

Lab 6

Examining Billing Data with BigQuery

Overview

In this lab, you learn how to use BigQuery to analyze billing data.


Objectives

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

- Sign in to BigQuery from the GCP Console
- Create a dataset
- Create a table
- Import data from a billing CSV file stored in a bucket
- Run complex queries on a larger dataset

Task 1: Use BigQuery to import data

Sign in to BigQuery and create a dataset

- In the GCP Console, on the **Products & Services** menu () , click **BigQuery**.
- Re-enter the Qwiklabs-provided student password.
- Click **Sign in**.
- If prompted to **Check your account recovery options**, leave all fields blank and click **Done**.

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.

- In the left pane, click on the down arrow next to your Project ID (starts with qwiklabs) and click **Create new dataset**.
- Specify the following:

Property	Value (type value or select option as specified)
Dataset ID:	imported_billing_data
Data location:	US
Data expiration:	In 1 days.

- Click **OK**. You should see **imported_billing_data** in the left pane.

Create a table and import

- Point to **imported_billing_data**, and then click **+** to create a new table.
- For **Source Data**, specify the following, and leave the remaining settings as their defaults:

Property	Value
----------	-------

	(type value or select option as specified)
Source Data:	Create from source
Location:	Google Cloud Storage
gs://	gs://cloud-training/archinfra/export-billing-example.csv
File format:	CSV

- For **Destination Table**, 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

- For **Schema**, click **Automatically detect**.
- Click **Create Table**. After the job is completed, the table appears below the dataset in the left pane.

Task 2: Examine the table

- 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.

- Click **Details**. As you can see in **Number of Rows**, this is a relatively small table with 44 rows.
- Click **Preview**.
- Locate the row that has the **Description**: Network Internet Ingress from EMEA to Americas.

What was the total consumption and units consumed?

Measurement1_Total_Consumption = 9,738,199

Measurement_Units = bytes

- Scroll to the **Cost** column.

The cost was 0.0, so with an ingress of 9.7 Mbytes, traffic from EMEA to the Americas had no charge.

- Locate the row that has the **Description**: Network Internet Egress from Americas to China.

Can you interpret the information?

5,542 bytes exited the Americas and was transferred to China at a charge of 1.0E-6.

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.

The query must specify the data using the following format, in which all the items are specified inside square brackets:

[DatasetID.TableID.Field]

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.

- In the left pane, click **Compose query**.
- For **New Query**, paste the following:

```
SELECT *  
FROM [imported_billing_data.sampleinfotable]  
WHERE (imported_billing_data.sampleinfotable.Cost > 0);
```

This query says select all (44) records where Cost is greater than 0.

- Click **Run query**.

How many rows involved non-zero charges?

The table shows 20 rows and they all have non-zero charges.

Task 4: Use larger data

In the next activity, you use BigQuery to analyze a sample dataset with 22,537 lines of billing data.

- For **New Query**, paste the following:

```
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]
```

- Click **Run query**. Verify that the resulting table has 22,537 lines of billing data.
- To find the first 100 records where there were changes, for **New Query**, paste the following:

```
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 ([cloud-training-prod-bucket.arch_infra.billing_data.cost] > 0)  
LIMIT 100
```

- Click **Run query**.
- To find all charges that were more than 3 dollars, for **New Query**, paste the following:

```
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 ([cloud-training-prod-bucket.arch_infra.billing_data.cost] > 3)
```

- Click **Run query**.
- To find the product with the most records in the billing data, for **New Query**, paste the following:

```
SELECT product, COUNT(*)  
FROM [cloud-training-prod-bucket.arch_infra.billing_data]  
GROUP BY product
```

- Click **Run query**. Cloud Pub/Sub has 10,271 records.
- To find the most frequently used product costing more than 1 dollar, for **New Query**, paste the following:

```
SELECT product, COUNT(*)  
FROM [cloud-training-prod-bucket.arch_infra.billing_data]  
WHERE ([cloud-training-prod-bucket.arch_infra.billing_data.cost] > 1)
```

- Click **Run query**. Compute Engine has 17 charges costing more than 1 dollar.
- To find the most commonly charged unit of measure, for **New Query**, paste the following:

```
SELECT usage_unit, COUNT(*)
FROM [cloud-training-prod-bucket.arch_infra.billing_data]
GROUP BY usage_unit
```

- Click **Run query**. Requests were the most commonly charged unit of measure with 6,539 requests.
- To find the product with the highest aggregate cost, for **New Query**, paste the following:

```
SELECT product, SUM(cost)
FROM [cloud-training-prod-bucket.arch_infra.billing_data]
GROUP BY product
ORDER BY f0_DESC
```

- Click **Run query**. Compute Engine has an aggregate cost of \$112.02.

Task 5: Review

In this lab, you imported billing data into BigQuery that had been generated as a CSV file. You ran a simple query on the file. Then you accessed a shared dataset containing more than 22,000 records of billing information. You ran a variety of queries on that data to explore how you can use BigQuery to ask and answer questions by running queries.

Cleanup

- In the **Cloud Platform Console**, sign out of the Google account.
- Close the browser tab.

Last Updated: 2018-01-10

End your lab

Lab 7

Resource Monitoring (Stackdriver)

Overview

In this lab, you learn how to use Stackdriver Monitoring to gain insight into applications that run on Google Cloud Platform.

Objectives


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

- Enable Stackdriver Monitoring
- Add charts to dashboards
- Create alerts with multiple conditions
- Create resource groups
- Create uptime checks

Task 1: Create resources to monitor

You need to create some resources in a project before you can monitor them with Stackdriver.

Create a few Nginx instances using a Bitnami Nginx Stack image, which includes a complete PHP, MySQL and Nginx development environment.

- In the GCP Console, to open Cloud Shell, click **Activate Google Cloud Shell** (). If prompted, click **Start Cloud Shell**.
- In Cloud Shell, paste and run the following commands to create a few instances:

```
for i in {1..3}; \
do \
  gcloud compute instances create "nginxstack-$i" \
  --machine-type "f1-micro" \
  --tags nginxstack-tcp-443,nginxstack-tcp-80 \
  --zone us-central1-f \
  --image "https://www.googleapis.com/compute/v1/projects/bitnami-launchpad/global/images/bitnami-nginxstack-1-10-2-0-linux-debian-8-x86-64" \
  --boot-disk-size "200" --boot-disk-type "pd-standard" \
  --boot-disk-device-name "nginxstack-$i"; \
done
```


- Run the following command to create a firewall rule to allow external traffic to the instances:

```
gcloud compute firewall-rules create nginx-firewall \
--allow tcp:80,tcp:443 \
--target-tags nginxstack-tcp-80,nginxstack-tcp-443
```

Task 2: Create a Stackdriver account

To use Stackdriver Monitoring with your project, do the following:

Launch Stackdriver Monitoring

- In the GCP Console, on the **Products & Services** menu () , click **Monitoring**.
- Click **Log in with Google**.
- Click the qwiklabs-generated student account to log in.

