

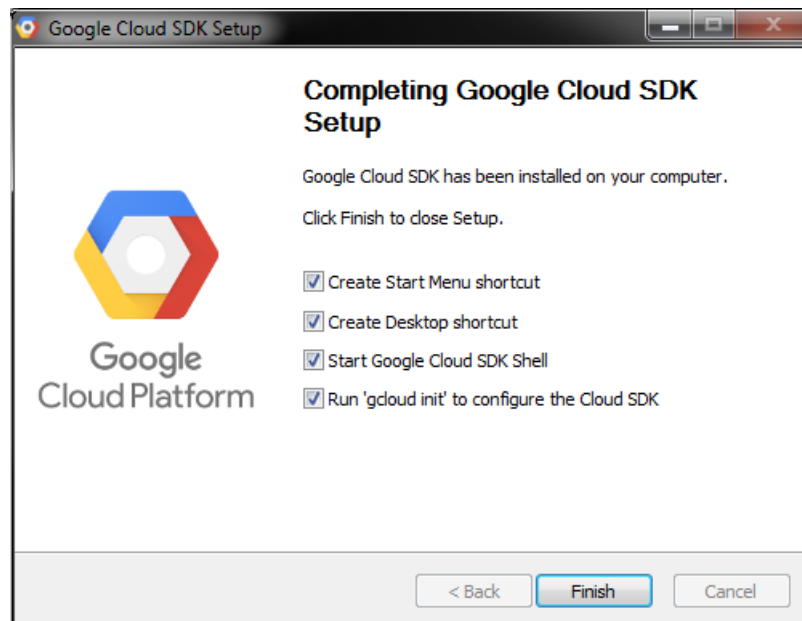
Ex.1: INSTALL AND CONFIGURE CLOUD SDK

AIM:

To install and configure cloud SDK.

PROCEDURE:

1. Download the Google Cloud CLI installer using the link
<https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe>
2. Launch the installer and follow the prompts.
3. Google Cloud CLI requires Python; supported versions are Python 3.8 to 3.12. By default, the Windows version of Google Cloud CLI comes bundled with Python 3.
4. After installation is complete, the installer gives you the option to create Start Menu and Desktop shortcuts, start the Google Cloud CLI shell, and configure the gcloud CLI.
5. Make sure that you leave the options to start the shell and configure your installation selected.



6. The installer starts a terminal window and runs the **gcloud init** command.
gcloud init initialize the gcloud CLI.
7. Create or select a configuration as prompted.
 - ✓ If it's a new installation, a configuration named "default" is created

and set as the active configuration.

- ✓ If you have existing configurations, choose to re-initialize the active one, switch to another and re-initialize, or create a new one.
8. Complete the authorization step when prompted. Depending on previous access, you may log in and grant access in a web browser or select an existing account.
 9. Once authorized, **gcloud init** sets the account property in the configuration to the specified account.
 10. Choose a current Google Cloud project if prompted.
 - ✓ If you have access to one project, it's automatically selected.
 - ✓ If you have multiple projects, select from a list or enter a project ID, create a new project, or list projects.
 11. Choose a default Compute Engine zone.
 12. **gcloud init** sets the region and zone properties in the configuration using the chosen zone.
 13. Verify the properties set by **gcloud init** using the command:
gcloud config list

Result:

The Google Cloud CLI is successfully installed, configured and is ready for use with Google Cloud services from the local environment.

Ex.2: GETTING STARTED WITH CLOUD SHELL AND GCLOUD

AIM:

To connect to the computing resources hosted on Google Cloud via Cloud Shell with the gcloud tool

PROCEDURE:

Access Cloud Shell and Configure Environment:

1. Open the Cloud shell either by launching the Google cloud console or through SDK.
2. In the Google cloud console, click on the "Activate Cloud Shell" icon located in the top-right corner.
3. To set the Region and the Zone, run the following commands:
[Replace REGION and ZONE with the region name and zone name]

```
gcloud config set compute/region REGION gcloud config  
set compute/zone ZONE
```
4. To view the project region and zone setting, run the following command:

```
gcloud config get-value compute/region gcloud config get-  
value compute/zone
```
5. In Cloud Shell, run the following gcloud command, to view the project id for your project:

```
gcloud config get-value project
```


Run the following gcloud command to view details about the project:

```
gcloud compute project-info describe --project $(gcloud config get-value project)
```
6. Environment variables define your environment and help save time when you write scripts that contain APIs or executables.

Create an environment variable to store your Project ID:

```
export PROJECT_ID=$(gcloud config get-value project)
```


Create an environment variable to store your Zone:

```
export ZONE=$(gcloud config get-value compute/zone)
```
7. Use the gcloud tool to create a new virtual machine (VM) instance. To create your VM, run the following command:

```
gcloud compute instances create gcelab2 --machine-type e2-medium --zone  
$ZONE
```

Output:

```
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-04-326fae68bc3d/zones/us-east1-c/instances/gcelab2].
```

NAME	ZONE	MACHINE_TYPE	PREEMPTIBLE	INTERNAL_IP	EXTERNAL_IP	STATUS
gcelab2	ZONE	e2-medium		10.128.0.2	34.67.152.90	RUNNING

8. Explore the gcloud commands:

Learn more about gcloud

commands:

```
gcloud -h
```

Detailed configuration help:

```
gcloud config --help
```

Additional configuration help:

```
gcloud help config
```

View the list of configurations in your environment:

```
gcloud config list
```

To see all properties and their settings:

```
gcloud config list --all
```

List installed components:

```
gcloud components list
```

Explore and Filter Command Output:

1. List the compute instance available in the project:

```
gcloud compute instances list
```

Output:

```
NAME: gcelab2
ZONE: ZONE

MACHINE_TYPE: e2-medium
PREEMPTIBLE:
INTERNAL_IP: 10.142.0.2
EXTERNAL_IP: 35.237.43.111
```

2. List the gcelab2 virtual machine:

```
gcloud compute instances list --filter="name=('gcelab2')"
```

Output:

```
NAME: gcelab2
ZONE: ZONE

MACHINE_TYPE: e2-medium
PREEMPTIBLE:
INTERNAL_IP: 10.142.0.2
EXTERNAL_IP: 35.237.43.111
```

3. List the firewall rules in the project:

`gcloud compute firewall-rules list`

Output:

NAME	NETWORK	DIRECTION	PRIORITY	ALLOW	DENY	DISABLED
default-allow-icmp	default	INGRESS	65534	icmp		False
default-allow-internal	default	INGRESS	65534	tcp:0-65535,udp:0-65535,icmp		False
default-allow-rdp	default	INGRESS	65534	tcp:3389		False
default-allow-ssh	default	INGRESS	65534	tcp:22		False
dev-net-allow-ssh	dev-network	INGRESS	1000	tcp:22		False
serverless-to-vpc-connector	dev-network	INGRESS	1000	icmp,udp:665-666,tcp:667		False
vpc-connector-egress	dev-network	INGRESS	1000	icmp,udp,tcp		False
vpc-connector-health-check	dev-network	INGRESS	1000	tcp:667		False
vpc-connector-to-serverless	dev-network	EGRESS	1000	icmp,udp:665-666,tcp:667		False

4. List the firewall rules for the default network:

`gcloud compute firewall-rules list -- filter="network='default'"`

Output:

NAME	NETWORK	DIRECTION	PRIORITY	ALLOW	DENY	DISABLED
default-allow-icmp	default	INGRESS	65534	icmp		False
default-allow-internal	default	INGRESS	65534	tcp:0-65535,udp:0-65535,icmp		False
default-allow-rdp	default	INGRESS	65534	tcp:3389		False
default-allow-ssh	default	INGRESS	65534	tcp:22		False

5. List the firewall rules for the default network where the allow rule matches an

ICMP rule:

`gcloud compute firewall-rules list --filter="NETWORK:'default' AND
ALLOW:'icmp'"`

Output:

NAME	NETWORK	DIRECTION	PRIORITY	ALLOW	DENY	DISABLED
default-allow-icmp	default	INGRESS	65534	icmp		False
default-allow-internal	default	INGRESS	65534	tcp:0-65535,udp:0-65535,icmp		False

Connect to VM Instance via SSH:

1. To connect to your VM with SSH, run the following command:

```
gcloud compute ssh gcelab2 --zone $ZONE
```

```
WARNING: The public SSH key file for gcloud does not exist.  
WARNING: The private SSH key file for gcloud does not exist.  
WARNING: You do not have an SSH key for gcloud.  
  
WARNING: [/usr/bin/ssh-keygen] will be executed to generate a key.  
This tool needs to create the directory
```

Output:

To continue, type Y.

```
Generating public/private rsa  
key pair. Enter passphrase
```

To leave the passphrase empty, press Enter twice.

Now, the connection to the virtual machine is established.

Result:

Thus, we have successfully connected to computing resources hosted on Google Cloud via Cloud Shell using the gcloud tool.

Ex.3:**CREATING A VIRTUAL MACHINE****AIM:**

To create virtual machine instances of various machine types using the Cloud Console and the command line and to connect an NGINX web server to the virtual machine.

PROCEDURE:**Create a new instance from the Cloud console:**

1. In the Cloud console, on the Navigation menu (Navigation menu icon), click Compute Engine > VM Instances.
2. To create a new instance, click CREATE INSTANCE.
3. Using the following parameters configure the new instance.

Field	Value
Name	gcelab
Region	<REGION>
Zone	<ZONE>
Series	E2
Machine Type	2 vCPU, 4 GB RAM
Boot Disk	New 10 GB balanced persistent disk OS Image: Debian GNU/Linux 11 (bullseye)
Firewall	Allow HTTP traffic

4. Click Create.
5. To use SSH to connect to the VM, click SSH to the right of the instance name, gcelab. This launches an SSH client directly from your browser.

Create a new instance with gcloud:

1. In the Cloud Shell, use gcloud to create a new VM instance from the command line: `gcloud compute instances create gcelab2 --machine-type e2-medium --zone=$ZONE`

Output:

```
Created [...gcelab2].

NAME: gcelab2
ZONE: Zone

MACHINE_TYPE: e2-medium
PREEMPTIBLE:
INTERNAL_IP: 10.128.0.3

EXTERNAL_IP: 34.136.51.150
```

2. You can also use SSH to connect to your instance via gcloud.

```
gcloud compute ssh gcelab2 --zone=$ZONE
```

Install an NGINX web server:

Install an NGINX web server, one of the most popular web servers in the world, to connect the VM to something.

