

CLOUD COMPUTING LAB (BCS601)

Experiment-01: Creating a Virtual Machine in Google Cloud

Step 1: Sign in to Google Cloud Console

1. **Go to Google Cloud Console:**
 - Visit: <https://cloud.google.com/free>
2. **Log in with your Google Account.**
3. **Select or create a new project:**
 - From the top navigation bar, select an existing project or create a new one.

Step 2: Open Compute Engine

1. **Navigate to Compute Engine:**
 - In the left sidebar, click on **"Compute Engine"**.
 - Then, select **"VM instances"**.
2. **Click "Create Instance":**
 - Click the **"Create Instance"** button to begin configuring your new virtual machine.

Step 3: Configure the Virtual Machine

1. **Name the VM:**
 - Enter a name for your VM instance (e.g., `my-vm-instance`).
 2. **Select the Region and Zone:**
 - Choose a region close to your target audience or users (e.g., `us-central1`).
 - Select an availability zone (e.g., `us-central1-a`).
 3. **Choose the Machine Configuration:**
 - Under **"Machine Configuration"**, choose:
 - **Series:** (e.g., E2, N1, N2, etc.)
 - **Machine type:** Select based on your CPU and RAM requirements.
 - Example:
 - **e2-medium (2 vCPU, 4GB RAM)**
 - **n1-standard-4 (4 vCPU, 16GB RAM)**
 - Click **"Customize"** if you need specific CPU and RAM configurations.
 4. **Select the Boot Disk (Operating System):**
 - Click **"Change"** under **Boot Disk**.
 - Choose an operating system (e.g., **Ubuntu, Windows, Debian**).
 - Set the disk size (e.g., **20GB** or more).
 5. **Networking and Firewall:**
 - Enable **"Allow HTTP Traffic"** or **"Allow HTTPS Traffic"** if needed for web access.
 - For advanced networking configurations, click on **"Advanced options"**.
-

Step 4: Create and Deploy the VM

1. **Review all configurations:**
 - Double-check all the settings you have selected for the VM.
2. **Click "Create":**
 - Once you are satisfied with the configuration, click **"Create"** to deploy the VM.
3. **Wait for the instance to be provisioned:**
 - Google Cloud will take a few minutes to provision your VM.

Step 5: Connect to the VM

1. **Using SSH (Web):**
 - Go to **Compute Engine** → **VM Instances**.
 - Click the **"SSH"** button next to your VM instance to connect via the browser-based SSH terminal.
2. **Using SSH (Terminal):**
 - Open **Google Cloud SDK (Cloud Shell)** or your local terminal.
 - Run the following command to connect:

```
gcloud compute ssh your-instance-name --zone=us-central1-a
```

Step 6: Verify and Use the VM

1. **Check CPU and Memory:**
 - To view CPU details, run:
2. **Install Required Software (e.g., Apache Web Server):**
 - Update your package list and install Apache by running:

```
lscpu
```

```
free -h
```

```
sudo apt update && sudo apt install apache2 -y
```

Step 7: Stop or Delete the VM (Optional)

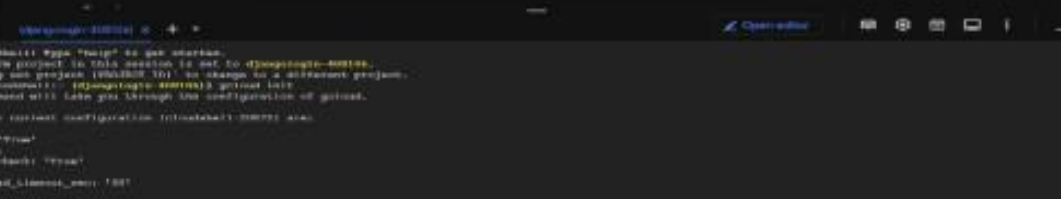
1. **Stop the VM:**
 - To stop the VM, run the following command:

```
gcloud compute instances stop your-instance-name --zone=us-central1-a
```

2. **Delete the VM:**
 - To delete the VM, run:

```
gcloud compute instances delete your-instance-name --zone=us-central1-a
```

Output:



```

Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to djangogpt-00196.
Use "gcloud config set project [PROJECT ID]" to change to a different project.
$ gcloud config set project djangogpt-00196
Warning: This command will take you through the configuration of gcloud.
Settings from your current configuration (cloudshell-b3072) and
cloudshell-b3072:
  account: "true"
  disable_usage_reporting: "false"
  disable_update_check: "true"
  compute:
    get_metadata_read_timeout_sec: 700
  dns:
    account: "aythasomathio@gmail.com"
    disable_usage_reporting: "false"
    provider: "cloudshell-b3072"
    service:
      dnslookup: default
Pick configuration to use:
[[1] Re-initialize this configuration (cloudshell-b3072) with new settings
[[2] Create a new configuration.
Please enter your numeric choice: 2
Please enter a value between 1 and 2: 1
Your current configuration has been set to: (cloudshell-b3072)
You can skip diagnostic steps time by using the following flag:
$ gcloud init --skip-diagnostics
Between diagnostic screens you find code between connection issues.
Checking network connection...done.
Reachability check passed.
Network diagnostic passed (1/1) checks passed.
Choose the account you want to use for this configuration.
To see a full list of accounts, exit this command and run it to the gcloud cli with your login configuration file, then run this command again.
select an account:
[[1] aythasomathio@gmail.com
[[2] Sign in with a new Google Account

```

```
Google Cloud | gcplogin | Search (/) for resources, docs, products and more | + [Icons] | [Profile]
```

```
gcp shell
Terminal | gcpshell-40810a1 x + - | Open editor | [Icons] | [Close]
```

```
[12] us-west1-c
[13] us-west1-a
[14] europe-west4-a
[15] europe-west4-b
[16] europe-west4-c
[17] europe-west1-b
[18] europe-west1-d
[19] europe-west1-c
[20] europe-west3-c
[21] europe-west3-a
[22] europe-west3-b
[23] europe-west2-c
[24] europe-west2-b
[25] europe-west2-a
[26] asia-east1-b
[27] asia-east1-a
[28] asia-east1-c
[29] asia-northeast1-b
[30] asia-northeast1-a
[31] asia-northeast1-c
[32] asia-northeast1-b
[33] asia-northeast1-c
[34] asia-northeast1-a
[35] asia-south1-c
[36] asia-south1-b
[37] asia-south1-a
[38] australia-southeast1-b
[39] australia-southeast1-c
[40] australia-southeast1-a
[41] southamerica-east1-b
[42] southamerica-east1-c
[43] southamerica-east1-a
[44] africa-south1-a
[45] africa-south1-b
[46] africa-south1-c
[47] asia-east2-a
[48] asia-east2-b
[49] asia-east2-c
[50] asia-northeast3-a

Did not print [51] options.
Too many options [125]. Enter "list" at prompt to print choices fully.
Please enter numeric choice or text value (must exactly match list item)
```

Experiment-02: 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-central1a**
 - 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