Configure Stackdriver to monitor the project

- Click **Create Account**.
- Click **Continue**.
- Click **Skip AWS Setup**.
- Click **Continue**.
- Select **No reports**, and then click **Continue**.
- Click **Launch monitoring**.
- If prompted, click **Continue with the trial**.

You should now see the Stackdriver Monitoring console. The information on the console varies depending on the resources you are monitoring.

Stackdriver performs an initial collection task on your resources. Please wait 2 minutes before proceeding with the next steps.

Task 3: Custom dashboards

Create a dashboard

- In the left pane, click **Dashboards > Create Dashboard**.
- Click **Untitled Dashboard**, type **Archinfra Dashboard**, and press **ENTER**.

Add a chart

- Click **Add Chart**.
- For **Title**, give your chart a name (you can revise this before you save based on the selections you make).
- For **Find resource type and metric**, select **GCE VM Instance**.
- 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.

- Click **Show more options** and explore using an aggregate function.
- Click **Filter** and explore the various options.
- Click **View Options** and explore adding a Threshold or changing the **Chart mode**.
- 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**.

- In the left pane, click **Resources > Metrics Explorer**.
- For **Find resource type and metric**, type a metric or resource name.
- 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 4: Alerting policies

Create an alert and add the first condition

- In the left pane, click **Alerting > Create a Policy**.
- Click **Add Condition**.
- For **Metric Threshold**, click **Select**.
- For **Resource Type**, select **Instance (GCE)**.

If you cannot locate the **Instance (GCE)** resource type, you might have to refresh the page.

- For **Applies to**, select **Single**.
- Choose one of your instances as the resource that will be part of the target.
- For **If metric**, click a metric you are interested in evaluating, such as **CPU (agent)**.
- For **Condition**, click **above**.
- 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**.
- Click **Save Condition**.

Add a second condition

- Click **+ Add Another Condition**.
- Repeat the steps above to specify the second condition for this policy. For example, repeat the condition for a different instance. Click **Save Condition**.
- In **Policy Triggers**, for **Trigger when**, click **ALL conditions are met**.

Configure notifications and finish the alerting policy

- In **Notifications**, click **Add Notification**.
 - Select **Email** as the notification channel, and enter an email address.
 - Skip the Documentation step.
 - For **Name this policy**, type a name for the policy.
- Policy names are used as subjects in notification emails, so use that to your advantage.
- Click **Save Policy**.

Task 5: Resource groups

- In the left pane, click **Groups > Create Group**.
- Enter a name for the group. For example: **GCE Central**
- Select **Region** in the dropdown populated with **Name**.
- For **region**, select **gce us-central1**.
- Click **Save Group**.
- Review the dashboard Stackdriver created for your group.

Task 6: Uptime monitoring

- In the Group Dashboard from the previous task, for **Uptime Checks**, click **Add**.
- Type a title for the uptime check.
- For **Check Type**, click **HTTP**.
- For **Resource Type**, click **Instance**.
- For **Check every**, click **5 minutes**.
- Click **Save**.

If the **Save** button is grayed out, you might have to refresh the page.

- Click **No thanks**.

Task 7: Review

In this lab, you learned how to:

- Monitor your projects
- Create a Stackdriver account
- Create alerts with multiple conditions
- Add charts to dashboards
- Create resource groups
- Create uptime checks for your services

Cleanup

- In the **Cloud Platform Console**, sign out of the Google account.
- Close the browser tab.

Last Updated: 2018-03-27

End your lab

Error Reporting and Debugging (Stackdriver)

Overview

In this lab, you learn how to use Stackdriver Error Reporting and integrated Stackdriver Debugger.

Objectives

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

- Launch a simple Google App Engine application
- Introduce an error into the application
- Explore Stackdriver Error Reporting
- Use Stackdriver Debugger to identify the error in the code
- Fix the bug and monitor in Stackdriver

Task 1: Create an application

Get and test the application

- In the GCP Console, launch Cloud Shell by clicking **Activate Google Cloud Shell** . If prompted, click **Start Cloud Shell**.
- 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/* .
```

- To run the application using the local development server in Cloud Shell, run the following command:

```
dev_appserver.py .
```

- In Cloud Shell, click **Web Preview > Preview on port 8080** to view the application. You may have to collapse the **Products & services** pane to access the **Web Preview** icon.

A new browser window opens to the localhost and displays the message **Hello, World!**

- In Cloud Shell, press **Ctrl+C** to exit the development server.

Deploy the application to App Engine

- To deploy the application to App Engine, run the following command:

```
gcloud app deploy app.yaml
```
- If prompted for a region, enter the number corresponding to a region.
- When prompted, type **Y** to continue.
- 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.

- If needed, press **Ctrl+C** to exit the development mode.

Introduce an error to break the application

- 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.

- 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
```

- 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

- 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.

- 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.

- If needed, press **Ctrl+C** to exit development mode.
- Leave Cloud Shell open.

Task 2: Explore Stackdriver Error Reporting

View Error Reporting and trigger additional errors

- In the GCP Console, on the **Products & Services** menu () click **Error Reporting**.

You should see an error regarding the failed import of webapp22.

- Click **Auto reload**.
- 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.

- 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.

View details and identify the cause

- 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**.

- For **Stack trace sample**, click **Parsed**. This opens the Stackdriver Debugger, showing the error in the code!

View the logs and fix the error

- 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.
- 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.
- On the **Stackdriver Error Reporting** page, ensure that **Auto Reload** is enabled to watch the addition of new errors.
- In Cloud Shell, fix the error by running the following command:

```
sed -i -e 's/webapp22/webapp2/' main.py
```

- To re-deploy the application to App Engine, run the following command:

```
gcloud app deploy app.yaml --quiet
```
- 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.

- On the **Stackdriver 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 Stackdriver Error Reporting to identify the issue, and then analyzed the problem, finding the root cause using Stackdriver Debugger. You modified the code to fix the problem, and then saw the results in Stackdriver.

