
18      kind: ReplicaSet
19      name: nginx-1-68bc45996d
20      uid: 5476abe6-0161-11e9-8f81-42010a800055
21   resourceVersion: '1952'
22   selfLink: /api/v1/namespaces/default/pods/nginx-1-68bc45996d-mng5x
23   uid: 5476c98-0161-11e9-8f81-42010a800055
24 spec:
25   containers:
26    - image: nginx:latest
27      imagePullPolicy: Always
28      name: nginx
29      resources:
30        requests:
31          cpu: 100m
32      terminationMessagePath: /dev/termination-log
33      terminationMessagePolicy: File
34      volumeMounts:
35        - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
36          name: default-token-57sww
37          readOnly: true
38      dnsPolicy: ClusterFirst
39      nodeName: gke-standard-cluster-1-default-pool-7c7bba57-m4nf
40      restartPolicy: Always
41      schedulerName: default-scheduler

```