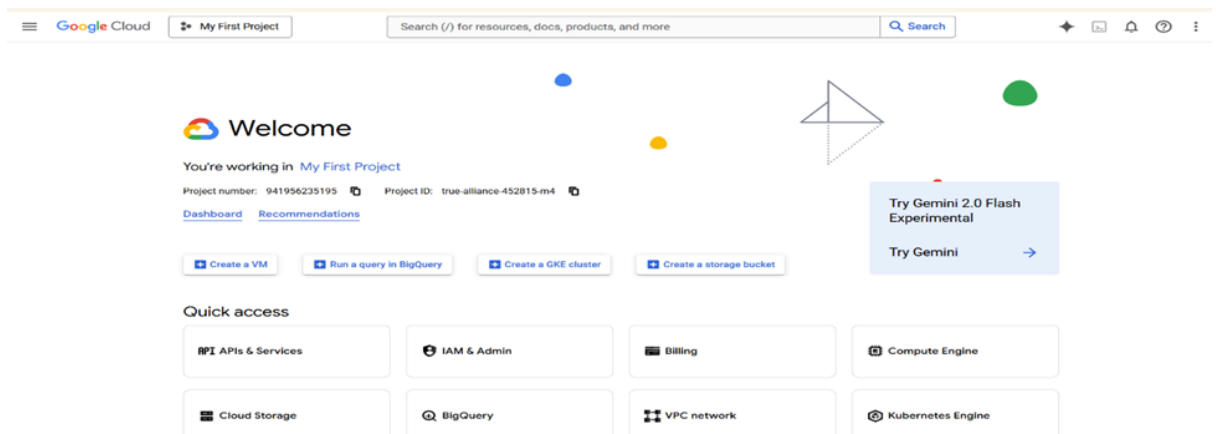


EXPERIMENT NUMBER 1

Creating a Virtual Machine: Configure and deploy a virtual machine with specific CPU and memory requirements in Google Cloud

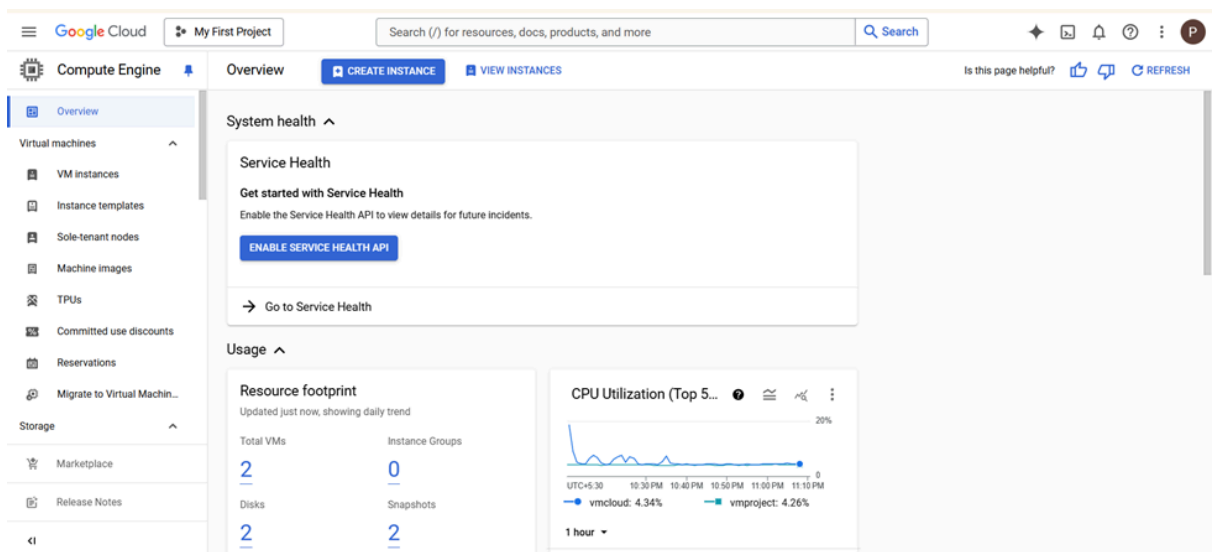
Step 1: Log in to Google Cloud Console

1. Go to Google Cloud Console.
2. Make sure you have a **Google Cloud account** and a **project** set up.



Step 2: Open Compute Engine

1. In the **Navigation Menu**, go to **Compute Engine > VM instances**.
2. Click **Create Instance**.



Step 3: Configure VM Instance

- **Name:** Give your VM a recognizable name.
- **Region & Zone:** Choose a region (e.g., us-central1) based on your needs.
- **Machine Configuration:**
 - Click **Machine type**.
 - Choose a predefined machine type or customize:
 - **Custom (vCPUs & Memory):** Set the desired number of CPUs and RAM.
 - Example: If you need **4 vCPUs and 16 GB RAM**, select a **custom machine type** and set:
 - vCPU: 4
 - Memory: 16 GB

The screenshot shows the 'Machine configuration' step of the 'Create an instance' wizard. The 'Name' field is set to 'projectvm'. The 'Region' is 'us-central1 (Iowa)' and the 'Zone' is 'us-central1-a'. The 'General purpose' tab is selected. A table of machine types is displayed, with 'E2' selected. The 'Monthly estimate' on the right is \$25.46.

Series	Description	vCPUs	Memory	CPU Platform
C4	Consistently high performance	2 - 192	4 - 1,488 GB	Intel Emerald Rapids
C4A	Arm-based consistently high performance	1 - 72	2 - 576 GB	Google Axion
N4	Flexible & cost-optimized	2 - 80	4 - 640 GB	Intel Emerald Rapids
C3	Consistently high performance	4 - 192	8 - 1,536 GB	Intel Sapphire Rapids
C3D	Consistently high performance	4 - 360	8 - 2,880 GB	AMD Genoa
E2	Low cost, day-to-day computing	0.25 - 32	1 - 128 GB	Intel Broadwell
N2	Balanced price & performance	2 - 128	2 - 864 GB	Intel Cascade Lake
N2D	Balanced price & performance	2 - 224	2 - 896 GB	AMD Milan

Step 4: Select Boot Disk

- Choose an OS (e.g., **Ubuntu, Debian, Windows Server**).
- Adjust **Boot disk size** if needed.

The screenshot shows the 'Operating system and storage' step of the 'Create an instance' wizard. The 'Name' is 'projectvm', 'Type' is 'New balanced persistent disk', 'Size' is '10 GB', 'Snapshot schedule' is 'default-schedule-1', 'License type' is 'Free', and 'Image' is 'Debian GNU/Linux 12 (bookworm)'. The 'Monthly estimate' on the right is \$25.46.

Step 5: Configure Networking (Optional)

- Allow HTTP/HTTPS traffic if needed.
- Modify firewall rules as required.

Google Cloud My First Project Search (/) for resources, docs, products, and more

Create an instance CREATE VM FROM...

Machine configuration
e2-medium, us-central1-a

OS and storage
Debian GNU/Linux 12 (bookworm)

Data protection
Snapshot schedules

Networking
1 firewall rule, 1 network interface

Observability
Install Ops Agent

Security

Advanced

Networking

Firewall
Add tags and firewall rules to allow specific network traffic from the Internet

☒ Allow HTTP traffic

☐ Allow HTTPS traffic

☐ Allow Load Balancer Health Checks

Network tags

Hostname
Set a custom hostname for this instance or leave it default. Choice is permanent

IP forwarding
☒ Enable

Network performance configuration

Network bandwidth
☒ Enable per VM Tier_1 networking performance

Maximum outbound network bandwidth: 2Gbps

VM ID: 12345678901234567890

Monthly estimate
\$25.46
That's about \$0.03 hourly

Pay for what you use: no upfront costs and per second billing

Item	Monthly estimate
2 vCPU + 4 GB memory	\$24.46
10 GB balanced persistent disk	\$1.00
Logging	Cost varies
Monitoring	Cost varies
Snapshot schedule	Cost varies
Total	\$25.46

[Compute Engine pricing](#)
[Cloud Operations pricing](#)
[LESS](#)

CREATE **CANCEL** **EQUIVALENT CODE**

Step 6: Create and Deploy

- Click **Create**.
- Wait for the VM to be provisioned.

Google Cloud My First Project Search (/) for resources, docs, products, and more

Compute Engine VM instances CREATE INSTANCE IMPORT VM REFRESH

INSTANCES OBSERVABILITY INSTANCE SCHEDULES

VM instances

Filter Enter property name or value

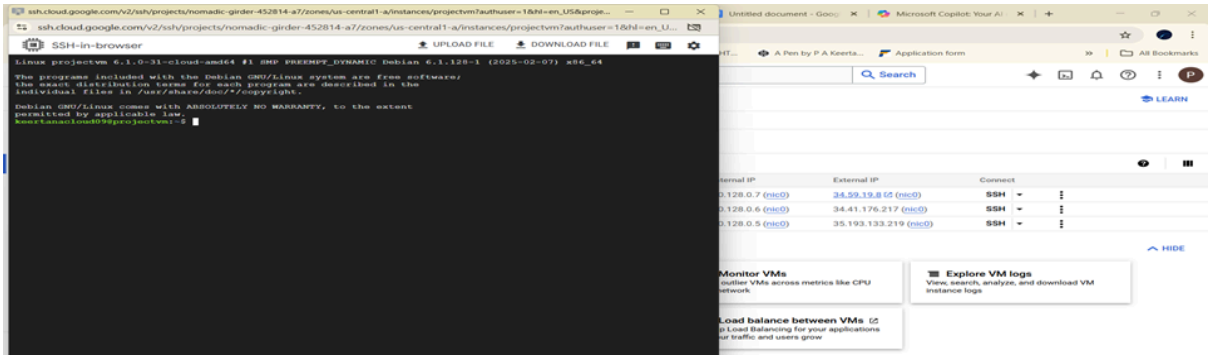
Status	Name	Zone	Recommendations	In use by	Internal IP	External IP	Connect
<input checked="" type="checkbox"/>	projectvm	us-central1-a			10.128.0.7 (nic0)	34.59.19.8 (nic0)	SSH
<input checked="" type="checkbox"/>	vmcloud	us-central1-b			10.128.0.6 (nic0)	34.41.176.217 (nic0)	SSH
<input checked="" type="checkbox"/>	vmproject	us-central1-a			10.128.0.5 (nic0)	35.193.133.219 (nic0)	SSH

Related actions

- Explore Backup and DR** NEW
Back up your VMs and set up disaster recovery
- View billing report**
View and manage your Compute Engine billing
- Monitor VMs**
View outlier VMs across metrics like CPU and network
- Explore VM logs**
View, search, analyze, and download VM instance logs
- Set up firewall rules**
Control traffic to and from a VM instance
- Patch management**
Schedule patch updates and view patch compliance on VM instances
- Load balance between VMs**
Set up Load Balancing for your applications as your traffic and users grow

Step 7: Connect to the VM

- In the **VM instances** list, click **SSH** to connect directly from the browser.