Cleanup

- In the **Cloud Platform Console**, sign out of the Google account.
- Close the browser tab.

Last Updated: 2018-03-27

End your lab

Lab 8

Virtual Private Networks (VPN)

Overview

In this lab, you create two networks in separate regions and establish VPN tunnels between them such that a VM in one network can ping a VM in the other network over its internal IP.

Objectives


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

- Create two custom networks and associated subnetwork
- Create VPN gateways in each network
- Establish static routes to enable the gateways to pass traffic
- Configure static routes to pass traffic to the VPN gateway
- Establish firewall rules to enable ICMP and SSH traffic

Task 1: Create the networks

Create two custom networks, with subnets, and start micro VMs in each.

Create the first network

- In the GCP Console, on the **Products & Services** menu () , click **VPC network** > **VPC networks**.
- Click **Create VPC network**.
- Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	vpn-network-1
Description	Enter an optional description
Subnet creation mode	Custom
Name	subnet-a
Region	us-east1
IP address range	10.5.4.0/24

- Click **Create**.

Create the second network

For the second network, choose a different region than the first network.

- Click **Create VPC network**.
- Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	vpn-network-2
Description	Enter an optional description
Subnet creation mode	Custom
Name	subnet-b

Region	europe-west1
IP address range	10.1.3.0/24

- Click **Create**.

Task 2: Create the utility VMs

Create the first instance

The first VM is created in the same region as **vpn-network-1**.

- On the **Products & Services** menu, click **Compute Engine > VM instances**.
- Click **Create**.
- Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	server-1
Zone	us-east1-b
Machine type	micro (1 shared vCPU)

- Click **Management, disks, networking, SSH keys**.
- 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	vpn-network-1
Subnetwork	subnet-a

- Click **Done**.
- Click **Create**.

Create the second instance

The second VM is created in the same region as **vpn-network-2**.

- In the Console, navigate to **Products & services > Compute Engine > VM instances**.
- Click **+ CREATE INSTANCE**.
- Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	server-2
Zone	europe-west1-b
Machine type	micro (1 shared vCPU)

- Click **Management, disks, networking, SSH keys**.
- 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	vpn-network-2
Subnetwork	subnet-b

- Click **Done**.
- Click **Create**.

Task 3: Create the firewall rules

Allow ICMP and SSH into each network.

Allow traffic to vpn-network-1

- On the **Products & Services** menu, click **VPC network > Firewall rules**.
- Click **Create firewall rule**.
- Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	allow-icmp-ssh-network-1
Network	vpn-network-1
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 Type: icmp; tcp:22

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

- Click **Create**.

Allow traffic to vpn-network-2

- Click **Create firewall rule**.
- Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	allow-icmp-ssh-network-2
Network	vpn-network-2
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 Type: icmp; tcp:22

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

- Click **Create**.

Task 4: Verify network connectivity

You should be able to ping the **external IP** of server-2, but not its **internal IP**.

Test connectivity from server-1 to server-2

- On the **Products & Services** menu, click **Compute Engine > VM instances**.
- Note the external and internal IP addresses for **server-2**.
- For **server-1**, click **SSH** to launch a terminal and connect.
- To test connectivity to server-2's external IP, run the following command, replacing server-2's external IP with the value noted earlier:

```
ping -c 3 <Enter server-2's external IP here>
```

- To test connectivity to server-2's internal IP, run the following command, replacing server-2's internal IP with the value noted earlier:

```
ping -c 3 <Enter server-2's internal IP here>
```

You should see 100% packet loss when pingging the internal IP.

- Exit the SSH terminal.

Test connectivity from server-2 to server-1

- Note the external and internal IP addresses for **server-1**.
- For **server-2**, click **SSH** to launch a terminal and connect.
- To test connectivity to server-1's external IP, run the following command, replacing server-1's external IP with the value noted earlier:

```
ping -c 3 <Enter server-1's external IP here>
```

- To test connectivity to server-1's internal IP, run the following command, replacing server-1's internal IP with the value noted earlier:

```
ping -c 3 <Enter server-1's internal IP here>
```

You should see similar results.

- Exit the SSH terminal.

Why are we testing both **server-1** to **server-2** and **server-2** to **server-1**?

For the purposes of this lab, the path from subnet-a to subnet-b is not the same as the path from subnet-b to subnet-a. You are using one tunnel to pass traffic in each direction. And if both tunnels are not established, you won't be able to ping the remote server on its internal IP. The ping might reach the remote server, but the response can't be returned.


This makes it much easier to debug the lab during class. In practice, a single tunnel could be used with symmetric configuration. However, it is more common to have multiple tunnels or multiple gateways and VPNs for production work, because a single tunnel could be a single point of failure.

Task 5: Create and prepare the VPN gateways

Create the VPN gateways and do all the setup work to establish the VPN tunnels. You will be doing this from the command line using Cloud Shell. Cloud Shell is used instead of the GCP Console so you can learn about the available options and how they fit together. The GCP Console conceals much of the complexity.

Create two VPN gateways, one in each region. Create forwarding rules for EPS, UDP:500, and UDP:4500 for each gateway.

Project ID

- In the GCP Console, on the **Products & Services** menu, click **Home**.
- Note the Project ID; it is referred to as [PROJECT_ID] in the following steps.
- Click **Activate Google Cloud Shell** (). If prompted, click **Start Cloud Shell**.
- To verify that gcloud is configured to [PROJECT_ID], run the following command:

```
gcloud config list project
```

If the project ID is undefined or does not match [PROJECT_ID], update it using `gcloud config set project <Enter PROJECT_ID here>`

Set up the VPN for both networks

- In Cloud Shell, to create the **vpn-1** gateway, run the following command:

```
gcloud compute target-vpn-gateways \
create vpn-1 \
--network vpn-network-1 \
--region us-east1
```

- To create the **vpn-2** gateway, run the following command:

```
gcloud compute target-vpn-gateways \
```



```
create vpn-2 \
--network vpn-network-2 \
--region europe-west1
```

Reserve a static IP for each network

- To reserve a Static IP for the **vpn-1** gateway, run the following command:
`gcloud compute addresses create --region us-east1 vpn-1-static-ip`
- To view the Static IP for the **vpn-1** gateway, run the following command:
`gcloud compute addresses list`
- To store the Static IP for the **vpn-1** gateway, in an environment variable, run the following command, and replace the IP address with the address from the output of the last command:
`export STATIC_IP_VPN_1=<Enter IP address for vpn-1 here>`
- To reserve a Static IP for the **vpn-2** gateway, run the following command:
`gcloud compute addresses create --region europe-west1 vpn-2-static-ip`
- To view the Static IP for the **vpn-2** gateway, run the following command:
`gcloud compute addresses list`
- To store the Static IP for the **vpn-2** gateway, in an environment variable, run the following command, and replace the IP address with the address from the output of the last command:
`export STATIC_IP_VPN_2=<Enter IP address for vpn-2 here>`

Create forwarding rules for both vpn gateways

The forwarding rules forward traffic arriving on the external IP to the VPN gateway. It connects them together. Create three forwarding rules for the protocols necessary for VPN.