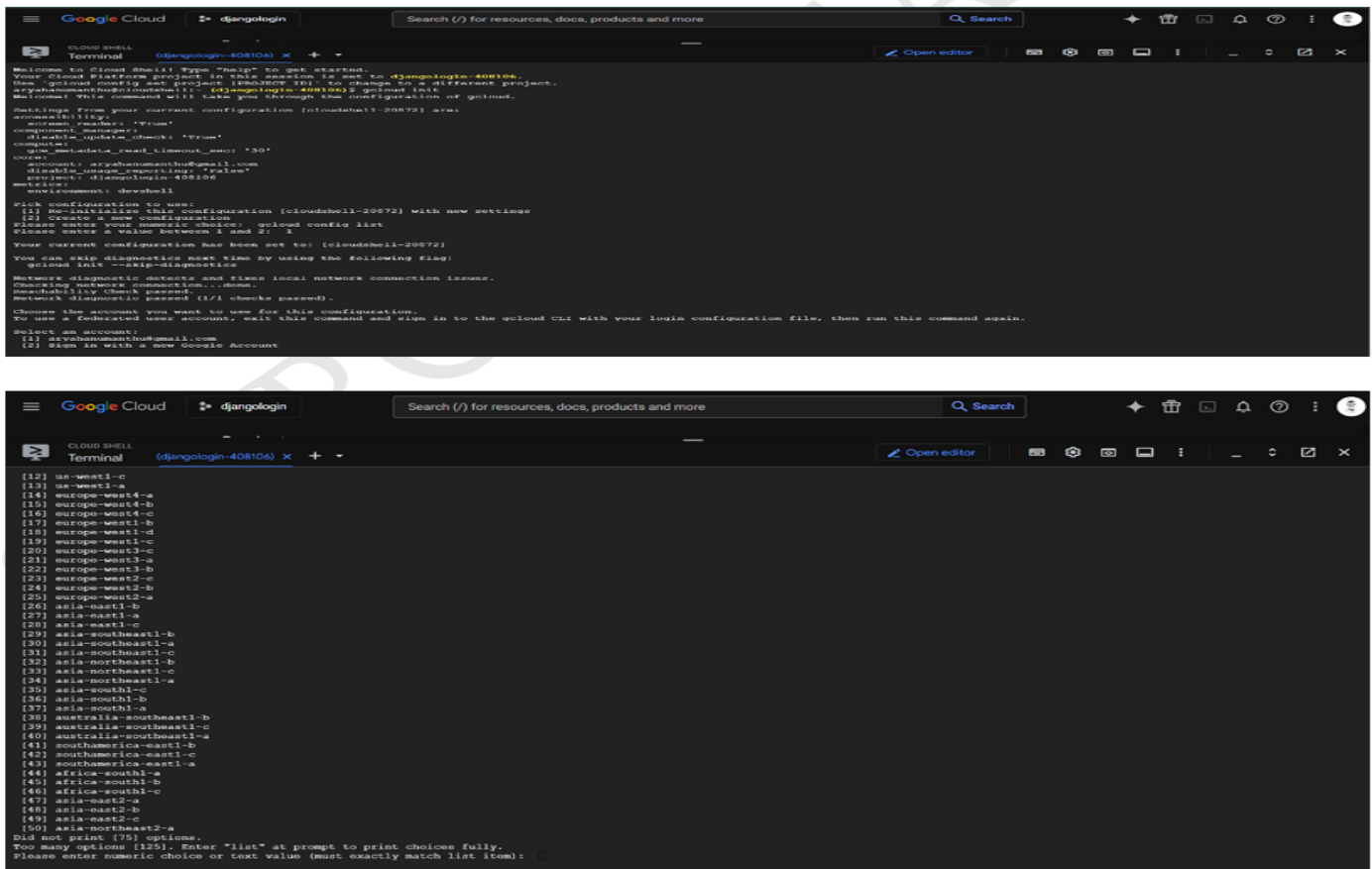
Check your billing accounts using:

- **gcloud beta billing accounts list**

Step 13: Exit Cloud Shell

Simply close the Cloud Shell tab to exit.

Output



Experiment-03 Cloud Functions: Create and deploy a Cloud Function to automate a specific task based on a Cloud Storage event. Step 1: Open Cloud Shell

Step 1:

Create a GCP Project: If you don't have one, create a project in the Google Cloud Console.

New Project



You have 19 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)

[Manage Quotas](#)

Project name *

Cloud-Functions-Demo



Project ID: cloud-functions-demo-454613. It cannot be changed later. [Edit](#)

Organization *

svit.co.in



Select an organization to attach it to a project. This selection can't be changed later.

Location *

 svit.co.in

[Browse](#)

Parent organization or folder

Create

Cancel

Step 2:

Enable Required APIs

Enable the Cloud Functions API and Cloud Storage API:

Go to the API Library in the Google Cloud Console.

Search for and enable "Cloud Functions API" and "Cloud Storage API."

or

- Run the following command to enable both necessary APIs:

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

Step 3:

Install gcloud CLI: Install and configure the Google Cloud SDK (gcloud CLI) on your local machine.

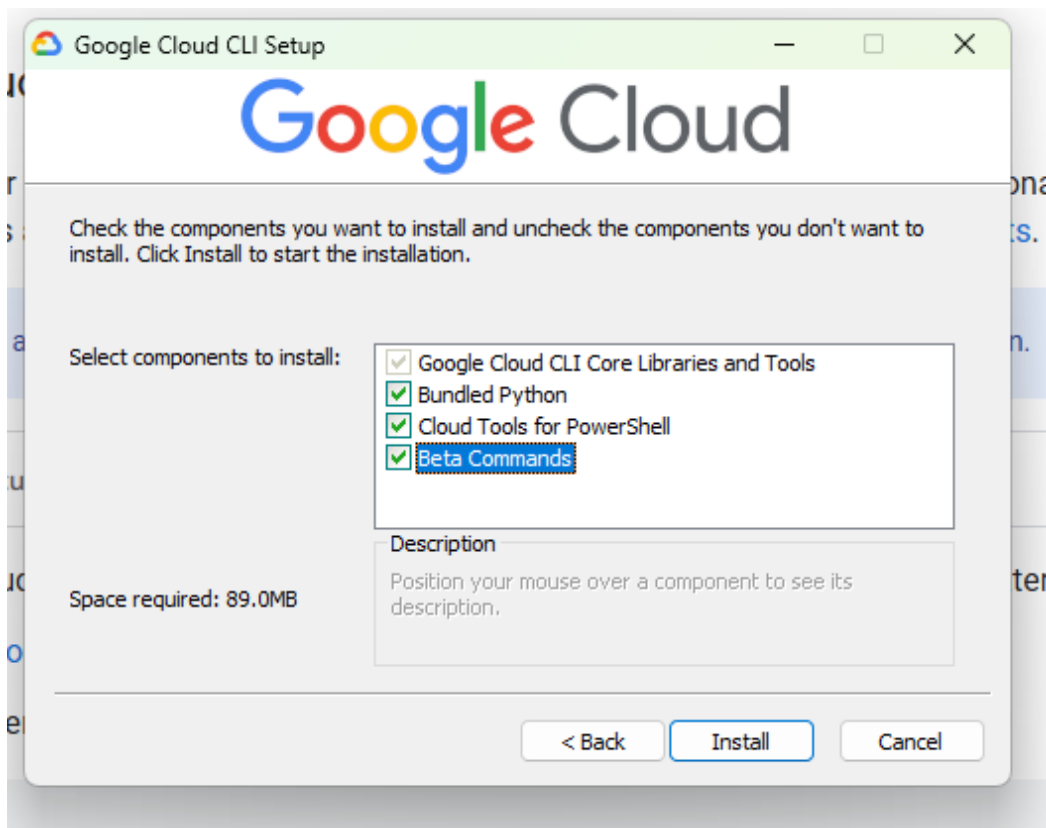
Follow the instructions mentioned as below

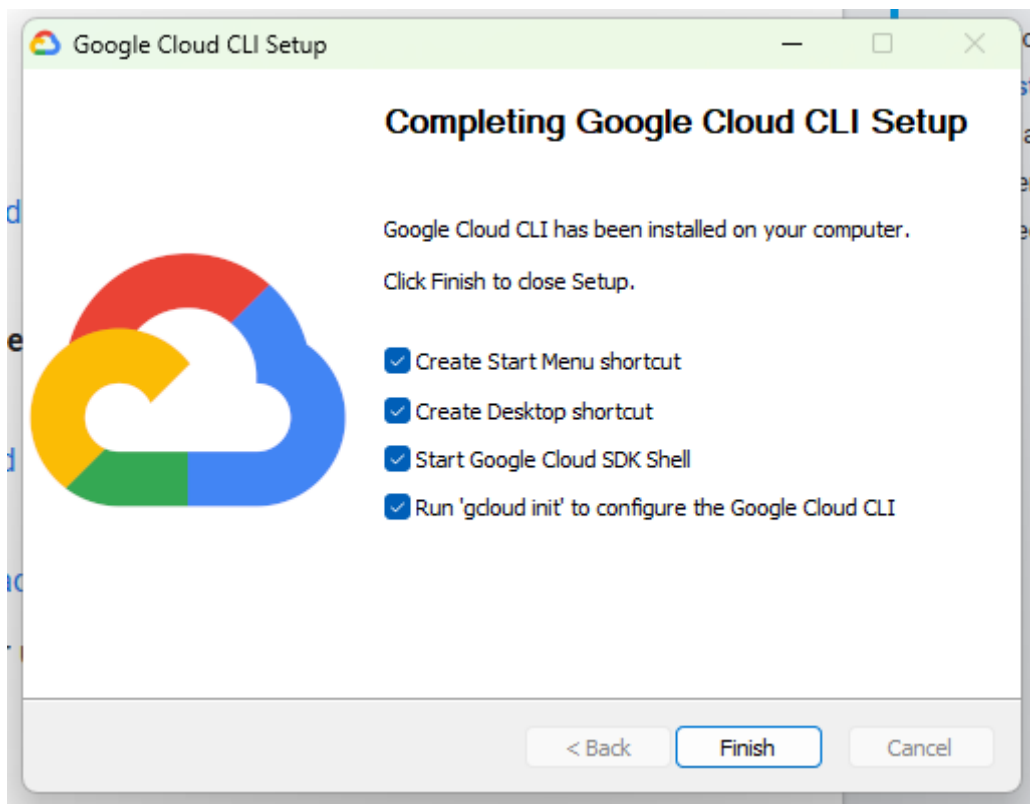
<https://cloud.google.com/sdk/docs/install>