Step 8: Verify Resources

Run the following command inside your VM to check CPU and memory:

```
sh
CopyEdit
lscpu # Check CPU details
free -h # Check memory usage
```



Additional Configurations (Optional)

- **Install software:** Use apt or yum depending on the OS.
- **Enable auto-start:** Configure startup scripts if required.
- **Attach storage:** Add persistent disks if needed.


EXPERIMENT NUMBER 2

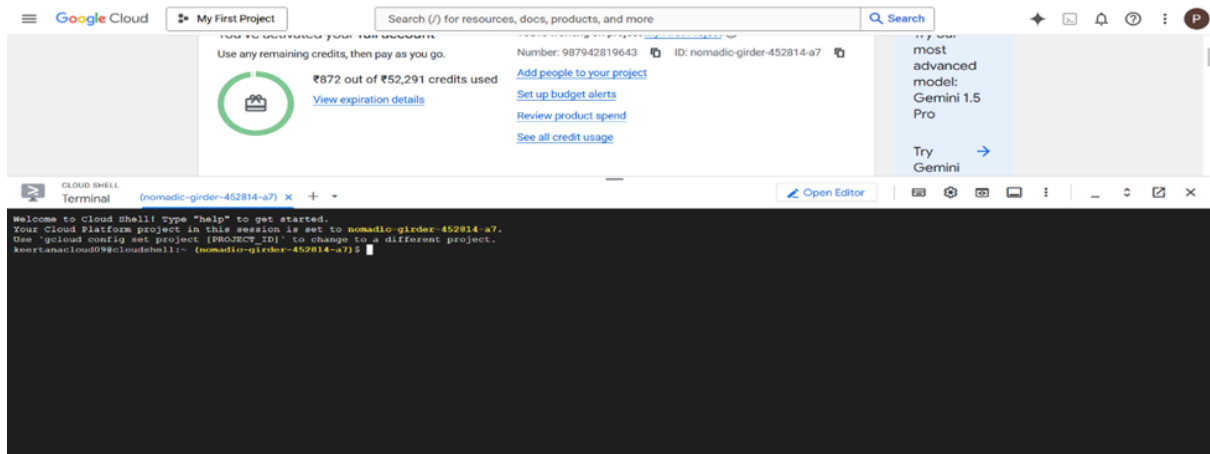
Getting Started with Cloud Shell and gcloud

What is Cloud Shell?

Cloud Shell is an interactive command-line environment in Google Cloud that lets you manage your resources using the `gcloud` command-line tool. It provides a pre-configured environment with essential Google Cloud tools.

Step 1: Open Cloud Shell

1. Go to the Google Cloud Console.
2. Click the **Cloud Shell icon** () in the top-right corner.
3. A terminal window will open at the bottom of the screen.



Step 2: Verify gcloud is Installed

Cloud Shell comes with `gcloud` pre-installed. Verify the installation with:

```
sh
CopyEdit
gcloud --version
```

You should see output similar to:

```
yaml
CopyEdit
Google Cloud SDK 123.0.0
bq 2.0.75
core 2023.10.01
gsutil 5.3
```


sh
CopyEdit
gcloud compute instances list
This shows all virtual machines (VMs) in the current project.

```
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7)$ gcloud compute instances list
NAME: vmproject
ZONE: us-central1-a
MACHINE_TYPE: e2-medium
PREEMPTIBLE:
INTERNAL_IP: 10.128.0.5
EXTERNAL_IP: 35.193.133.219
STATUS: RUNNING
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7)$
```

2. Create a VM Instance

sh
CopyEdit
gcloud compute instances create my-vm \
--machine-type=e2-medium \
--image-family=debian-11 \
--image-project=debian-cloud \
--zone=us-central1-a

- my-vm → Name of the VM
- --machine-type=e2-medium → Defines CPU and memory
- --image-family=debian-11 → OS Image
- --zone=us-central1-a → The zone where the VM is created

```
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7)$ gcloud compute instances create vmcloud --machine-type=e2-medium --image-family=debian-11 --image-project=debian-cloud --zone=us-central1-b
Created [https://www.googleapis.com/compute/v1/projects/nomadic-girder-452814-a7/zones/us-central1-b/instances/vmcloud].
NAME: vmcloud
ZONE: us-central1-b
MACHINE_TYPE: e2-medium
PREEMPTIBLE:
INTERNAL_IP: 10.128.0.6
EXTERNAL_IP: 34.41.176.217
STATUS: RUNNING
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7)$
```

3. SSH into the VM

sh
CopyEdit
gcloud compute ssh my-vm --zone=us-central1-a

```
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7)$ gcloud compute ssh vmcloud --zone=us-central1-b
Warning: Permanently added 'compute-2842001874580786179' (2025:519) to the list of known hosts.
Linux vmcloud 5.10.0-33-cloud-amd64 #1 SMP Debian 5.10.226-1 (2024-10-03) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
keertanacloud09@vmcloud:~$
```

4. Delete a VM

sh
CopyEdit
gcloud compute instances delete my-vm --zone=us-central1-a

```
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7)$ gcloud compute instances delete instance-20250305-140800 --zone=us-central1-a
The following instances will be deleted. Any attached disks configured to be auto-deleted will be deleted unless they are attached to any other instances or the '--keep-disks' flag is given
and specifies them for keeping. Deleting a disk is irreversible and any data on the disk will be lost.
- [instance-20250305-140800] in [us-central1-a]

Do you want to continue (Y/n)? y
Deleted [https://www.googleapis.com/compute/v1/projects/nomadic-girder-452814-a7/zones/us-central1-a/instances/instance-20250305-140800].
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7)$
```

Step 5: Manage Storage Buckets

1. List Storage Buckets

sh
CopyEdit
gcloud storage buckets list

```
Deleted [https://www.googleapis.com/compute/v1/projects/nomadic-girder-452814-a7/zones/us-central1-a/instances/instance-20250305-140800].
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7)$ gcloud storage buckets list
listed 0 items
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7)$ gcloud storage buckets create my-bucket --location=us-central1
```

2. Create a New Storage Bucket

```
sh
CopyEdit
gcloud storage buckets create my-bucket --location=us-central1
```

```
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7)$ gcloud storage buckets create gs://my-vmbucket --location=us-central1
Creating gs://my-vmbucket/...
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7)$
```

3. Upload a File to a Bucket

```
sh
CopyEdit
gcloud storage cp myfile.txt gs://my-bucket/
```

4. List Files in a Bucket

```
sh
CopyEdit
gcloud storage ls gs://my-bucket/
```

```
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7)$ gcloud storage cp testfile.txt gs://my-vmbucket/
Copying file://testfile.txt to gs://my-vmbucket/testfile.txt
Completed files 1/1 | 20.0B/20.0B
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7)$ gcloud storage ls gs://my-vmbucket/
gs://my-vmbucket/testfile.txt
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7)$
```

5. Delete a Bucket

```
sh
CopyEdit
gcloud storage buckets delete my-bucket
```

```
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7)$ gcloud storage rm -r gs://my-vmbucket/
Removing objects:
Removing gs://my-vmbucket/testfile.txt#1741365459369821...
Completed 1/1
Removing buckets:
Removing gs://my-vmbucket/...
Completed 1/1
```

Step 6: Managing IAM (Permissions)

1. List IAM Policies

```
sh
CopyEdit
gcloud projects get-iam-policy PROJECT_ID
```

```
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7)$ gcloud projects get-iam-policy nomadic-girder-452814-a7
bindings:
- members:
  - serviceAccount:service-987942819643@compute-system.iam.gserviceaccount.com
  role: roles/compute.serviceAgent
- members:
  - serviceAccount:987942819643-compute@developer.gserviceaccount.com
  - serviceAccount:987942819643@cloudservices.gserviceaccount.com
  role: roles/editor
- members:
  - user:keertanacloud09@gmail.com
  role: roles/owner
etag: BwYvMOY_Ti4=
version: 1
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7)$
```

2. Add an IAM Role to a User

```
sh
CopyEdit
gcloud projects add-iam-policy-binding PROJECT_ID \
--member="user:example@gmail.com" \
--role="roles/editor"
```

```
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7)$ gcloud projects add-iam-policy-binding nomadic-girder-452814-a7 \
--member="user:keertanacloud09@gmail.com" \
--role="roles/editor"
Updated IAM policy for project [nomadic-girder-452814-a7].
bindings:
- members:
  - serviceAccount:service-987942819643@compute-system.iam.gserviceaccount.com
  role: roles/compute.serviceAgent
- members:
  - serviceAccount:987942819643-compute@developer.gserviceaccount.com
  - serviceAccount:987942819643@cloudservices.gserviceaccount.com
  - user:keertanacloud09@gmail.com
  role: roles/editor
- members:
  - user:keertanacloud09@gmail.com
  role: roles/owner
etag: BwYv6UblFw=
version: 1
```