- To create ESP forwarding for **vpn-1**, run the following command:

```
gcloud compute \
forwarding-rules create vpn-1-esp \
--region us-east1 \
--ip-protocol ESP \
--address $STATIC_IP_VPN_1 \
--target-vpn-gateway vpn-1
```
- To create ESP forwarding for **vpn-2**, run the following command:

```
gcloud compute \
forwarding-rules create vpn-2-esp \
--region europe-west1 \
--ip-protocol ESP \
--address $STATIC_IP_VPN_2 \
--target-vpn-gateway vpn-2
```
- To create UDP500 forwarding for **vpn-1**, run the following command:

```
gcloud compute \
forwarding-rules create vpn-1-udp500 \
--region us-east1 \
--ip-protocol UDP \
--ports 500 \
--address $STATIC_IP_VPN_1 \
--target-vpn-gateway vpn-1
```
- To create UDP500 forwarding for **vpn-2**, run the following command:

```
gcloud compute \
forwarding-rules create vpn-2-udp500 \
--region europe-west1 \
--ip-protocol UDP \
```

```
--ports 500 \
--address $STATIC_IP_VPN_2 \
--target-vpn-gateway vpn-2
```

- To create UDP4500 forwarding for **vpn-1**, run the following command:


```
gcloud compute \
forwarding-rules create vpn-1-udp4500 \
--region us-east1 \
--ip-protocol UDP --ports 4500 \
--address $STATIC_IP_VPN_1 \
--target-vpn-gateway vpn-1
```

- To create UDP4500 forwarding for **vpn-2**, run the following command:

```
gcloud compute \
forwarding-rules create vpn-2-udp4500 \
--region europe-west1 \
--ip-protocol UDP --ports 4500 \
--address $STATIC_IP_VPN_2 \
--target-vpn-gateway vpn-2
```

Verify the external IP addresses and VPN gateways

They should be in use by the forwarding rules you just created.

- In the GCP Console, on the **Products & Services** menu () click **VPC network** > **External IP addresses**.
- Verify that both regions have an external IP address reserved and that all three forwarding rules are displayed in the **In use by** column.

Alternatively, you can reserve static addresses and set forwarding rules through this section of the GCP Console.

- In Cloud Shell, to verify the VPN gateways, run the following command:

```
gcloud compute target-vpn-gateways list
```

You should see the VPN gateways.

Task 6: Create tunnels

Create the tunnels between the VPN gateways. After the tunnels exist, create a static route to enable traffic to be forwarded into the tunnel. If this is successful, you can ping a local VM in one location on its internal IP from a VM in a different location.

- To create the tunnel for traffic from **Network-1** to **Network-2**, run the following command:

```
gcloud compute \
vpn-tunnels create tunnel1to2 \
--peer-address $STATIC_IP_VPN_2 \
--region us-east1 \
--ike-version 2 \
--shared-secret gcprocks \
--target-vpn-gateway vpn-1 \
--local-traffic-selector 0.0.0.0/0 \
--remote-traffic-selector 0.0.0.0/0
```

- To create the tunnel for traffic from **Network-2** to **Network-1**, run the following command:

```
gcloud compute \
vpn-tunnels create tunnel2to1 \
--peer-address $STATIC_IP_VPN_1 \
--region europe-west1 \
--ike-version 2 \
```

```
--shared-secret gcprocks \  
--target-vpn-gateway vpn-2 \  
--local-traffic-selector 0.0.0.0/0 \  
--remote-traffic-selector 0.0.0.0/0
```

- To verify that the tunnels are created, run the following command:

```
gcloud compute vpn-tunnels list
```

It may take a couple of minutes for the VPNs to connect to their peers. If the connection fails, it means something was entered incorrectly in the previous commands. Be very careful about spaces or copy-paste errors.

At this point, the gateways are connected and communicating. But there is no method to direct traffic from one subnet to the other. You must establish static routes.

Task 7: Create static routes

- To create a static route from **Network-1** to **Network-2**, run the following command:

```
gcloud compute \  
routes create route1to2 \  
--network vpn-network-1 \  
--next-hop-vpn-tunnel tunnel1to2 \  
--next-hop-vpn-tunnel-region us-east1 \  
--destination-range 10.1.3.0/24
```

- To create a static route from **Network-2** to **Network-1**, run the following command:

```
gcloud compute \  
routes create route2to1 \  
--network vpn-network-2 \  
--next-hop-vpn-tunnel tunnel2to1 \  
--next-hop-vpn-tunnel-region europe-west1 \  
--destination-range 10.5.4.0/24
```

Task 8: Verify VPN connectivity

Verify server-1 to server-2 connectivity

- In the GCP Console, on the **Products & Services** menu, click **Compute Engine > VM instances**.

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

- To test connectivity to **server-2**'s internal IP, run the following command:

```
ping -c 3 <insert server-2's internal IP here>
```

- Exit the **server-1** SSH terminal.

- For **server-2**, click **SSH** to launch a terminal and connect.

- To test connectivity to **server-1**'s internal IP, run the following command:

```
ping -c 3 <insert server-1's internal IP here>
```

Task 9: Review

You set up virtual private networking (VPN) between two subnets in different regions. This lab required you to perform most of the configuration from the command line. When you configure VPN using the GCP Console, many of the steps are automated. One purpose of this lab is to show you how to configure VPN manually, so that you will better understand what the GCP Console does automatically. This can help in troubleshooting a configuration.

Cleanup

- In the **Cloud Platform Console**, sign out of the Google account.

- Close the browser tab.

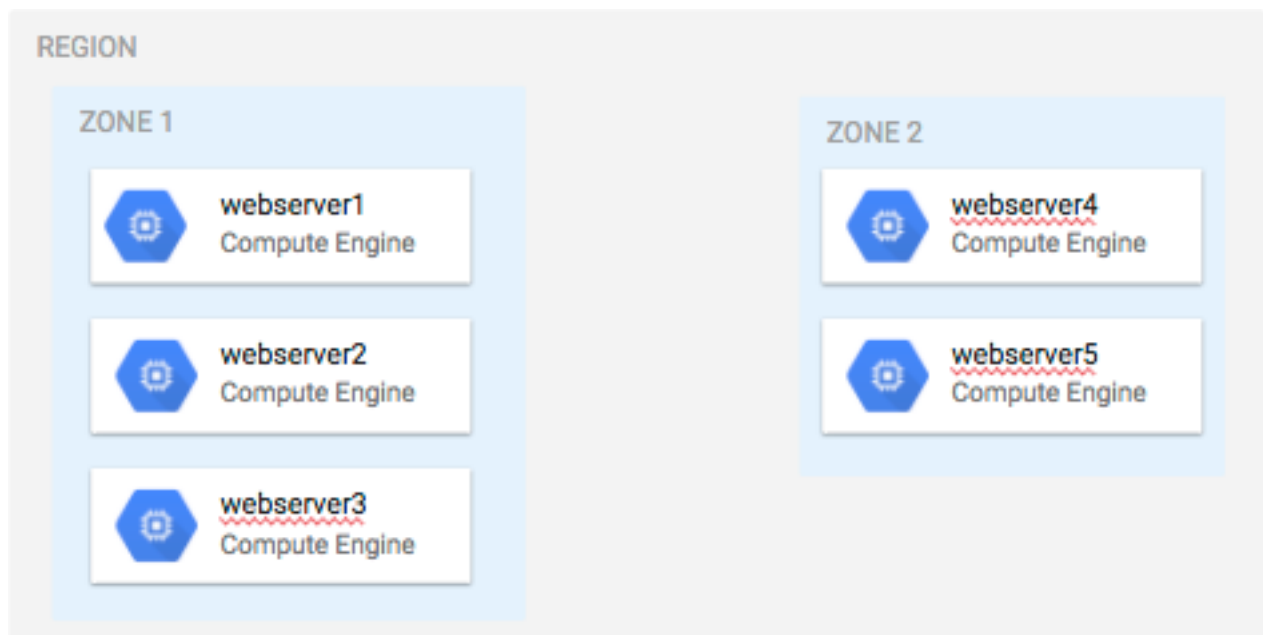
Last Updated: 2017-11-27

End your lab

Lab 9

Virtual Machine Automation and Load Balancing

Overview



In this lab, you exercise several features of virtual machines to create a pool of worker machines fed by a network load balancer. The pool provides distributed workload for scalable capacity and high availability, a common solution pattern. You also configure boot-time installation of software via a startup script and use metadata to both initiate the script and pass user-defined values to the script.

The Project ID is referred to later in this lab as [PROJECT_ID].

Choose a region and two zones within that region for this lab. They are identified as [YOUR_REGION], [YOUR_ZONE1], and [YOUR_ZONE2]. You create environment variables for these values later in the lab.

For more information, see:

Regions and Zones <https://cloud.google.com/compute/docs/regions-zones/regions-zones>

Load balancing <https://cloud.google.com/compute/docs/load-balancing/>

Network Load Balancing (external) <https://cloud.google.com/compute/docs/load-balancing/network/>

Internal Load Balancing <https://cloud.google.com/compute/docs/load-balancing/internal/>

Objectives


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

- Create a pool of VMs
- Configure an external load balancer to use the pool
- Place a load on the service and stop a VM to simulate an outage
- Launch two more VMs in a secondary zone
- Configure an internal load balancer

Test the internal load balancer

Task 1: Create three VMs using the startup script

Create the first VM

- In the GCP Console, on the **Products & Services** menu () , click **Compute Engine > VM instances**.
- Click **Create**.
- Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	webserver1
Zone	[YOUR_ZONE1]
Access scopes	Set access for each API For Storage , ensure that Read Only is set*
Firewall	Allow HTTP traffic

* This allows this VM to read from the Cloud Storage bucket during the boot process.

- Click **Management, disks, networking, SSH keys**.
- For **Metadata**, add the following key-value pair:

Property	Value (type value or select option as specified)
Key	startup-script-url
Value	gs://cloud-training/archinfra/mystartupscript

- Click **Add item**.
- For **Metadata**, add another key-value pair, this time with the unique name of your web server:

Property	Value (type value or select option as specified)
Key	my-server-id
Value	WebServer-1

- Click **Create**.
- On the **VM instances** page, for **webserver1**, click the **External IP**. This opens a new index page with the name "WebServer-1." If you get a "This site can't be reached" error, you might have to refresh the page: for all VMs created in this lab, it may take a minute or two before the web server starts serving content. Wait a minute and then refresh the page.

Create the second and third VMs

Create two more web servers. The only difference is the names and the second metadata value.

- In the GCP Console, click **Create instance**.
- Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	webserver2
Zone	[YOUR_ZONE1]
Access scopes	Set access for each API For Storage , ensure that Read Only is set*
Firewall	Allow HTTP traffic

* This allows this VM to read from the Cloud Storage bucket during the boot process.

- Click **Management, disks, networking, SSH keys**.

-

Property	Value (type value or select option as specified)
Key	startup-script-url
Value	gs://cloud-training/archinfra/mystartupscript

For **Metadata**, add the following key-value pair:

- Click **Add item**.
- For **Metadata**, add another key-value pair, this time with the unique name of your web server:

Property	Value (type value or select option as specified)
Key	my-server-id
Value	WebServer-2

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

Property	Value (type value or select option as specified)
Name	webserver3
Zone	[YOUR_ZONE1]
Access scopes	Set access for each API For Storage , ensure that Read Only is set*
Firewall	Allow HTTP traffic

* This allows this VM to read from the Cloud Storage bucket during the boot process.

- Click **Management, disks, networking, SSH keys**.
- For **Metadata**, add the following key-value pair:

Property	Value (type value or select option as specified)
Key	startup-script-url
Value	gs://cloud-training/archinfra/mystartupscript

- Click **Add item**.
- For **Metadata**, add another key-value pair, this time with the unique name of your web server:

Property	Value
----------	-------

	(type value or select option as specified)
Key:	my-server-id
Value:	WebServer-3

- Click **Create**.
- In the VM instances page, for **webserver2** and **webserver3**, click the external IP values. Verify that the unique names are being displayed on the home page. Again, if you get a "This site can't be reached" error, you might have to wait a minute and refresh the page.