Based on personal computer operating system

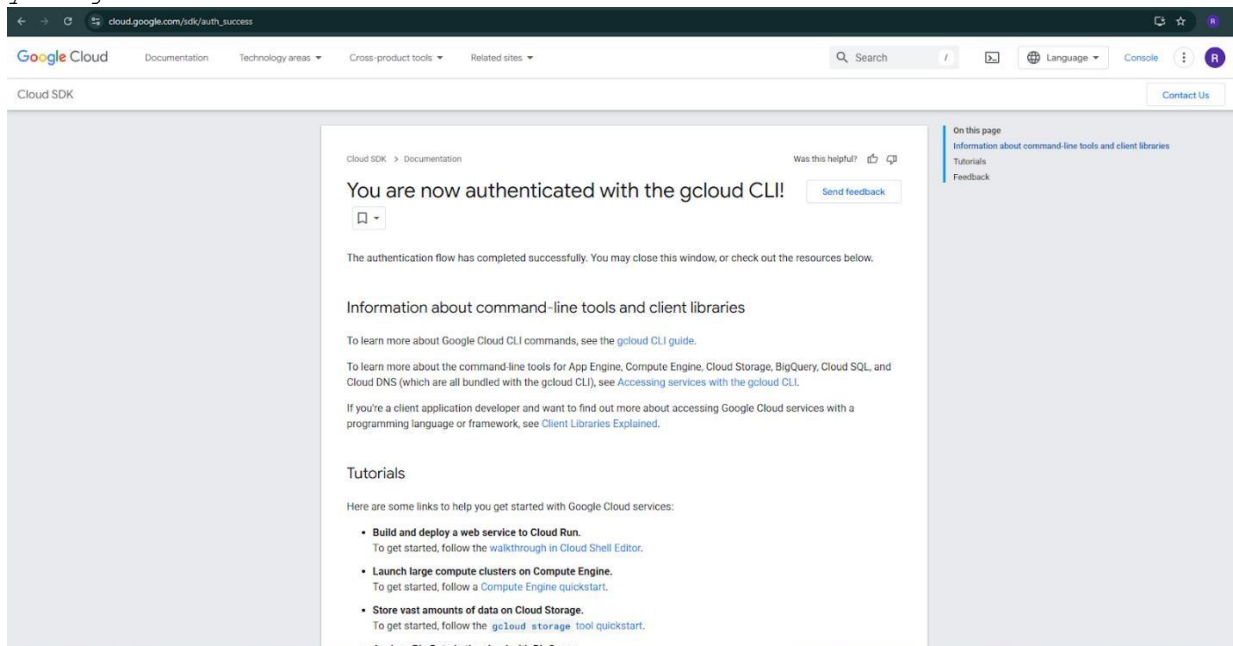
For windows

<https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe>





After finishing it will prompt a command prompt and it will ask you to authenticate with your gmail id.



```
C:\WINDOWS\SYSTEM32\cmd X + v
You can skip diagnostics next time by using the following flag:
gcloud init --skip-diagnostics

Network diagnostic detects and fixes local network connection issues.
Checking network connection...done.
Reachability Check passed.
Network diagnostic passed (1/1 checks passed).

You must sign in to continue. Would you like to sign in (Y/n)? y

Your browser has been opened to visit:

https://accounts.google.com/o/oauth2/auth?response_type=code&client_id=32555940559.apps.googleusercontent.com&redirect_uri=http%3A%2F%2Flocalhost%3A8085%2F&scope=openid+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloud-platform+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fappengine.admin+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fsqlservice.login+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcompute+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Faccounts.reauth&state=A3Ju2CqjmKA00UmzzieynR0nzBK00F&access_type=offline&code_challenge=3ZDfh8swH2aphksTnYDPH_f5xX0wWhBhfzdIWAzNdY0&code_challenge_method=S256

You are signed in as: [ramanjinamma.g@saividya.ac.in].

Pick cloud project to use:
[1] booming-flash-451416-q5
[2] cloud-functions-demo-454613
[3] plexiform-arc-451408-c4
[4] ramavm-453008
[5] Enter a project ID
[6] Create a new project
Please enter numeric choice or text value (must exactly match list item): |
```

Complete the authentication process.

Step 4: Create a Cloud Storage Bucket

1. **Create a Cloud Storage bucket** (if you don't already have one):
 - o Run the following command to create a bucket:

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

- o Replace BUCKET_NAME with a unique name for your bucket. Make sure the bucket name is globally unique, as Google Cloud Storage bucket names are unique across all users.

Step 5: Write the Cloud Function Code

1. **Create a working directory:**
 - o Run the following commands to create a directory for the Cloud Function and navigate into it:

```
mkdir gcs-function && cd gcs-function
```

2. **Create and open the main.py file:**
 - o Run the following command to create and open the file in the google cloud CLI:

```
notepad main.py
```

3. **Write the Cloud Function code:**
 - o Inside main.py, add the following Python code:

```
import functions_framework
from google.cloud import storage

@functions_framework.cloud_event
def process_storage_event(cloud_event):
    """Triggered when a new file is uploaded to Cloud Storage."""
```

```
data = cloud_event.data
bucket_name = data["bucket"]
file_name = data["name"]

print(f"File '{file_name}' was uploaded to bucket '{bucket_name}'.")

# Example: Perform an automated task (e.g., rename the file)
rename_uploaded_file(bucket_name, file_name)

def rename_uploaded_file(bucket_name, file_name):
    """Renames the uploaded file by adding 'processed_' prefix."""
    storage_client = storage.Client()
    bucket = storage_client.bucket(bucket_name)
    blob = bucket.blob(file_name)

    new_file_name = f"processed_{file_name}"
    new_blob = bucket.rename_blob(blob, new_file_name)

    print(f"File renamed to: {new_blob.name}")
```

Save and close the file:

Step 6: Create a `requirements.txt` File

1. Create and open the `requirements.txt` file:

- Run the following command to create and open the file:

```
notepad requirements.txt
```

2. Add the required dependency:

- Add the following line to `requirements.txt` to include the `functions-framework` library:

```
functions-framework  
google-cloud-storage
```

3. Save and close the file:

Step 7: Deploy the Cloud Function

1. Deploy the Cloud Function:

- Run the following command to deploy the Cloud Function:

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

- Replace `BUCKET_NAME` with the actual name of your Cloud Storage bucket.
- This command specifies that the function will trigger on a file upload to the specified Cloud Storage bucket and will use Python 3.11 runtime in the `us-central1` region.

Step 8: Test the Cloud Function

1. Upload a file to the Cloud Storage bucket:

- Run the following command to upload a test file to your bucket:

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

- Replace `BUCKET_NAME` with the name of your Cloud Storage bucket.

2. Check the function logs:

- After uploading the file, check the logs to verify that the function executed successfully:

```
gcloud functions logs read gcs_trigger --region=us-central1
```

- This will show logs for the function `gcs_trigger` to confirm if the Cloud Function was triggered when the file was uploaded.