3. Remove an IAM Role

```
sh
CopyEdit
gcloud projects remove-iam-policy-binding PROJECT_ID \
  --member="user:example@gmail.com" \
  --role="roles/editor"
```

```
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7)$ gcloud projects remove-iam-policy-binding nomadic-girder-452814-a7 --member="user:keertanacloud09@gmail.com" --role="roles/editor"
Updated IAM policy for project [nomadic-girder-452814-a7].
bindings:
- members:
  - serviceAccount:service-987942819643@compute-system.iam.gserviceaccount.com
    role: roles/compute.serviceAgent
- members:
  - serviceAccount:987942819643-compute@developer.gserviceaccount.com
  - serviceAccount:987942819643@cloudservices.gserviceaccount.com
    role: roles/editor
- members:
  - user:keertanacloud09@gmail.com
    role: roles/owner
etag: BwYvw-IRPli=
version: 1
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7)$
```

Step 7: Configuring gcloud Defaults

Set default region and zone:

```
sh
CopyEdit
gcloud config set compute/region us-central1
gcloud config set compute/zone us-central1-a
```

To check all configurations:

```
sh
CopyEdit
gcloud config list
```

```
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7)$ gcloud config set compute/region us-central1
gcloud config set compute/zone us-central1-a
WARNING: Property validation for compute/region was skipped.
Updated property [compute/region].
WARNING: Property validation for compute/zone was skipped.
Updated property [compute/zone].
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7)$ gcloud config list
[accessibility]
screen_reader = True
[component manager]
disable_update_check = True
[compute]
gce_metadata_read_timeout_sec = 30
region = us-central1
zone = us-central1-a
[core]
account = keertanacloud09@gmail.com
disable_usage_reporting = False
project = nomadic-girder-452814-a7
[metrics]
environment = devshell

Your active configuration is: [cloudshell-22185]
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7)$
```

Step 8: Exit Cloud Shell

Simply **close the browser tab** or type:

```
sh
CopyEdit
exit
```

Conclusion

Cloud Shell and `gcloud` CLI make managing Google Cloud resources easy and efficient. You can create and manage VMs, storage, IAM policies, and more—all from the terminal.

EXPERIMENT NUMBER 3

Deploying a Cloud Function to Automate a Task Based on a Cloud Storage Event

Google Cloud Functions allows you to run serverless code in response to Cloud Storage events (such as file uploads, deletions, or updates). In this guide, we will create and deploy a **Cloud Function** that triggers when a file is uploaded to a **Cloud Storage bucket**.

Step 1: Enable Required APIs

Before deploying a Cloud Function, enable the required APIs:

```
sh
CopyEdit
gcloud services enable cloudfunctions.googleapis.com storage.googleapis.com
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7) $ gcloud services enable cloudfunctions.googleapis.com storage.googleapis.com
Operation "operations/acat.p2-987942819643-e1a7e774-2b85-4a07-8a4e-6d6b429e81b3" finished successfully.
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7) $ gcloud storage buckets create my-cloud-function-bucket --location=us-central1
```

Step 2: Create a Cloud Storage Bucket

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

```
sh
CopyEdit
gcloud storage buckets create my-cloud-function-bucket --location=us-central1
Replace my-cloud-function-bucket with your bucket name.
```

```
try again.
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7) $ gcloud storage buckets create gs://keerclocloudbucket --location=us-central1
Creating gs://keerclocloudbucket/...
```

Step 3: Write the Cloud Function Code

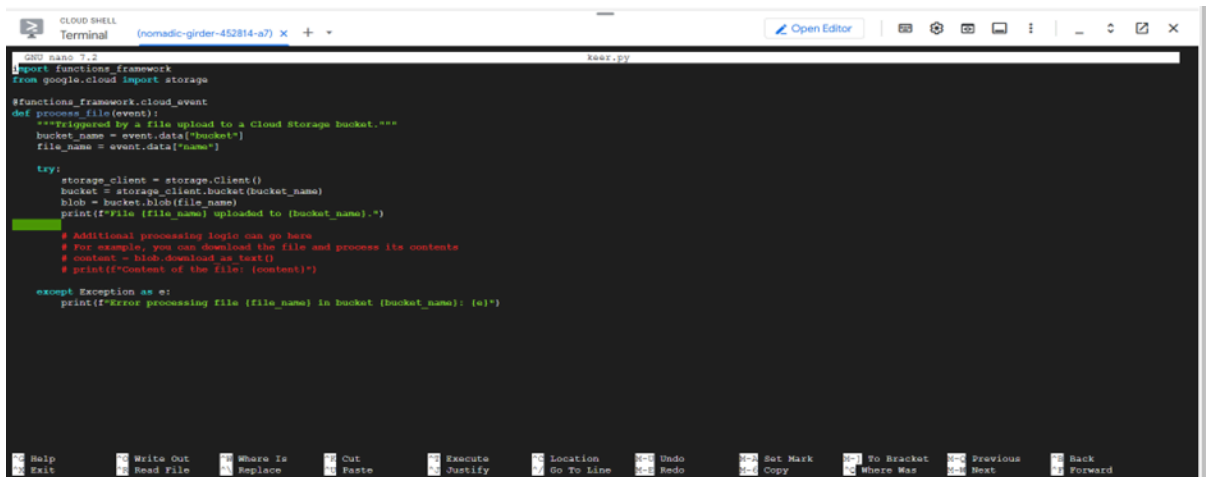
Create a new directory and navigate into it:

```
sh
CopyEdit
mkdir cloud-function-storage && cd cloud-function-storage
keertanacloud09@cloudshell:~ (nomadic-girder-452814-a7) $ mkdir keerstorage && cd keerstorage
keertanacloud09@cloudshell:~/keerstorage (nomadic-girder-452814-a7) $ nano keerp.py
```

Create a **Python script (main.py)** for the function:

```
python
CopyEdit
import functions_framework

@functions_framework.cloud_event
def process_file(event):
    """Triggered by a file upload to a Cloud Storage bucket."""
    bucket = event.data["bucket"]
    file_name = event.data["name"]
    print(f"File {file_name} uploaded to {bucket}")
```



```
#!/usr/bin/env python3
import functions_framework
from google.cloud import storage

@functions_framework.cloud_event
def process_file(event):
    """Triggered by a file upload to a Cloud Storage bucket."""
    bucket_name = event.data["bucket"]
    file_name = event.data["name"]

    try:
        storage_client = storage.Client()
        bucket = storage_client.bucket(bucket_name)
        blob = bucket.blob(file_name)
        print(f"File {file_name} uploaded to {bucket_name}.")

        # Additional processing logic can go here
        # For example, you can download the file and process its contents
        # content = blob.download_as_text()
        # print(f"Content of the file: {content}")

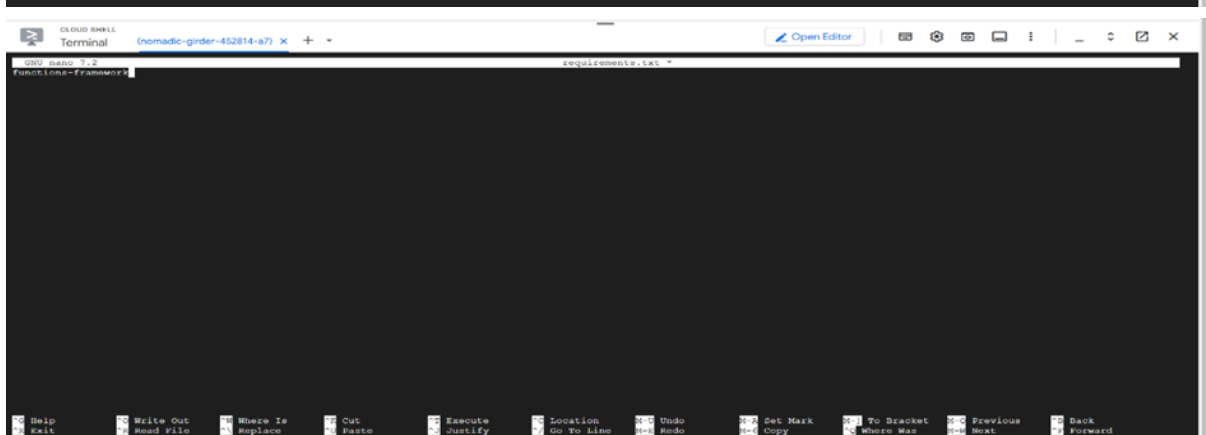
    except Exception as e:
        print(f"Error processing file {file_name} in bucket {bucket_name}: {e}")
```

Create a **requirements file (requirements.txt)** to specify dependencies:

```
pgsql
CopyEdit
functions-framework
```



```
keertanacloud09@cloudshell:~/keerstorage (nomadic-girder-452814-a7)$ nano requirements.txt
keertanacloud09@cloudshell:~/keerstorage (nomadic-girder-452814-a7)$
```



```
keertanacloud09@cloudshell:~/keerstorage (nomadic-girder-452814-a7)$ nano requirements.txt
keertanacloud09@cloudshell:~/keerstorage (nomadic-girder-452814-a7)$
```

Step 4: Deploy the Cloud Function

Deploy the function using the `gcloud` CLI:

```
sh
CopyEdit
gcloud functions deploy process_file \
  --runtime python310 \
  --trigger-event google.storage.object.finalize \
  --trigger-resource my-cloud-function-bucket \
  --entry-point process_file \
  --region us-central1 \
  --gen2
```

Explanation of Parameters:

- `process_file` → Name of the function.
- `--runtime python310` → Runtime environment (Python 3.10).
- `--trigger-event google.storage.object.finalize` → Trigger event (fires when a file is uploaded).
- `--trigger-resource my-cloud-function-bucket` → Storage bucket to monitor.

