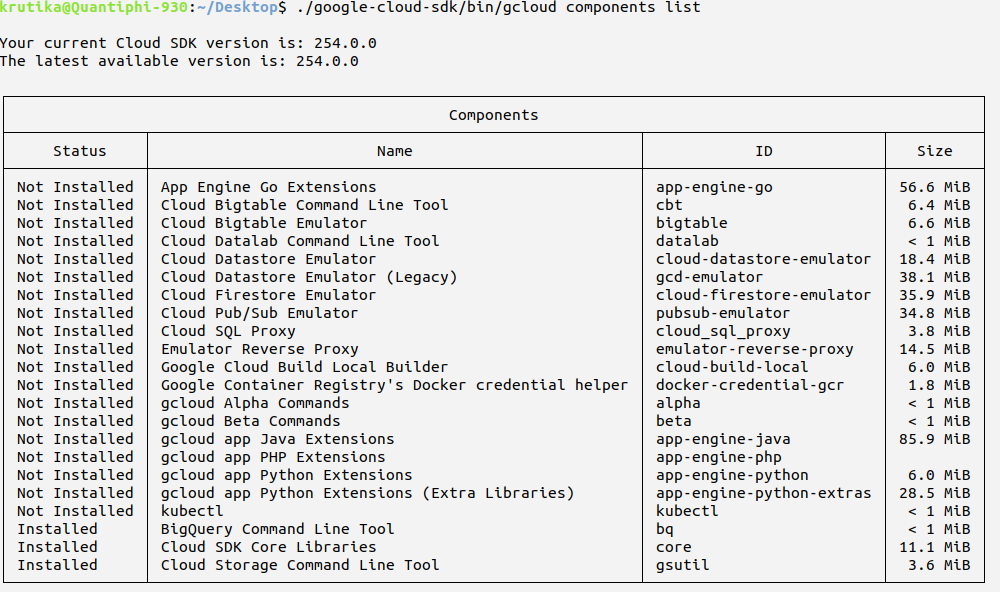
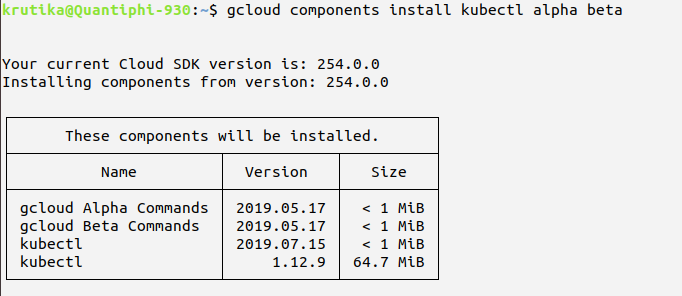
GCP Assessment 1

1. **Setup the gcloud sdk and provide the commands to check which of the gcloud components are installed. Install the components [gsutil, kubectl, bq, gcloud Alpha commands, gcloud beta commands] if not already installed.**

**gcloud components list**

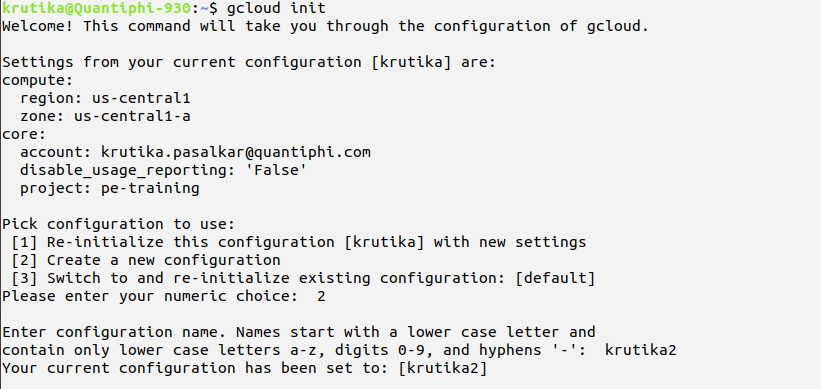


**gcloud components install kubectl alpha beta**

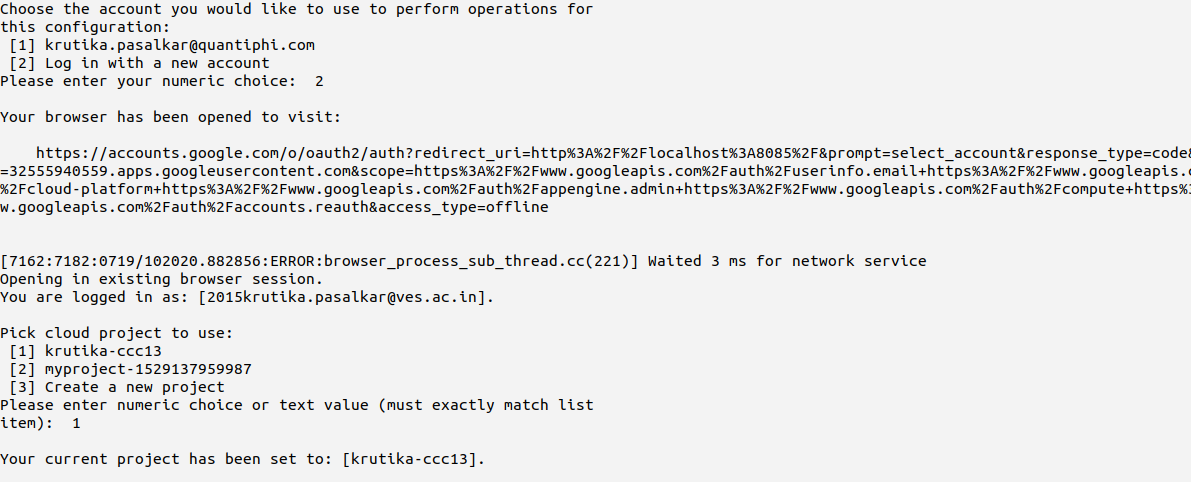


1. **Create multiple gcloud configurations with different configurations such as different email id’s or different projects etc. and list the commands to switch between two accounts**

