Experiment:1
Step 1: Sign in to Google Cloud Console
1. Go to Google Cloud Console: https://console.cloud.google.com/
2. Log in with your Google Account.
3. Select or create a new project from the top navigation bar.
Step 2: Open Compute Engine
1. In the left sidebar, navigate to "Compute Engine" → Click "VM instances".
2. Click "Create Instance".
Step 3: Configure the Virtual Machine
1. Name the VM
* Enter a name for your VM instance.
2. Select the Region and Zone
* Choose a region close to your target audience or users.
* Choose an availability zone (e.g., us-central1-a).
3. Choose the Machine Configuration
* Under "Machine Configuration", select:
o Series (E2, N1, N2, etc.)
o Machine type (Select based on your CPU & RAM needs)
: Example:
▪ e2-medium (2 vCPU, 4GB RAM)
▪ n1-standard-4 (4 vCPU, 16GB RAM)
▪ Click "Customize" if you want specific CPU & RAM.
4. Boot Disk (Operating System)
* Click "Change" under Boot Disk.
* Choose an Operating System (e.g., Ubuntu, Windows, Debian).
* Select disk size (e.g., 20GB or more).
5. Networking and Firewall
* Enable "Allow HTTP Traffic" or "Allow HTTPS Traffic" if needed.
* Click "Advanced options" for networking configurations.
Step 4: Create and Deploy the VM
1. Review all the configurations.
2. Click "Create" to deploy the VM.
3. Wait for the instance to be provisioned.
Step 5: Connect to the VM
1. Using SSH (Web)
* Go to Compute Engine → VM Instances.
* Click "SSH" next to your VM instance.
2. Using SSH (Terminal)
* Open Google Cloud SDK (Cloud Shell) or your local terminal.
* Run:
gcloud compute ssh your-instance-name --zone=us-central1-a
Step 6: Verify and Use the VM
* Check CPU and Memory:
lscpu # CPU details
free -h # Memory details
* Install required software (example: Apache web server)
sudo apt update && sudo apt install apache2 -y
Step 7: Stop or Delete the VM (Optional)
* Stop the VM:
gcloud compute instances stop your-instance-name --zone=us-central1-a
* Delete the VM:
gcloud compute instances delete your-instance-name –zone=us-central1-a

Experiment 3

Step 1: Enable Required APIs

Before deploying the Cloud Function, enable the necessary APIs:

gcloud services enable cloudfunctions.googleapis.com storage.googleapis.com

Step 2: Create a Cloud Storage Bucket

If you don't have a Cloud Storage bucket, create one:

gcloud storage buckets create BUCKET_NAME --location=us-central1

Replace BUCKET_NAME with a unique name for your bucket.

Step 3: Write the Cloud Function Code

1. Open Cloud Shell and create a working directory:

mkdir gcs-function && cd gcs-function

2. Create and open a new Python file (main.py):

nano main.py

3. Add the following code inside main.py:

```
import functions_framework
 @functions_framework.cloud_event
def gcs_trigger(cloud_event):
"""Triggered when a file is uploaded to Cloud Storage."""
data = cloud_event.data
bucket = data["bucket"]
file_name = data["name"]
print(f"
File {file_name} uploaded to {bucket}")
```

4. Save and close the file (CTRL + X, Y, Enter).

Step 4: Create a requirements.txt File

Create and open a requirements.txt file:

nano requirements.txt

Add the required dependency:

functions-framework

Save and close (CTRL + X, Y, Enter).

Step 5: Deploy the Cloud Function

Run the following command to deploy the function

gcloud functions deploy gcs_trigger \ --gen2 \ --runtime=python311 \ --region=us-central1 \ --source=. \ --entry-point=gcs_trigger \ --trigger-event-filters="type=google.cloud.storage.object.v1.finalized" \ --trigger-event-filters="bucket=BUCKET_NAME" \ --allow-unauthenticated

Replace BUCKET_NAME with your actual Cloud Storage bucket name.

Step 6: Test the Cloud Function

1. Upload a file to the Cloud Storage bucket:

gcloud storage cp test-file.txt gs://BUCKET_NAME

2. Check logs to verify function execution:

gcloud functions logs read gcs_trigger –region=us-central1

Experiment:2

Step 1: Open Cloud Shell

1. Sign in to the Google Cloud Console:
https://console.cloud.google.com/
2. Click the Cloud Shell icon (Terminal icon) in the top-right corner.
3. A terminal will open at the bottom of the page.

Step 2: Initialize gcloud CLI

1. Run the following command in Cloud Shell:
gcloud init
2. Follow the prompts to:
o Authenticate your Google account
o Select a Google Cloud project

Step 3: Verify gcloud Setup

To check if gcloud is properly configured, run:
gcloud config list
This displays your current project, account, and region settings.

Step 4: List Available Projects

Run the following command to view all projects associated with your Google account:
gcloud projects list

Step 5: Set Active Project

To set a specific project as the active one, run:
gcloud config set project PROJECT_ID
Replace PROJECT_ID with your actual project ID.

Step 6: Check Authentication Status

Run this command to verify that you're authenticated:
gcloud auth list
This will show the currently logged-in Google account.

Step 7: Create a Virtual Machine (VM) Instance

Launch a new Compute Engine VM instance:
gcloud compute instances create my-vm --zone=us-central1-a
* my-vm → Name of the instance
* --zone=us-central1-a → Choose a different zone if needed

Step 8: List Running VM Instances

To check all running VM instances, run:
gcloud compute instances list

Step 9: Delete a VM Instance

If you no longer need a VM, delete it using:
gcloud compute instances delete my-vm
Confirm the deletion when prompted.

Step 10: Enable an API (Example: Compute Engine API)

To enable an API, such as the Compute Engine API, run:
gcloud services enable compute.googleapis.com

Step 11: Deploy an Application to App Engine

If you have an application ready, deploy it using:
gcloud app deploy
Follow the instructions to deploy and access your app.

Step 12: View Active Billing Accounts

Check your billing accounts using:
gcloud beta billing accounts list

Step 13: Exit Cloud Shell

Simply close the Cloud Shell tab to exit.

Experiment:4

Step 1: Enable Required APIs

Before deploying your application, enable the App Engine API:

gcloud services enable appengine.googleapis.com

Step 2: Create an App Engine Application

Run the following command to create an App Engine application in your project:

gcloud app create --region=us-central1

You can replace us-central1 with another region if needed.

Step 3: Create a Simple Web Application

1. Open Cloud Shell and create a project directory:

mkdir app-engine-demo && cd app-engine-demo

2. Create a Python file (main.py):

nano main.py

3. Add the following simple Flask application code:

```
from flask import Flask
app = Flask(_name_)
@app.route('/')
def home():
return "
if _name_ == '_main_':
Welcome to Google App Engine with Auto Scaling!"
app.run(host='0.0.0.0', port=8080)
```

4. Save and close (CTRL + X, Y, Enter).

Step 3: Create a Simple Web Application

1. Open Cloud Shell and create a project directory:

mkdir app-engine-demo && cd app-engine-demo

2. Create a Python file (main.py):

nano main.py

3. Add the following simple Flask application code:

```
from flask import Flask
app = Flask(_name_)
@app.route('/')
def home():
return "
if _name_ == '_main_':
Welcome to Google App Engine with Auto Scaling!"
app.run(host='0.0.0.0', port=8080)
```

4. Save and close (CTRL + X, Y, Enter).

Step 3: Create a Simple Web Application

1. Open Cloud Shell and create a project directory:

mkdir app-engine-demo && cd app-engine-demo

2. Create a Python file (main.py):

nano main.py

3. Add the following simple Flask application code:

```
from flask import Flask
app = Flask(_name_)
@app.route('/')
def home():
return "
if _name_ == '_main_':
Welcome to Google App Engine with Auto Scaling!"
```

```
app.run(host='0.0.0.0', port=8080)
```

4. Save and close (CTRL + X, Y, Enter).

Experiment:5

Step 1: Open Google Cloud Console

1. Go to Google Cloud Console.
2. If not already logged in, sign in with your Google account.
3. Ensure that you have an active Google Cloud Project.

o If not, click on the project dropdown (top bar) and select an existing project or create a new project.

Step 2: Enable Cloud Storage API (If Not Enabled)

left.

1. In the Google Cloud Console, click the Navigation Menu (☰) on the top
2. Go to APIs & Services → Library.
3. Search for Cloud Storage API.
4. Click Enable if it is not already enabled.

Step 3: Create a Cloud Storage Bucket

1. In the Navigation Menu (☰), go to Storage → Buckets.
2. Click Create.
3. Enter a globally unique bucket name (e.g., your-unique-bucket-name).
4. Choose a Location (e.g., us-central1 for the USA).
5. Select a Storage Class (Choose based on your needs):

o Standard (Frequent access, low latency)
o Nearline (Access once a month)
o Coldline (Rare access, backup storage)
o Archive (Long-term storage)

6. Choose Access Control:

o Fine-grained (More detailed control)

7. Click Create.

o Uniform (Simpler access control)

Your bucket is now ready!

Step 4: Upload a File to the Bucket

1. Open your bucket from Storage → Buckets.
2. Click the Upload Files button.
3. Select a file from your computer and click Open.
4. Wait for the file to upload.

Your file is now stored in Cloud Storage!

Step 5: Download a File from the Bucket

1. Open your bucket in Storage → Buckets.
2. Click on the file you want to download.
3. Click Download to save the file to your computer.

Step 6: Make a File Public (Optional)

1. Open your bucket and click on the file.
2. Click the Permissions tab.
3. Click Add Principal.
4. In the New Principals field, enter:

allUsers

5. Select the Role:

o Storage Object Viewer (roles/storage.objectViewer)

6. Click Save.

Now your file is publicly accessible!

You will see a public URL like:

https://storage.googleapis.com/your-unique-bucket-name/your-file-name

Anyone can access the file using this link.

Step 7: Delete a File or Bucket (Optional)

* To delete a file, click on the file and select Delete.
* To delete a bucket, go to Storage → Buckets, select the bucket, and click Delete.
(You must first delete all files inside before deleting the bucket.)

Experiment:6

Key Features of Cloud SQL for MySQL

Automated Management

Automatic Backups – Cloud SQL provides daily automated backups and point-in-time recovery.

Automatic Updates & Patching – Google automatically applies security patches.

standby node if the primary fails.

High Availability (HA)

standby instance.

Automatic Failover – High-availability instances automatically switch to a

Regional Replication – Cloud SQL offers multi-zone high availability.

Failover Support – If a zone fails, the system automatically switches to a

Read Replicas – You can create read replicas for load balancing and performance improvement.

Security & Compliance

IAM-Based Access Control – Secure access via Identity and Access Management (IAM).

Encryption – Data is encrypted at rest and in transit.

VPC Peering & Private IPs – Secure database connections using private networking.

Scalability & Performance

Automatic Storage Increase – If storage runs out, Cloud SQL expands automatically.

Vertical Scaling – You can increase CPU and memory as needed.

Read Replicas – Scale reads by distributing queries across replicas.

Set Up Cloud SQL for MySQL

Step 1: Enable Cloud SQL API

1. Open Google Cloud Console.
2. Go to APIs & Services → Library.
3. Search for Cloud SQL Admin API and click Enable.

Step 2: Create a Cloud SQL for MySQL Instance

1. Go to Navigation Menu (☰) → SQL.
2. Click Create Instance → Choose MySQL.
3. Set:
   - o Instance ID (e.g., my-mysql-instance)
   - o Password (for root user)
   - o Region & Zone (choose near your app)
   - o Machine Type (choose appropriate CPU & RAM)
   - o Storage Capacity (set auto-increase if needed)
4. Click Create and wait for the instance to initialize.

Step 3: Connect to Cloud SQL

Using Cloud Console

1. Open SQL → Click on your instance.
2. Under Connections, find Public IP or Private IP.
3. Use the Cloud SQL Auth Proxy or MySQL client to connect.

Using MySQL Client

```
gcloud sql connect my-mysql-instance --user=root
```

or

```
mysql -u root -p -h [INSTANCE_IP]
```

Replace [INSTANCE_IP] with the actual instance IP.

Using Django/Flask

```
DATABASES = {
'default': {
'ENGINE': 'django.db.backends.mysql',
'NAME': 'your-db-name',
'USER': 'root',
'PASSWORD': 'your-password',
'HOST': '/cloudsql/your-project-id:your-region:your-instance',
'PORT': '3306',
}
}
```

Step 4: Enable High Availability (HA) (Optional)

1. Open your instance → Click Edit.
2. Enable High Availability and select a standby zone.
3. Save changes.

Step 5: Create a Read Replica (Optional)

1. Open your instance → Click Create Read Replica.
2. Select the region and name.
3. Click Create.

Step 6: Backup & Restore

Enable Automated Backups

1. Open your instance → Click Backups.
2. Click Edit → Enable automatic backups.

Manually Create a Backup

```
gcloud sql backups create --instance=my-mysql-instance
```

Restore from Backup

```
gcloud sql backups restore BACKUP_ID –instance=my-mysql-instance
```

Experiment:7

Google Cloud Pub/Sub is a fully managed messaging service that enables asynchronous, real-time communication between distributed applications. It follows a publish-subscribe model where publishers send messages to topics and subscribers receive them via push or pull delivery.

Why Use Cloud Pub/Sub?

Real-time messaging – Delivers messages instantly across services.
Decouples components – Microservices can communicate asynchronously.
High availability & scalability – Handles millions of messages per second.
Guaranteed delivery – Retries messages until they are acknowledged.
Security – Integrated with IAM for access control.

Cloud Pub/Sub Architecture

Publisher – Sends messages to a Topic.
Topic – A named channel where messages are published.
Subsc…

Step-by-Step: Experimenting with Cloud Pub/Sub

Step 1: Enable Cloud Pub/Sub API

1. Open Google Cloud Console → Navigation Menu (☰) → APIs & Services → Library.
2. Search for Cloud Pub/Sub API and click Enable.

Step 2: Create a Pub/Sub Topic

1. Go to Navigation Menu (☰) → Pub/Sub → Topics.
2. Click Create Topic.
3. Enter a Topic ID (e.g., my-topic).
4. Click Create.

Your topic is now ready!

Step 3: Create a Subscription

1. Click on your Topic → Create Subscription.
2. Enter a Subscription ID (e.g., my-subscription).
3. Choose a Delivery Type:

o Pull – Messages are manually fetched by the subscriber.
o Push – Messages are automatically sent to an HTTP endpoint.

4. Click Create.

Your subscription is now linked to the topic!

Step 4: Publish a Message (Using gcloud CLI)

Run the following command in Cloud Shell:

```
gcloud pubsub topics publish my-topic --message "Hello, Pub/Sub!"
```

Message published successfully!

Step 5: Pull Messages from Subscription (Using gcloud CLI)

Run the following command:

```
gcloud pubsub subscriptions pull my-subscription --auto-ack
```

This will retrieve and acknowledge messages from my-subscription.
You will see the message: Hello, Pub/Sub!

Step 6: Publish & Subscribe Using Python (Optional)

Install Google Cloud Pub/Sub SDK

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

Publisher Code (Python)

```
from google.cloud import pubsub_v1
project_id = "your-project-id"
topic_id = "my-topic"
publisher = pubsub_v1.PublisherClient()
topic_path = publisher.topic_path(project_id, topic_id)
```

```
message = "Hello, Pub/Sub from Python!"
future = publisher.publish(topic_path, message.encode("utf-8"))
print(f"Published message ID: {future.result()}")
```

Subscriber Code (Python)

```
from google.cloud import pubsub_v1
project_id = "your-project-id"
subscription_id = "my-subscription"
subscriber = pubsub_v1.SubscriberClient()
subscription_path = subscriber.subscription_path(project_id, subscription_id)
def callback(message):
print(f"Received: {message.data.decode('utf-8')}")
 message.ack()
subscriber.subscribe(subscription_path, callback=callback)
print("Listening for messages...")
import time
while True:
time.sleep(10)
```

Now, whenever you publish a message, the subscriber will receive it in real-time!