Task 2: Configure external network load balancing

Add tags to identify the VMs for external network load balancing

- In the GCP Console, click **webserver1**.
- Click **Edit**.
- For **Network tags**, type **network-lb**. That is a lowercase "L" for load balancer.
- Click **Save**.
- Repeat steps 3–5 for **webserver2** and **webserver3**.

The load balancer has a frontend and a backend. The frontend consists of an external static IP address and a forwarding rule. The backend consists of a pool of VMs and a health check that is used to verify that the VMs are operational. You create these components in the next steps and configure them into the load balancer.

Reserve an external Static IP for use with the external load balancer

- On the **Products & Services** menu () , click **VPC network > External IP addresses**.
- Click **Reserve static address**.
- Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	network-lb-ip
Description	IP address to be used with network load balancer
Region	[YOUR_REGION]

- Click **Reserve**.

Create a health check

- On the **Products & Services** menu, click **Compute Engine > Health checks**.
- Click **Create a health check**.
- For **Name**, type **webserver-health**.
- Leave the remaining settings as their defaults, and click **Create**.

You will complete the setup of the external load balancer from the Cloud Shell command line and verify the configuration from the console.

- Click **Activate Google Cloud Shell** () to Open Cloud Shell. If prompted, click **Start Cloud Shell**.

Configure the external load balancer

- Choose a region and zone from the list of available regions and zones, by running these commands in Cloud Shell:

```
gcloud compute regions list
gcloud compute zones list
```

- In Cloud Shell, to create environments for your selected region and zones, run the following commands:

```
export MY_REGION=<Enter YOUR_REGION here>
export MY_ZONE1=<Enter YOUR_ZONE1 here>
export MY_ZONE2=<Enter YOUR_ZONE2 here>
```

- The target pool is named extloadbalancer and is connected to the health check you created in the previous step. Run the following command:

```
gcloud compute target-pools create extloadbalancer \
--region $MY_REGION --http-health-check webserver-health
```

- Add the three VMs, webserver1, webserver2, and webserver3, into the target pool extloadbalancer. Run the following command:

```
gcloud compute target-pools add-instances extloadbalancer \
--instances webserver1,webserver2,webserver3 \
--instances-zone=$MY_ZONE1
```

This completes the external load balancer backend configuration.

Create the forwarding rule

- In Cloud Shell, to get the reserved static IP address, run the following command:

```
gcloud compute addresses list
```


- To create an environment variable for the static external IP listed, run the following command, replacing [YOUR_STATIC_IP] with the value in the output of the last command:

```
export STATIC_EXTERNAL_IP=[YOUR_STATIC_IP]
```

- To complete the configuration of the forwarding rule, run the following command:

```
gcloud compute forwarding-rules create webserver-rule \
--region $MY_REGION --ports 80 \
--address $STATIC_EXTERNAL_IP --target-pool extloadbalancer
```

Verify the load balancing configuration in the GCP Console

- In the GCP Console, on the **Products & Services** menu () , click **VPC network** > **External IP addresses**. You should see that the static IP is now in use by a forwarding rule.
- On the **Products & Services** menu, click **Network services** > **Load balancing**. You should see the **extloadbalancer**.

Task 3: Test the solution

Send continuous traffic to the external load balancer and force an outage

- In Cloud Shell, to start sending repeated requests to the external load balancer, run the following command:

```
while true; do curl -m1 $STATIC_EXTERNAL_IP; done
```

You should see the webserver name changing in the results, thus proving that load balancing is working.

- Leave the test running.
- In the GCP Console, on the **Products & Services** menu, click **Compute Engine > VM instances**.
- Select **webserver1**, click **Delete**, and then click **Delete** to confirm.

For a minute, you'll see misses and then it stabilizes. Recall that this policy was set when you created the health check—how long to wait and how many failures constitutes an out-of-service VM. After a minute or so, the external load balancer shifts the load to the remaining servers.

You created a scalable and highly available solution. You could add additional VMs into the pool to increase capacity. In this example, you used an Apache web server for simplicity and also to help reveal how the infrastructure is working. However, this is a network tcp external load balancer solution, and you can use it with any other kind of traffic or software.

View the external load balancer in the GCP Console and end the test

- On the **Products & Services** menu, click **Network services > Load balancing**.
- Click **extloadbalancer** to see details. You should see that **webserver1** is listed, but marked as out of service.
- In Cloud Shell, press **Ctrl+C** to exit the test script.

Task 4: Configure internal load balancing

The following configuration can be done in the GCP Console or by using `gcloud`. For more information, see: <https://cloud.google.com/compute/docs/load-balancing/internal/#setting-up-internal-load-balancing>

A production system would normally use managed instance groups based on instance templates, but this setup is quicker for initial testing.

For more information, see:

Managed instance groups: <https://cloud.google.com/compute/docs/instance-groups/>

Instance templates: <https://cloud.google.com/compute/docs/instance-templates>

Previously, you created three web servers in the first zone (**webserver1**, **webserver2**, and **webserver3**). **Webserver2** and **webserver3** are still running. You modify **webserver2** and **webserver3**, and you create two more servers (**webserver4** and **webserver5**) in the second zone.

Modify webserver2 and webserver3

- On the **Products & Services** menu, click **Compute Engine > VM instances**.
- Click **webserver2** to access the VM instance details.
- Click **Edit**.
- For **Network tags**, remove **network-lb** and add **int-lb**.
- Click **Save**.
- Repeat steps 2–5 for **webserver3**.