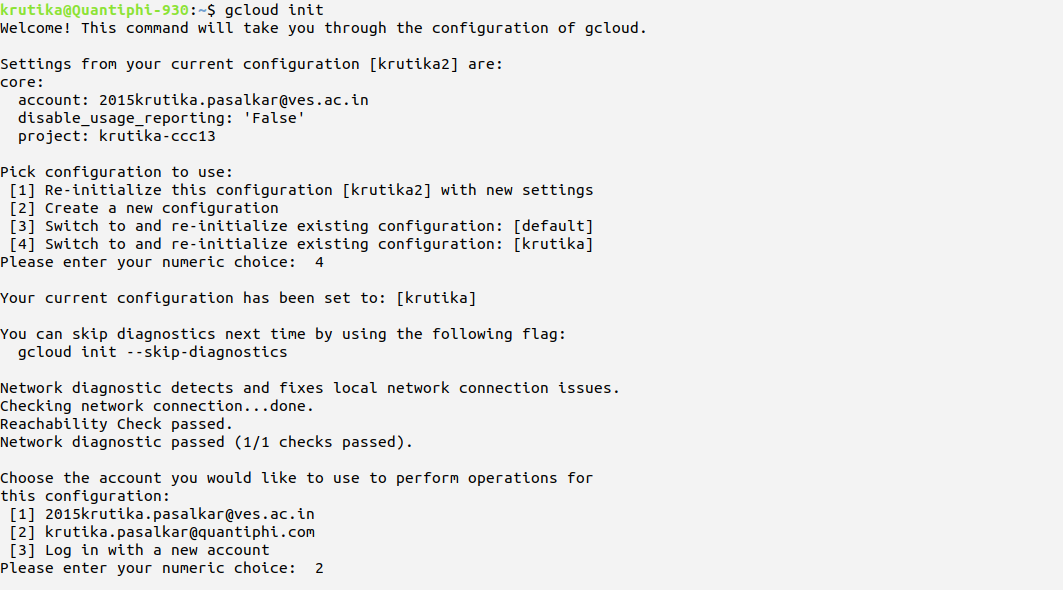
**Created new configuration krutika2**

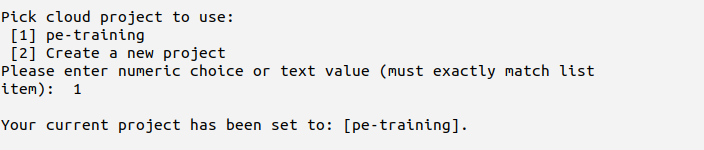


**Choosing new account in krutika2 and selecting new project in krutika2**

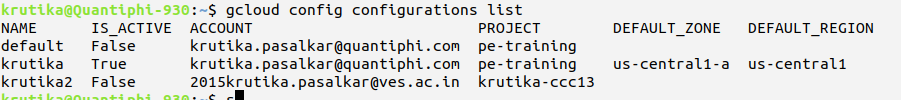


**Switching to krutika configuration and project pe-training**



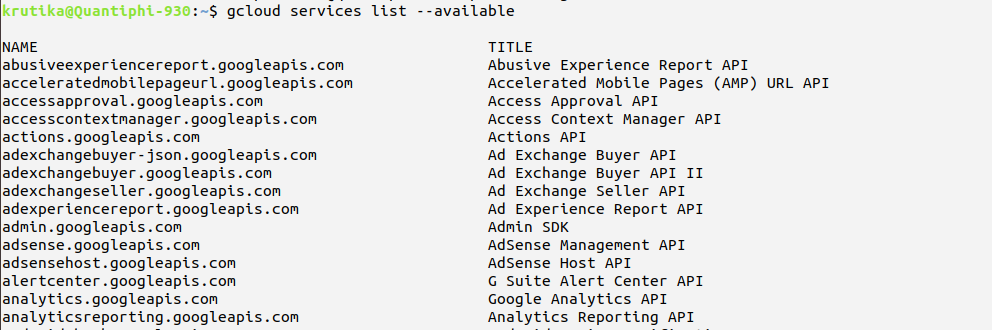


**List of configurations**

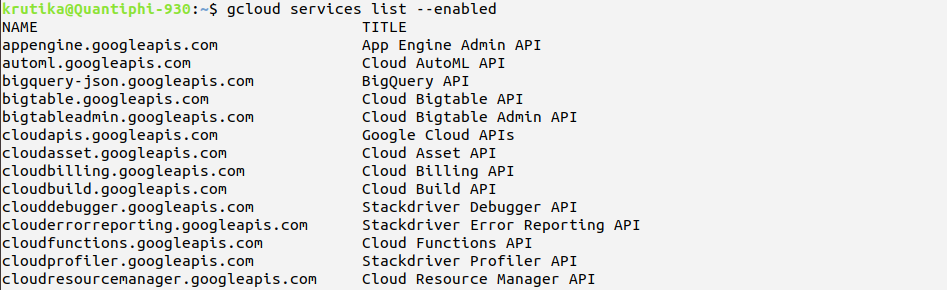


1. **Write commands to list out the available API Services and enabled API services**

**Available API services**



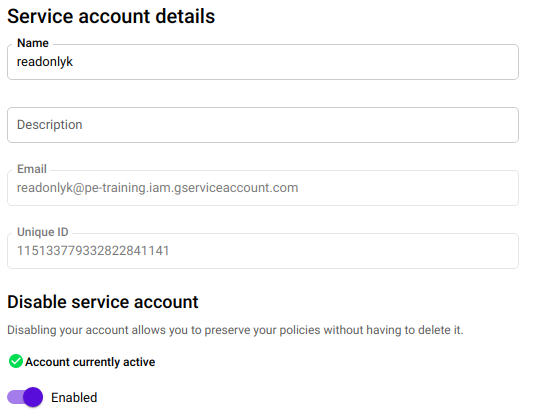
**Enabled API services**



1. **Create a GCE VM instance with a custom service account that has only read permissions to the cloud storage. Use console**

**To create new service account:**

* Goto IAM
* Goto service accounts -> create new service account
* Enter Service account details (name) -> in service account permissions choose role as **storage object viewer**
* Create service account