Output:

```
Google Cloud  djangologin Search (/) for resources, docs, products and more Search
CLOUD SHELL
Terminal djangologin-408106 + - Open editor
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to djangologin-408106.
Use 'gcloud config set project [PROJECT_ID]' to change to a different project.
aryahammanthu@cloudshell:~ (djangologin-408106) $ gcloud services enable cloudfunctions.googleapis.com storage.googleapis.com
Operation "operations/acf.p2-238240780218-31034c13-7420-48fe-bc73-37edafe0f2b4" finished successfully.
aryahammanthu@cloudshell:~ (djangologin-408106) $ gcloud storage buckets create BUCKET_NAME --location=us-central1
ERROR: (gcloud.storage.buckets.create) "gcloud storage buckets create" only accepts bucket URIs.
Example: "gs://bucket"
Received: "file:///BUCKET_NAME"
aryahammanthu@cloudshell:~ (djangologin-408106) $ gcloud storage buckets create main --location=us-central1
ERROR: (gcloud.storage.buckets.create) "gcloud storage buckets create" only accepts bucket URIs.
Example: "gs://bucket"
Received: "file:///main"
aryahammanthu@cloudshell:~ (djangologin-408106) $ mkdir gcs-function && cd gcs-function
aryahammanthu@cloudshell:~/gcs-function (djangologin-408106) $ ls
aryahammanthu@cloudshell:~/gcs-function (djangologin-408106) $ ls
aryahammanthu@cloudshell:~/gcs-function (djangologin-408106) $ nano main.py
aryahammanthu@cloudshell:~/gcs-function (djangologin-408106) $ nano requirements.txt
aryahammanthu@cloudshell:~/gcs-function (djangologin-408106) $
```

Experiment-04 App Engine: Deploy a web application on App Engine with automatic scaling enabled.

Step 1: Enable Required APIs

Before deploying your application, enable the necessary APIs for Google Cloud services.

1. Open your terminal (Cloud Shell or local terminal with `gcloud` installed).
2. Run the following command to enable the App Engine API:

```
gcloud services enable appengine.googleapis.com
```

Step 2: Create an App Engine Application

To create an App Engine application in your Google Cloud project, run the following command:

1. Replace `us-central1` with your preferred region, if necessary.

```
gcloud app create --region=us-central1
```

Step 3: Create a Simple Web Application

1. **Open Cloud Shell** (or your local terminal) and create a project directory:

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

2. **Create the `main.py` file** for your Flask application:

```
nano main.py
```

3. **Add the following code** to the `main.py` file:

```
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 the file:** Press CTRL + X, then Y, then Enter.

Step 4: Create a `requirements.txt` File

1. Create the `requirements.txt` file to list dependencies for the application:

```
nano requirements.txt
```

2. **Add the following content** to the `requirements.txt` file:

```
Flask
unicorn
```

3. **Save and close the file:** Press CTRL + X, then Y, then Enter.

Step 5: Create an app.yaml File for App Engine

1. **Create the app.yaml configuration file:**

```
nano app.yaml
```

2. **Add the following content** to the app.yaml file:

```
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
```

3. **Save and close the file:** Press CTRL + X, then Y, then Enter.

Step 6: Deploy the Web Application

1. Now, deploy your application to Google App Engine using the following command:

```
gcloud app deploy
```

2. You will be prompted to confirm the deployment. Type Y to confirm.

Step 7: Access the Deployed Application

Once the deployment is complete, you can access your deployed web application:

1. Run the following command to open the web app in a browser:

```
gcloud app browse
```

This command 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

1. **Check the logs** of your application to see any potential errors or messages:

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

2. **Monitor the deployed services** to view the status and scaling information:

```
gcloud app services list
```

output:

```
Google Cloud  django  Search (/) for resources, docs, products and more  Search  ✦ 🗑️ 📄 🔔 ⓘ ⋮ ⚙️

CLOUD SHELL
Terminal (django-408106) x +
Open editor  📄 ⚙️ 📄 🗑️ ⓘ ⋮ ✕

Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to django-408106.
Use 'gcloud config set project [PROJECT_ID]' to change to a different project.
aryahanmanthu@cloudshell:~ (django-408106) $ gcloud services enable appengine.googleapis.com
Operation "operations/acat.p2-238240780218-0f79a5ef-8eal-45c1-8d73-851168750010" finished successfully.
aryahanmanthu@cloudshell:~ (django-408106) $ gcloud app create --region=us-central1
You are creating an app for project [django-408106].
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>.

mkdir app-engine-demo && cd app-engine-demo
^C

Command killed by keyboard interrupt

aryahanmanthu@cloudshell:~ (django-408106) $ mkdir app-engine-demo && cd app-engine-demo
aryahanmanthu@cloudshell:~/app-engine-demo (django-408106) $ nano main.py
aryahanmanthu@cloudshell:~/app-engine-demo (django-408106) $ nano requirements.txt
aryahanmanthu@cloudshell:~/app-engine-demo (django-408106) $ nano app.yaml
aryahanmanthu@cloudshell:~/app-engine-demo (django-408106) $ gcloud app deploy
Services to deploy:

descriptor:      [/home/aryahanmanthu/app-engine-demo/app.yaml]
source:          [/home/aryahanmanthu/app-engine-demo]
target project:  [django-408106]
target service:  [default]
target version:  [20250131t153405]
target url:      [https://django-408106.uc.r.appspot.com]
target service account: [django-408106@appspot.gserviceaccount.com]