Launch two new servers in the second zone

- In Cloud Shell, to launch **webserver4**, run the following command:

```
gcloud compute instances create webserver4 \
  --image-family debian-9 \
```

```
--image-project debian-cloud \
--tags int-lb \
--zone $MY_ZONE2 \
--subnet default \
--metadata startup-script-url="gs://cloud-training/archinfra/mystartupscript",my-
server-id="WebServer-4"
```

- To launch webserver5, run the following command:

```
gcloud compute instances create webserver5 \
--image-family debian-9 \
--image-project debian-cloud \
--tags int-lb \
--zone $MY_ZONE2 \
--subnet default \
--metadata startup-script-url="gs://cloud-training/archinfra/mystartupscript",my-
server-id="WebServer-5"
```

Create an instance group for each zone and add the instances

- To create **ig1** in your first zone, run the following command:

```
gcloud compute instance-groups unmanaged create ig1 \
--zone $MY_ZONE1
```

- To add **webserver2** and **webserver3** to **ig1**, run the following command:

```
gcloud compute instance-groups unmanaged add-instances ig1 \
--instances=webserver2,webserver3 --zone $MY_ZONE1
```

- To create **ig2** in your second zone, run the following command:

```
gcloud compute instance-groups unmanaged create ig2 \
--zone $MY_ZONE2
```

- To add **webserver4** and **webserver5** to **ig2**, run the following command:

```
gcloud compute instance-groups unmanaged add-instances ig2 \
--instances=webserver4,webserver5 --zone $MY_ZONE2
```

Configure the load balancer

Internal load balancing spreads incoming connections based on the session affinity setting. If session affinity is not set, the load balancer spreads all connections across all available instances as evenly as possible, regardless of current load.

- To create a health check, run the following command:

```
gcloud compute health-checks create tcp my-tcp-health-check \
--port 80
```

- To create a backend service, run the following command:

```
gcloud compute backend-services create my-int-lb \
--load-balancing-scheme internal \
--region $MY_REGION \
--health-checks my-tcp-health-check \
--protocol tcp
```

- To add **ig1** to your backend service, run the following command:

```
gcloud compute backend-services add-backend my-int-lb \
--instance-group ig1 \
--instance-group-zone $MY_ZONE1 \
--region $MY_REGION
```

- To add **ig2** to your backend service, run the following command:

```
gcloud compute backend-services add-backend my-int-lb \
--instance-group ig2 \
--instance-group-zone $MY_ZONE2 \
```

```
--region $MY_REGION
```

- To create a forwarding rule, run the following command:

```
gcloud compute forwarding-rules create my-int-lb-forwarding-rule \
--load-balancing-scheme internal \
--ports 80 \
--network default \
--subnet default \
--region $MY_REGION \
--backend-service my-int-lb
```

- To configure a firewall rule to allow traffic to the load balancer and from the load balancer to the instances, run the following command:

```
gcloud compute firewall-rules create allow-internal-lb \
--network default \
--source-ranges 0.0.0.0/0 \
--target-tags int-lb \
--allow tcp:80,tcp:443
```

- To configure another firewall rule to allow health check probes from the health checker, run the following command:

```
gcloud compute firewall-rules create allow-health-check \
--network default \
--source-ranges 130.211.0.0/22,35.191.0.0/16 \
--target-tags int-lb \
--allow tcp
```

Create a standalone client

For testing purposes, create a standalone client instance in the same region as the load balancer.

- To create the client, run the following command:


```
gcloud compute instances create standalone-instance-1 \
--image-family debian-9 \
--image-project debian-cloud \
--zone $MY_ZONE1 \
--tags standalone \
--subnet default
```

- To create a firewall rule to allow SSH connections to the standalone client instance, run the following command:

```
gcloud compute firewall-rules create allow-ssh-to-standalone \
--network default \
--target-tags standalone \
--allow tcp:22
```

Delete the external IPs from your web servers

When you created the instances, they needed external IPs to download and install Apache. Now that Apache is installed, and because these instances are going to serve an internal load balancer, they do not need access to the internet, so you can remove the external IPs from the instances.

- In the GCP Console, on the **Products & Services** menu () , click **Compute Engine > VM instances**.
- Click **webserver2**.
- Click **Edit**.
- For **Network interfaces**, click the pencil to edit.
- For **External IP**, select **None**.

- Click **Done**.
- Click **Save**.
- Repeat steps 2–7 for **webserver3**, **webserver4**, and **webserver5**.
- In the left pane, click **VM instances**. Only **standalone-instance-1** should have an external IP.

View the internal load balancer status in the GCP Console

- On the **Products & Services** menu, click **Network services > Load balancing**.
- Click **my-int-lb** to see the details. You should see the configured backend with **ig1** and **ig2** displaying 2/2 healthy instances each.
- Note the IP address of the internal load balancer. It will be used in the next Task.

Task 5: Test the internal load balancer

- On the **Products & Services** menu, click **Compute Engine > VM instances**.
- For **standalone-instance-1**, click **SSH**.
- Store the IP address of the internal load balancer in an environment variable:


```
export ILB_IP=<Enter the IP address of the internal load balancer>
```

 - Curl the IP address of the load balancer several times:


```
for ((i=1;i<=10;i++)); do curl $ILB_IP; done
```

You should see the HTML output for the different backends with their web server id populated.

You can optionally delete one of the web servers and check the listed number of healthy instances in **my-int-lb**.

Task 6: Review

In this lab you created a pool of web servers and directed traffic to them through an external network load balancer, thus distributing work among the servers. Then you tested for high availability by shutting down one of the servers. You then launched two additional web servers in a secondary zone and configured an internal load balancer for work distribution and availability.

Cleanup

- 1 In the **Cloud Platform Console**, sign out of the Google account.
- 2 Close the browser tab.

Last Updated: 2018-06-14

End your lab

Lab 10

Autoscaling

Overview

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.

Autoscaling works by scaling your instance group in or out. That is, it adds more instances to your instance group when there is more load (scaling out) and removes instances when the need for instances is lowered (scaling in).


Objectives

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

- Create a custom image for a web server
- Create an instance template based on the custom image
- Create a managed instance group
- Create a load balancer
- Stress test the autoscaler

Task 1: Create a custom image for a web server

Create a VM

- In the GCP Console, on the **Products & Services** menu () , click **Compute Engine > VM instances**.
- Click **Create**.
- Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	webserver
Zone	Choose a zone
Machine type	micro (1 shared vCPU)
Firewall	Allow HTTP traffic Allow HTTPS traffic

- Click **Management, disks, networking, SSH keys**.
- Click **Disks**, and disable **Delete boot disk when instance is deleted**.
- Click **Create**.

Customize the VM

- For **webserver**, click **SSH** to launch a terminal and connect.
- To install Apache2, run the following commands:

```
sudo apt-get update  
sudo apt-get install -y apache2
```
- To start the apache server, run the following command:

```
sudo service apache2 start
```
- To enable SSL and restart the apache server, run the following commands:

```
sudo a2ensite default-ssl  
sudo a2enmod ssl  
sudo service apache2 restart
```
- In the GCP Console, for **webserver**, click the **External IP** address.
- Click through the warning to see the actual page. For example, in Chrome, click **Advanced**, and then click **Proceed to External IP Address**. The default page for the Apache2 server should be displayed.