**Granting service account to vm instance:**

* In google compute engine -> go to VM instance
* Create new instance -> give name, select m/c type, cpu platform, boot disk
* In service account choose the service account created and create instance

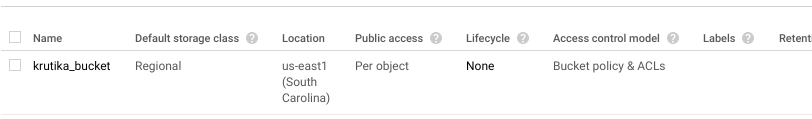
1. **Perform all basic kind of operations of objects (copy | move| upload | download | delete) using gsutil sdk and python client library**

**USING gsutil sdk**

**Create bucket:**

krutika@Quantiphi-930:~$ gsutil mb -l us-east1 gs://krutika\_bucket/

Creating gs://krutika\_bucket/…



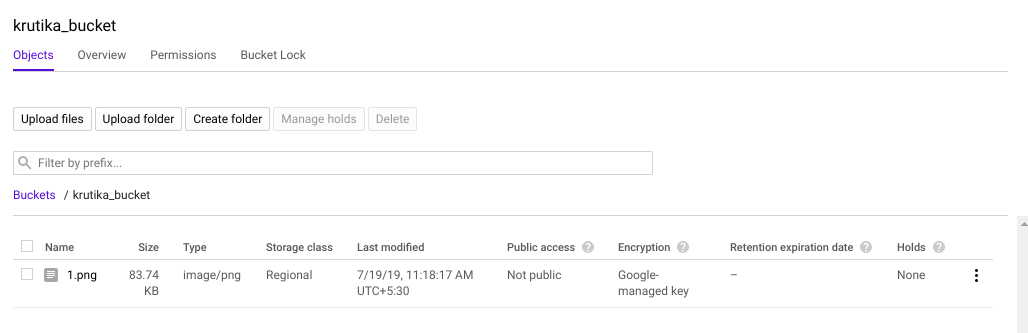
**Upload object in bucket:**

krutika@Quantiphi-930:~$ gsutil cp Desktop/1.png gs://krutika\_bucket

Copying file://Desktop/1.png [Content-Type=image/png]...

/ [0 files][ 0.0 B/ 83.7 KiB] / [1 files][ 83.7 KiB/ 83.7 KiB]

Operation completed over 1 objects/83.7 KiB.



**Download object from bucket:**

krutika@Quantiphi-930:~$ gsutil cp gs://krutika\_bucket/1.png Download/2.png

Copying gs://krutika\_bucket/1.png...

/ [0 files][ 0.0 B/ 83.7 KiB] / [0 files][ 83.7 KiB/ 83.7 KiB] - [1 files][ 83.7 KiB/ 83.7 KiB]

Operation completed over 1 objects/83.7 KiB.

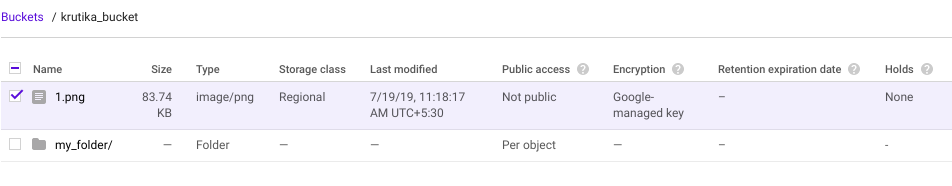
**Copy object in bucket:**

krutika@Quantiphi-930:~$ gsutil cp gs://krutika\_bucket/1.png gs://krutika\_bucket/my\_folder/2.png

Copying gs://krutika\_bucket/1.png [Content-Type=image/png]...

/ [0 files][ 0.0 B/ 83.7 KiB] - [1 files][ 83.7 KiB/ 83.7 KiB]

Operation completed over 1 objects/83.7 KiB.



**Move object:**

krutika@Quantiphi-930:~$ gsutil mb -l us-east1 gs://krutika2\_bucket/

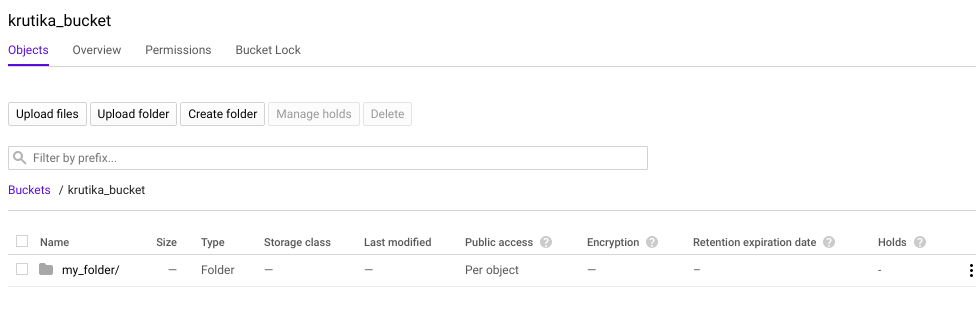
Creating gs://krutika2\_bucket/...

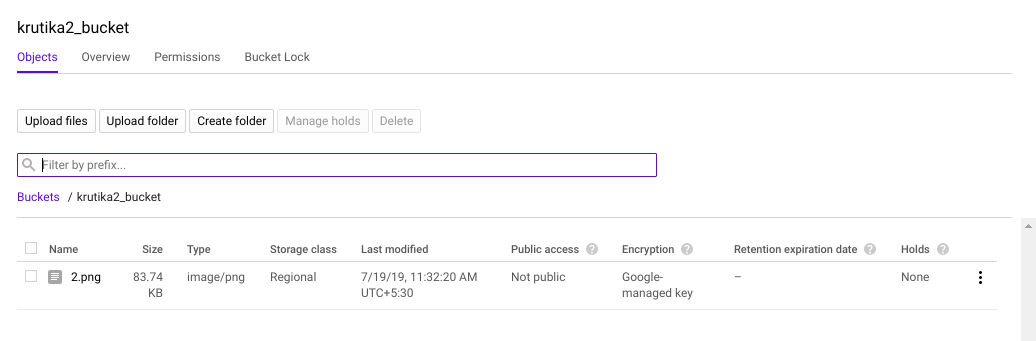
krutika@Quantiphi-930:~$ gsutil mv gs://krutika\_bucket/1.png gs://krutika2\_bucket/2.png

Copying gs://krutika\_bucket/1.png [Content-Type=image/png]...

/ [0 files][ 0.0 B/ 83.7 KiB] - [1 files][ 83.7 KiB/ 83.7 KiB] Removing gs://krutika\_bucket/1.png...

Operation completed over 1 objects/83.7 KiB.





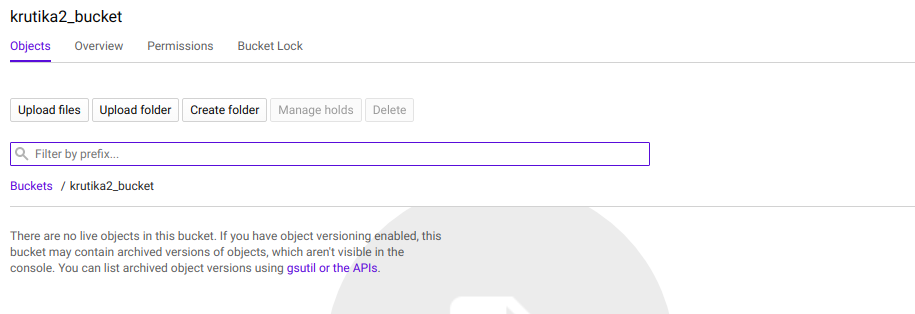
**Delete object in bucket:**

krutika@Quantiphi-930:~$ gsutil rm gs://krutika2\_bucket/2.png

Removing gs://krutika2\_bucket/2.png...

/ [1 objects]

Operation completed over 1 objects.



**USING python library**

Create role and download private keys

**Create bucket:**

# Imports the Google Cloud client library

from google.cloud import storage

# Instantiates a client

storage\_client = storage.Client()

# The name for the new bucket

bucket\_name = 'kru\_1'

# Creates the new bucket

bucket = storage\_client.create\_bucket(bucket\_name)

print('Bucket {} created.'.format(bucket.name))

**o/p:**

krutika@Quantiphi-930:~$ python q5\_1.py

Bucket kru\_1 created.

**Upload object in bucket:**

from google.cloud import storage

def upload\_blob(bucket\_name, source\_file\_name, destination\_blob\_name):

"""Uploads a file to the bucket."""

storage\_client = storage.Client()

bucket = storage\_client.get\_bucket(bucket\_name)

blob = bucket.blob(destination\_blob\_name)

blob.upload\_from\_filename(source\_file\_name)

print('File {} uploaded to {}.'.format(

source\_file\_name,

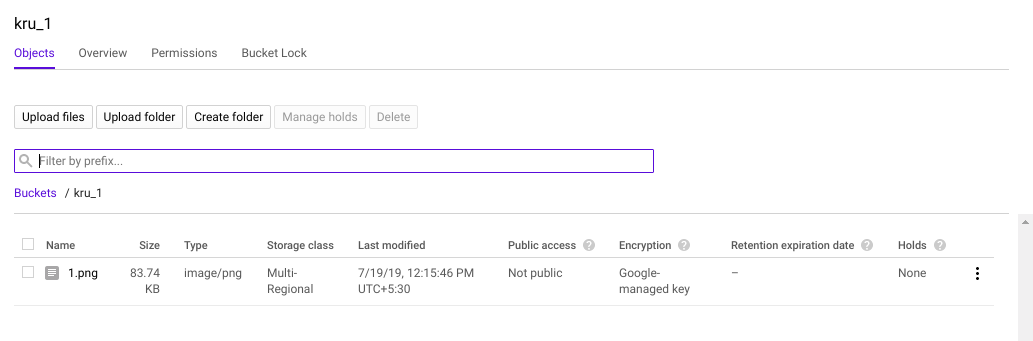
destination\_blob\_name))

upload\_blob('kru\_1','/home/krutika/Pictures/1.png','1.png')

**o/p:**

krutika@Quantiphi-930:~$ python q5\_2.py

File /home/krutika/Pictures/1.png uploaded to 1.png.



**Download object from bucket:**

from google.cloud import storage

def download\_blob(bucket\_name, source\_blob\_name,destination\_file\_name):

"""Downloads a blob from the bucket."""

storage\_client = storage.Client()

bucket = storage\_client.get\_bucket(bucket\_name)

blob = bucket.blob(source\_blob\_name)

blob.download\_to\_filename(destination\_file\_name)

print('Blob {} downloaded to {}.'.format(

source\_blob\_name,

destination\_file\_name))

download\_blob('kru\_1','1.png','/home/krutika/Pictures/download.png')

**O/p:**

krutika@Quantiphi-930:~$ python q2\_3.py

Blob 1.png downloaded to /home/krutika/Pictures/download.png.

**Copying object from one bucket to other:**

from google.cloud import storage

def copy\_blob(bucket\_name, blob\_name, new\_bucket\_name, new\_blob\_name):

"""Copies a blob from one bucket to another with a new name."""

storage\_client = storage.Client()

source\_bucket = storage\_client.get\_bucket(bucket\_name)

source\_blob = source\_bucket.blob(blob\_name)

destination\_bucket = storage\_client.get\_bucket(new\_bucket\_name)

new\_blob = source\_bucket.copy\_blob(

source\_blob, destination\_bucket, new\_blob\_name)

print('Blob {} in bucket {} copied to blob {} in bucket {}.'.format(

source\_blob.name, source\_bucket.name, new\_blob.name,

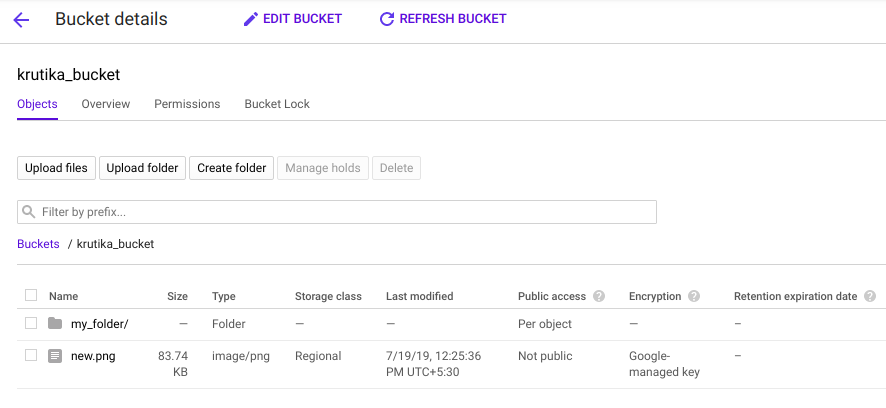
destination\_bucket.name))

copy\_blob('kru\_1','1.png','krutika\_bucket','new.png')

**o/p:**

krutika@Quantiphi-930:~$ python q5\_4.py

Blob 1.png in bucket kru\_1 copied to blob new.png in bucket krutika\_bucket.



**Deleting object from bucket:**

from google.cloud import storage

def delete\_blob(bucket\_name, blob\_name):

"""Deletes a blob from the bucket."""

storage\_client = storage.Client()

bucket = storage\_client.get\_bucket(bucket\_name)

blob = bucket.blob(blob\_name)

blob.delete()

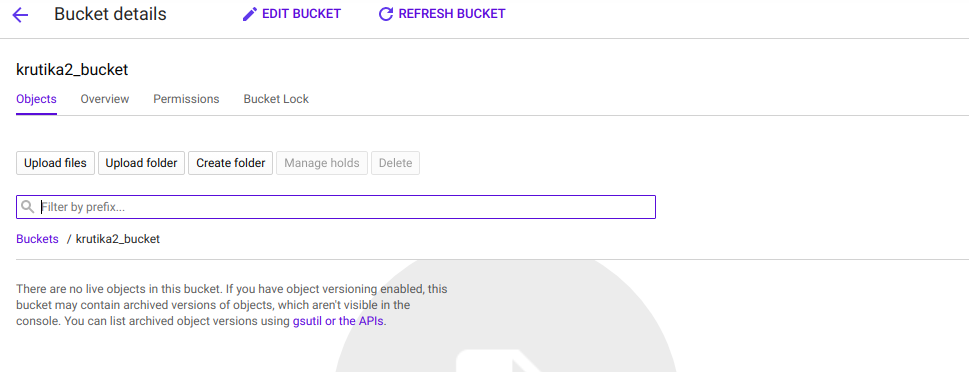
print('Blob {} deleted.'.format(blob\_name))

delete\_blob('krutika2\_bucket','gcp\_1\_a.png')

**o/p:**

krutika@Quantiphi-930:~$ python q5\_5.py

Blob gcp\_1\_a.png deleted.



**Move object from one bucket to another:**

from google.cloud import storage

def copy\_blob(bucket\_name, blob\_name, new\_bucket\_name, new\_blob\_name):

"""Copies a blob from one bucket to another with a new name."""

storage\_client = storage.Client()

source\_bucket = storage\_client.get\_bucket(bucket\_name)

source\_blob = source\_bucket.blob(blob\_name)

destination\_bucket = storage\_client.get\_bucket(new\_bucket\_name)

new\_blob = source\_bucket.copy\_blob(

source\_blob, destination\_bucket, new\_blob\_name)

def delete\_blob(bucket\_name, blob\_name):

"""Deletes a blob from the bucket."""

storage\_client = storage.Client()

bucket = storage\_client.get\_bucket(bucket\_name)

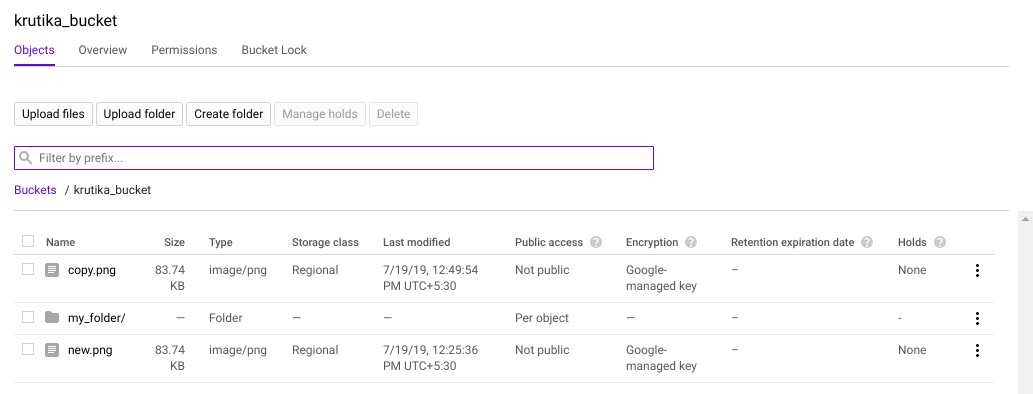
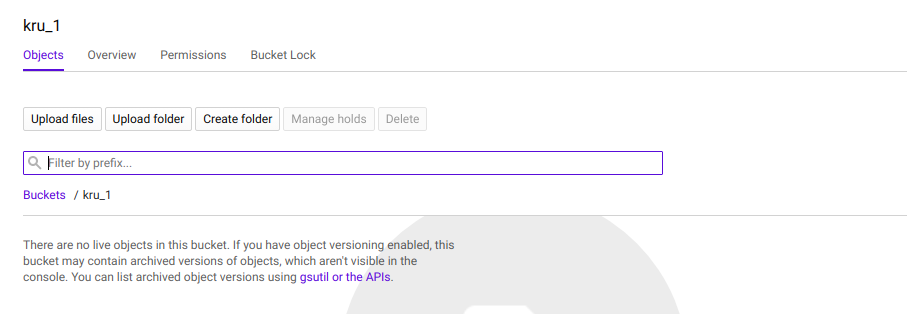
blob = bucket.blob(blob\_name)

blob.delete()

copy\_blob('kru\_1','1.png','krutika\_bucket','copy.png')