Do you want to continue (Y/n)? y

Beginning deployment of service [default]...
Created .gcloudignore file. See 'gcloud topic gcloudignore' for details.
Uploading 3 files to Google Cloud Storage
33%
67%
100%
100%
File upload done.
Updating service [default]...failed.
```


Experiment-05

Cloud Storage: Qwikstart: Google Cloud Storage provides scalable and secure object storage for managing data, accessible via the Cloud Console or gsutil CLI. OR Creating Lambda Functions Using the AWS SDK for Python

Step 1: Open Google Cloud Console

1. Go to <https://console.cloud.google.com>
2. Sign in with your Google account, if not already signed in.
3. Ensure you have an active **Google Cloud Project**:
 - Click the **project dropdown** in the top navigation bar.
 - Select an existing project or **create a new one**.

Step 2: Enable Cloud Storage API (if not enabled)

1. 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).
5. Select a **Storage Class**:
 - **Standard** (frequent access)
 - **Nearline** (monthly access)
 - **Coldline** (infrequent access)
 - **Archive** (long-term storage)
6. Choose **Access Control**:
 - **Uniform** (simpler)
 - **Fine-grained** (more detailed)
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. Choose a file from your computer and click **Open**.
4. Wait for the upload to complete.

✓ Your file is now stored in the cloud.

Step 5: Download a File from the Bucket

1. Go to your bucket in **Storage → Buckets**.
2. Click on the file you want to download.
3. Click the **Download** button.

✓ The file will be saved to your computer.

◆ Step 6: Make a File Public (Optional)

1. Open your bucket and click on the file.
2. Go to the **Permissions** tab.
3. Click **Add Principal**.
4. In the **New principals** field, type:

allUsers

5. Select the role:

Storage Object Viewer (roles/storage.objectViewer)

6. Click **Save**.

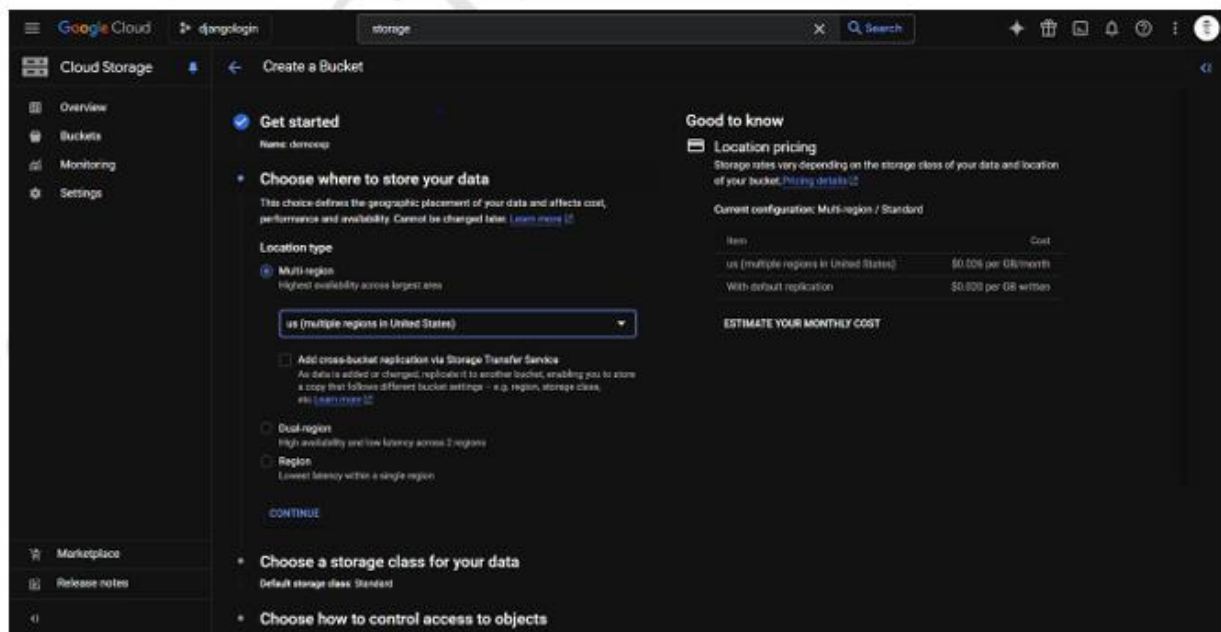
<https://storage.googleapis.com/your-unique-bucket-name/your-file-name>

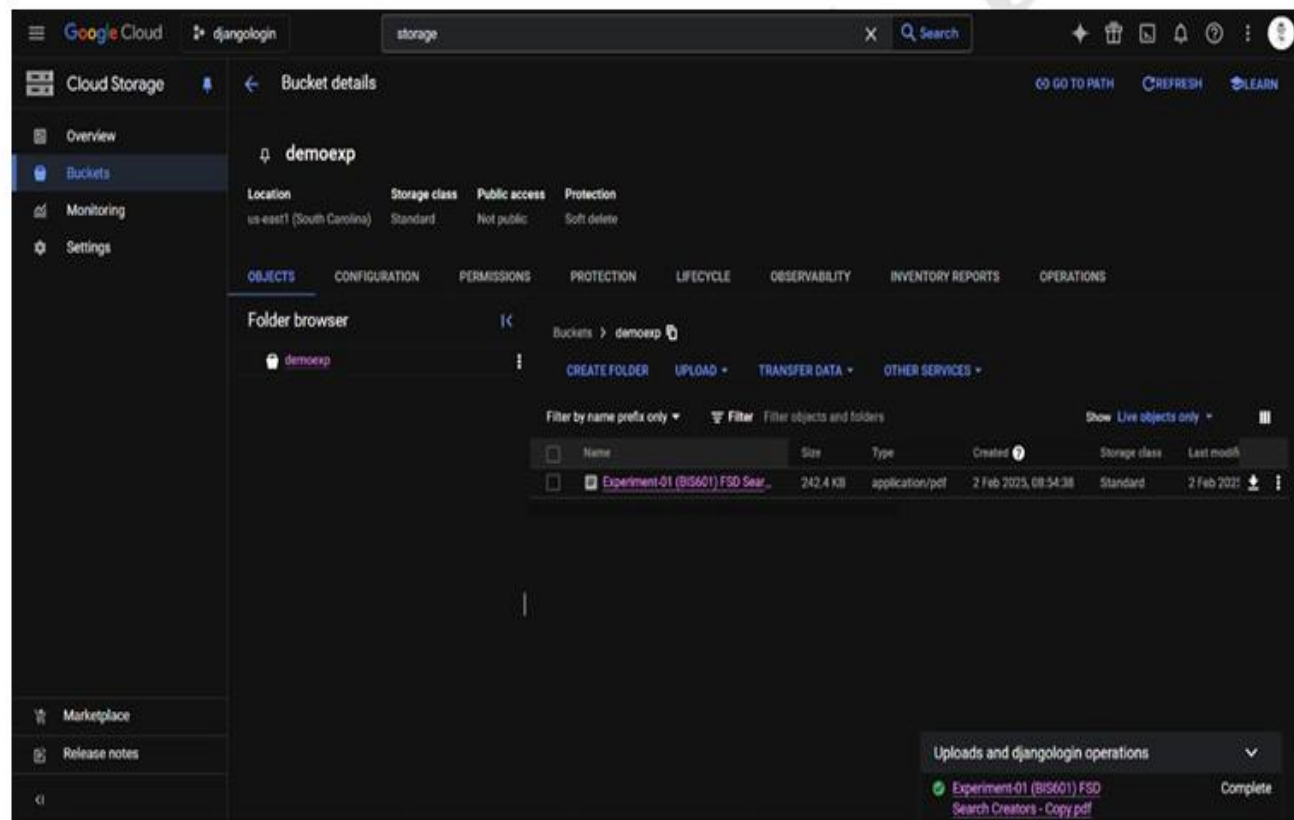
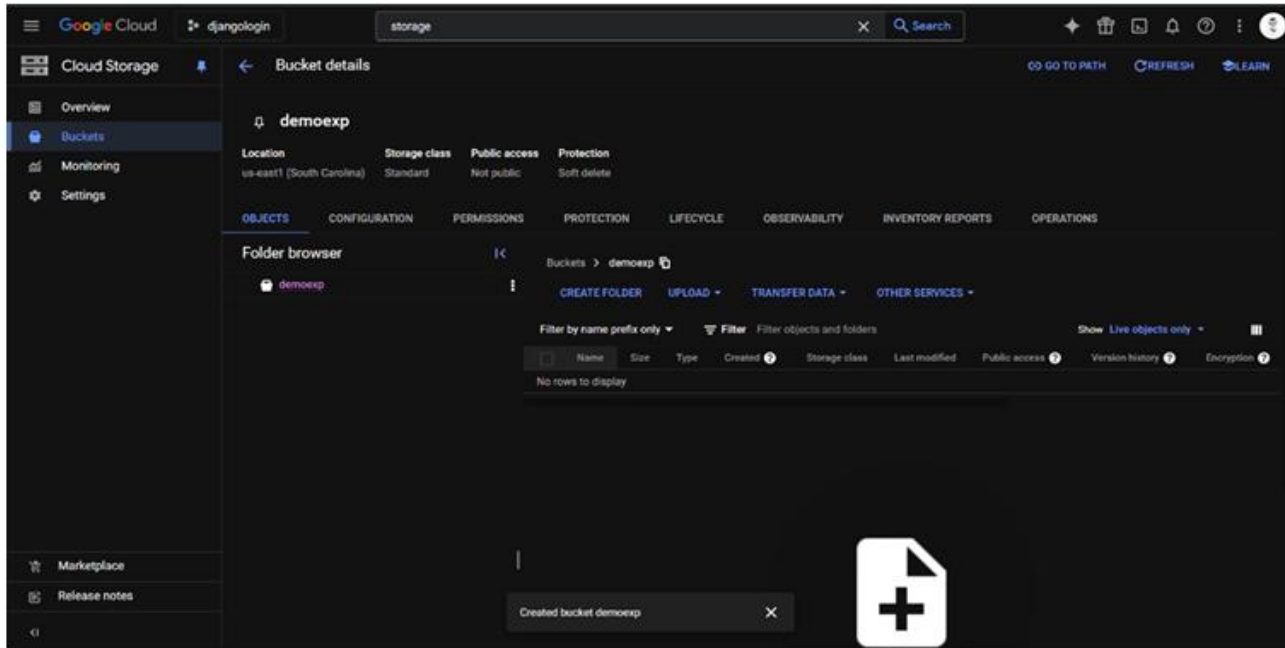
Anyone can access your file using this link.

Step 7: Delete a File or Bucket (Optional)

- To delete a **file**, click the file and select **Delete**.
- To delete a **bucket**, go to **Storage** → **Buckets**, select the bucket, and click **Delete**.

Output





Experiment-06

Cloud SQL for MySQL: Discover how Google Cloud SQL for MySQL provide automated management and high availability for MySQL databases? OR Migrating a Web Application to Docker Containers.

Key Features of Cloud SQL for MySQL

Automated Management:

1. **Automatic Backups** – Cloud SQL provides daily automated backups and **point-in-time recovery**.
2. **Automatic Updates & Patching** – Google automatically applies **security patches**.
3. **High Availability (HA):**
 - **Automatic Failover** – High-availability instances automatically switch to a **standby node** if the primary fails.
 - **Regional Replication** – Cloud SQL offers **multi-zone high availability** for disaster recovery.
 - **Failover Support** – In case a zone fails, the system automatically switches to a standby zone.
 - **Read Replicas** – You can create **read replicas** for **load balancing** and **performance improvement**.

Security & Compliance:

1. **IAM-Based Access Control** – Secure access via **Identity and Access Management (IAM)**.
2. **Encryption** – Data is encrypted **at rest** and **in transit**.
3. **VPC Peering & Private IPs** – Secure database connections using **private networking**.

Scalability & Performance:

1. **Automatic Storage Increase** – If storage runs out, Cloud SQL will automatically expand storage capacity.
2. **Vertical Scaling** – You can **increase CPU** and **memory** as needed.
3. **Read Replicas** – Scale reads by distributing queries across replicas to improve performance.

Set Up Cloud SQL for MySQL

◆ Step 1: Enable Cloud SQL API

1. Open **Google Cloud Console**: <https://console.cloud.google.com>
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 the **Navigation Menu (≡)** → **SQL**.
2. Click **Create Instance** → Choose **MySQL**.
3. Configure the following:
 - **Instance ID**: Set a unique name (e.g., my-mysql-instance).
 - **Password**: Set a password for the root user.
 - **Region & Zone**: Choose the region and zone closest to your application for better performance.
 - **Machine Type**: Choose an appropriate **CPU** and **RAM** for your needs.
 - **Storage Capacity**: Set the storage size and enable **auto-increase** if necessary.
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 the **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:

In your DATABASES settings (for 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. Click **Save**.

◆ Step 5: Create a Read Replica (Optional)

1. Open your instance → Click **Create Read Replica**.
2. Select the **region** and give the replica a **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
```

Replace BACKUP_ID with the actual backup ID you wish to restore.

Output

The screenshot displays the Google Cloud SQL console interface. At the top, there's a navigation bar with the Google Cloud logo, a search bar, and user information. The main content area is titled 'Cloud SQL' and describes it as a fully managed database service for MySQL, PostgreSQL, and SQL Server. Below this, there are two buttons: 'CREATE INSTANCE WITH YOUR FREE CREDITS' and 'MIGRATE DATABASE'. The 'Get started' section is divided into three columns: 'MySQL', 'POSTGRESQL', and 'SQL SERVER'. Each column has a 'Sandbox' section with details about minimally provisioned instances, a 'Development' section with details about performance and cost, and a 'Production' section with details about high availability and durability. On the right side, there's a 'Recommended for you' section with links to various documentation pages.

Cloud SQL
Cloud SQL offers a fully managed database service for MySQL, PostgreSQL, and SQL Server, reducing your overall cost of operations and freeing up teams to focus on innovation.

[CREATE INSTANCE WITH YOUR FREE CREDITS](#) [MIGRATE DATABASE](#)

Get started

[MYSQL](#) [POSTGRESQL](#) [SQL SERVER](#)

Sandbox
Minimally provisioned, cost-effective instances. Use to learn about and test Cloud SQL.

Cloud SQL, Enterprise edition
MySQL 8.0
2 vCPUs, 8 GB RAM, 10 GB SSD storage
Single zone
Up to 7 days PITR

[→ Start with Sandbox](#)

Development
Performant but not highly available, while reducing cost by provisioning less compute.

Cloud SQL, Enterprise Plus edition
MySQL 8.0
4 vCPUs, 32 GB RAM, 250 GB SSD storage
Single zone
Up to 35 days PITR
Data cache for enhancing read throughput

[→ Start with Development](#)

Production
Optimised for the most critical workloads. Highly available, performant and durable.

Cloud SQL, Enterprise Plus edition
MySQL 8.0
8 vCPUs, 64 GB RAM, 250 GB SSD storage
Multi-zone (99.99% availability SLA)
Up to 35 days PITR
Data cache for enhancing read throughput

[→ Start with Production](#)

Recommended for you

- [Cloud SQL overview](#) [Help document](#)
Learn about Cloud SQL.
- [Cloud SQL features by database engine](#) [Help document](#)
Learn about the Cloud SQL features for each database engine.
- [Cloud SQL for MySQL features](#) [Help document](#)
Learn about Cloud SQL for MySQL features.
- [Cloud SQL for PostgreSQL features](#) [Help document](#)
Learn about Cloud SQL for PostgreSQL features.
- [Cloud SQL for SQL Server features](#) [Help document](#)
Learn about Cloud SQL for SQL Server features.

[All Cloud SQL documentation](#)

Experiment-07

Cloud Pub/Sub: Experiment how Google Cloud Pub/Sub facilitate real-time messaging and communication between distributed applications. OR Caching Application Data with ElastiCache, Caching with Amazon CloudFront, Caching Strategies.

Cloud Pub/Sub Overview

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**.
- **Subscribers** receive messages from the topics 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.
- **Subscription:** Defines how messages are delivered to subscribers.
- **Subscriber:** Reads messages from a subscription.
- **Delivery Mechanism:** Either **Pull** (manual retrieval) or **Push** (automatic HTTP delivery).

Step 1: Enable Cloud Pub/Sub API

1. Open **Google Cloud Console**: <https://console.cloud.google.com>
 2. Navigate to **Navigation Menu (☰) → APIs & Services → Library**.
 3. Search for **Cloud Pub/Sub API**.
 4. Click **Enable** to activate the Cloud Pub/Sub API for your project.
-

Step 2: Create a Pub/Sub Topic

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

✓ Your **topic** is now ready for use.

Step 3: Create a Subscription

1. Click on the **Topic** you just created (e.g., my-topic).
2. Click **Create Subscription**.
3. Enter a **Subscription ID** (e.g., my-subscription).
4. Choose a **Delivery Type**:
- 5.

- **Pull:** Messages are manually fetched by the subscriber.
 - **Push:** Messages are automatically sent to an HTTP endpoint.
6. Click **Create**.

✓ Your **subscription** is now linked to the topic and ready to receive messages.

Step 4: Publish a Message Using gcloud CLI

1. Open **Cloud Shell** or use your local terminal (ensure that **gcloud** is configured).
2. Run the following command to publish a message to your topic:

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

✓ **Message successfully published** to the topic.

Step 5: Pull Messages from Subscription Using gcloud CLI

1. Run the following command to pull and acknowledge messages from your subscription:

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

✓ You will see the message: Hello, Pub/Sub! pulled from the subscription.

Step 6: Publish and Subscribe Using Python (Optional)

Install Google Cloud Pub/Sub SDK

Before writing the code, ensure you have the required Python package:

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

Publisher Code (Python)

```
from google.cloud import pubsub_v1
```

```
# Define project ID and topic ID
project_id = "your-project-id"
topic_id = "my-topic"
```

```
# Initialize Publisher client
publisher = pubsub_v1.PublisherClient()
```

```
# Define the topic path
topic_path = publisher.topic_path(project_id, topic_id)
```

```
# Define the message to publish
message = "Hello, Pub/Sub from Python!"
```

```
# Publish the message
future = publisher.publish(topic_path, message.encode("utf-8"))
```

```
# Output the message ID
print(f"Published message ID: {future.result()}")
```

This will publish the message "Hello, Pub/Sub from Python!" to the topic.

Subscriber Code (Python)

```
from google.cloud import pubsub_v1

# Define project ID and subscription ID
project_id = "your-project-id"
subscription_id = "my-subscription"

# Initialize Subscriber client
subscriber = pubsub_v1.SubscriberClient()

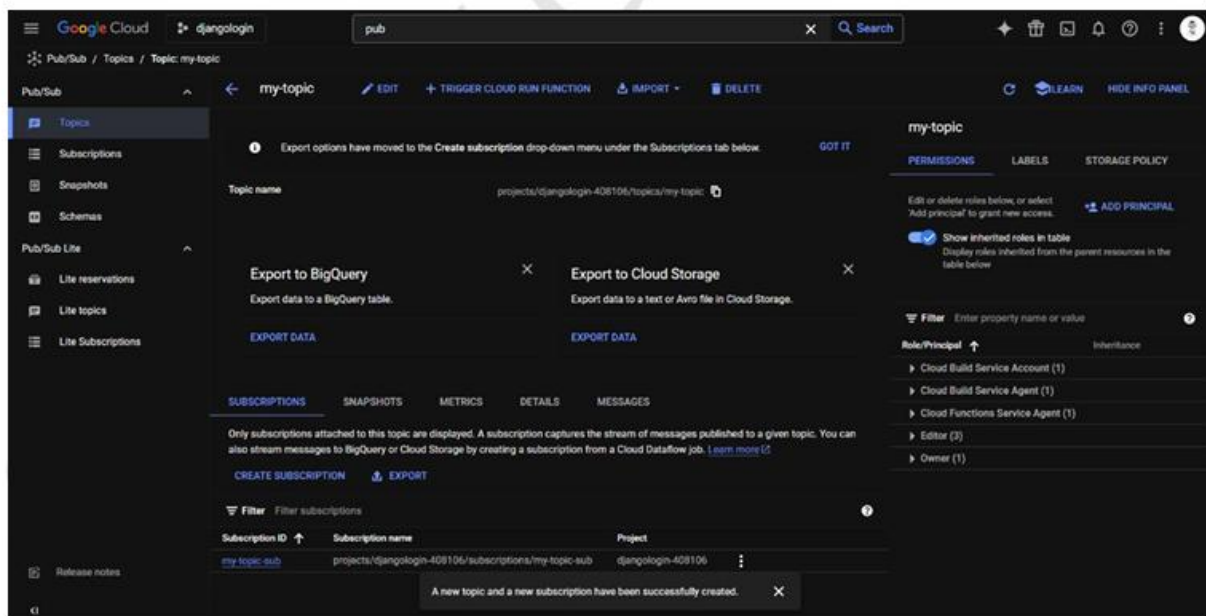
# Define the subscription path
subscription_path = subscriber.subscription_path(project_id, subscription_id)

# Define a callback function to handle received messages
def callback(message):
    print(f"Received: {message.data.decode('utf-8')}")
    message.ack() # Acknowledge the message

# Subscribe to the topic
subscriber.subscribe(subscription_path, callback=callback)

# Keep the subscriber alive
print("Listening for messages...")
import time
while True:
    time.sleep(10) # Keep the script running to listen for messages
```

Output



Google Cloud Pub/Sub console interface for a subscription named "my-subscription".

Subscription Details:

- Subscription name: projects/djangologin-408106/subscriptions/my-subscription
- Subscription state: active
- Topic name: projects/djangologin-408106/topics/my-topic

Actions: EDIT, CREATE SNAPSHOT, REPLAY MESSAGES, PURGE MESSAGES, DETACH, DELETE, LEARN, HIDE INFO PANEL.

Metrics: Overview, Health, Pull. A graph shows "Oldest unacked message a..." with a warning: "No data is available for the selected time frame."

Permissions: Cloud Build Service Agent (1), Cloud Functions Service Agent (1), Editor (2), Owner (1).

Release notes: Subscription added successfully.

Google Cloud Cloud Shell terminal interface showing the successful creation of a Pub/Sub topic and subscription.

```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to djangologin-408106.
Use 'gcloud config set project [PROJECT_ID]' to change to a different project.
aryahananthu@cloudshell:~ (djangologin-408106) $ gcloud pubsub topics publish my-topic --message "Hello, Pub/Sub!"
messageId:
- '13571364796449056'
aryahananthu@cloudshell:~ (djangologin-408106) $
```

Experiment-08

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

Google Cloud, Virtual Private Cloud (VPC) allows you to define and control networking environments for your resources. You can have multiple VPC networks, each isolated from one another or interconnected for specific use cases. Managing multiple VPCs helps you scale, secure, and organize resources efficiently.

Benefits of Using Multiple VPC Networks

Resource Isolation & Security

- **Network Isolation** – You can isolate resources within different VPC networks to improve security and control traffic between services.
- **Private Connectivity** – Each VPC can have private IPs that do not communicate with other VPCs unless explicitly allowed, keeping sensitive data isolated.
- **Granular Firewall Rules** – Define specific firewall rules for each VPC, limiting access to resources within a VPC or between multiple VPCs.

Organizational Structure & Management

- **Separation by Department or Service** – Different teams or services (e.g., dev, test, production) can operate within their own VPCs, helping to organize and manage resources based on logical groupings.
- **Custom Subnetting** – Each VPC can have its own subnet structure tailored to the needs of specific projects or services.

Traffic Control

- **VPC Peering** – You can allow traffic between two or more VPCs by creating VPC peering connections. This gives you flexibility in managing traffic flow while maintaining network isolation.
- **Shared VPC** – A Shared VPC allows multiple projects to connect to a common VPC network, enabling central management of network resources.
- **Private Google Access** – For certain services, you can configure access to Google Cloud services without using public IPs, enhancing security.

Scaling Flexibility

- **Scalability for Different Environments** – As projects or environments grow, you can add more VPCs, allowing the architecture to scale without impacting other parts of the system.
- **Cross-Region Connectivity** – Create VPCs in different regions for disaster recovery and global distribution of your resources. Google Cloud provides the ability to set up global VPCs and establish secure connections across regions.

Enhanced Network Performance

- **Low Latency Communication** – By grouping resources that need high throughput and low latency within a specific VPC, you can optimize performance for specific workloads.
- **Dedicated Resources** – Certain VPCs can be dedicated to specific high performance workloads (e.g., compute-intensive tasks), while others may be used for general workloads, ensuring efficient resource use.

Use Cases for Multiple VPCs

Multi-Tier Applications

You can deploy multi-tier architectures where each tier (e.g., web, app, database) resides in separate VPC networks, enabling better isolation and security between tiers.

Cross-Region Architecture

You can deploy resources in multiple regions for disaster recovery or to meet local compliance requirements while maintaining network isolation between regions. For instance, a production VPC in one region and a disaster recovery VPC in another.

Hybrid Cloud or Multi-Cloud

If you're integrating on-premises infrastructure or other cloud platforms with Google Cloud, using separate VPCs for each environment allows secure and controlled network communication across different systems.

Managed Service Integration You might have managed services (like Cloud SQL or BigQuery) in one VPC while using compute instances or other resources in another, optimizing resource placement.

Set Up Multiple VPC Networks in Google Cloud

Step 1: Create a VPC Network

1. Go to Google Cloud Console → Navigation Menu (☰) → VPC Network → Create 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.

Step 3: 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 between 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.

This screenshot shows the Google Cloud VPC networks overview page. The left sidebar contains a navigation menu with options like IP addresses, Internal ranges, Bring your own IP, Firewall, Routes, VPC network peering, Shared VPC, Serverless VPC access, Packet mirroring, and VPC flow logs. The main content area features a 'Get started with real-time analytics' section with a list of benefits and a 'TRY NOW' button. Below this is a table of VPC networks. A notification banner at the top right indicates that SMTP port 25 is disallowed in the project.

Get started with real-time analytics

Use Network Intelligence Center for comprehensive monitoring and troubleshooting. [Learn more](#)

- ✓ Visualise your network resources
- ✓ Diagnose and prevent connectivity issues
- ✓ View packet loss and latency metrics
- ✓ Keep your firewall rules strict and efficient

[TRY NOW](#) [REMIND ME LATER](#)

SMTP port 25 disallowed in this project. [Learn more](#)

VPC networks

Filter: Enter property name or value

Name	Subnets	MTU	Mode	IPv6 ULA range	Gateways	Firewall rules	Global dynamic routing
default	41	1460	Auto			4	Off

This screenshot shows the Google Cloud VPC network details page for the 'default' network. The left sidebar is the same as the overview page. The main content area has a breadcrumb trail: VPC network / VPC networks / Network: default. Below this is a tabbed interface with tabs for OVERVIEW, SUBNETS, STATIC INTERNAL IP ADDRESSES, FIREWALLS, FIREWALL ENDPOINTS, ROUTES, VPC NETWORK PEERING, and PRIVATE SE. The 'OVERVIEW' tab is active, showing an 'EDIT' button and a list of configuration details.