- --entry-point process_file → Function entry point.
- --region us-central1 → Deployment region.
- --gen2 → Deploys Cloud Function in **2nd generation (Gen 2)** for better performance.

```
keertanac@cloudshell:~/keertanac$ gcloud functions deploy process_file \
--runtime python310 \
--trigger-event google.storage.object.finalize \
--trigger-resource keertanac-test \
--entry-point process_file \
--region us-central1 \
--gen2
Preparing Function...done
* Updating Function (may take a while)...
OK [Build] Logs are available at [https://console.cloud.google.com/cloud-build/builds?region=us-central1&fd953d-c453-443d-8fab-6060c6d1de9?project=967942819443]
[Service]
OK [Triggers]
- [ArtifactRegistry]
- [Healthcheck]
- [TriggerCheck]
Completed with warnings:
[INFO] A new revision will be deployed serving with 0% traffic.
You can view your function in the Cloud Console here: https://console.cloud.google.com/functions/details/us-central1/process_file?project=967942819443&w=
BuildConfig:
automaticUpdatePolicy: {}
build: projects/967942819443/locations/us-central1/builds/94fd953d-c453-443d-8fab-6060c6d1de9
dockerRegistry: ARTIFACT_REGISTRY
dockerRepository: projects/967942819443/locations/us-central1/repositories/gcf-artifacts
entryPoint: process_file
runtime: python310
serviceAccount: projects/967942819443/serviceAccounts/967942819443-compute@developer.gserviceaccount.com
source:
storageSource:
  bucket: gcf-v2-source-967942819443-us-central1
  generation: '174188396471220'
  object: process_file/function-source.zip
storageSource:
  resolvedStorageSource:
    bucket: gcf-v2-source-967942819443-us-central1
    generation: '174188396471220'
    object: process_file/function-source.zip
createTime: '2023-03-13T06:39:18.427000Z'
environment: GEN_2
eventTrigger:
  eventFilters:
    - attribute: bucket
      value: keertanac-test
  eventType: google.cloud.storage.object.v1.finalize
  eventTrigger: projects/967942819443/locations/us-central1/eventarc/eventarc-us-central1-process-file-321379-084
  entryPolicy: HTTP_POLICY_DO_NOT_RETRY
  serviceAccount: projects/967942819443/serviceAccounts/967942819443-compute@developer.gserviceaccount.com
  triggers: projects/967942819443/locations/us-central1/triggers/process-file-321379
  triggerRegion: us-central1
labels:
```

Step 5: Test the Function

Upload a test file to the bucket:

```
sh
CopyEdit
gcloud storage cp test-file.txt gs://my-cloud-function-bucket/
```

```
keertanac@cloudshell:~/keertanac$ gcloud storage cp /home/keertanac@cloudshell:/keertanac/test-file.txt gs://keertanac-test/
Copying file:///home/keertanac@cloudshell:/keertanac/test-file.txt to gs://keertanac-test-file.txt
Completed files 1/1 | 0B
```

Check the function logs to verify execution:

```
sh
CopyEdit
gcloud functions logs read process_file --region=us-central1
```

```
keertanac@cloudshell:~/keertanac$ gcloud functions logs read process_file --region=us-central1
LEVEL: INFO
NAME: process-file
EXECUTION_ID: JxKovv123PM
TIME_UTC: 2023-03-13 16:47:11.585
LOG: File processed successfully

LEVEL: INFO
NAME: process-file
EXECUTION_ID:
TIME_UTC: 2023-03-13 16:47:11.548
LOG: Default STARTUP TCP probe succeeded after 1 attempt for container "worker" on port 8080.

LEVEL: INFO
NAME: process-file
EXECUTION_ID:
TIME_UTC: 2023-03-13 16:47:10.278
LOG:

LEVEL: WARNING
NAME: process-file
EXECUTION_ID:
TIME_UTC: 2023-03-13 07:22:03.457
LOG: The request was not authenticated. Either allow unauthenticated invocations or set the proper Authorization header. Read more at https://cloud.google.com/run/docs/securing/authenticating Additional troubleshooting documentation can be f
ound at https://cloud.google.com/run/docs/troubleshooting#unauthenticated-client

LEVEL: WARNING
NAME: process-file
EXECUTION_ID:
TIME_UTC: 2023-03-13 07:22:03.313
LOG: The request was not authenticated. Either allow unauthenticated invocations or set the proper Authorization header. Read more at https://cloud.google.com/run/docs/securing/authenticating Additional troubleshooting documentation can be f
ound at https://cloud.google.com/run/docs/troubleshooting#unauthenticated-client

LEVEL: INFO
NAME: process-file
EXECUTION_ID:
TIME_UTC: 2023-03-13 07:22:02.212
LOG: Default STARTUP TCP probe succeeded after 1 attempt for container "worker" on port 8080.

LEVEL: INFO
NAME: process-file
EXECUTION_ID:
TIME_UTC: 2023-03-13 06:43:41.039
LOG: Default STARTUP TCP probe succeeded after 1 attempt for container "worker" on port 8080.
keertanac@cloudshell:~/keertanac$
```

Step 6: Update or Delete the Function

Update the Function

If you modify main.py, redeploy with:

```
sh
CopyEdit
gcloud functions deploy process_file --region=us-central1
```

```
keertanacloud@cloudshell:~/keertanage (nomadic-girder-452814-a7) $ gcloud functions deploy process_file --region=us-central1
Preparing function...done.
X Updating function (may take a while)...
OK [Build] Logs are available at [https://console.cloud.google.com/cloud-build/builds;region=us-central1/37996695-362f-44fc-9581-1b58d13de281?project=987942819643]
[Service]
- [Trigger]
- [ArtifactRegistry]
- [Healthcheck]
- [Triggercheck]
Completed with warnings:
[INFO] A new revision will be deployed serving with 0% traffic.
You can view your function in the Cloud Console here: https://console.cloud.google.com/functions/details/us-central1/process_file?project=nomadic-girder-452814-a7
buildConfig:
  automaticUpdatePolicy: {}
  build: projects/987942819643/locations/us-central1/builds/37996695-362f-44fc-9581-1b58d13de281
  dockerRegistry: ARTIFACT_REGISTRY
  dockerRepository: projects/nomadic-girder-452814-a7/locations/us-central1/repositories/gcf-artifacts
  entryPoint: process_file
  runtime: python310
  serviceAccount: projects/nomadic-girder-452814-a7/serviceaccounts/987942819643-compute@developer.gserviceaccount.com
  source:
    storageSource:
      bucket: gcf-v2-sources-987942819643-us-central1
      generation: '1741884537189626'
      object: process_file/function-source.zip
    sourceFromStorage:
      bucket: gcf-v2-sources-987942819643-us-central1
      generation: '1741884537189626'
      object: process_file/function-source.zip
  createdAt: 2025-03-13T06:39:18.427508362Z
  environment: GCM_2
  eventTrigger:
```

Delete the Function

```
sh
CopyEdit
gcloud functions delete process_file --region=us-central1
```

```
keertanacloud@cloudshell:~/keertanage (nomadic-girder-452814-a7) $ gcloud functions delete process_file --region=us-central1
2nd gen function [projects/nomadic-girder-452814-a7/locations/us-central1/functions/process_file] will be deleted.
Do you want to continue (Y/n)? y
Preparing function...done.
OK Deleting function...
OK [Artifact Registry]
OK [Service]
OK [Trigger]
Done.
Deleted [projects/nomadic-girder-452814-a7/locations/us-central1/functions/process_file].
keertanacloud@cloudshell:~/keertanage (nomadic-girder-452814-a7) $
```

Conclusion

You have successfully created and deployed a **Cloud Function** that triggers when a file is uploaded to **Cloud Storage**. You can extend this by processing images, sending notifications, or integrating with other Google Cloud services.

EXPERIMENT NUMBER 4

Deploying a Web Application on Google App Engine with Automatic Scaling

Google **App Engine** is a fully managed serverless platform that allows you to deploy and scale web applications automatically. In this guide, we will deploy a simple **Flask web application** on **App Engine (Standard Environment)** with automatic scaling enabled.

Step 1: Enable App Engine API

Before deploying, enable the App Engine API:

```
sh
CopyEdit
gcloud services enable appengine.googleapis.com
Then, initialize App Engine for your project:
```

```
sh
CopyEdit
gcloud app create --region=us-central
Replace us-central with your preferred region.
```

```
keertanacloud09@cloudshell:~ (refined-sunup-455515-d0) $ gcloud services enable appengine.googleapis.com
Operation "operations/acat.p2-372816833289-3244bd6f-5cb7-4f3a-8a25-7c6bf09e54f3" finished successfully.
keertanacloud09@cloudshell:~ (refined-sunup-455515-d0) $ gcloud app create --region=us-central
You are creating an app for project [refined-sunup-455515-d0].
WARNING: Creating an App Engine application for a project is irreversible and the region
cannot be changed. More information about regions is at
<https://cloud.google.com/appengine/docs/locations>.

WARNING: Starting from March, 2025, App Engine sets the automatic scaling maximum instances
default for standard environment deployments to 20. This change doesn't impact
existing apps. To override the default, specify the new max_instances value in your
app.yaml file, and deploy a new version or redeploy over an existing version.
For more details on max instances, see
<https://cloud.google.com/appengine/docs/standard/reference/app-yaml.md#scaling_elements>.

Creating App Engine application in project [refined-sunup-455515-d0] and region [us-central]...done.
Success! The app is now created. Please use 'gcloud app deploy' to deploy your first app.
keertanacloud09@cloudshell:~ (refined-sunup-455515-d0) $
```

Step 2: Create a Simple Web Application

Create a new project directory and navigate into it:

```
sh
CopyEdit
mkdir app-engine-app && cd app-engine-app
```

```
keertanacloud09@cloudshell:~/app-engine-app (refined-sunup-455515-d0) $ nano main.py
```

Create a **Python web application** using **Flask**:

1. Install Flask

```
sh
CopyEdit
pip install flask
```

```
keertanacloud09@cloudshell:~/app-engine-app (refined-sunup-455515-d0) $ pip install flask
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: flask in /usr/local/lib/python3.12/dist-packages (3.1.0)
Requirement already satisfied: Werkzeug>=3.1 in /usr/local/lib/python3.12/dist-packages (from flask) (3.1.3)
Requirement already satisfied: Jinja2>=3.1.2 in /usr/local/lib/python3.12/dist-packages (from flask) (3.1.5)
Requirement already satisfied: itsdangerous>=2.2 in /usr/local/lib/python3.12/dist-packages (from flask) (2.2.0)
Requirement already satisfied: click>=8.1.3 in /usr/local/lib/python3.12/dist-packages (from flask) (8.1.8)
Requirement already satisfied: blinker>=1.9 in /usr/local/lib/python3.12/dist-packages (from flask) (1.9.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from Jinja2>=3.1.2->flask) (3.0.2)
```

2. Create `main.py` (Web Application)

Create a Python file (`main.py`) with the following content:

```
python
```

```
CopyEdit
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return "Hello, World! Welcome to App Engine with Auto Scaling."

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



Step 3: Define App Engine Configuration

Create an `app.yaml` file to configure the App Engine environment:

```
yaml
CopyEdit
runtime: python310 # Define the runtime
entrypoint: gunicorn -b :$PORT main:app

automatic_scaling:
  min_instances: 1
  max_instances: 5
  target_cpu_utilization: 0.65
  target_throughput_utilization: 0.75

handlers:
- url: /*
  script: auto
```

Explanation:

- **runtime: python310** → Specifies Python 3.10 as the runtime.
- **entrypoint: gunicorn -b :\$PORT main:app** → Uses Gunicorn to run the app.
- **automatic_scaling:**
 - **min_instances: 1** → Keeps at least one instance running.
 - **max_instances: 5** → Scales up to 5 instances based on demand.
 - **target_cpu_utilization: 0.65** → Scales based on CPU load.
 - **target_throughput_utilization: 0.75** → Scales based on request load.
- **handlers** → Routes all incoming requests to the application.

```
meertancloud579@cloudshell:~/app-engine-demo (refined-mwmp-455515-d0) $ nano app.yaml
```

Step 6: Access the Application

Once deployed, open your app in a browser:

```
sh
CopyEdit
gcloud app browse
Alternatively, visit:
https://<your-project-id>.appspot.com/
```

```
ksertanacloud09@cloudshell:~/app-engine-demo (refined-munip-455515-d0) $ gcloud app browse
Did not detect your browser. Go to this link to view your app:
https://refined-munip-455515-d0.us.r.appspot.com
ksertanacloud09@cloudshell:~/app-engine-demo (refined-munip-455515-d0) $
```

Hello, World! Welcome to App Engine with Auto Scaling.

Step 7: Monitor and Scale

1. Check App Engine Instances

```
sh
CopyEdit
gcloud app instances list
```

2. View Logs

```
sh
CopyEdit
gcloud app logs tail -s default
```

```
ksertanacloud09@cloudshell:~/app-engine-demo (refined-munip-455515-d0) $ gcloud app instances list
INSTANCE default
VERSION 20250401152839
ID 02e2e0c134639235m1F224dc0c9246a2e0F9213b346a4d10c272e1001058ed73c0b0b6FE1235aF0ea30Fc004500F0441b03ac20c4Ed1726d8a2ee287a02e91F8ac0b93c09a2ee3463030a
ON 0FA000 N/A
No LIVELOGS
Running 0 instances
ksertanacloud09@cloudshell:~/app-engine-demo (refined-munip-455515-d0) $ gcloud app logs tail -s default
Waiting for new log entries...
2025-04-01 15:33:49 default(20250401152839) "GET / HTTP/1.1" 200
2025-04-01 15:33:50 default(20250401152839) [2025-04-01 15:33:50 +0000] [11] [INFO] Starting gunicorn 23.0.0
2025-04-01 15:33:50 default(20250401152839) [2025-04-01 15:33:50 +0000] [11] [INFO] Listening at: http://0.0.0.0:8081 (1)
2025-04-01 15:33:50 default(20250401152839) [2025-04-01 15:33:50 +0000] [11] [INFO] Using worker: sync
2025-04-01 15:33:50 default(20250401152839) [2025-04-01 15:33:50 +0000] [14] [INFO] Booting worker with pid: 14
2025-04-01 15:33:50 default(20250401152839) "GET /favicon.ico HTTP/1.1" 404
```

3. Monitor Scaling

Monitor the deployed services:

```
gcloud app services list
```

```
ksertanacloud09@cloudshell:~/app-engine-demo (refined-munip-455515-d0) $ gcloud app services list
SERVICE default
URL_VERSIONS 2
```

Conclusion

You have successfully deployed a **Flask web application** on **Google App Engine** with **automatic scaling** enabled! 🚀

App Engine will now scale your app based on traffic demand automatically.

EXPERIMENT NUMBER 5

Qwikstart: Google Cloud Storage (GCS)

Google Cloud Storage (GCS) is a **scalable, secure, and durable object storage** solution that allows you to store and manage data efficiently. You can interact with GCS using the **Google Cloud Console**, **gsutil CLI**, or the **gcloud CLI**.

Step 1: Enable Cloud Storage API

Before using Cloud Storage, ensure the API is enabled:

```
sh
CopyEdit
gcloud services enable storage.googleapis.com
```

```
keertanacloud09@cloudshell:~ (refined-summp-455515-d0) $ gcloud services enable storage.googleapis.com
```

Step 2: Create a Cloud Storage Bucket

A **bucket** is a container for storing objects (files). Buckets have globally unique names.

Using the Google Cloud Console

1. Navigate to **Cloud Storage**
2. Click **Create bucket**
3. Enter a **globally unique name** (e.g., my-unique-bucket-123)
4. Select a **storage class** (e.g., **Standard** for frequent access)
5. Choose a **location** (e.g., us-central1)
6. Click **Create**

Using the gcloud CLI

```
sh
CopyEdit
gcloud storage buckets create my-bucket --location=us-central1
```

Replace my-bucket with your unique bucket name.

```
keertanacloud09@cloudshell:~ (refined-summp-455515-d0) $ gcloud storage buckets create gs://my-unique-bucket35 --location=us-central1
Creating gs://my-unique-bucket35/...
keertanacloud09@cloudshell:~ (refined-summp-455515-d0) $ g`C
```

Step 3: Upload a File to Cloud Storage

Using Google Cloud Console

1. Open **Cloud Storage** in the **Google Cloud Console**
2. Click your **bucket**
3. Click **Upload files**
4. Select a file and upload

Using the gcloud CLI

```
sh
CopyEdit
gcloud storage cp myfile.txt gs://my-bucket/
```

Replace `myfile.txt` with your file and `my-bucket` with your bucket name.

```
keertanacloud09@cloudshell:~ (refined-summp-455515-d0)$ gcloud storage cp testfile.txt gs://my-unique-bucket35/
Copying file://testfile.txt to gs://my-unique-bucket35/testfile.txt
Completed files 1/1 | 20.0B/20.0B
keertanacloud09@cloudshell:~ (refined-summp-455515-d0)$ gsutil cp testfile.txt gs://my-bucket/
```

Using gsutil (Cloud Storage CLI)

```
sh
CopyEdit
gsutil cp myfile.txt gs://my-bucket/
```

Step 4: List Files in a Bucket

Using the gcloud CLI

```
sh
CopyEdit
gcloud storage ls gs://my-bucket/
```

```
keertanacloud09@cloudshell:~ (refined-summp-455515-d0)$ gcloud storage ls gs://my-unique-bucket35/
gs://my-unique-bucket35/AI_IAB_06.ipynb
gs://my-unique-bucket35/testfile.txt
```

Using gsutil

```
sh
CopyEdit
gsutil ls gs://my-bucket/
```

Step 5: Download a File from Cloud Storage

Using the gcloud CLI

```
sh
CopyEdit
gcloud storage cp gs://my-bucket/myfile.txt .
```

```
keertanacloud09@cloudshell:~ (refined-summp-455515-d0)$ gcloud storage cp gs://my-unique-bucket35/testfile.txt .
Copying gs://my-unique-bucket35/testfile.txt to file://./testfile.txt
Completed files 1/1 | 20.0B/20.0B
```

Using gsutil

```
sh
CopyEdit
gsutil cp gs://my-bucket/myfile.txt .
```

Step 6: Delete a File or Bucket

Delete a File

```
sh
CopyEdit
gcloud storage rm gs://my-bucket/myfile.txt
```

```
keertanacloud09@cloudshell:~ (refined-summp-455515-d0)$ gcloud storage rm gs://my-unique-bucket35/testfile.txt
Removing objects:
Removing gs://my-unique-bucket35/testfile.txt...
Completed 1/1
```

Uploads and My Project 92210 operations ▾

Delete a Bucket (Ensure it's empty first)

```
sh
CopyEdit
gcloud storage buckets delete my-bucket
```

```
keertanacloud09@cloudshell:~ (refined-sunup-455515-d0) $ gcloud storage buckets delete gs://my-unique-bucket35
Removing gs://my-unique-bucket35/...
Completed 1/1
keertanacloud09@cloudshell:~ (refined-sunup-455515-d0) $
```

Uploads and My Project 92210 operations

All 1 AR 06 min 17h Complete

Step 7: Make a File Public (Optional)

To share a file publicly, set its permissions:

```
sh
CopyEdit
gcloud storage objects add-iam-policy-binding gs://my-bucket/myfile.txt \
  --role=roles/storage.objectViewer \
  --member=allUsers
```

Now, anyone can access it via:

```
perl
CopyEdit
https://storage.googleapis.com/my-bucket/myfile.txt
```

```
keertanacloud09@cloudshell:~ (refined-sunup-455515-d0) $ gcloud storage buckets add-iam-policy-binding gs://my-unique-bucket35 \
  --role=roles/storage.objectViewer \
  --member=allUsers
bindings:
- members:
  - projectEditor:refined-sunup-455515-d0
  - projectOwner:refined-sunup-455515-d0
  role: roles/storage.legacyBucketOwner
- members:
  - projectViewer:refined-sunup-455515-d0
  role: roles/storage.legacyBucketReader
- members:
  - allUsers
  role: roles/storage.objectViewer
etag: CAI=
kind: storage#policy
resourceId: projects/_/buckets/my-unique-bucket35
version: 1
```

Uploads and My Project 92210 operations

Conclusion

You have successfully created a Cloud Storage bucket, uploaded files, listed objects, and managed storage using Cloud Console, gcloud, and gsutil! 🎉

EXPERIMENT NUMBER 6

Cloud SQL for MySQL: Managed MySQL with High Availability in Google Cloud

Google **Cloud SQL for MySQL** is a **fully managed relational database** service that automates tasks such as **setup, maintenance, backups, and scaling**. It ensures **high availability** and integrates seamlessly with Google Cloud services.

Key Features of Cloud SQL for MySQL

- ✓ **Automated Backups & Patching** – No need to manage database updates manually.
 - ✓ **High Availability (HA)** – Supports automatic failover in case of outages.
 - ✓ **Automatic Scaling** – Adjusts resources based on workload demands.
 - ✓ **Security** – Provides encryption, IAM-based access, and private IP support.
 - ✓ **Integration** – Works with App Engine, Compute Engine, Kubernetes (GKE), and BigQuery.
-

Step 1: Enable Cloud SQL API

Before creating a database instance, enable the Cloud SQL API:

```
sh
CopyEdit
gcloud services enable sqladmin.googleapis.com
```



Step 2: Create a Cloud SQL for MySQL Instance

Using the Google Cloud Console

1. Go to **Cloud SQL**
2. Click **Create Instance**
3. Select **MySQL**
4. Choose a **MySQL version** (e.g., **MySQL 8.0**)
5. Enter an **instance ID** (e.g., **my-mysql-instance**)
6. Select a **region** and **zone**
7. Choose the **Machine Type** (e.g., **2 vCPUs, 8GB RAM**)
8. Enable **High Availability (HA)** if required
9. Set a **root password**
10. Click **Create**

Using the gcloud CLI

```
sh
CopyEdit
gcloud sql instances create my-mysql-instance \
  --database-version=MYSQL_8_0 \
  --tier=db-n1-standard-2 \
  --region=us-central1 \
```

--root-password=mysecurepassword

- ◆ Replace my-mysql-instance with your instance name.
- ◆ Change --tier=db-n1-standard-2 for different CPU/RAM configurations.
- ◆ Replace mysecurepassword with a strong password.

```
keertanacloud0%cloudshell:~ (refined-rump-455515-d0)$ gcloud sql instances create my-mysql1 --data
base-version=MYSQL_8_0 --tier=db-n1-standard-2 --region=us-central1 --root-password@cloud15
Creating Cloud SQL instance for MYSQL_8_0...done.
Created (https://sqladmin.googleapis.com/sql/v1beta4/projects/refined-rump-455515-d0/instances/my-mysql1).
Name: my-mysql1
Database Version: MYSQL_8_0
Location: us-central1-c
Tier: db-n1-standard-2
Primary Address: 34.123.125.143
Private Address: -
Status: RUNNABLE
keertanacloud0%cloudshell:~ (refined-rump-455515-d0)$
```

Step 3: Connect to Cloud SQL for MySQL

Once your instance is ready, connect using one of the following methods.

Option 1: Cloud SQL Proxy (Recommended for Security)

1. Install the **Cloud SQL Proxy**:

```
sh
CopyEdit
curl -o cloud_sql_proxy
https://dl.google.com/cloudsql/cloud_sql_proxy.linux.amd64
chmod +x cloud_sql_proxy
```

2. Start the proxy:

```
sh
CopyEdit
./cloud_sql_proxy
--instances=<PROJECT-ID>:us-central1:my-mysql-instance=tcp:3306
```

3. Connect using MySQL CLI:

```
sh
CopyEdit
mysql -h 127.0.0.1 -u root -p
```

Option 2: Direct Connection via Public IP

1. Get the **public IP address** of your instance:

```
sh
CopyEdit
gcloud sql instances describe my-mysql-instance
--format="value(ipAddresses.ipAddress)"
```

```
keertanacloud0%cloudshell:~ (refined-rump-455515-d0)$ gcloud sql instances describe my-mysql1 \
--format="value(ipAddresses.ipAddress)"
34.45.248.153
```

2. Use MySQL CLI to connect:

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

```

kcs@tanakacloud09@cloudshell:~$ (terraform output -var=MYSQL_HOSTNAME) gcloud sql connect my-mysql35 --user-root
Allocating your IP for incoming connection for 5 minutes...done.
Connecting to database with MySQL user [root].Enter password:
Welcome to the MySQL monitor. Commands end with ; or \q.
Your MySQL connection id is 55
Server version: 8.0.27-google (Google)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database mydb;
Query OK, 1 row affected (0.21 sec)

mysql> use mydb;
Database changed
mysql> create table users(id int auto_increment primary key, name varchar(100), email varchar(100) unique);
Query OK, 0 rows affected (0.24 sec)

mysql> insert into users(name,email) values ('Alice','alice@gmail.com');
Query OK, 1 row affected (0.21 sec)

mysql> select * from users;
+----+-----+-----+
| id | name | email |
+----+-----+-----+
| 1 | Alice | alice@gmail.com |
+----+-----+-----+
1 row in set (0.21 sec)

mysql> ^C
kcs@tanakacloud09@cloudshell:~$ (terraform output -var=MYSQL_HOSTNAME) gcloud sql instances patch my-mysql35 --availability

```

3. Enter your password when prompted.

Step 4: Create a Database & Table

After connecting, create a database:

```

sql
CopyEdit
CREATE DATABASE mydatabase;
USE mydatabase;
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100),
  email VARCHAR(100) UNIQUE
);
INSERT INTO users (name, email) VALUES ('Alice', 'alice@example.com');
SELECT * FROM users;

```

```

kcs@tanakacloud09@cloudshell:~$ (terraform output -var=MYSQL_HOSTNAME) gcloud sql connect my-mysql35 --user-root
Allocating your IP for incoming connection for 5 minutes...done.
Connecting to database with MySQL user [root].Enter password:
Welcome to the MySQL monitor. Commands end with ; or \q.
Your MySQL connection id is 55
Server version: 8.0.27-google (Google)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database mydb;
Query OK, 1 row affected (0.21 sec)

mysql> use mydb;
Database changed
mysql> create table users(id int auto_increment primary key, name varchar(100), email varchar(100) unique);
Query OK, 0 rows affected (0.24 sec)

mysql> insert into users(name,email) values ('Alice','alice@gmail.com');
Query OK, 1 row affected (0.21 sec)

mysql> select * from users;
+----+-----+-----+
| id | name | email |
+----+-----+-----+
| 1 | Alice | alice@gmail.com |
+----+-----+-----+
1 row in set (0.21 sec)

mysql> ^C
kcs@tanakacloud09@cloudshell:~$ (terraform output -var=MYSQL_HOSTNAME) gcloud sql instances patch my-mysql35 --availability

```

Step 5: Enable High Availability (Optional)

To enable **High Availability (HA)** for automatic failover:

```

sh
CopyEdit
gcloud sql instances patch my-mysql-instance --availability-type=REGIONAL
This ensures automatic failover to a standby instance in case of failure.

```

Step 6: Backup and Restore

Enable Automated Backups

```
sh
CopyEdit
gcloud sql instances patch my-mysql-instance --backup-start-time=02:00
```

Manually Create a Backup

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

```
kernatancloud9@cloudshell:~ (refined-sump-455515-d0) $ gcloud sql instances patch my-mysql35 --backup-start-time=02:00
The following message will be used for the patch API method.
{"name": "my-mysql35", "project": "refined-sump-455515-d0", "settings": {"backupConfiguration": {"backupRetentionSettings": {"retainedBackups": 7, "retentionUnit": "COUNT"}, "enabled": true, "startTime": "02:00", "transactionLogRetentionDays": 7, "transactionLogStorageState": "TRANSACTIONAL_LOG_STORAGE_STATE_UNSPECIFIED"}}}
Patching Cloud SQL instance...done.
Updated [https://sqladmin.googleapis.com/sql/v1beta4/projects/refined-sump-455515-d0/instances/my-mysql35].
kernatancloud9@cloudshell:~ (refined-sump-455515-d0) $ gcloud sql backups create --instance=my-mysql35
Backing up Cloud SQL instance...done.
[https://sqladmin.googleapis.com/sql/v1beta4/projects/refined-sump-455515-d0/instances/my-mysql35] backed up.
```

Restore a Backup

```
sh
CopyEdit
gcloud sql backups restore BACKUP_ID --instance=my-mysql-instance
Replace BACKUP_ID with the ID of the backup.
```

```
kernatancloud9@cloudshell:~ (refined-sump-455515-d0) $ gcloud sql backups restore projects/refined-sump-455515-d0/backups/1744300221406 --restore-instance=my-mysql35
1. All current data on the instance will be lost when the backup is restored to an existing instance.
2. If restoring to a new instance, settings will be applied from the backup unless they are overridden.
Do you want to continue (Y/n)? y
```

Step 7: Monitor and Scale Cloud SQL

Check Instance Status

```
sh
CopyEdit
gcloud sql instances list
```

View Database Metrics

Use **Cloud Monitoring** to track CPU, memory, and storage usage.

Upgrade the Instance (Increase CPU/RAM)

```
sh
CopyEdit
gcloud sql instances patch my-mysql-instance --tier=db-n1-standard-4
```

Step 8: Delete Cloud SQL Instance (If Needed)

```
sh
CopyEdit
gcloud sql instances delete my-mysql-instance
```

```
kernatancloud9@cloudshell:~ (refined-sump-455515-d0) $ gcloud sql instances list
NAME: my-mysql35
DATABASE_VERSION: MYSQL_8_0
LOCATION: us-central1-e
TIER: db-n1-standard-2
PRIMARY_ADDRESS: 34.42.240.152
PRIVATE_ADDRESS: -
STATUS: RUNNING
kernatancloud9@cloudshell:~ (refined-sump-455515-d0) $ gcloud sql instances patch my-mysql35 --tier=db-n1-standard-4
The following message will be used for the patch API method.
{"name": "my-mysql35", "project": "refined-sump-455515-d0", "settings": {"tier": "db-n1-standard-4"}}
WARNING: This patch modifies a value that requires your instance to be restarted. Submitting this patch will immediately restart your instance if it's running.
Do you want to continue (Y/n)? y
Patching Cloud SQL instance...done.
Updated [https://sqladmin.googleapis.com/sql/v1beta4/projects/refined-sump-455515-d0/instances/my-mysql35].
kernatancloud9@cloudshell:~ (refined-sump-455515-d0) $ gcloud sql instances delete my-mysql35
All of the instance data will be lost when the instance is deleted.
Do you want to continue (Y/n)? y
Deleting Cloud SQL instance...done.
Deleted [https://sqladmin.googleapis.com/sql/v1beta4/projects/refined-sump-455515-d0/instances/my-mysql35].
kernatancloud9@cloudshell:~ (refined-sump-455515-d0) $
```


Conclusion

- ✓ **Cloud SQL for MySQL** simplifies database management with **automated scaling, high availability, and security**.
- ✓ Supports **manual and automatic backups, private networking, and IAM-based access control**.
- ✓ Easily integrates with **App Engine, Compute Engine, Kubernetes, and BigQuery**.

EXPERIMENT NUMBER 7

Cloud Pub/Sub: Real-Time Messaging and Communication

Google **Cloud Pub/Sub** is a **fully managed messaging service** that enables **real-time event-driven communication** between distributed applications. It follows a **publisher-subscriber** model, making it ideal for **asynchronous message processing, event streaming, and data ingestion pipelines**.

How Cloud Pub/Sub Works?

- ✓ **Publishers** → Send messages to a **topic**.
 - ✓ **Subscribers** → Receive messages from a **subscription**.
 - ✓ **Message Delivery** → Ensures **at-least-once delivery** with automatic retries.
 - ✓ **Event-Driven** → Triggers Cloud Functions, Dataflow, or other services.
-

Step 1: Enable Cloud Pub/Sub API

Before using Pub/Sub, enable the API:

```
sh
CopyEdit
gcloud services enable pubsub.googleapis.com
```

Step 2: Create a Pub/Sub Topic

A **topic** is where messages are sent.

Using Google Cloud Console

1. Open **Cloud Pub/Sub**
2. Click **Create Topic**
3. Enter a topic name (e.g., `my-topic`)
4. Click **Create**

Using gcloud CLI

```
sh
CopyEdit
gcloud pubsub topics create my-topic
```

Step 3: Create a Subscription

Subscribers need a **subscription** to receive messages.

Using Google Cloud Console

1. Open **Cloud Pub/Sub**
2. Click **Subscriptions** → **Create Subscription**
3. Choose the topic (`my-topic`)
4. Set **Delivery Type**:
 - **Pull** → Manual message retrieval

- **Push** → Automatic HTTP delivery to a service (e.g., Cloud Function)
5. Click **Create**

Using gcloud CLI

```
sh
CopyEdit
gcloud pubsub subscriptions create my-subscription --topic=my-topic
```

Step 4: Publish a Message

Using gcloud CLI

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

Step 5: Receive Messages

Pull Subscription (Manual Retrieval)

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

☑ **--auto-ack** ensures the message is acknowledged and removed from the queue.

Push Subscription (Automatic Delivery)

If using a push subscription, Pub/Sub will **send messages to a configured endpoint** (e.g., Cloud Function, HTTP API).

Step 6: Integrate with Cloud Functions (Event-Driven Processing)

Deploy a Cloud Function to **process messages automatically**:

❶ Create a Python function (`main.py`):

```
python
CopyEdit
import functions_framework

@functions_framework.cloud_event
def process_pubsub_event(event):
    """Triggered by a Pub/Sub message."""
    message = event.data["message"]["data"]
    decoded_message = base64.b64decode(message).decode("utf-8")
    print(f"Received message: {decoded_message}")
```

❷ Deploy the function:

```
sh
CopyEdit
gcloud functions deploy process_pubsub_event \
    --runtime python310 \
    --trigger-topic my-topic \
    --region us-central1
```

📌 **Now, whenever a message is published, this function will execute automatically.**

Step 7: Monitor & Manage Pub/Sub

List Topics

```
sh
CopyEdit
gcloud pubsub topics list
```

List Subscriptions

```
sh
CopyEdit
gcloud pubsub subscriptions list
```

Delete a Topic

```
sh
CopyEdit
gcloud pubsub topics delete my-topic
```

Delete a Subscription

```
sh
CopyEdit
gcloud pubsub subscriptions delete my-subscription
```

Use Cases for Cloud Pub/Sub



Real-time event processing → Process streaming events in real time.



Asynchronous microservices communication → Decouple services using Pub/Sub.



Data ingestion for BigQuery & Dataflow → Efficiently move data across services.



Trigger serverless functions → Automate tasks using Cloud Functions.

Conclusion

You've successfully experimented with **Cloud Pub/Sub** for **real-time messaging** and **event-driven processing**! 🎉

EXPERIMENT NUMBER 8

Multiple VPC Networks in Google Cloud: Benefits & Use Cases

Google Cloud **Virtual Private Cloud (VPC)** allows you to create and manage **multiple VPC networks** to better organize and isolate resources. Using multiple VPCs improves **security, scalability, and flexibility** in cloud architectures.

♦ Why Use Multiple VPC Networks?

- ✓ **Isolation & Security** – Separate resources for better access control (e.g., production vs. development).
 - ✓ **Multi-Tenancy** – Assign different VPCs to different teams, customers, or business units.
 - ✓ **Hybrid & Multi-Cloud Networking** – Connect on-premises infrastructure or other cloud providers securely.
 - ✓ **Compliance & Governance** – Keep sensitive workloads isolated for regulatory requirements.
 - ✓ **Traffic Segmentation** – Control and monitor internal communication between different workloads.
-

Step 1: Create Multiple VPC Networks

Using Google Cloud Console

1. Navigate to **VPC Networks**
2. Click **Create VPC Network**
3. Provide a **name** (e.g., `vpc-prod`, `vpc-dev`)
4. Choose **Custom Subnet Mode** to define subnets manually
5. Click **Create**

Using gcloud CLI

```
sh
CopyEdit
gcloud compute networks create vpc-prod --subnet-mode=custom
gcloud compute networks create vpc-dev --subnet-mode=custom
♦ --subnet-mode=custom allows you to define subnets explicitly instead of auto-created ones.
```

Step 2: Create Subnets in Each VPC

Each VPC network requires **subnets** in different regions.

```
sh
CopyEdit
gcloud compute networks subnets create subnet-prod \
  --network=vpc-prod \
  --region=us-central1 \
  --range=10.0.1.0/24

gcloud compute networks subnets create subnet-dev \
```

```
--network=vpc-dev \  
--region=us-central1 \  
--range=10.0.2.0/24
```

- ◆ `--range=10.0.1.0/24` defines the IP address range for the subnet.
-

◆ Step 3: Connecting Multiple VPCs

Since VPCs are isolated by default, you need **VPC Peering** or **VPN** to enable communication.

Option 1: VPC Peering (For Low-Latency Private Communication)

```
sh  
CopyEdit  
gcloud compute networks peerings create vpc-prod-to-dev \  
--network=vpc-prod \  
--peer-network=vpc-dev
```

```
gcloud compute networks peerings create vpc-dev-to-prod \  
--network=vpc-dev \  
--peer-network=vpc-prod
```

- ◆ This allows **private communication** between `vpc-prod` and `vpc-dev`.

Option 2: Cloud VPN (For Secure Communication Across Regions or Cloud Providers)

❶ Create VPN Gateway in `vpc-prod`

```
sh  
CopyEdit  
gcloud compute vpn-gateways create vpn-gateway-prod \  
--network=vpc-prod \  
--region=us-central1
```

❷ Create a VPN Tunnel to `vpc-dev`

```
sh  
CopyEdit  
gcloud compute vpn-tunnels create vpn-tunnel-prod-to-dev \  
--region=us-central1 \  
--peer-gcp-gateway=vpn-gateway-dev \  
--ike-version=2 \  
--shared-secret=my-vpn-secret
```

- ◆ Replace `my-vpn-secret` with a strong **pre-shared key**.
-

◆ Step 4: Set Up Firewall Rules for Communication

By default, VPC networks have **strict firewall rules**. You need to explicitly **allow traffic**.

```
sh  
CopyEdit  
gcloud compute firewall-rules create allow-internal \  
--network=vpc-prod \  
--allow tcp,udp,icmp \  
--source-ranges=10.0.0.0/16
```

```
gcloud compute firewall-rules create allow-internal \  
--network=vpc-dev \  

```

```
--allow tcp,udp,icmp \  
--source-ranges=10.0.0.0/16
```

- ◆ This allows communication **inside each VPC**.
-

◆ Step 5: Verify Connectivity

To check if multiple VPCs can communicate:

1 Create two VM instances in separate VPCs

```
sh  
CopyEdit  
gcloud compute instances create vm-prod \  
  --network=vpc-prod \  
  --zone=us-central1-a \  
  --subnet=subnet-prod  
  
gcloud compute instances create vm-dev \  
  --network=vpc-dev \  
  --zone=us-central1-a \  
  --subnet=subnet-dev
```

2 SSH into vm-prod and ping vm-dev

```
sh  
CopyEdit  
ping 10.0.2.2  
(Replace with the actual private IP of vm-dev.)
```

If successful, your **VPC peering or VPN is correctly configured!** 🎉

◆ Best Practices for Multiple VPC Networks

- ✓ Use **VPC Peering** for **low-latency private networking** between VPCs.
 - ✓ Use **Cloud VPN or Interconnect** for **cross-region or hybrid cloud setups**.
 - ✓ **Segment workloads** based on environments (e.g., **prod, dev, test**).
 - ✓ **Define IAM policies** to control access to VPCs based on team roles.
 - ✓ **Monitor network traffic** using **VPC Flow Logs & Cloud Logging**.
-

🎯 Conclusion

You have successfully explored **multiple VPC networks**, configured **subnets**, set up **VPC Peering**, and ensured **secure communication** between VPCs! 🚀

EXPERIMENT NUMBER 9

Cloud Monitoring in Google Cloud: Tracking & Analyzing Performance

Google **Cloud Monitoring** helps you track, analyze, and optimize the **performance, availability, and health** of your cloud resources. It provides **real-time metrics, dashboards, alerts, and logs** to detect and resolve issues quickly.

♦ Key Features of Cloud Monitoring

✓ **Real-time Metrics** – Collects performance data from Google Cloud services, VMs, and applications.

✓ **Dashboards** – Customizable dashboards for visualizing resource health.

✓ **Alerts & Notifications** – Set up alerts for CPU, memory, network usage, etc.

✓ **Logs Integration** – Works with **Cloud Logging** to analyze errors and troubleshoot issues.

✓ **Uptime Monitoring** – Ensures your websites and applications are available globally.

✓ **SLO & SLA Tracking** – Define and monitor **Service Level Objectives (SLOs)** to meet business goals.

Step 1: Enable Cloud Monitoring

Before using Cloud Monitoring, enable the API:

```
sh
CopyEdit
gcloud services enable monitoring.googleapis.com logging.googleapis.com
```

Alternatively, enable it in the **Google Cloud Console**:

1. Open **Cloud Monitoring**
 2. Click **Enable Cloud Monitoring**
-

Step 2: Create a Monitoring Dashboard

Using Google Cloud Console

1. Navigate to **Cloud Monitoring** → **Dashboards**
2. Click **Create Dashboard**
3. Add **widgets** (CPU, memory, network, etc.)
4. Click **Save**

Using gcloud CLI

To create a dashboard with CPU usage:

```
sh
CopyEdit
gcloud monitoring dashboards create --config-from-file=my-dashboard.json
(Requires a JSON configuration file defining metrics.)
```

Step 3: Set Up Alerts for Performance Issues

Alerts notify you of potential issues (e.g., high CPU, low memory).

Create an Alert in Cloud Console

1. Go to **Cloud Monitoring** → **Alerting**
2. Click **Create Policy**
3. Select **Condition** (e.g., CPU utilization > 80%)
4. Set a **Notification Channel** (Email, Slack, PagerDuty, etc.)
5. Click **Create**

Using gcloud CLI

To create an alert for **high CPU usage**:

```
sh
CopyEdit
gcloud alpha monitoring policies create \
  --display-name="High CPU Usage Alert" \
  --conditions="metric.type=compute.googleapis.com/instance/cpu/utilization,
threshold_value=0.8" \
  --notification-channels=projects/my-project/notificationChannels/12345
♦ This sends an alert when CPU usage exceeds 80%.
```

Step 4: Monitor Logs for Troubleshooting

Cloud Monitoring integrates with **Cloud Logging** to analyze system errors and application logs.

View Logs in Cloud Console

1. Go to **Cloud Logging** → **Logs Explorer**
2. Select a **resource type** (e.g., VM instance, Cloud Function)
3. Apply **filters** (e.g., severity=ERROR)
4. Analyze logs to troubleshoot issues

Using gcloud CLI

To view logs for a specific VM:

```
sh
CopyEdit
gcloud logging read "resource.type=gce_instance AND severity>=ERROR"
--limit=10
♦ Fetches the last 10 error logs from a Compute Engine instance.
```

Step 5: Set Up Uptime Checks

Uptime checks monitor website and service availability.

Create an Uptime Check in Cloud Console

1. Go to **Cloud Monitoring** → **Uptime Checks**
2. Click **Create Uptime Check**
3. Enter the **URL or IP address** of your service
4. Set **Frequency & Locations**
5. Click **Create**

Using gcloud CLI

sh

CopyEdit

```
gcloud monitoring uptime-checks create http \
  --display-name="My Website Uptime" \
  --host="example.com" \
  --path="/" \
  --check-interval=60s
```

- ♦ Checks **example.com** every 60 seconds.
-

♦ Best Practices for Cloud Monitoring

- ✓ **Use Dashboards** – Visualize key performance metrics in real time.
 - ✓ **Set Alerts** – Get notified when performance thresholds are exceeded.
 - ✓ **Enable Logging** – Capture and analyze logs for troubleshooting.
 - ✓ **Monitor Uptime** – Ensure websites and APIs are accessible globally.
 - ✓ **Define SLOs** – Track **Service Level Objectives (SLOs)** for reliability.
-

Conclusion

Google **Cloud Monitoring** helps you track, analyze, and maintain **optimal performance** for cloud resources. 🚀

EXPERIMENT NUMBER 10

Kubernetes Engine: Quick Start - Deploying a Containerized Application

Google **Kubernetes Engine (GKE)** allows you to deploy, manage, and scale containerized applications using **Kubernetes**. This guide will help you deploy a **containerized application** on a **GKE cluster** quickly.

♦ Step 1: Enable Required APIs

Before using GKE, enable the necessary APIs:

```
sh
CopyEdit
gcloud services enable container.googleapis.com
```

Or enable it via **Google Cloud Console**:

1. Navigate to **APIs & Services**
 2. Search for **Kubernetes Engine API**
 3. Click **Enable**
-

Step 2: Create a GKE Cluster

A **Kubernetes cluster** is required to deploy applications.

Using Google Cloud Console

1. Go to **Kubernetes Engine** → **Clusters**
2. Click **Create Cluster**
3. Choose **Standard Cluster**
4. Set the **name** (e.g., my-cluster)
5. Choose a **region** (e.g., us-central1)
6. Click **Create**

Using gcloud CLI

```
sh
CopyEdit
gcloud container clusters create my-cluster \
  --num-nodes=3 \
  --zone=us-central1-a
```

- ♦ Creates a **3-node cluster** in us-central1-a.

Verify the cluster:

```
sh
CopyEdit
gcloud container clusters list
```

Step 3: Connect to the Cluster

Once the cluster is created, configure `kubectl` to connect:

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

Check cluster nodes:

```
sh
CopyEdit
kubectl get nodes
```

- ◆ This ensures **Kubernetes is running** correctly.
-

Step 4: Deploy a Containerized Application

For this quick start, we will deploy an **Nginx web server**.

Create a Deployment

```
sh
CopyEdit
kubectl create deployment my-app --image=nginx
```

- ◆ This deploys **Nginx** in the cluster.

Verify the deployment:

```
sh
CopyEdit
kubectl get deployments
```

Expose the Deployment

To access the app, expose it with a **LoadBalancer** service:

```
sh
CopyEdit
kubectl expose deployment my-app --type=LoadBalancer --port=80
```

Check the service:

```
sh
CopyEdit
kubectl get services
```

Look for the **EXTERNAL-IP** of your app. If pending, wait a few minutes:

```
sh
CopyEdit
kubectl get services my-app --watch
```

Once available, **open in a browser**:

```
cpp
CopyEdit
http://EXTERNAL-IP
```

You should see the **Nginx Welcome Page!** 🎉

◆ Step 5: Clean Up Resources

To avoid unnecessary costs, **delete the cluster** when done:

```
sh
CopyEdit
gcloud container clusters delete my-cluster --zone=us-central1-a
```

♦ Best Practices for Deploying on GKE

- ✓ **Use Managed GKE** – Reduces operational overhead.
 - ✓ **Deploy via YAML** – Define deployments in `deployment.yaml` for better control.
 - ✓ **Use Ingress** – For advanced traffic management instead of `LoadBalancer`.
 - ✓ **Enable Auto-scaling** – Optimize resource usage with **HPA (Horizontal Pod Autoscaler)**.
 - ✓ **Monitor & Secure** – Use **Cloud Monitoring & IAM** roles to enhance security.
-

Conclusion

You have successfully **created a GKE cluster, deployed a containerized app, and exposed it to the internet!** 🚀