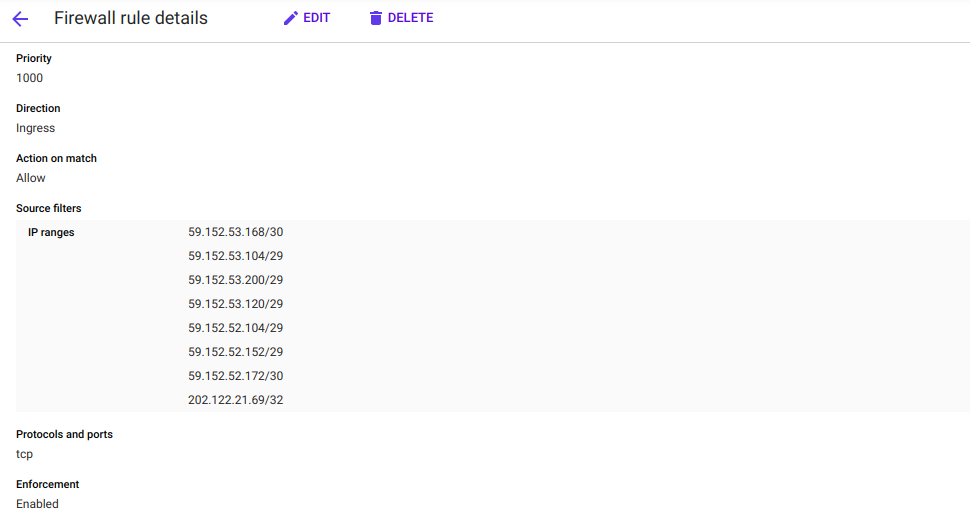
delete\_blob('kru\_1','1.png')

**o/p:**

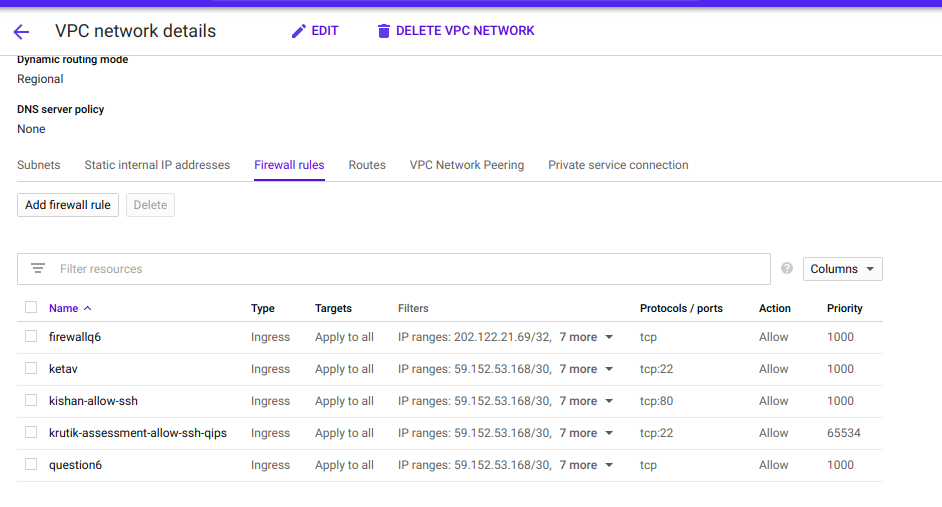


1. **Create new VPC network with automatic subnets. Configure the firewall rule to allow ssh access from only Quantiphi's IP. Use Console**

* Goto console -> VPC network
* Goto firewall rule -> create new firewall
* Give name, in source filter select IP ranges, give quantiphi IP ranges in source IP ranges, in protocols and port select tcp protocol

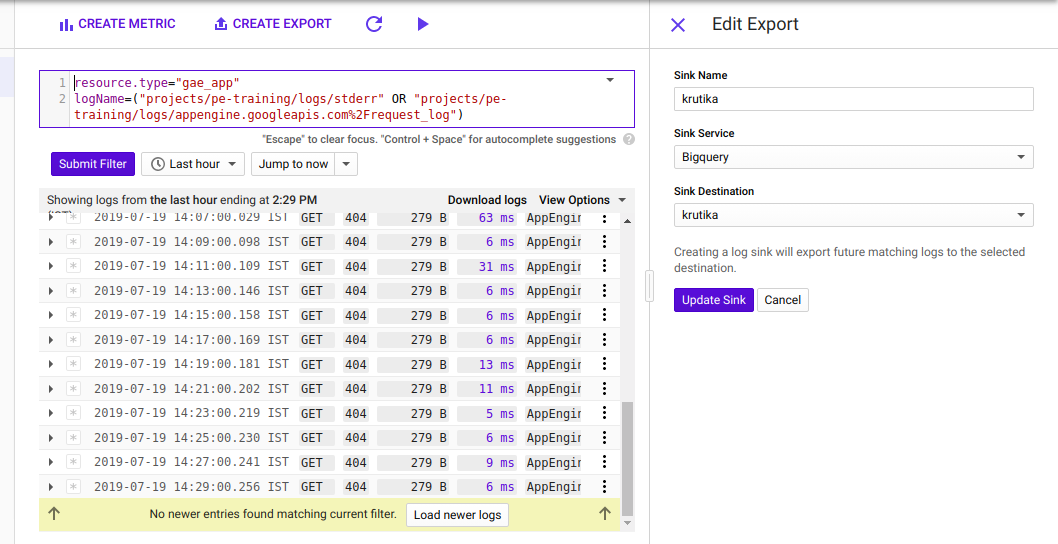


* Create new VPC network-> give name -> add firewall rules to vpc network



1. **Steps to export all the logs related to firewall rules to BigQuery for further analysis. Use console**

* Go to the Firewall rules page in the Google Cloud Platform Console.
* Select the firewall rule that we want to update.
* Click Edit.
* For the Logs setting, select On.
* Click Save.
* Goto Stackdriver -> logging -> logs viewer -> create export
* In Edit Export, give Sink Name, Sink Service-> bigquery, Sink Destination-> create new database.



1. **Create an auto scaling instance group that hosts the apache server that displays the name of the instance that has served the current request to the user. Stress the Autoscaling group artificially using the stress tool (sudo apt-get install stress) and demonstrate the group being autoscaled. Use cli/client library**

**Starter script:**

#!/bin/bash

apt-get update -y

apt-get install apache2 -y

apt-get install php7.0 -y

mv /var/www/html/index.html /var/www/html/index.php

cat <<EOF> /var/www/html/index.php

<html>

<body>HEllO WORLD<br></br>

<?php

echo gethostname();

?>

</body>

</html>

EOF

**For creating instance template:**

krutika@Quantiphi-930:~$ gcloud compute --project=pe-training instance-templates create krutika --machine-type=n1-standard-1 --network=projects/pe-training/global/networks/default --network-tier=PREMIUM --metadata=startup-script=\#\!/bin/bash$'\n'apt-get\ update\ -y$'\n'apt-get\ install\ apache2\ -y$'\n'apt-get\ install\ php7.0\ -y$'\n'mv\ /var/www/html/index.html\ /var/www/html/index.php$'\n'cat\ \<\<EOF\>\ /var/www/html/index.php$'\n'\<html\>$'\n'\<body\>HEllO\ WORLD\<br\>\</br\>$'\n'\<\?php$'\n'echo\ gethostname\(\)\;$'\n'\?\>$'\n'\</body\>$'\n'\</html\>$'\n'EOF --maintenance-policy=MIGRATE --service-account=912623308461-compute@developer.gserviceaccount.com --scopes=https://www.googleapis.com/auth/devstorage.read\_only,https://www.googleapis.com/auth/logging.write,https://www.googleapis.com/auth/monitoring.write,https://www.googleapis.com/auth/servicecontrol,https://www.googleapis.com/auth/service.management.readonly,https://www.googleapis.com/auth/trace.append --tags=http-server --image=debian-9-stretch-v20190618 --image-project=debian-cloud --boot-disk-size=10GB --boot-disk-type=pd-standard --boot-disk-device-name=krutika

WARNING: You have selected a disk size of under [200GB]. This may result in poor I/O performance. For more information, see: https://developers.google.com/compute/docs/disks#performance.

Created [https://www.googleapis.com/compute/v1/projects/pe-training/global/instanceTemplates/krutika].

NAME MACHINE\_TYPE PREEMPTIBLE CREATION\_TIMESTAMP

krutika n1-standard-1 2019-07-19T04:12:45.386-07:00

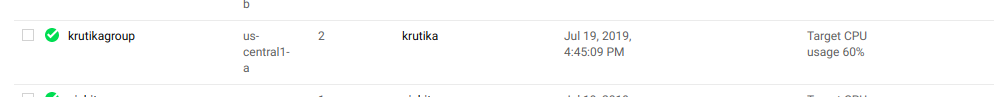
**For creating instance group:**

krutika@Quantiphi-930:~$ gcloud compute --project=pe-training instance-groups managed create krutikagroup --base-instance-name=krutikagroup --template=krutika --size=1 --zone=us-central1-a

Created [https://www.googleapis.com/compute/v1/projects/pe-training/zones/us-central1-a/instanceGroupManagers/krutikagroup].

NAME LOCATION SCOPE BASE\_INSTANCE\_NAME SIZE TARGET\_SIZE INSTANCE\_TEMPLATE AUTOSCALED

krutikagroup us-central1-a zone krutikagroup 0 1 krutika no



**For enabling autoscaling:**

krutika@Quantiphi-930:~$ gcloud beta compute --project "pe-training" instance-groups managed set-autoscaling "krutikagroup" --zone "us-central1-a" --cool-down-period "60" --max-num-replicas "3" --min-num-replicas "1" --target-cpu-utilization "0.6"

Created [https://www.googleapis.com/compute/beta/projects/pe-training/zones/us-central1-a/autoscalers/krutikagroup-jvx7].

---

autoscalingPolicy:

coolDownPeriodSec: 60

cpuUtilization:

utilizationTarget: 0.6

maxNumReplicas: 3

minNumReplicas: 1

creationTimestamp: '2019-07-19T04:15:33.279-07:00'

id: '8187163311492307130'

kind: compute#autoscaler

name: krutikagroup-jvx7

selfLink: https://www.googleapis.com/compute/beta/projects/pe-training/zones/us-central1-a/autoscalers/krutikagroup-jvx7

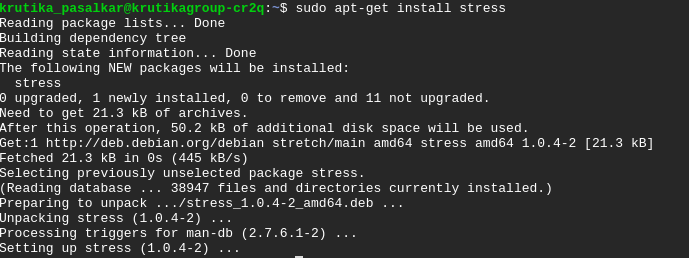
status: ACTIVE

target: https://www.googleapis.com/compute/beta/projects/pe-training/zones/us-central1-a/instanceGroupManagers/krutikagroup

zone: https://www.googleapis.com/compute/beta/projects/pe-training/zones/us-central1-a

**Ssh into the instance group**

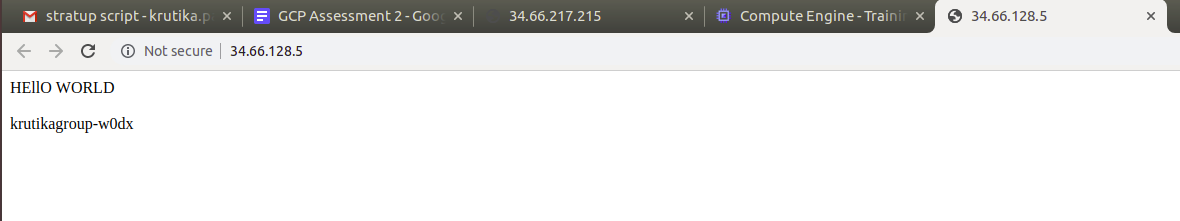
Install stress



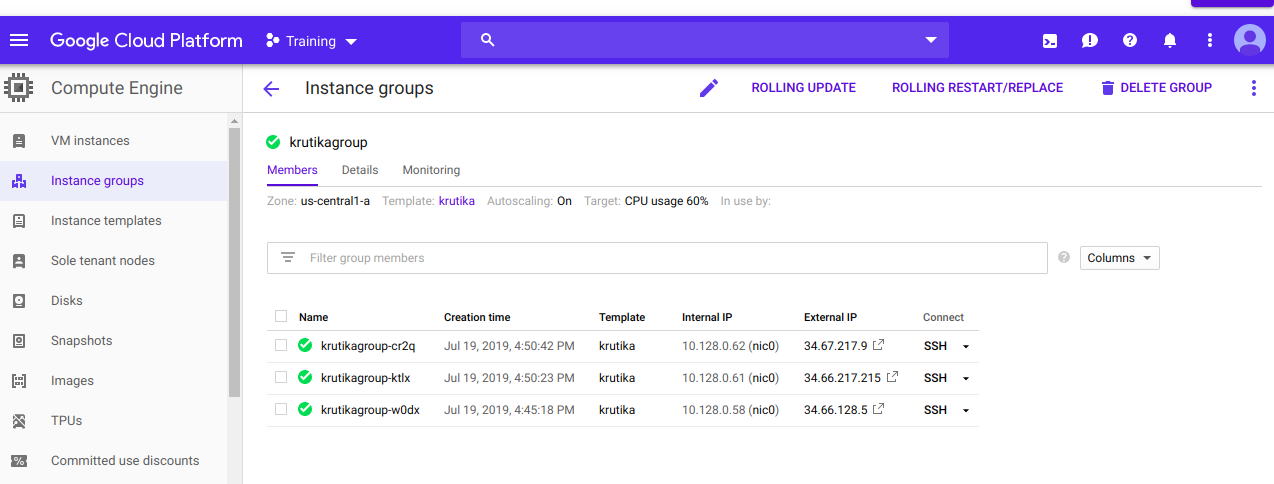
Apply stress for 60 seconds

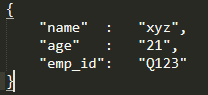


index.html



Autoscaling to 3 instances



1. **Create a Cloud Function which reads the messages from Pub/Sub subscription. The pubsub message contains the bucket name and object path of a json file. The Cloud Function should store the contents of the json file into an appropriate database. Use console. (Explore multiple ways of integrating cloud function with Pub/Sub)  
   Example contents of an object file(sample.json):**  
    

* Create above json file.

Create bucket and upload the json file.

* Go to Pubsub

Create topic

* Go to Big query

create dataset

enter the name

* Go to Cloud Function

Create a new function

Enter the name

Trigger: cloud pub/sub

Select the topic created above

runtime: python 3.7

Put the following code in main.py

import base64

from google.cloud import bigquery

def hello\_pubsub(event, context):

message = base64.b64decode(event['data']).decode('utf-8')

message\_list = message.splitlines()

bucket\_name = message\_list[0]

object\_path = message\_list[1]

print(bucket\_name)

print(object\_path)

client = bigquery.Client()

dataset\_ref = client.dataset('krutikaq9\_bq')

job\_config = bigquery.LoadJobConfig()

job\_config.autodetect = True

job\_config.source\_format = bigquery.SourceFormat.NEWLINE\_DELIMITED\_JSON

uri = "gs://" + bucket\_name + "/" + object\_path

load\_job = client.load\_table\_from\_uri(uri, dataset\_ref.table("gcptable"), job\_config=job\_config) # API request

print("Starting job {}".format(load\_job.job\_id))

load\_job.result()

Enter the following line in requirements.txt

google-cloud-bigquery

* Click create
* Go to Pubsub topic created

Publish message

Enter the bucket name <Enter>

Enter the object name (json file)

Publish the message

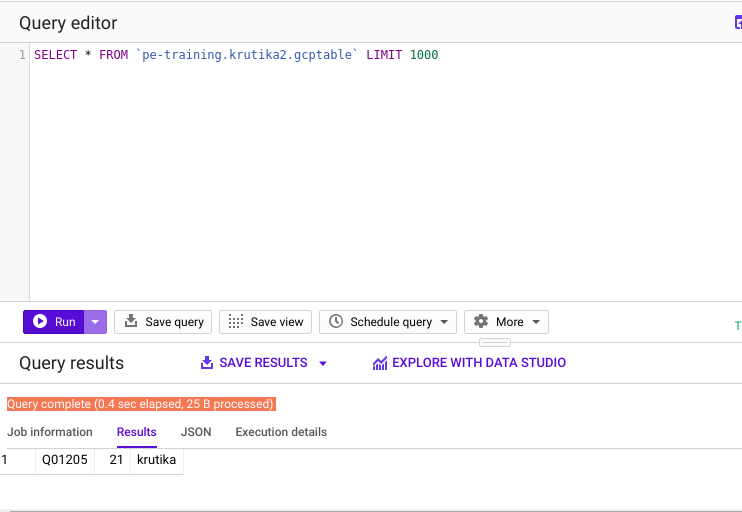
* Go to Big query

Go to table->gcp table will be created there

Click on query table->put \* after select

Run the query

* Table will be displayed



1. **Configure Apache2 HTTP server on a GCE VM instance and setup an email alert notification which triggers when the health check of the instance fails. Use console**

**Starter script:**

#!/bin/bash

apt-get update -y

apt-get install apache2 -y

apt-get install php7.0 -y

mv /var/www/html/index.html /var/www/html/index.php

cat <<EOF> /var/www/html/index.php

<html>

<body>HEllO WORLD<br></br>

</body>

</html>

EOF

1. In vm instance -> Create new vm instance
2. Create an instance which allows HTTP traffic and give starter script.
3. SSH the instance
4. Note the instance id
5. Open the external IP in browser to check if Apache is installed
6. Go to Stackdriver
7. Create alert policy
8. Add condition
9. Go to Process Healthcheck
10. Select resource type as single and add the instance-id
11. Click apply and save
12. Go to notifications
13. Select email and enter email-id
14. Click Add
15. Enter the resource policy name
16. Click save