1. Update the OS:

```
sudo apt-get update
```

Output:

```
Get:1 http://security.debian.org stretch/updates InRelease [94.3 kB]
Ign http://deb.debian.org stretch InRelease
Get:2 http://deb.debian.org stretch-updates InRelease [91.0 kB]
```

2. Install NGINX:

```
sudo apt-get install -y nginx
```

Output:

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
```

3. Confirm that NGINX is running:

```
ps auwx | grep nginx Output:
```

```
root      2330  0.0  0.0 159532 1628 ?        Ss      14:06   0:00 nginx: master process
/usr/sbin/nginx -g daemon on; master_process on;
www-data  2331   0.0  0.0 159864 3204 ?        S       14:06   0:00 nginx: worker
process
```


www-data	2332	0.0	0.0	159864	3204	?	S	14:06	0:00	nginx: worker
process										
root	2342	0.0	0.0	12780	988	pts/0	S+	14:07	0:00	grep nginx

To see the web page, return to the Cloud console and click the External IP link in the row for your machine, or add the External IP value to `http://EXTERNAL_IP/` in a new browser window or tab.

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Result:

Thus the virtual machine instances of various machine types have been created using the Cloud Console and the command line and also connected an NGINX web server to the virtual machine.

Ex.4**CREATE A SMALL APP ENGINE APPLICATION THAT
DISPLAYS A SHORT MESSAGE****AIM:**

To create a small App Engine application that displays a short message using Python

PROCEDURE:

This lab exercise is divided into:

1. Enabling Google App Engine Admin API
2. Download the Hello World App
3. Test the application
4. Change the application
5. Deploy the application
6. View your application

After initial set up of GCP Account, activate Cloud Shell using following steps

1. Click Activate Cloud Shell at the top of the Google Cloud console.
2. You can list the active account name with this command: `a. gcloud auth list`
3. Click Authorize.
4. You can list the project ID with this command:
5. `gcloud config list project`

Enabling Google App Engine Admin API

The App Engine Admin API enables developers to provision and manage their App Engine Applications

1. In the left Navigation menu, click APIs & Services > Library.
2. Type "App Engine Admin API" in the search box.
3. Click the App Engine Admin API card.
4. Click Enable. If there is no prompt to enable the API, then it is already enabled and no action is needed.

Download the Hello World App

1. Enter the following command to copy the Hello World sample app repository to your Google Cloud instance: `git clone https://github.com/GoogleCloudPlatform/python-docssamples.git`
2. Go to the directory that contains the sample code: `cd python-docs-samples/appengine/standard_python3/hello_world`

Test the application

- From within your helloworld directory where the app's app.yaml configuration file is located, start the Google Cloud development server with the following command:
`dev_appserver.py app.yaml`

The development server is now running and listening for requests on port 8080.

View the results by clicking the Web preview () > Preview on port 8080. You'll see this in a new browser window

Change the application

You can leave the development server running while you develop your application. The development server watches for changes in your source files and reloads them if necessary.

Let's try it. Leave the development server running. We'll open another command line window, then edit main.py to change "Hello World!" to "Hello, Cruel World!".

1. Click the (+) next to your Cloud Shell tab to open a new command line session.
2. Enter this command to go to the directory that contains the sample code: `cd python-docs-samples/appengine/standard_python3/hello_world`
3. Enter the following to open main.py in nano to edit the content: `nano main.py`
4. Change "Hello World!" to "Hello, Cruel World!".
5. Save the file with CTRL-S and exit with CTRL-X.
6. Reload the Hello World! Browser or click the Web Preview () > Preview on port 8080 to see the result

Deploy the application

1. To deploy your app to App Engine, run the following command from within the root directory of your application where the app.yaml file is located: `gcloud app deploy`
2. Enter the number that represents your region: `<REGION>`
3. The App Engine application will then be created.
4. Enter Y when prompted to confirm the details and begin the deployment of service.

View the application

1. To launch your browser enter the following command, then click on the link it provides:
`gcloud app browse`.

Result:

Thus, the procedure to create a small App Engine application that displays a short message using Python has been executed successfully

Ex.5 CREATE, DEPLOY, AND TEST A CLOUD FUNCTION USING THE CLOUD SHELL COMMAND LINE.

AIM:

To create, deploy, and test a cloud function using the cloud shell command line.

PROCEDURE:

This lab exercise is divided into:

1. Creating a function
2. Creating a cloud storage bucket
3. Deploy your function
4. Test the function
5. View Logs

After initial set up of GCP Account, activate Cloud Shell using following steps:

1. Click Activate Cloud Shell at the top of the Google Cloud console.
2. You can list the active account name with this command:
 - a. `gcloud auth list`
3. Click Authorize.
4. You can list the project ID with this command:
5. `gcloud config list project`

Creating a function:

First, you're going to create a simple function named helloWorld. This function writes a message to the Cloud Functions logs. It is triggered by cloud function events and accepts a callback function used to signal completion of the function.

For this lab the cloud function event is a cloud pub/sub topic event. A pub/sub is a messaging service where the senders of messages are decoupled from the receivers of messages. When a message is sent or posted, a subscription is required for a receiver to be alerted and receive the message. To learn more about pub/subs, in Cloud Pub/Sub Guides, see [Google Cloud Pub/Sub: A Google-Scale Messaging Service](#).

To learn more about the event parameter and the callback parameter, in Cloud Functions Documentation, see Background Functions. To create a cloud function:

1. In Cloud Shell, run the following command to set the default region:

```
gcloud config set compute/region REGION
```

2. Create a directory for the function code:

```
mkdir gcf_hello_world
```

3. Move to the gcf_hello_world directory:

```
cd gcf_hello_world
```

4. Create and open index.js to edit:

```
nano index.js
```

5. Copy the following into the index.js file:

```
/**
 * Background Cloud Function to be triggered by Pub/Sub.
 * This function is exported by index.js, and executed when
 * the trigger topic receives a message.
 * @param {object} data The event payload.
 * @param {object} context The event metadata.
 */

exports.helloWorld = (data, context) => {
  const pubSubMessage = data;
  const name = pubSubMessage.data
  ? Buffer.from(pubSubMessage.data, 'base64').toString() : "Hello World";
  console.log(`My Cloud Function: ${name}`); };
```

6. Exit nano (Ctrl+x) and save (Y) the file.

Create a cloud storage bucket

- Use the following command to create a new cloud storage bucket for your

function:

```
gsutil mb -p [PROJECT_ID] gs://[BUCKET_NAME]
```

- PROJECT_ID is the Project ID in the lab details panel on the left of this lab: <filled in at lab start>
- BUCKET_NAME is the name you give to the bucket. You can use the Project ID as the bucket name to ensure a globally unique name: <filled in at lab start>

Deploy your function:

When deploying a new function, you must specify `--trigger-topic`, `--trigger-bucket`, or `--trigger-http`. When deploying an update to an existing function, the function keeps the existing trigger unless otherwise specified.

For this lab, you'll set the `--trigger-topic` as `hello_world`.

1. Deploy the function to a pub/sub topic named **hello_world**,

replacing [BUCKET_NAME] with the name of your bucket:

```
gcloud functions deploy helloWorld \  
--stage-bucket [BUCKET_NAME] \  
--trigger-topic hello_world \  
--runtime nodejs20
```

Note: If you get `OperationError`, ignore the warning and re-run the command.

If prompted, enter Y to allow unauthenticated invocations of a new function

2. Verify the status of the function: `gcloud functions describe helloWorld` An **ACTIVE** status indicates that the function has been deployed.

Every message published in the topic triggers function execution, the message contents are passed as input data.

Test the function:

After you deploy the function and know that it's active, test that the function writes a message to the cloud log after detecting an event.

- Enter this command to create a message test of the function: `DATA=$(printf 'Hello World!'|base64) && gcloud functions call helloWorld --data '{"data":"$DATA"}'`

The cloud tool returns the execution ID for the function, which means a message has been written in the log.

Example output:

View logs to confirm that there are log messages with that execution ID.

View logs:

Check the logs to see your messages in the log history: gcloud functions logs read helloWorld If the function executed successfully, messages in the log

Result:

Thus the procedure to create, deploy, and test a cloud function using the cloud shell command line has been executed successfully.

Ex.6**DEPLOY A CONTAINERIZED APPLICATION****AIM:**

To deploy a containerized application using Google Kubernetes Engine.

PROCEDURE:

This lab exercise is divided into:

1. Set a default compute zone
2. Creating a GKE cluster
3. Get authentication credentials for the cluster
4. Deploying an application to the cluster
5. Deleting the cluster

After initial set up of GCP Account, activate Cloud Shell using following steps

1. Click Activate Cloud Shell at the top of the Google Cloud console.
2. You can list the active account name with this command: `gcloud auth list`
3. Click Authorize.
4. You can list the project ID with this command:
`gcloud config list project`

Set a default compute zone:

Your `compute_zone` is an approximate regional location in which your clusters and their resources live.

For example, `us-central1-a` is a zone in the `us-central1` region.

In your Cloud Shell session, run the following commands.

1. Set the default compute region:

```
gcloud config set compute/region "REGION"
```

2. Set the default compute zone:

```
gcloud config set compute/zone "ZONE"
```

Creating a GKE cluster:

A cluster consists of at least one cluster master machine and multiple worker machines called nodes. Nodes are Compute Engine virtual machine (VM) instances that run the Kubernetes processes necessary to make them part of the cluster.

Note: Cluster names must start with a letter and end with an alphanumeric, and cannot be longer than 40 characters.

Run the following command:

Create a cluster:

```
gcloud container clusters create --machine-type=e2-medium -- zone=ZONE lab-cluster
```

You can ignore any warnings in the output. It might take several minutes to finish creating the cluster.

Get authentication credentials for the cluster:

After creating your cluster, you need authentication credentials to interact with it.

- Authenticate with the cluster:

```
gcloud container clusters get-credentials lab-cluster
```

Deploy an application to the cluster:

You can now deploy a containerized application to the cluster. For this lab, you'll run hello-app in your cluster.

GKE uses Kubernetes objects to create and manage your cluster's resources. Kubernetes provides the Deployment object for deploying stateless applications like web servers. Service objects define rules and load balancing for accessing your application from the internet.

1. To create a new Deployment hello-server from the helloapp container image, run the following kubectl create command:

```
kubectl create deployment hello-server --image=gcr.io/googlesamples/hello-app:1.0
```

Expected output:

deployment.apps/hello-server created This Kubernetes command creates a deployment object that represents hello-server. In this case, --image specifies a container image to deploy. The command pulls the example image from a Container Registry bucket. gcr.io/google-samples/hello-app:1.0 indicates the specific image version to pull. If a version is not specified, the latest version is used.

2. To create a Kubernetes Service, which is a Kubernetes resource that lets you expose your application to external traffic, run the following kubectl expose command:

```
kubectl expose deployment hello-server --type=LoadBalancer -- port 8080
```

In this command:

- `--port` specifies the port that the container exposes.
- `type="LoadBalancer"` creates a Compute Engine load balancer for your container.

Expected output:

```
service/hello-server exposed
```

3. To inspect the hello-server Service, run `kubectl get`:

```
kubectl get service
```

Note: It might take a minute for an external IP address to be generated. Run the previous command again if the EXTERNAL-IP column status is pending

4. To view the application from your web browser, open a new tab and enter the following address, replacing [EXTERNAL IP] with the EXTERNAL-IP for hello-server.