VPC network details [DELETE VPC NETWORK](#)

default

[EDIT](#)

Description
Default network for the project

Maximum transmission unit
1460

VPC network ULA internal IPv6 range
Disabled

Subnet creation mode
Auto subnets

Dynamic routing mode
Regional

Best path selection mode
Legacy

Tags
-

[EQUIVALENT REST](#)

Experiment-09

Cloud Monitoring: Discover how Cloud Monitoring help in tracking and analyzing the performance and health of cloud resources? OR Orchestrating Serverless Functions with AWS Step Functions

Steps for Setting Up Google Cloud Monitoring

1. Enable Cloud Monitoring API:

- Navigate to Google Cloud Console → Menu (☰) → APIs & Services → Library.
- Search for "Cloud Monitoring API" and enable it.

2. Set Up Monitoring:

- Go to Menu (☰) → Monitoring → Dashboards.
- Create a custom dashboard and select the services and metrics to track.
- Adjust visualizations (e.g., line charts, heat maps).

3. Set Up Alerts:

- Go to Menu (☰) → Monitoring → Alerting.
- Create alert policies based on conditions (e.g., CPU usage > 80%).
- Configure notification channels (e.g., email, SMS, Slack) and escalation policies.

4. Review Logs:

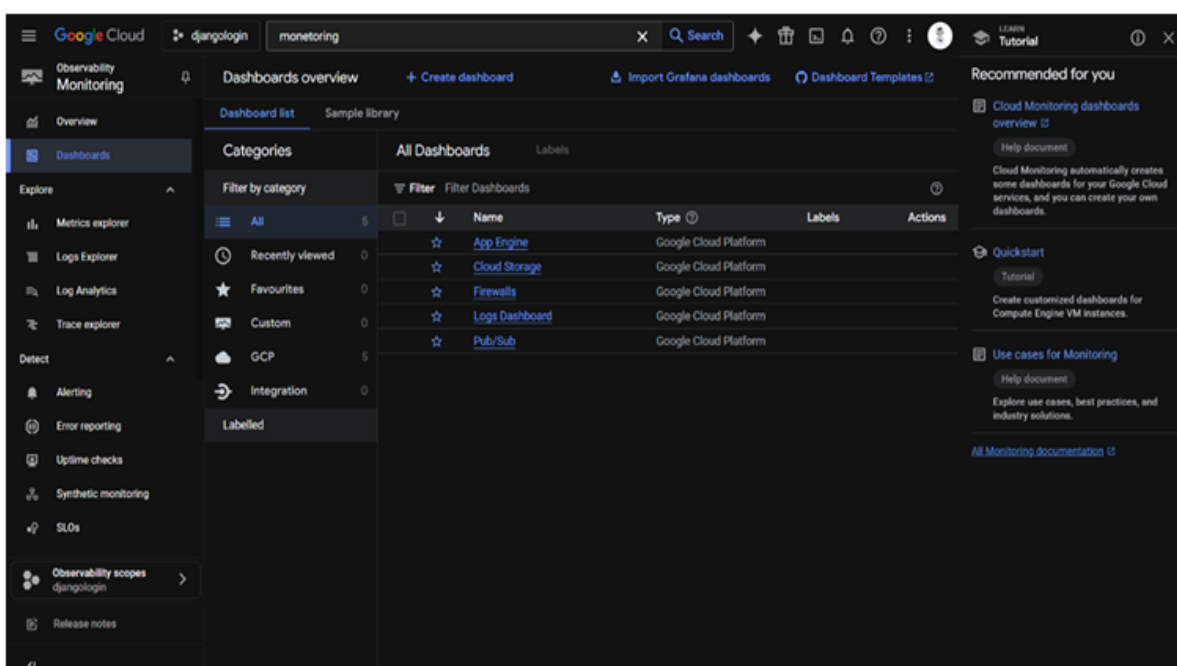
- Navigate to Menu (☰) → Logging.
- Define log filters and use Log Explorer to search for specific logs like errors or performance warnings.

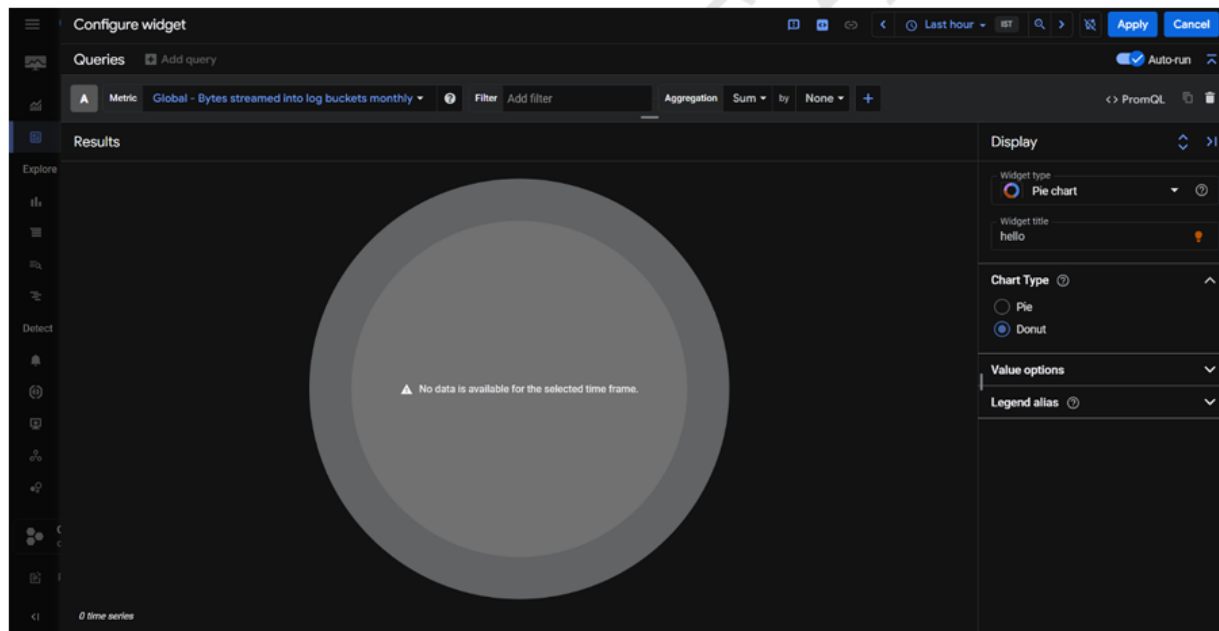
5. Set Up Distributed Tracing (Optional):

- Access Cloud Trace via the Menu (☰).
- Enable trace collection using Cloud Trace SDK for your services.
- Analyze trace data to identify system bottlenecks.

This process ensures effective monitoring, alerting, and log analysis for your Google Cloud resources.

Output





Experiment-10

Kubernetes Engine: Qwik Start: Deploy a containerized application to a Kubernetes Engine cluster. OR Automating Application Deployment Using a CI/CD Pipeline

Steps to Deploy a Containerized Application to GKE:

1. Set Up Google Cloud SDK and Kubernetes Tools:

- Install the Google Cloud SDK and configure it.
- Use the kubectl tool (it comes preinstalled with the Cloud SDK).

2. Create a Google Cloud Project:

- Create a new project in Google Cloud Console or select an existing project.
- Set the project in the Cloud SDK using:

gcloud config set project PROJECT_ID

3. Enable Required APIs:

- Enable the Kubernetes Engine API:
- gcloud services enable container.googleapis.com
- Enable the Compute Engine API (if not already enabled):

gcloud services enable compute.googleapis.com

4. Ceate a Kubernetes Cluster:

- Create the cluster:
- gcloud container clusters create my-cluster --zone us-central1-a
- Configure kubectl to use the cluster:

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

5. Create a Containerized Application:

- Write a Dockerfile for your application.
- Build the Docker image:

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

- Push the image to Google Container Registry:

docker push gcr.io/PROJECT_ID/my-app:v1

6. Create a Kubernetes Deployment:

- Define a deployment.yaml for your application (include details like replicas, container image, ports, etc.).
- