Experiment-01

1. Creating a virtual machine: Configure and deploy a virtual machine with specific CPU and memory requirements on Google cloud.

⇒

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. (eg:- us-central-a).

3. choose the Machine Configuration

- Under "Machine Configuration", select:
  - series (E2, N1, N2, etc.)
  - 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 HTTP 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
```

04/03/23

2. Getting started with Cloud shell and gcloud: Discover the use of gcloud commands to manage Google cloud resources from cloud shell.

⇒

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:
   - Authenticate your Google account
   - 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 delection 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.googleapies.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: ~~Exit~~ View Active Billing Accounds

check your billing accounts using:

gcloud beta billing account list

Step 13: ~~Exit~~ cloud Shell

simply close the cloud shell tab to exit

18/03/25

3. Cloud Functions: Create and deploy a Cloud Function to automate a specific task based on a Cloud Storage event.

=>

#. 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="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
```

25/03/25

2. check logs to verify function execution:

```
gcloud functions logs read gcs-trigger --region=us-central1
```

Experiment - 04

4. App Engine: Deploy a web application on App Engine with automatic scaling enabled.

=>

#. Step 1: Enable Required API's

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 sample Flask application code:

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return "Welcome to google App Engine with Auto scaling!";

if __name__ == '__main__':
    app.run(host='0.0.0.0', Port=8080)
```

4. SAVE and Close (CTRL + X, Y, Enter).

Step 4 : Create a requirements.txt file

Create and open a requirements.txt file:

```
nano requirements.txt
```

Add the following dependencies:

```
Flask
gunicorn
```

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

#. Step 5 : Create an app.yaml file for App Engine

Create an app.yaml file to configure App Engine:

nano app.yaml

Add the following content:

```
runtime: python311

entrypoint: gunicorn -b :$PORT main:app

automatic_scaling:

min_instances: 1

max_instances: 5

target_cpu_utilization: 0.65

target_throughput_utilization: 0.75
```

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

Step 6 : Deploy the web Application

Run the following command to deploy the application:

gcloud app deploy

- Confirm the deployment when prompted (Y).

Step 7: Access the Deployed Application

once deployed, open your web app in a browser:

gcloud app browse

This will return a URL (e.g. https://your-Project-id.appspot.com), where you can view your running app.

Step 8 : View Logs and Monitor Scaling

Check the logs of your application:

gcloud app logs tail -s default

Monitor the deployed services:

gcloud app services list

5. cloud storage: Quickstart: Google Cloud Storage provides scalable and secure object storage for managing data, accessible via the cloud Console or gsutil CLI.

⇒

#. 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.
   - If not, click on the Project dropdown (top bar) & select an existing Project or create a new Project.

#. Step 2 : Enable cloud Storage API (If Not Enabled)

1. In the Google cloud console, click the Navigation menu (☰) on the top left.
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):
   - Standard (Frequent access, low latency)
   - Nearline (Access once a month)
   - Coldline (Rare access, backup storage)
   - Archive (Long-term storage)
6. Choose Access Control:
   - Fine-grained (more detailed control)
   - Uniform (simpler access control)
7. Click Create.

☑ 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:
   - 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 → Bucket, select the bucket and click Delete.
  (You must first delete all the files inside before deleting the Bucket.)

08/04/25

Experiment-06

6. Cloud SQL for MySQL : Discover how Google cloud SQL for MySQL Provide automated management and high availability for MySQL databases?

=>

#. 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:
   - Instance ID (e.g, my-mysql-instance)
   - Password (for root user)
   - Region & Zone (choose near your app)
   - Machine Type (choose appropriate CPU & RAM)
   - 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.

15/04/25

7. Cloud Pub/Sub : Experiment how Google cloud Pub/Sub facilitate real-time messaging and communication between distributed applications.

⇒

#. 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 (eg, my-subscription).

3. Choose a Delivery Type:

o Pull - messages are manually fetched by the subscriber.

- 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, PubSub!"
```

☑ 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 retrive and acknowledge message 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()

subscriber_path = subscriber.subscription_path(project_id, subscrip
                  -tion_id)

def callback(message):

    PrintCf"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)
```

22/04/25

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

Experiment-08

8. Multiple VPC Networks: Explore benefits of using multiple VPC networks in Google cloud for organizing and isolating resources.

⇒

Step 1: Create a VPC Network

1. Go to Google Cloud ~~Network~~/Console - Navigation Menu (☰) → VPC Network → VPC Network
2. Specify the name, region, and subnet configuration for your VPC.
3. Click create.

Step 2: Create Additional VPC Networks

1. You can repeat the process to create as many VPCs as needed.
2. Choose Custom subnet mode to define your own subnets or Auto mode for auto-assigned subnets.

Step3: Set up VPC Peering (optional)

1. Go to VPC Network Peering → Create Peering Connection
2. Select the Source VPC and Destination VPC.
3. Define the network and routes that can be shared across the VPCs.
4. Click Create.

Step 4 : Create Firewall Rules (Optional)

1. Go to Firewall Rules → Create Firewall Rule.

2. Define the source and destination VPCs, and configure the firewall to allow or deny traffic b/w the VPCs.

Step 5 : set up Shared VPC (Optional)

1. Go to VPC Networks → Shared VPC → set up a shared VPC.

2. Select the host project and service projects.

3. Share the VPC resources with other projects.

29/04/25

9. Cloud Monitoring: Discover how cloud monitoring help in tracking and analyzing the performance and health of cloud resources?

⇒

#. Step 1: Enable Cloud Monitoring API

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

#. Step 2: Set Up Monitoring for Your Resources

1. Go to Navigation menu (☰) → Monitoring → Dashboards.

2. Click on create Dashboard to start building your custom dadabord.

3. Select Metrics from the dropdown and choose the service you want to monitor (e.g. Compute Engine, Cloud Storage, Cloud Functions).

4. Add the required metrics to your dashboard and adjust visualizations like line charts, heat maps, or bar charts.

#. Step 3: Set UP Alerts

1. Go to Navigation Menu (☰) → Monitoring → Alerting.

2. Click Create Policy.

3. Choose the condition (e.g. CPU usage > 80%).

4. Select the notification channels (e.g. email, Slack, SMS) where the alert should be sent.

5. Set up escalation policies to ensure the right team is notified.

6. Click create to finish the alert policy.

Step 4 : Review Logs

1. Go to Navigation Menu (☰) → Logging.

2. Choose the resource type (e.g. VM instances, Kubernets Engine) and define your log filter.

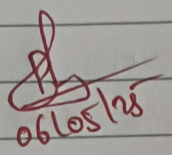
3. Use Log Explorer to search for specific logs, such as error message or performance warnings, and correlate them with metrics in cloud monitoring.

Step 5 : set up Distributed Tracing (optional)

1. Open Cloud Trace from the Navigation Menu.

2. Enable trace collection on your services (e.g. by using the Cloud Trace SDK for your app).

3. View trace data to identify latency or bottlenecks in your system.



Experiment-10

10. Kubernets Engine: Quick start: Deploy a containerized applicati -on to a kubernetes Engine cluster.

=>

Steps to Deploy a containerized Application to GKE

set up Google cloud SDK and kubernetes Tools

1. Install Google cloud SDK

If you haven't already installed the Google cloud SDK, follow the instructions here:

Google cloud SDK Installation

2. Install kubectl

Kubectl is the kubernetes command-line tool used to manage kubernetes clusters. The cloud SDK includes kubectl, so if you have the SDK intalled, you already have kubectl.

Create a Google cloud Project

1. Create a new Google cloud Project (if you don't already have one):

- Go to the Google cloud Console.

- Click on Select a Project > New Project
- Name your Project and click Create.

2. set your Project in the gcloud CLI:

gcloud config set project PROJECT_ID

Enable Required APIs

1. Enable Kubernetes Engine API:

gcloud services enable container.googleapis.com

2. Enable Compute Engine API (if not aleardy enabled):

gcloud services enable compute.googleapis.com

#. Create a kubernetes cluster

1. Create the Kubernetes Engine cluster:

gcloud container clusters create my-cluser --zone us-central-a

Replace my-cluster with your desired cluster name and adjust the zone if necessary.

2. Get the credentials for your cluster: This command configures kubectl to use the cluster you just created.

gcloud container clusters get-credentials my-cluster --zone us-central1-a

Create a containerized Application

1. Create a Dockerfile for your application. Below is an example for a simple web application using Node.js:

Dockerfile:

dockerfile

Copy Edit

```
FROM node:14

WORKDIR /usr/src/app

COPY..

RUN npm install

EXPOSE 8080
```

CMD ["npm", "start"]

2. Build the Docker image:

docker build -t gcr.io/PROJECT_ID/my-app:v1.

Replace PROJECT_ID with your Google Cloud Project ID

3. Push the image to Google Container Registry:

docker push gcr:io/PROJECT-ID/my-app:v1

#. Create a Kubernetes Deployment

1. Create a Kubernetes Deployment configuration file (deployment.yaml) for your containerized application.

deployment.yaml:

apiVersion: apps/v1

Kind: Deployment

metadata:

name: my-app

```
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
        - name: my-app
          image: gcr.io/PROJECT_ID/my-app:v1
          ports:
```

```
- containerPort: 8080
```

Replace PROJECT_ID with your project ID.

2. Apply the Deployment to the kubernetes cluster:

```
kubectl apply -f deployment.yaml
```

Expose the Application via a service

1. Create a service to expose the application (either Load-Balancer or ClusterIP for internal access.).

Example service.yaml for external exposure (Load Balancer Type):

```
apiVersion: v1

kind: Service

metadata:

name: my-app-service

Spec:

 Selector:
```

app: my-app

Ports:

- Protocol: TCP

Port: 80

targetPort: 8080

type: LoadBalancer

2. Apply the service configuration:

kubectl apply -f service.yaml

3. Get the external IP address: It may take a few moments for the LoadBalancer to be provisioned.

kubectl get svc

The EXTERNAL-IP column will show the public IP once the LoadBalancer is provisioned.

4. Verify the Application

1. Open a web browser and navigate to the external IP address to verify your application is running.

2. You can also use kubectl to get the status of your Pods and Services:

```
kubectl get pods

kubectl get svc
```