```
http://[EXTERNAL-IP]:8080
```

Expected output: The browser tab displays the message Hello, world! as well as the version and hostname.

Deleting the cluster

1. To delete the cluster, run the following command:

```
gcloud container clusters delete lab-cluster
```

2. When prompted, type Y to confirm.

Deleting the cluster can take a few minutes. For more information on deleting GKE clusters from the Google Kubernetes Engine (GKE) article, [Deleting a cluster](#).

Result:

Thus the procedure to deploy a containerized app using Google Kubernetes Engine has been executed successfully.

**Ex.7 CREATE A STORAGE BUCKET, UPLOAD OBJECTS TO IT,
CREATE FOLDERS AND SUBFOLDERS IN IT, AND MAKE
OBJECTS PUBLICLY ACCESSIBLE USING THE CLOUD
COMMAND LINE.**

AIM:

To create a storage bucket in cloud and make objects publicly accessible using cloud command line.

PROCEDURE:

Task 1. Create a bucket

Buckets are the basic containers that hold your data in Cloud

Storage. To create a bucket:

- In the Cloud Console, go to **Navigation menu > Cloud Storage > Buckets**.
- Click + **Create**:
- Enter your bucket information and click **Continue** to complete each step:
 - **Name your bucket:** Enter a unique name for your bucket. For this lab, you can use your **Project ID** as the bucket name because it will always be unique.

Bucket naming rules:

- Choose **Region** for **Location type** and <filled in at lab start> for **Location**.
- Choose **Standard** for **default storage class**.
- Choose **Uniform** for **Access control** and **uncheck** *Enforce public access prevention on this bucket* to turn it off.
- Leave the rest of the fields as their default values and click

Create. Task 2. Upload an object into the bucket



To upload the image above into your new bucket:

Right-click on the image above and download it to your computer. Save the image as `kitten.png`, renaming it on download.

- In the Cloud Storage browser page, click the name of the bucket that you created.
- In the **Objects** tab, click **Upload files**.
- In the file dialog, go to the file that you downloaded and select it.
- Ensure the file is named **kitten.png**. If it is not, click the **three dot** icon for your file, select **Rename** from the dropdown, and rename the file to **kitten.png**.

Task 3. Share a bucket publicly

To allow public access to the bucket and create a publicly accessible URL for the image:

- Click the **Permissions** tab above the list of files.
- Ensure the view is set to **Principals**. Click **Grant Access** to view the **Add principals** pane.
- In the **New principals** box, enter *allUsers*.
- In the **Select a role** drop-down, select **Cloud Storage > Storage Object Viewer**.
- Click **Save**.
- In the **Are you sure you want to make this resource public?** window, click **Allow public access**.

Task 4. Create folders

- In the **Objects** tab, click **Create folder**.
- Enter **folder1** for **Name** and click **Create**.

You should see the folder in the bucket with an image of a folder icon to distinguish it from objects.

Create a subfolder and upload a file to it:

- Click **folder1**.
 - Click **Create folder**.
 - Enter **folder2** for **Name** and click **Create**.
 - Click **folder2**.
 - Click **Upload files**.
 - In the file dialog, navigate to the screenshot that you downloaded and select it.
- After the upload completes, you should see the file name and information about the file, such as its size and type.

Task 5. Delete a folder

- Click the arrow next to **Bucket details** to return to the buckets level.
- Select the bucket.
- Click on the **Delete** button.
- In the window that opens, type DELETE to confirm the deletion of the folder.
- Click **Delete** to permanently delete the folder and all objects and subfolders in it.

Result:

Thus the procedure to create a storage bucket in cloud and make objects publicly accessible using cloud command line was executed successfully.

Ex 8. DEPLOY A SAMPLE API WITH ANY OF THE API SERVICES

AIM:

To deploy an API managed by Cloud Endpoints

PROCEDURE:

Step 1: Getting the sample code

a) Enter the following command in Cloud Shell to get the sample API and scripts:

```
gsutil cp gs://spls/gsp164/endpoints-quickstart.zip .
```

```
unzip endpoints-quickstart.zip
```

b) Change to the directory that contains the sample code:

```
cd endpoints-quickstart
```

Step 2: Deploying the Endpoints configuration

To deploy the Endpoints configuration...

a) In the endpoints-qwikstart directory, enter the following:

```
cd scripts
```

b) Run the following script, which is included in the sample:

```
./deploy_api.sh
```

Step 3: Deploying the API backend

To deploy the API backend, make sure you are in the endpoints-quickstart/scripts directory.

Then, run the following script:

```
./deploy_app.sh
```

Step 4: Sending requests to the API

1. After deploying the sample API, you can send requests to it by running the following script:

```
./query_api.sh
```

2. To test, run this example in Cloud Shell:

```
./query_api.sh JFK
```

Step 5: Tracking API activity

1. Run this traffic generation script in Cloud Shell to populate the graphs and logs:

```
./generate_traffic.sh
```

2. Enter CTRL+C in Cloud Shell to stop the script

Step 6: Add a quota to the API

1. Deploy the Endpoints configuration that has a quota:

```
./deploy_api.sh ../openapi_with_ratelimit.yaml
```

2. Redeploy your app to use the new Endpoints configuration (this may take a few minutes):

```
./deploy_app.sh
```

3. Click **Create credentials** and choose **API key**. A new API key is displayed on the screen.

4. In Cloud Shell, type the following. Replace YOUR-API-KEY with the API key you just created:

```
export API_KEY=YOUR-API-KEY
```

5. Send your API a request using the API key variable you just created:

```
./query_api_with_key.sh $API_KEY
```

The API now has a limit of 5 requests per second. Run the following command to send traffic to the API and trigger the quota limit: `./generate_traffic_with_key.sh $API_KEY`

After running the script for 5-10 seconds, enter CTRL+C in Cloud Shell to stop the script.

Send another authenticated request to the API:

```
./query_api_with_key.sh $API_KEY
```


Output:

Congratulations! You've successfully rate-limited your API. You can also set varying limits on different API methods, create multiple kinds of quotas, and keep track of which consumers use which APIs.

Result:

Thus the procedure to deploy an API managed by Cloud Endpoints was executed successfully.

Ex.9. PUBLISH MESSAGES WITH MANAGED MESSAGE SERVICE USING THE PYTHON CLIENT LIBRARY

AIM:

To create publish and receive messages in Pub/Sub by using a client library

PROCEDURE:

Step 1. Create a virtual environment:

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

a) Install the virtualenv environment:

```
sudo apt-get install -y virtualenv
```

b) Build the virtual environment:

```
python3 -m venv venv
```

c) Activate the virtual environment:

```
source venv/bin/activate
```

Step 2. Install the client library:

a) Run the following to install the client library:

```
pip install --upgrade google-cloud-pubsub
```

b) Get the sample code by cloning a GitHub repository:

```
git clone https://github.com/googleapis/python-pubsub.git
```

c) Navigate to the directory:

```
cd python-pubsub/samples/snippets
```

Step 3. Create a topic:

a) In Cloud Shell, your Project ID should automatically be stored in the environment variable

GOOGLE_CLOUD_PROJECT:

```
echo $GOOGLE_CLOUD_PROJECT
```

b) Ensure the output is the same as the Project ID in your CONNECTION DETAILS.

```
cat publisher.py
```

c) For information about the publisher script:

```
python publisher.py -h
```

d) Run the publisher script to create Pub/Sub Topic:

```
python publisher.py $GOOGLE_CLOUD_PROJECT create MyTopic
```

e) This command returns a list of all Pub/Sub topics in a given project:

```
python publisher.py $GOOGLE_CLOUD_PROJECT list
```

f) Navigate to **Navigation menu > Pub/Sub > Topics**.

You should see MyTopic.

Step 4. Create a subscription:

a) Create a Pub/Sub subscription for topic with subscriber.py script:

```
python subscriber.py $GOOGLE_CLOUD_PROJECT create MyTopic MySub
```

b) This command returns a list of subscribers in given project:

```
python subscriber.py $GOOGLE_CLOUD_PROJECT list-in-project
```

c) For information about the subscriber script:

```
python subscriber.py -h
```

Step 5. Publish messages

a) Publish the message "Hello" to MyTopic:

```
gcloud pubsub topics publish MyTopic --message "Hello"
```

b) Publish a few more messages to MyTopic—run the following commands (replacing <YOUR NAME> with your name and <FOOD> with a food you like to eat):

```
gcloud pubsub topics publish MyTopic --message "Publisher's name is <YOUR NAME>"
```

```
gcloud pubsub topics publish MyTopic --message "Publisher likes to eat <FOOD>"
```

```
gcloud pubsub topics publish MyTopic --message "Publisher thinks Pub/Sub is awesome"
```

Step 6. View messages:

a) Use MySub to pull the message from MyTopic:

```
python subscriber.py $GOOGLE_CLOUD_PROJECT receive MySub
```

b) Click **Ctrl+c** to stop listening.

OUTPUT:

You used Python to create a Pub/Sub topic, published to the topic, created a subscription, then used the subscription to pull data from the topic.

Result:

Thus the procedure to create Publish and receive messages in Pub/Sub by using a client library was executed successfully.

Ex.10:**EXPLORING IAM ROLES****AIM:**

To create users and to explore granting and revoking Cloud IAM roles to the users.

PROCEDURE:**Step 1: Create Two Users**

1. Log in to the Google Cloud Console with your credentials.
2. Navigate to IAM & Admin > IAM.
3. Click on +ADD MEMBER.
4. Enter the email address of the first user [User1] and assign the Project Owner role.
5. Repeat steps 3-4 for the second user [User 2], assigning the Project Viewer role.

Step 2: Verify Project Viewer Access

1. Switch to the User 2 console.
2. Navigate to IAM & Admin > IAM.
3. Ensure that User 2 has the "Viewer" role assigned.

Step 3: Create a Cloud Storage Bucket

1. Switch back to the User 1 console.
2. Navigate to Navigation menu > Cloud Storage > Buckets.
3. Click +CREATE.
4. Fill in the required fields for bucket creation (unique name, Multi-Region, etc.).
5. Click CREATE.
6. Upload a sample file to the bucket (e.g., 'sample.txt').

Step 4: Verify Project Viewer Access to Bucket

1. Switch to the User 2 console.
2. Navigate to Navigation menu > Cloud Storage > Buckets.
3. Verify that User 2 can see the created bucket.

Step 5: Remove Project Viewer Access for User 2

1. Switch back to the User 1 console.
2. Navigate to IAM & Admin > IAM.
3. Locate User 2 in the Member list.
4. Click the pencil icon next to User 2.
5. Remove the Project Viewer role for User 2.
6. Click SAVE.

Step 6: Verify Revoked Access for User 2

1. Switch to the User 2 console.
2. Navigate to Navigation menu > Cloud Storage > Buckets.
3. Observe the permission error indicating the lack of access.

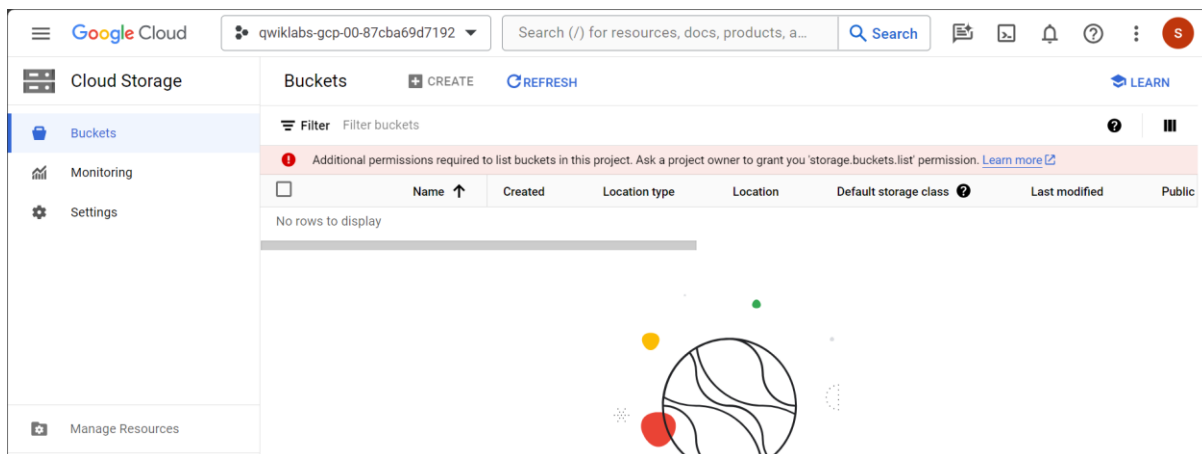
Step 7: Add Storage Permissions for User 2

1. Copy User 2's name from the Lab Connection panel.
2. Switch back to the User 1 console.
3. Navigate to IAM & Admin > IAM.
4. Click +GRANT ACCESS.
5. Paste User 2's name into the New principals field.
6. In the Select a role field, choose Cloud Storage > Storage Object Viewer.
7. Click SAVE.

Step 8: Verify Storage Access for User 2

1. Switch to the User 2 console.
2. Open Cloud Shell.
3. Run the command: ``gsutil ls gs://[YOUR_BUCKET_NAME]``.
4. Verify that User 2 has view access to the Cloud Storage bucket.

Output:



IAM & Admin

- IAM
- Identity & Organization
- Policy Troubleshooter
- Policy Analyzer **NEW**
- Organization Policies
- Service Accounts
- Workload Identity Federati...
- Workforce Identity Federa...
- Labels
- Manage Resources
- Release Notes

Type	Principal	Name	Role	Security insights
	794080652991-compute@developer.gserviceaccount.com	Compute Engine default service account	Editor	8025/8025 excess permissions
	admiral@qwiklabs-services-prod.iam.gserviceaccount.com		Owner	9153/9162 excess permissions 2 service account impersonations
	qwiklabs-gcp-00-87cba69d7192@qwiklabs-gcp-00-87cba69d7192.iam.gserviceaccount.com	Qwiklabs User Service Account	BigQuery Admin Owner Storage Admin	110/110 excess permissions 9162/9162 excess permissions 47/47 excess permissions
	student-03-2c7db3f3dc8b@qwiklabs.net	student-9be26e2a	App Engine Admin BigQuery Admin Editor Owner Viewer	
	student-03-31bd407c6b8e@qwiklabs.net		Storage Object Viewer	

Policy updated

Cloud Storage

- Buckets
- Monitoring
- Settings

Buckets [CREATE](#) [REFRESH](#) [LEARN](#)

Filter Filter buckets

Name	Created	Location type	Location	Default storage class	Last modified
ajgrmkcslab-1	Feb 24, 2024, 3:17:42 PM	Multi-region	us	Standard	Feb 24, 2024, 3:17:42 PM

Cloud Storage

- Buckets
- Monitoring
- Settings

Bucket details [REFRESH](#) [LEARN](#)

ajgrmkcslab-1

Location: us (multiple regions in United States) | Storage class: Standard | Public access: Not public | Protection: None

OBJECTS CONFIGURATION PERMISSIONS PROTECTION LIFECYCLE OBSERVABILITY INVENTORY REPORTS

Buckets > ajgrmkcslab-1

[UPLOAD FILES](#) [UPLOAD FOLDER](#) [CREATE FOLDER](#) [TRANSFER DATA](#) [MANAGE HOLDS](#) [EDIT RETENTION](#) [DOWNLOAD](#) [DELETE](#)

Filter by name prefix only Filter Filter objects and folders Show deleted data

Name	Size	Type	Created	Storage class	Last modified	Public access
Sample.txt	71.6 KB	application/pdf	Feb 24, 2024, 3:18:24 PM	Standard	Feb 24, 2024, 3:18:24 PM	Not public

Result:

Thus, the users are created and explored granting and revoking Cloud IAM roles to the users.

Ex.11: MULTIPLE VPC NETWORKS

AIM:

To create several VPC networks and VM instances and test connectivity across networks.

PROCEDURE:

1. Create an auto mode network mynetwork.
2. Create two custom networks managementnet and privatenet.
3. Create firewall rules to allow SSH, ICMP, and RDP ingress traffic to VM instances on all the networks.
4. Create two VM instances:
 - a. managementsubnet-europe-west1-vm in managementsubnet-europe-west1
 - b. privatesubnet-europe-west1-vm in privatesubnet-europe-west1
5. Explore the connectivity between the VM instances to determine the effect of having VM instances in the same zone versus having instances in the same VPC network.

GCLOUD COMMANDS & OUTPUT:

Create mynetwork network

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

Create managementnet network

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

Created	[https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-04-382957804a55/global/networks/managementnet].
NAME:	managementnet
SUBNET_MODE:	CUSTOM
BGP_ROUTING_MODE:	REGIONAL
IPV4_RANGE:	
GATEWAY_IPV4:	

Create managementsubnet europe-west1 subnet

```
gcloud compute networks subnets create managementsubnet-europe-west1 \
  --network=managementnet \
  --region=europe-west1 \
  --range=10.130.0.0/20
```



```
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-04-382957804a55/regions/europe-west1/subnetworks/managementsubnet-europe-west1].
NAME: managementsubnet-europe-west1
REGION: europe-west1
NETWORK: managementnet
RANGE: 10.130.0.0/20
STACK_TYPE: IPV4_ONLY
IPV6_ACCESS_TYPE:
INTERNAL_IPV6_PREFIX:
EXTERNAL_IPV6_PREFIX:
```

```
# Create privatenet network
```

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

```
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-04-382957804a55/global/networks/privatenet].
NAME: privatenet
SUBNET_MODE: CUSTOM
BGP_ROUTING_MODE: REGIONAL
IPV4_RANGE:
GATEWAY_IPV4:
```

```
# Create privatesubnet- europe-west1 subnet
```

```
gcloud compute networks subnets create privatesubnet-europe-west1 \
--network=privatenet \
--region=europe-west1 \
--range=172.16.0.0/24
```

```
NAME: privatesubnet-europe-west1
REGION: europe-west1
NETWORK: privatenet
RANGE: 172.16.0.0/24
STACK_TYPE: IPV4_ONLY
IPV6_ACCESS_TYPE:
INTERNAL_IPV6_PREFIX:
EXTERNAL_IPV6_PREFIX:
```

```
# Create privatesubnet- us-central1 subnet
```

```
gcloud compute networks subnets create privatesubnet-us-central1 \
--network=privatenet \
--region=us-central1 \
--range=172.20.0.0/24
```

```
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-04-382957804a55/regions/us-central1/subnetworks/privatesubnet-us-central1].
NAME: privatesubnet-us-central1
REGION: us-central1
NETWORK: privatenet
RANGE: 172.20.0.0/24
STACK_TYPE: IPV4_ONLY
IPV6_ACCESS_TYPE:
INTERNAL_IPV6_PREFIX:
EXTERNAL_IPV6_PREFIX:
```

#List the available VPC networks

```
gcloud compute networks list
```

```
NAME: default
SUBNET_MODE: AUTO
BGP_ROUTING_MODE: REGIONAL
IPV4_RANGE:
GATEWAY_IPV4:
```

```
NAME: managementnet
SUBNET_MODE: CUSTOM
BGP_ROUTING_MODE: REGIONAL
IPV4_RANGE:
GATEWAY_IPV4:
```

```
NAME: mynetwork
SUBNET_MODE: AUTO
BGP_ROUTING_MODE: REGIONAL
IPV4_RANGE:
GATEWAY_IPV4:
```

```
NAME: privatenet
SUBNET_MODE: CUSTOM
BGP_ROUTING_MODE: REGIONAL
IPV4_RANGE:
GATEWAY_IPV4:
```

#Create the firewall rules for managementnet network

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

#Create the firewall rules for privatenet network

```
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
```

```
Creating firewall...working..Created
[https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-04-
382957804a55/global/firewalls/privatenet-allow-icmp-ssh-rdp].
Creating firewall...done.
NAME: privatenet-allow-icmp-ssh-rdp
NETWORK: privatenet
DIRECTION: INGRESS
PRIORITY: 1000
ALLOW: icmp,tcp:22,tcp:3389
DENY:
DISABLED: False
```

Create managementnet-europe-west1-vm instance

```
gcloud compute instances create managementnet-europe-west1-vm \
--zone=europe-west1-d \
--machine-type=e2-micro \
--subnet=managementsubnet-europe-west1
```

```
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-04-
382957804a55/zones/europe-west1-d/instances/managementnet-europe-west1-vm].
NAME: managementnet-europe-west1-vm
ZONE: europe-west1-d
MACHINE_TYPE: e2-micro
PREEMPTIBLE:
INTERNAL_IP: 10.130.0.2
EXTERNAL_IP: 35.195.13.233
STATUS: RUNNING
```

Create privatenet-europe-west1-vm instance

```
gcloud compute instances create privatenet-europe-west1-vm \
--zone=europe-west1-d --machine-type=e2-micro \
--subnet=privatesubnet-europe-west1
```

```
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-04-
382957804a55/zones/europe-west1-d/instances/privatenet-europe-west1-vm].
NAME: privatenet-europe-west1-vm
```

```
ZONE: europe-west1-d  
MACHINE_TYPE: e2-micro  
PREEMPTIBLE:  
INTERNAL_IP: 172.16.0.2  
EXTERNAL_IP: 34.78.251.30  
STATUS: RUNNING
```

#List all the VM instances

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

```
NAME: managementnet-europe-west1-vm  
ZONE: europe-west1-d  
MACHINE_TYPE: e2-micro  
PREEMPTIBLE:  
INTERNAL_IP: 10.130.0.2  
EXTERNAL_IP: 35.195.13.233  
STATUS: RUNNING
```

```
NAME: mynet-europe-west1-vm  
ZONE: europe-west1-d  
MACHINE_TYPE: e2-medium  
PREEMPTIBLE:  
INTERNAL_IP: 10.132.0.2  
EXTERNAL_IP: 35.233.66.106  
STATUS: RUNNING
```

```
NAME: privatenet-europe-west1-vm  
ZONE: europe-west1-d  
MACHINE_TYPE: e2-micro  
PREEMPTIBLE:  
INTERNAL_IP: 172.16.0.2  
EXTERNAL_IP: 34.78.251.30  
STATUS: RUNNING
```

```
NAME: mynet-us-central1-vm  
ZONE: us-central1-b  
MACHINE_TYPE: e2-medium  
PREEMPTIBLE:  
INTERNAL_IP: 10.128.0.2  
EXTERNAL_IP: 34.133.206.222  
STATUS: RUNNING
```

Google Cloud VM instances page showing a list of VM instances. The table includes columns for Status, Name, Zone, Recommendations, In use by, Internal IP, External IP, and Connect. Below the table, there are related actions such as Explore Backup and DR, Monitor VMs, Explore VM logs, Set up firewall rules, Patch management, and Load balance between VMs.

#Ping the external IP addresses

ping -c 3 34.133.206.22

```
PING 34.133.206.222 (34.133.206.222) 56(84) bytes of data:
64 bytes from 34.133.206.222: icmp_seq=1 ttl=51 time=207 ms
64 bytes from 34.133.206.222: icmp_seq=2 ttl=51 time=205 ms
64 bytes from 34.133.206.222: icmp_seq=3 ttl=51 time=205 ms

--- 34.133.206.222 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 204.986/205.552/206.600/0.741 ms
```

#Ping the internal IP addresses

ping -c 3 10.128.0.2

```
PING 10.128.0.2 (10.128.0.2) 56(84) bytes of data:

--- 10.128.0.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2039ms
```

Result:

Thus, multiple VPC networks and VM instances are created and the connectivity across the networks were tested.

Ex.12: VPC NETWORKS - CONTROLLING ACCESS

AIM:

To create two nginx web servers on the default VPC network and control external HTTP access to the web servers using tagged firewall rules.

PROCEDURE:

1. Create two web servers (blue and green) in the default VPC network.
2. Create the blue server with a network tag web-server.
 - a. Go to "Compute Engine" > "VM instances" in the Google Cloud Console.
 - b. Click "Create Instance."
 - c. Set Name to "blue," choose a Region, and a Zone.
 - d. Click "Advanced Options" > "Networking."
 - e. Under "Network tags," type web-server.
 - f. Click "Create."
3. Create the green server without a network tag.
 - a. Click "Create Instance" again.
 - b. Set Name to "green," choose a Region, and a Zone.
 - c. Click "Create."
4. Install nginx on both VM instances and modify the welcome page to distinguish the servers.
 - a. In the VM instances dialog, locate the "blue" instance and click "SSH" to launch a terminal and connect.
 - b. Run the following command to install Nginx:

```
sudo apt-get install nginx-light -y
```
 - c. Close the SSH terminal to the blue server.
 - d. In the VM instances dialog, locate the "green" instance and click "SSH" to launch a terminal and connect.
 - e. Run the following command to install Nginx:

```
sudo apt-get install nginx-light -y
```
 - f. Close the SSH terminal to the green server.
5. Create a firewall rule that applies to VM instances with the web-server network tag.

Create firewall rule for web-server

```
gcloud compute firewall-rules create allow-http-web-server \  
  --network=default \  
  --action=ALLOW \  
  --direction=INGRESS \  
  --target-tags=web-server \  
  --source-ranges=0.0.0.0/0 \  
  --rules=tcp:80,icmp
```

6. Create a test-vm instance using the Cloud Shell command line.

```
gcloud compute instances create test-vm --machine-type=e2-micro --subnet=default --  
zone=ZONE
```

7. From the test-vm, curl the internal and external IP addresses of blue and green.
8. Go to "Compute Engine" > "VM instances" in the Google Cloud Console.
9. Note down the internal and external IP addresses of the "blue" and "green" servers.
10. For "test-vm," click "SSH" to launch a terminal and connect.
11. Test HTTP connectivity to blue's internal IP and green's internal IP.
12. Test HTTP connectivity to blue's external IP and green's external IP.
13. The external IP of the blue server works due to the allow-http-web-server rule with the web-server tag.
14. The HTTP access to the external IP address of the green server doesn't work.

Result:

Thus, two nginx web servers are created on the default VPC network and external HTTP access to the web servers are controlled using tagged firewall rules.

Ex.13:**HTTP LOAD BALANCER****AIM:**

To configure a HTTP Load Balancer with global backends and to stress test the Load Balancer and denylist the stress test IP.

PROCEDURE:**Step 1: Configure HTTP and Health Check Firewall Rules**

- HTTP Firewall Rule:
 - Navigate to VPC Network > Firewall in the Cloud Console.
 - Create a firewall rule to allow HTTP traffic to the backends.
 - Specify the appropriate source, target tags, and protocols.
- Health Check Firewall Rule:
 - Create another firewall rule to allow health check probes from designated IP ranges.
 - Define source IP ranges, target tags, and required protocols.

Step 2: Configure Instance Templates and Create Instance Groups

- Instance Templates:
 - Go to Compute Engine > Instance templates.
 - Create instance templates specifying machine types, network configurations, and optional startup scripts.
- Managed Instance Groups (MIGs):
 - Navigate to Compute Engine > Instance groups.
 - Create managed instance groups using the previously defined instance templates.
 - Configure autoscaling policies and other group-specific settings.

Step 3: Configure the HTTP Load Balancer

- Start Configuration:
 - Go to Network Services > Load balancing.
 - Click "Create load balancer."
 - Select the load balancing type (e.g., HTTP Load Balancer).

- **Configure Frontend:**
 - Define frontend settings such as protocol, IP version, and port configurations.
- **Configure Backend:**
 - Set up backend services, associating them with the previously created managed instance groups.
 - Define balancing modes, capacities, and other backend-specific settings.
- **Configure Health Check:**
 - Create health checks to monitor the instances' health.
 - Specify protocols, ports, and health check intervals.
- **Review and Create:**
 - Review the load balancer configuration.
 - Confirm that all settings align with the requirements.
 - Click "Create" to initiate the HTTP Load Balancer creation.

Step 4: Test the HTTP Load Balancer

- **Access Load Balancer:**
 - After creation, note the Load Balancer's IP address or domain.
 - Access the Load Balancer in a web browser to ensure proper functionality.
- **Stress Test Load Balancer:**
 - Create a new virtual machine to simulate load (e.g., siege-vm).
 - Install a stress testing tool (e.g., Siege).
 - Run stress tests against the Load Balancer to evaluate its performance.

Step 5: Denylist an IP Address with Cloud Armor

- **Create Cloud Armor Security Policy:**
 - Navigate to Compute Engine > VM instances.
 - Identify the external IP of the stress testing VM (e.g., siege-vm).
 - Go to Network Security > Cloud Armor Policies.
 - Create a security policy with a denylist rule for the stress testing VM's IP.
 - Set the rule to deny and respond with an appropriate status code (e.g., 403).
- **Verify Security Policy:**
 - Confirm that the stress testing VM is denied access to the Load Balancer.

- Access the Load Balancer from a different source to ensure continued functionality.

COMMAND LINE EXECUTION:

Create HTTP Firewall Rule:

```
gcloud compute firewall-rules create default-allow-http \
  --network=default \
  --action=allow \
  --direction=ingress \
  --target-tags=http-server \
  --source-ranges=0.0.0.0/0 \
  --rules=tcp:80
```

Create Health Check Firewall Rule:

```
gcloud compute firewall-rules create default-allow-health-check \
  --network=default \
  --action=allow \
  --direction=ingress \
  --target-tags=http-server \
  --source-ranges=130.211.0.0/22,35.191.0.0/16 \
  --rules=tcp
```

Create Instance Template for us-west1:

```
gcloud compute instance-templates create us-west1-template \
  --machine-type=e2-micro \
  --network=default \
  --tags=http-server \
  --subnet=default \
  --metadata=startup-script-url=gs://cloud-training/gcpnet/http1b/startup.sh
```

Create Instance Template for us-east1:

```
gcloud compute instance-templates create us-east1-template \
  --machine-type=e2-micro \
  --network=default \
  --tags=http-server \
```

```
--subnet=default \  
--metadata=startup-script-url=gs://cloud-training/gcpnet/http1b/startup.sh
```

OUTPUT:

```
Creating firewall...working..Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-a8579e7ae2f3/global/firewalls/http-firewall-rule].  
Creating firewall...done.  
  
NAME: http-firewall-rule  
NETWORK: default  
DIRECTION: INGRESS  
PRIORITY: 1000  
ALLOW: tcp:80  
DENY:  
DISABLED: False
```

```
Creating firewall...working..Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-a8579e7ae2f3/global/firewalls/health-check-firewall-rule].  
Creating firewall...done.  
NAME: health-check-firewall-rule  
NETWORK: default  
DIRECTION: INGRESS  
PRIORITY: 1000  
ALLOW: tcp  
DENY:  
DISABLED: False  
student_03_ded7fd24cdc3@cloudshell:~ (qwiklabs-gcp-01-a8579e7ae2f3)$
```

```
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-a8579e7ae2f3/global/instanceTemplates/us-west1-template].  
NAME: us-west1-template  
MACHINE_TYPE: e2-micro  
PREEMPTIBLE:  
CREATION_TIMESTAMP: 2024-03-05T00:42:59.562-08:00  
student_03_ded7fd24cdc3@cloudshell:~ (qwiklabs-gcp-01-a8579e7ae2f3)$
```

```
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-a8579e7ae2f3/global/instanceTemplates/us-east1-template].  
NAME: us-east1-template  
MACHINE_TYPE: e2-micro  
PREEMPTIBLE:  
CREATION_TIMESTAMP: 2024-03-05T00:44:09.173-08:00  
student_03_ded7fd24cdc3@cloudshell:~ (qwiklabs-gcp-01-a8579e7ae2f3)$
```

Google Cloud

qwiklabs-gcp-01-a8579e7ae2f3

Search (/) for resources, docs, products, and more

Q Search

5

?

⋮

S

Compute Engine

Instance groups

CREATE INSTANCE GROUP

REFRESH

DELETE

LEARN

Instance groups

Instance groups are collections of VM instances that use load balancing and automated services, like autoscaling and autohealing. [Learn more](#)

Filter

Enter property name or value

Status

Name

Instances

Template

Group type

Creation time

Recommendation

Autoscaling

Zone

In Use By

us-east1-mig

1

us-east1-template

Managed

Mar 5, 2024, 2:35:53 PM UTC+05:30

On: Target CPU utilization 80%

us-east1 (3/3)

us-west1-mig

1

us-west1-template

Managed

Mar 5, 2024, 2:28:47 PM UTC+05:30

On: Target CPU utilization 80%

us-west1 (3/3)

Google Cloud

qwiklabs-gcp-01-a8579e7ae2f3

Search (/) for resources, docs, products, and more

Q Search

5

?

⋮

S

Compute Engine

VM instances

CREATE INSTANCE

IMPORT VM

REFRESH

LEARN

INSTANCES

OBSERVABILITY

INSTANCE SCHEDULES

VM instances

Filter

Enter property name or value

Status

Name

Zone

Recommendations

In use by

Internal IP

External IP

Connect

us-east1-mig-9z02

us-east1-c

10.142.0.3 (nic0)

34.73.165.176 (nic0)

SSH

us-west1-mig-2lz2

us-west1-c

10.138.0.3 (nic0)

34.168.152.93 (nic0)

SSH

Related actions

Explore Backup and DR

Monitor VMs

Explore VM logs

Set up firewall rules

Patch management

Load balance between VMs

Google Cloud

qwiklabs-gcp-01-a8579e7ae2f3

load

Q Search

6

?

⋮

S

Network services

Load balancing

CREATE LOAD BALANCER

REFRESH

DELETE

LEARN

LOAD BALANCERS

BACKENDS

FRONTENDS

Filter

Enter property name or value

Name

Load balancer type

Access type

Protocols

Region

Backends

http-lb

Application

External

HTTP

1 backend service (2 instance groups, 0 network endpoint groups)

To view or delete load balancing resources like forwarding rules and target proxies, go to the [load balancing components view](#).

44

Google Cloud | qwiklabs-gcp-01-a8579e7ae2f3 | cloud arm | Search

Network Security

- Secure Web Proxy
- Cloud Armor
 - Cloud Armor policies**
 - Adaptive Protection
 - Managed Protection
- Cloud IDS
 - IDS Dashboard
 - IDS Endpoints
 - IDS Threats
- Cloud NGFW
 - Firewall policies

Cloud Armor policies | CREATE POLICY | DELETE POLICY

Security policies let you control access to your Google Cloud Platform resources at your network's edge.

You can use security policies to protect workloads using global external Application Load Balancers, global external proxy Network Load Balancers, external passthrough Network Load Balancers, Protocol Forwarding, or Instances with Public IP addresses. [Learn more](#)

Filter: Enter property name or value

<input type="checkbox"/>	Name ↑	Type	Scope	Rules	Targets	Description
<input type="checkbox"/>	default-security-policy-for-backend-service-http-backend	Backend security policy	global	2	0	Default security policy for: http-backend
<input type="checkbox"/>	denylist-siege	Backend security policy	global	2	1	

Result:

Thus, the HTTP Load Balancer is configured with global backends. Then, stress tested the Load Balancer with a VM and denylisted the IP address of that VM with Cloud Armor.

Ex.14 CREATE TWO MANAGED INSTANCE GROUPS IN THE SAME REGION. THEN, CONFIGURE AND TEST AN INTERNAL LOAD BALANCER WITH THE INSTANCES GROUPS AS THE BACKENDS.

AIM:

To create two managed instance groups in the same region. Then, configure and test an Internal Load Balancer with the instances groups as the backends.

PROCEDURE:

This lab exercise is divided into:

1. Configure HTTP and health check firewall rules.
2. Configure instance templates and create instance groups.
3. Configure the internal Load Balancer
4. Test the Internal Load Balancer

Configure HTTP and health check firewall rules:

Configure firewall rules to allow HTTP traffic to the backends and TCP traffic from the Google Cloud health checker.

Explore the my-internal-app network

The network my-internal-app with subnet-a and subnet-b along with firewall rules for RDP, SSH, and ICMP traffic have been configured for you.

1. In the Console, navigate to Navigation menu > VPC network > VPC networks.
2. Scroll down and 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 subneta and subnet-b. Both subnets are in the REGION 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 HTTP firewall rule

Create a firewall rule to allow HTTP traffic to the backends from the Load Balancer and the internet (to install Apache on the backends).

1. Still in VPC network, in the left pane click Firewall.
2. Notice the app-allow-icmp and app-allow-ssh-rdp firewall rules. These firewall rules have been created for you.
3. Click Create Firewall Rule.
4. Set the following values, leave all other values at their defaults:
5. Click Create.

Create the health check firewall rules

Health checks determine which instances of a Load Balancer can receive new connections. For Internal 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.

1. Still in the Firewall rules page, click Create Firewall Rule.
2. Set the following values, leave all other values at their defaults:
3. Click Create. Click Check my progress to verify the objective.

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. In the Console, navigate to Navigation menu > Compute Engine > Instance templates.
2. Click Create instance template.
3. For Name, type instance-template-1.
4. For Series, select E2.
5. Click advanced options.
6. Click Networking.
7. For Network tags, specify lb-backend. Note: The network tag lb-backend ensures that the HTTP and Health Check firewall rules apply to these instances.
8. For Network interfaces, click the dropdown icon to edit.

9. Set the following values, leave all other values at their defaults:

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

Network: my-internal-app

Subnetwork: subnet-a

10. Click Done.

11. Click Management.

12. Under Metadata, click Add item and specify the following:

Key 1: Value 1

startupscripturl gs://cloudtraining/gcpnet/ilb/startup.sh

Note: The startup-script-url specifies a script that will be 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. Feel free to explore this script.

13. Click Create.

14. Wait for the instance template to be created.

Configure the next instance template:

Create another instance template for subnet-b by copying instancetemplate-1:

1. Still in Instance templates, check the box next to instancetemplate-1, then click Copy. You will see the instance is named instance-template-2.

2. Click Advanced options.

3. Click the Networking tab.

4. For Network interfaces, click the dropdown icon to edit.

5. Select subnet-b as the Subnetwork.

6. Click Done and then click Create.

Verify that VM instances are being created in both subnets and create a utility VM to access the backends' HTTP sites.

1. Still in Compute Engine, click VM instances.

2. Notice two instances that start with instance-group1 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.

3. Click Create instance.

4. Set the following values, leave all other values at their defaults:
5. Click Advanced options.
6. Click Networking.
7. For Network interfaces, click the dropdown icon to edit.
8. Set the following values, leave all other values at their defaults:
9. Click Done and then click Create. Click Check my progress to verify the objective.
10. Note that the internal IP addresses for the backends are 10.10.20.2 and 10.10.30.2. Note: If these IP addresses are different, replace them in the two curl commands below.
11. For utility-vm, click SSH to launch a terminal and connect.
12. To verify the welcome page for instance-group-1-xxxx, run the following command: curl 10.10.20.2
13. To verify the welcome page for instance-group-2-xxxx, run the following command: curl 10.10.30.2 verifying that the Internal Load Balancer sends traffic to both backends.
14. Close the SSH terminal to utility-vm: exit

Configure the Internal Load Balancer:

Start the configuration

1. In the Cloud Console, navigate to Navigation menu > Network Services > Load balancing, and then click Create load balancer.
2. Under Network Load Balancer (TCP/SSL), click on Start configuration.
3. For Internet facing or internal only, select Only between my VMs. Note: Choosing Only between my VMs makes this Load Balancer internal. This choice requires the backends to be in a single region and does not allow offloading TCP processing to the Load Balancer.
4. Click Continue.
5. For Name, type my-ilb.
6. For Region, select REGION.
7. For Network, select my-internal-app.

Test the Internal Load Balancer

Verify that the my-ilb IP address forwards traffic to instance-group1 and instance-group-2.
Access the Internal Load Balancer

1. In the Cloud Console, navigate to Navigation menu > 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`
4. Run the same command a couple more times. In the output, you should be able to see responses from instancegroup-1 in Zone and instance-group-2 in the different zone of same region.

Result:

Thus, procedure to create two managed instance groups in the same region. Then, configure and test an Internal Load Balancer with the instances groups as the backends has been executed successfully.

Ex.15 MONITOR A COMPUTE ENGINE VIRTUAL MACHINE (VM) INSTANCE WITH CLOUD MONITORING BY CREATING UPTIME CHECK, ALERTING POLICY, DASHBOARD AND CHART

AIM:

To monitor a Compute Engine virtual machine (VM) instance with Cloud Monitoring by creating uptime check, alerting policy, dashboard and chart.

PROCEDURE:

This lab exercise is divided into:

1. Create a Compute Engine Instance
2. Add Apache2 HTTP Server to your instance
3. Create an uptime check
4. Create an alerting policy
5. Create a dashboard and chart
6. View your logs
7. Check the uptime check results and triggered alerts

Create a Compute Engine instance

1. In the Cloud Console dashboard, go to Navigation menu > Compute Engine > VM instances, then click Create instance.
2. Fill in the fields as follows, leaving all other fields at the default value
3. Click Create. Wait a couple of minutes, you'll see a green check when the instance has launched. Click Check my progress below. A green check confirms you're on track.

Add Apache2 HTTP Server to your instance

1. In the Console, click SSH in line with lamp-1-vm to open a terminal to your instance.
2. Run the following commands in the SSH window to set up Apache2 HTTP Server:

`sudo apt-get update`
`sudo apt-get install apache2 php7.0`
3. When asked if you want to continue, enter Y.
4. Return to the Cloud Console, on the VM instances page. Click the External IP for lamp-1-vm instance to see the Apache2 default page for this instance.

Create an uptime check:

Uptime checks verify that a resource is always accessible. For practice, create an uptime check to verify your VM is up.

1. In the Cloud Console, in the left menu, click Uptime checks, and then click Create Uptime Check.
2. For Protocol, select HTTP.
3. For Resource Type, select Instance.
4. For Instance, select lamp-1-vm.
5. For Check Frequency, select 1 minute.
6. Click Continue.
7. In Response Validation, accept the defaults and then click Continue.
8. In Alert & Notification, accept the defaults, and then click Continue.
9. For Title, type Lamp Uptime Check.
10. Click Test to verify that your uptime check can connect to the resource. When you see a green check mark everything can connect.
11. Click Create.

The uptime check you configured takes a while for it to become active. Continue with the lab, you'll check for results later. While you wait, create an alerting policy for a different resource

Create an alerting policy

Use Cloud Monitoring to create one or more alerting policies.

1. In the left menu, click Alerting, and then click +Create Policy.
2. Click on Select a metric dropdown. Uncheck the Active.
3. Type Network traffic in filter by resource and metric name and click on VM instance > Interface. Select Network traffic (agent.googleapis.com/interface/traffic) and click Apply. Leave all other fields at the default value.
4. Click Next.
5. Set the Threshold position to Above threshold, Threshold value to 500 and Advanced Options > Retest window to 1 min. Click Next.
6. Click on the drop down arrow next to Notification Channels, then click on Manage Notification Channels. A Notification channels page will open in a new tab.
7. Scroll down the page and click on ADD NEW for Email.

8. In the Create Email Channel dialog box, enter your personal email address in the Email Address field and a Display name.
9. Click on Save.
10. Go back to the previous Create alerting policy tab.
11. Click on Notification Channels again, then click on the Refresh icon to get the display name you mentioned in the previous step
12. Click on Notification Channels again if necessary, select your Display name and click OK.
13. Add a message in documentation, which will be included in the emailed alert.
14. Mention the Alert name as Inbound Traffic Alert.
15. Click Next.
16. Review the alert and click Create Policy. You've created an alert! While you wait for the system to trigger an alert, create a dashboard and chart, and then check out Cloud Logging. Click Check my progress below. A green check confirms you're on track.

Create a dashboard and chart

You can display the metrics collected by Cloud Monitoring in your own charts and dashboards. In this section you create the charts for the lab metrics and a custom dashboard.

1. In the left menu select Dashboards, and then +Create Dashboard.
2. Name the dashboard Cloud Monitoring LAMP Qwik Start Dashboard.

Add the first chart

Add the second chart

View your logs

1. Select Navigation menu > Logging > Logs Explorer.
2. Select the logs you want to see, in this case, you select the logs for the lamp-1-vm instance you created at the start of this lab:

- Click on Resource.
- Select VM Instance > lamp-1-vm in the Resource dropdown menu.
- Click Apply.
- Leave the other fields with their default values.
- Click the Stream logs. You see the logs for your VM instance.

Check out what happens when you start and stop the VM instance.

Check the uptime check results and triggered alerts:

1. In the Cloud Logging window, select Navigation menu > Monitoring > Uptime checks. This view provides a list of all active uptime checks, and the status of each in different locations.

You will see Lamp Uptime Check listed. Since you have just restarted your instance, the regions are in a failed status. It may take up to 5 minutes for the regions to become active. Reload your browser window as necessary until the regions are active.

2. Click the name of the uptime check, Lamp Uptime Check.

Since you have just restarted your instance, it may take some minutes for the regions to become active. Reload your browser window as necessary.

Check if alerts have been triggered:

1. In the left menu, click Alerting.

2. You see incidents and events listed in the Alerting window.

3. Check your email account. You should see Cloud Monitoring Alerts.

Result:

Thus, procedure to monitor a Compute Engine virtual machine (VM) instance with Cloud Monitoring by creating uptime check, alerting policy, dashboard and chart has been executed successfully.

Ex.16 CREATE A CLUSTER, RUN A SIMPLE APACHE SPARK JOB IN THE CLUSTER, THEN MODIFY THE NUMBER OF WORKERS IN THE CLUSTER.

AIM:

To create a cluster, run a simple Apache Spark job in the cluster, then modify the number of workers in the cluster.

PROCEDURE:

This lab exercise is divided into:

1. Create a cluster.
2. Submit a job.
3. Update the cluster.

Create a cluster:

1. In Cloud Shell, run the following command to set the Region: `gcloud config set dataproc/region Region`
2. Dataproc creates staging and temp buckets that are shared among clusters in the same region. Since we're not specifying an account for Dataproc to use, it will use the Compute Engine default service account, which doesn't have storage bucket permissions by default. Let's add those.

- First, run the following commands to grab the `PROJECT_ID` and `PROJECT_NUMBER`:

```
PROJECT_ID=$(gcloud config get-value project) && \
gcloud config set project $PROJECT_ID
PROJECT_NUMBER=$(gcloud projects describe $PROJECT_ID --
format='value(projectNumber)')
```

- Now run the following command to give the Storage Admin role to the Compute Engine default service account: `gcloud projects add-iam-policy-binding $PROJECT_ID \ --member=serviceAccount:$PROJECT_NUMBERcompute@developer.gserviceaccount.com \ --role=roles/storage.admin`

3. Run the following command to create a cluster called `examplecluster` with `e2-standard-4` VMs and default Cloud Dataproc settings:

```
gcloud dataproc clusters create example-cluster --worker-boot-disksize 500 --worker-machine-type=e2-standard-4 --master-machinetype=e2-standard-4
```

4. If asked to confirm a zone for your cluster. Enter Y. Your cluster will build for a couple of minutes.

Waiting for cluster creation operation...done.

Created [... example-cluster]

When you see a "Created" message, you're ready to move on.

Submit a job:

- Run this command to submit a sample Spark job that calculates a rough value for pi:

```
gcloud dataproc jobs submit spark --cluster example-cluster \  
--class org.apache.spark.examples.SparkPi \  
--jars file:///usr/lib/spark/examples/jars/spark-examples.jar -- 1000
```

The command specifies:

- That you want to run a spark job on the example-cluster cluster
- The class containing the main method for the job's pi-calculating application
- The location of the jar file containing your job's code
- The parameters you want to pass to the job—in this case, the number of tasks, which is 1000

The job's running and final output is displayed in the terminal window:

Waiting for job output... ...

Pi is roughly 3.14118528 ...

state: FINISHED

Update a cluster:

1. To change the number of workers in the cluster to four, run the following command:

```
gcloud dataproc clusters update example-cluster --num-workers
```

Your cluster's updated details are displayed in the command's output:

```
Waiting on operation [projects/qwiklabs-  
gcp7f7aa0829e65200f/regions/global/operations/b86892cc-e71d-4e7baa5e-6030c945ea67].
```

Waiting for cluster update operation...done.

2. You can use the same command to decrease the number of worker nodes:

```
gcloud dataproc clusters update example-cluster --num-workers
```


3. Now you can create a Dataproc cluster and adjust the number of workers from the gcloud command line on the Google Cloud.

Result:

Thus, procedure to create a cluster, run a simple Apache Spark job in the cluster, then modify the number of workers in the cluster has been executed successfully.

Ex.17 CREATE A STREAMING PIPELINE USING ONE OF THE CLOUD SERVICES

AIM:

To create a streaming pipeline using the cloud service.

PROCEDURE:

Task 1. Create a Cloud Storage bucket:

Create a Cloud Storage bucket using your Project ID as the bucket name: <filled in at lab start>

Task 2. Create a BigQuery dataset and table:

a) Create a BigQuery dataset called BigQuery dataset name in the region named US (multi region).

b) In the created dataset, create a table called BigQuery table name.

Task 3. Set up a Pub/Sub topic:

a) Create a Pub/Sub topic called Pub/Sub topic name.

Use the default settings, which has enabled the checkbox for Add a default subscription.

Task 4. Run a Dataflow pipeline to stream data from Pub/Sub to BigQuery:

a) Create and run a Dataflow job called Dataflow job name to stream data from **Pub/Sub to BigQuery**, using the Pub/Sub topic and BigQuery table you created in the previous tasks.

b) Use the Pub/Sub topic that you created in a previous task: Pub/Sub topic name

c) Use the Cloud Storage bucket that you created in a previous task as the temporary location: <filled in at lab start>

d) Use the BigQuery dataset and table that you created in a previous task as the output table: BigQuery dataset name.BigQuery table name

e) Use **Region** as the regional endpoint.

Task 5. Publish a test message to the topic and validate data in BigQuery:

- a) Publish a message to your topic using the following code syntax for **Message**: {"data": "73.4 F"}

Note: 73.4 F can be replaced with any value.

- b) Run a SELECT statement in BigQuery to see the test message populated in your table.

Output:

Congratulations! In this lab, you have successfully created a Cloud Storage bucket, a BigQuery dataset and table, a Pub/Sub topic, and a Dataflow job to stream data from Pub/Sub to BigQuery. You also published a test message to the topic and validated the data in BigQuery.

Result:

Thus the Procedure to create a streaming pipeline using the cloud service was executed successfully.

**Ex. 18. SET UP YOUR PYTHON DEVELOPMENT ENVIRONMENT,
GET THE RELEVANT SDK FOR PYTHON, AND RUN AN
EXAMPLE PIPELINE USING THE CLOUD CONSOLE**

AIM:

To create a python development environment to run a pipeline using cloud console

PROCEDURE:

Set up your environment

To install Python and then create a virtual environment, follow these steps:

1. Check that you have Python 3 and pip running in

your system: `python --version`
`python -m pip --version`

Get the Apache Beam SDK

To download and install the Apache Beam SDK, follow these steps:

1. Verify that you are in the Python virtual environment that you created in the preceding section. Ensure that the prompt starts with `<env_name>`, where `env_name` is the name of the virtual environment.
2. Install the [Python wheel](#) packaging

standard: `pip install wheel`

3. Install the latest version of the Apache Beam SDK for

Python: `pip install 'apache-beam[gcp]'`

On Microsoft Windows, use the following

command: `pip install apache-
beam[gcp]`

Depending on the connection, your installation might take a while.

Run the pipeline locally

To see how a pipeline runs locally, use a ready-made Python module for the wordcount example that is included with the `apache_beam` package.

The wordcount pipeline example does the following:

1. Takes a text file as input.

This text file is located in a Cloud Storage bucket with the resource name `gs://dataflow-samples/shakespeare/kinglear.txt`.

2. Parses each line into words.
3. Performs a frequency count on the

tokenized words. To stage the wordcount pipeline

locally, follow these steps:

1. From your local terminal, run the wordcount example:

```
python -m apache_beam.examples.wordcount \
--output outputs
```

2. View the output of the

```
pipeline: more
outputs*
```

3. To exit, press q.

Run the pipeline on the Dataflow service

In this section, run the wordcount example pipeline from the `apache_beam` package on the Dataflow service. This example specifies `DataflowRunner` as the parameter for `--runner`.

- Run the pipeline:

```
python -m apache_beam.examples.wordcount \
--region DATAFLOW_REGION \
--input gs://dataflow-samples/shakespeare/kinglear.txt \
--output gs://BUCKET_NAME/results/outputs \
--runner DataflowRunner \
--project PROJECT_ID \
--temp_location
```

gs://*BUCKET_NAME*/tmp/ Replace the following:

- *DATAFLOW_REGION*: the [regional endpoint](#) where you want to deploy the Dataflow job—for example, europe-west1
The --region flag overrides the default region that is set in the metadata server, your local client, or environment variables.
- *BUCKET_NAME*: the Cloud Storage bucket name that you copied earlier
- *PROJECT_ID*: the Google Cloud project ID that you copied earlier

View your results

To view your results in Google Cloud console, follow these steps:

1. In the Google Cloud console, go to the Dataflow

Jobs page. [Go to Jobs](#)

The **Jobs** page displays details of your wordcount job, including a status of **Running** at first, and then **Succeeded**.

2. Go to the Cloud Storage

Buckets page. [Go to Buckets](#)

3. From the list of buckets in your project, click the storage bucket that you created earlier. In the wordcount directory, the output files that your job created are displayed.

Modify the pipeline code

The wordcount pipeline in the previous examples distinguishes between uppercase and lowercase words. The following steps show how to modify the pipeline so that the wordcount pipeline is not case-sensitive.

1. On your local machine, download the latest copy of the [wordcount code](#) from the Apache Beam GitHub repository.
2. From the local terminal, run the

```
pipeline: python  
wordcount.py --output  
outputs
```

3. View the results:

more outputs*

4. To exit, press q.
5. In an editor of your choice, open the wordcount.py file.
6. Inside the run function, examine the

```
pipeline steps: counts = (  
    lines  
    | 'Split' >> (beam.ParDo(WordExtractingDoFn()).with_output_types(str))  
    | 'PairWithOne' >> beam.Map(lambda x: (x, 1))  
    | 'GroupAndSum' >>
```

beam.CombinePerKey(sum)) After split, the
lines are split into words as strings.

7. To lowercase the strings, modify the line after split:

```
counts =  
(  
    lines  
    | 'Split' >> (beam.ParDo(WordExtractingDoFn()).with_output_types(str))  
    | 'lowercase' >> beam.Map(str.lower)  
    | 'PairWithOne' >> beam.Map(lambda x: (x, 1))  
    | 'GroupAndSum' >> beam.CombinePerKey(sum))
```

This modification maps the str.lower function onto every word. This line is
equivalent to beam.Map(lambda word: str.lower(word)).

8. Save the file and run the modified

```
wordcount job: python wordcount.py --  
  
output outputs
```

9. View the results of the modified pipeline:

more outputs*

10. To exit, press q.
11. Run the modified pipeline on the Dataflow

```
service: python wordcount.py \  
    --region DATAFLOW_REGION \  
    --input gs://dataflow-samples/shakespeare/kinglear.txt \  
    --output gs://BUCKET_NAME/results/outputs \  
  
    --runner DataflowRunner \  
    --project PROJECT_ID \  

```

`--temp_location`

`gs://BUCKET_NAME/tmp/` Replace the

following:

- *DATAFLOW_REGION*: the [regional endpoint](#) where you want to deploy the Dataflow job
- *BUCKET_NAME*: your Cloud Storage bucket name
- *PROJECT_ID*: your Google Cloud project ID

Clean up

1. In the Google Cloud console, go to the Cloud Storage **Buckets** page. [Go to Buckets](#)
2. Click the checkbox for the bucket that you want to delete.
3. To delete the bucket, click delete **Delete**, and then follow the instructions.
4. If you keep your project, revoke the roles that you granted to the Compute Engine default service account. Run the following command once for each of the following IAM roles:
 - `roles/dataflow.admin`
 - `roles/dataflow.worker`
 - `roles/storage.objectAdmin`

`gcloud projects remove-iam-policy-binding PROJECT_ID \`

`--member=serviceAccount:PROJECT_NUMBER-
compute@developer.gserviceaccount.com \`
`--role=SERVICE_ACCOUNT_ROLE`

5. Optional: Revoke the authentication credentials that you created, and delete the local credential file.

`gcloud auth application-default revoke`

6. Optional: Revoke credentials from the

`gcloud CLI. gcloud auth revoke`

Result:

Thus the procedure to create a python development environment to run a pipeline using cloud console was executed successfully.

**Ex. 19 USE CLOUD-BASED DATA PREPARATION TOOL TO
MANIPULATE A DATASET. IMPORT DATASETS,
CORRECT MISMATCHED DATA, TRANSFORM
DATA, AND JOIN DATA**

AIM:

To create a cloud based data preparation tool to manipulate a dataset.

PROCEDURE:

Task 1. Create a Cloud Storage bucket in your project

2. In the Cloud Console, select **Navigation menu**(≡) > **Cloud Storage** > **Buckets**.
3. Click **Create bucket**.
4. In the **Create a bucket** dialog, **Name** the bucket a unique name. Leave other settings at their default value.
5. Uncheck **Enforce public access prevention on this bucket** for Choose how to control access to objects.
6. Click **Create**.

Task 2. Initialize Cloud Dataprep

1. Open **Cloud Shell** and run the following command:

```
gcloud beta services identity create --service=dataprep.googleapis.com
```
2. Select **Navigation menu** > **Dataprep**.
3. Check to accept the Google Dataprep Terms of Service, then click **Accept**.
4. Check to authorize sharing your account information with Trifacta, then click **Agree and Continue**.
5. Click **Allow** to allow Trifacta to access project data.
6. Click your student username to sign in to Cloud Dataprep by Trifacta. Your username is the **Username** in the left panel in your lab.
7. Click **Allow** to grant Cloud Dataprep access to your Google Cloud lab account.
8. Check to agree to Trifacta Terms of Service, and then click **Accept**.
9. Click **Continue** on the **First time setup** screen to create the default storage location.

Task 3. Create a flow

Cloud Dataprep uses a flow workspace to access and manipulate datasets.

1. Click **Flows** icon, then the **Create** button, then select **Blank Flow** :
2. Click on **Untitled Flow**, then name and describe the flow. Since this lab uses 2016 data from the [United States Federal Elections Commission 2016](#), name the flow "FEC-2016", and then describe the flow as "United States Federal Elections Commission 2016".
3. Click **OK**.

Task 4. Import datasets

In this section you import and add data to the FEC-2016 flow.

1. Click **Add Datasets**, then select the **Import Datasets** link.
2. In the left menu pane, select **Cloud Storage** to import datasets from Cloud Storage, then click on the pencil to edit the file path.
3. Type gs://spl/gsp105 in the **Choose a file or folder** text box, then click **Go**.
4. Click **us-fec/**.
5. Click the + icon next to cn-2016.txt to create a dataset shown in the right pane. Click on the title in the dataset in the right pane and rename it "Candidate Master 2016".
6. In the same way add the itcont-2016-orig.txt dataset, and rename it "Campaign Contributions 2016".
7. Both datasets are listed in the right pane; click **Import & Add**

Task 5. Prep the candidate file

1. By default, the Candidate Master 2016 dataset is selected. In the right pane, click **Edit Recipe**.
2. Each of the column heads have a Name and value that specify the data type. To see data types, click the column icon:
3. Notice also that when you click the name of the column, a **Details** panel opens on the right.
4. Click **X** in the top right of the Details panel to close the **Details** panel.

In the following steps you explore data in the grid view and apply transformation steps to your recipe.

1. Column5 provides data from 1990-2064. Widen column5 (like you would

on a spreadsheet) to separate each year. Click to select the tallest bin, which represents the year 2016.

2. In the **Suggestions** panel on the right, in the **Keep rows** section, click **Add** to add this step to your recipe.

The Recipe panel on the right now has the following step:

Keep rows where (DATE(2016, 1, 1) <= column5) && (column5 < DATE(2018, 1, 1))

3. In Column6 (State), hover over and click on the mismatched (red) portion of the header to select the mismatched rows.
4. To correct the mismatch, click **X** in the top of the Suggestions panel to cancel the transformation, then click on the flag icon in Column6 and change it to a "String" column.
5. Filter on just the presidential candidates, which are those records that have the value "P" in column7. In the histogram for column7, hover over the two bins to see which is "H" and which is "P". Click the "P" bin.
6. In the right Suggestions panel, click **Add** to accept the step to the recipe.

Task 6. Wrangle the Contributions file and join it to the Candidates file

On the Join page, you can add your current dataset to another dataset or recipe based on information that is common to both datasets.

Before you join the Contributions file to the Candidates file, clean up the Contributions file.

1. Click on **FEC-2016** (the dataset selector) at the top of the grid view page.
2. Click to select the grayed out **Campaign Contributions 2016**.
3. In the right pane, click **Add > Recipe**, then click **Edit Recipe**.
4. Click the **recipe** icon at the top right of the page, then click **Add New Step**.
5. Insert the following Wrangle language command in the

Search box: replacepatterns col: * with: " on:

`{start}"|"{end}` global: true

6. Click **Add** to add the transform to the recipe.
7. Add another new step to the recipe. Click **New Step**, then type "Join" in the Search box.
8. Click **Join datasets** to open the Joins page.
9. Click on "Candidate Master 2016" to join with Campaign Contributions 2016, then **Accept** in the bottom right.
10. On the right side, hover in the Join keys section, then click on the pencil (Edit icon).

11. In the Add Key panel, in the Suggested join keys section, click **column2 = column11**.
12. Click **Save and Continue**.
13. Click **Next**, then check the checkbox to the left of the "Column" label to add all columns of both datasets to the joined dataset.
14. Click **Review**, and then **Add to Recipe** to return to the grid view.

Task 7. Summary of data

Generate a useful summary by aggregating, averaging, and counting the contributions in Column 16 and grouping the candidates by IDs, names, and party affiliation in Columns 2, 24, 8 respectively.

1. At the top of the Recipe panel on the right, click on **New Step** and enter the following formula in the **Transformation** search box to preview the aggregated data.

pivot value:sum(column16),average(column16),countif(column16 > 0) group:
column2,column24,column8

2. Click **Add** to open a summary table of major US presidential candidates and their 2016 campaign contribution metrics.

Task 8. Rename columns

You can make the data easier to interpret by renaming the columns.

1. Add each of the renaming and rounding steps individually to the recipe by clicking **New Step**, then enter:

rename type: manual mapping: [column24,'Candidate_Name'],
[column2,'Candidate_ID'],[column8,'Party_Affiliation'],
[sum_column16,'Total_Contribution_Sum'],
[average_column16,'Average_Contribution_Sum'],
[countif,'Number_of_Contributions']

2. Then click **Add**.

3. Add in this last **New Step** to round the Average Contribution amount:

set col: Average_Contribution_Sum value: round(Average_Contribution_Sum)

4. Then

click **Add**.

Output:

RBC	Candidate_ID	RBC	Candidate_Name	RBC	Party_Affiliation	#	Total_Contribution_Sum
19 Categories		19 Categories		2 Categories		25 - 996.03k	
C00573519		CARSON, BENJAMIN S SR MD		IND			244843
C00574624		CRUZ, RAFAEL EDWARD "TED"		IND			348112
C00575795		CLINTON, HILLARY RODHAM / TIMOTHY MICHAEL KAINE		IND			996034
C00577130		SANDERS, BERNARD		IND			217178
C00575449		PAUL, RAND		IND			54078
C00577312		FIORINA, CARLY		IND			63046
C00578757		GRAHAM, LINDSEY O		IND			19592
C00580399		CHRISTIE, CHRISTOPHER J		IND			97220
C00580480		WALKER, SCOTT		IND			40965
C00579450		BUSH, JEB		IND			340301
C00581215		WEBB, JAMES		IND			2350
C00581876		KASICH, JOHN R		IND			65832
C00580587		PERRY, JAMES R (RICK)		IND			21400
C00578658		O'MALLEY, MARTIN JOSEPH		IND			43823
C00581199		STEIN, JILL		IND			350
C00580159		JINDAL, BOBBY		IND			15365
C00578492		SANTORUM, RICHARD J.		IND			7665
C00578245		PATAKI, GEORGE E		IND			5100
C00575795		CLINTON, HILLARY RODHAM / TIMOTHY MICHAEL KAINE		ORG			1500
C00573519		CARSON, BENJAMIN S SR MD		ORG			100
C00580655		WELLS, ROBERT CARR JR					25

Result:

Thus the procedure to create a cloud based data preparation tool to manipulate a dataset was executed successfully.

Ex.20 UTILIZE A CLOUD-BASED DATA PROCESSING AND ANALYSIS TOOL FOR DATA EXPLORATION AND USE A MACHINE LEARNING PLATFORM TO TRAIN AND DEPLOY A CUSTOM TENSORFLOW REGRESSOR MODEL FOR PREDICTING CUSTOMER LIFETIME VALUE.

AIM:

To create a cloud based data processing and analysis tool for data exploration using machine learning platform.

PROCEDURE:

Task 1. Enable Google Cloud services

- In Cloud Shell, use gcloud to enable the services used

```
in the lab: gcloud services enable \  
compute.googleapis.com \  
iam.googleapis.com \  
iamcredentials.googleapis.com \  
monitoring.googleapis.com \  
logging.googleapis.com \  
notebooks.googleapis.com \  
aiplatform.googleapis.com \  
bigquery.googleapis.com \  
artifactregistry.googleapis.com \  
cloudbuild.googleapis.com \  
container.googleapis.com
```

Task 2. Create Vertex AI custom service account for Vertex Tensorboard integration

Create custom service account:

```
SERVICE_ACCOUNT_ID=vertex-custom-training-sa  
gcloud iam service-accounts create $SERVICE_ACCOUNT_ID \  
--description="A custom service account for Vertex custom training  
with Tensorboard" \  

```

```
--display-name="Vertex AI Custom Training"
```

Grant it access to Cloud Storage for writing and retrieving

Tensorboard logs: `PROJECT_ID=$(gcloud config get-value core/project)`

```
gcloud projects add-iam-policy-binding $PROJECT_ID \
member=serviceAccount:$SERVICE_ACCOUNT_ID@$PROJECT_ID.iam.gserviceaccount.com \
```

```
--role="roles/storage.admin"
```

Grant it access to your BigQuery data source to read data into your

TensorFlow model: `gcloud projects add-iam-policy-binding $PROJECT_ID \`

```
--
```

```
member=serviceAccount:$SERVICE_ACCOUNT_ID@$PROJECT_ID.iam.gserviceaccount.com \
```

```
--role="roles/bigquery.admin"
```

Grant it access to Vertex AI for running model training, deployment, and explanation jobs:

```
gcloud projects add-iam-policy-binding $PROJECT_ID \
```


```
--
```

```
member=serviceAccount:$SERVICE_ACCOUNT_ID@$PROJECT_ID.iam.gserviceaccount.com \
```

```
--role="roles/aiplatform.user"
```

Task 3. Launch Vertex AI Workbench notebook

Note: Please use TensorFlow Enterprise 2.13 to complete this lab. To create and launch a Vertex AI Workbench notebook:

1. In the **Navigation Menu** , click **Vertex AI > Workbench**.
2. On the **Workbench** page, click **Enable Notebooks API** (if it isn't enabled yet).
3. Click on **User-Managed Notebooks** tab then, click **Create New**.
4. Name the notebook.
5. Set **Region** to REGION and **Zone** to ZONE.

6. In the **New instance** menu, choose the latest version of **TensorFlow Enterprise 2.11** in **Environment**.
7. Click **Advanced Options** to edit the instance properties.
8. Click **Machine type** and then select **e2-standard-2** for Machine type.
9. Leave the remaining fields at their default and click **Create**.

After a few minutes, the **Workbench** page lists your instance, followed by **Open JupyterLab**.

Click **Open JupyterLab** to open JupyterLab in a new tab. If you get a message saying beatrix jupyterlab needs to be included in the build, just ignore it.

Task 4. Clone the lab repository

To clone the training-data-analyst repository in your JupyterLab instance:

1. In JupyterLab, click the **Terminal** icon to open a new terminal.
2. At the command-line prompt, type the following command and press **ENTER**:

```
git clone --depth=1 https://github.com/GoogleCloudPlatform/training-data-analyst
```
3. To confirm that you have cloned the repository, in the left panel, double click the training-data-analyst folder to see its contents.

Task 5. Install lab dependencies

- Run the following to go to the training-data-analyst/self-paced-labs/vertex-ai/vertex-ai-qwikstart folder, then pip3 install requirements.txt to install lab dependencies:

```
cd training-data-analyst/self-paced-labs/vertex-ai/vertex-ai-qwikstart  
pip3 install --user -r requirements.txt
```

Navigate to lab notebook

1. In your notebook, navigate to **training-data-analyst > self-paced-labs > vertex- ai > vertex-ai-qwikstart**, and open **lab_exercise.ipynb**.

📁 / ... / vertex-ai / vertex-ai-qwikstart /

Name ▲	Last Modified
📁 images	3 minutes ago
📁 online-retail-clv-3M	3 minutes ago
📁 utils	3 minutes ago
📄 lab_exercise.ipynb	3 minutes ago
📄 README.md	3 minutes ago
📄 requirements.txt	3 minutes ago

2. Continue the lab in the notebook, and run each cell by clicking the **Run** icon at the top of the screen.

Alternatively, you can execute the code in a cell with **SHIFT + ENTER**.

Congratulations!

In this lab, you ran a machine learning experimentation workflow using Google Cloud BigQuery for data storage and analysis and Vertex AI machine learning services to train and deploy a TensorFlow model to predict customer lifetime value. You progressed from training a TensorFlow model locally to training on the cloud with Vertex AI and leveraged several new unified platform capabilities such as Vertex TensorBoard and prediction feature attributions.

Result:

Thus the procedure to create a cloud based data processing and analysis tool for data exploration using machine learning platform was executed successfully.