In this test setup, the instance is using self-signed certificates. Therefore, you will see a warning in your browser the first time you access a page. Alternatively, you can copy the IP address and access the page in a new tab using `http://<External IP address>/`

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.

- In the webserver SSH terminal, set the service to start on boot:

```
sudo update-rc.d apache2 enable
```
 - In the GCP Console, select **webserver**, and then click **Reset**.
 - 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.
- For **webserver**, click the **External IP** address of the instance to verify that the Apache service is available. You should see the default page.
 - You can also check the server by connecting via SSH to the VM and entering the following command:

```
sudo service apache2 status
```
 - The result should show **Started The Apache HTTP Server**.

Prepare the disk to create a custom image

- On the VM instances page, click **webserver** to view the VM instance details and verify that **Delete boot disk when instance is deleted** is disabled.
- Return to the VM instances page, click **webserver**, and click **Delete**.
- In the confirmation dialog, click **Delete**.
- In the left pane, click **Disks** and verify that the **webserver** disk exists.

Create the custom image

- In the left pane, click **Images**.
- Click **Create image**.

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

Property	Value (type value or select option as specified)
Name	mywebserver1
Source	Disk
Source disk	webserver

- Click **Create**.

You have created a custom image from which multiple identical webserver can be started. The next step is to use that image to define an Instance Template that can be used in a managed instance group.

Task 2: Create an Instance Template based on the custom image

- In the left pane, click **Instance templates**.
- Click **Create instance template**.
- Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	webserver-template
Machine type	micro (1 shared vCPU)

- For **Boot disk**, click **Change**.
- Click **Custom images**.
- Select **mywebserver1**.
- Click **Select**.
- For **Firewall**, enable **Allow HTTP traffic** and **Allow HTTPS traffic**.
- Click **Create**.

You created an instance template from the custom image. Now you can use it in a Managed Instance Group.

Task 3: Create a managed instance group

- In the left pane, click **Instance groups**.
- 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	mywebserver-group
Location	Multi-zone
Region	Choose a region
Instance template	webserver-template
Autoscaling	On
Autoscale based on	HTTP load balancing usage
Maximum number of instances	5

- For **Health check**, select **Create a health check**.
- For **Name**, type **webserver-healthcheck**
- Click **Save and continue**.

- 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.
- Click **Create**. A warning tells you that there is no load balancer. That's OK; you are going to create and attach one to the Managed Instance Group in the next section.
- Click **OK**.
- In the left pane, click **VM instances**.
- Test the VM by clicking on the **External IP** address of the instance in the console.
- Click through the warning to see the actual page. For example, in Chrome, click **Advanced**, and then click **Proceed to External IP Address**.

In this test setup, the instance is using self-signed certificates. Therefore, you see a warning in your browser the first time you access this new External IP address. Alternatively, you can copy the IP address and access the page in a new tab using `http://<External IP address>/`

Task 4: Create a load balancer

- On the **Products & services** menu, click **Network services > Load balancing**.
- Click **Create load balancer**.
- In **HTTP(S) Load Balancing**, click **Start configuration**.
- Click **Frontend configuration**.
- For **Name**, type **mywebserver-frontend**
- Leave the remaining settings as their defaults, and click **Done**.
- Click **Backend configuration**.
- Click **Create or select a backend service & backend buckets > Backend services > Create a backend service**.
- For **Name**, type **mywebserver-backend**
- In **Backends > New backend**, for **Instance group**, click **mywebserver-group**.
- Leave the remaining settings as their defaults, and click **Done**.
- For **Health check**, click **webserver-healthcheck**.
- Click **Create**.
- Enter a name for your HTTP(S) load balancer: **webserver-load-balancer**
- Click **Create**.

Note: Creating the backend automatically set a **Host and path rule** to deliver all traffic to the backend.

- Click **webserver-load-balancer**.
- Find the **External IP** that was assigned to the Frontend, which is later referred to as **[YOUR_LB_IP]**.
- On the **Products & services** menu, click **Compute Engine > Instance groups**. mywebserver-group may show a red icon indicating that there is no backend attached to the group. It may take a minute or two for the backend configuration to register. Click Refresh, and it should change.

Now you should see a warning icon indicating that there is no traffic to the site yet. This is expected.

- Open a new browser tab or window and browse to the load balancer's IP using `http://[YOUR_LB_IP]/`. You should see the Apache default page.
- If you get a server error, wait 1 minute and refresh the page. There might be a delay

in the load balancer response.

Task 5: Stress test the Autoscaler

The entire configuration is working, as evidenced by the fact that you could browse to the load balancer's IP and view the default page on the web server. However, you need to see if the Autoscaler is working and will actually start new VMs in response to a load.

To test this you need software that can send repeated requests to a web server. Fortunately, free web server benchmarking software, called Apache Bench, is part of the Apache2 package.

That means that when you created the webserver custom image, you also installed and created the image with pre-installed software for a benchmark server.

- In the GCP Console, on the **Products & Services** menu, click **Compute Engine > VM instances**.
- Click **Create instance**.
- Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	stress-test
Zone	Choose a zone
Machine type	micro (1 shared vCPU)

- For **Boot Disk**, click **Change**.
- Click **Custom images**.
- Select **mywebserver1**.
- Click **Select**, and then click **Create**.
- On the **VM instances** page, for **stress-test**, click **SSH** to launch a terminal and connect.
- To create an environment variable for your load balancer IP address, run the following command:

```
export LB_IP=<Enter [YOUR_LB_IP] here>
```
- To place a load on the load balancer, run the following command:

```
ab -n 50000 -c 1000 http://$LB_IP/
```
- In the GCP Console, in the left pane, click **Instance groups**.
- Click **mywebserver-group**. Verify that new instances have been created.

Feel free to repeat the command a couple of times to create 5 instances (maximum number of instances defined in the Instance Group).

Task 6: Review

In this lab you set up an HTTP(S) load balancer with autoscaling and verified that it was working. To do this, you first created a VM, then you customized it by installing software and changing a configuration setting (making Apache start on boot). You used the custom image in an instance template, and then used the image template to make a managed instance group. After all the backend and frontend parts were connected together, you stress-tested the system and triggered autoscaling using Apache bench.

Cleanup

- In the **Cloud Platform Console**, sign out of the Google account.
- Close the browser tab.

Last Updated: 2018-03-27
End your lab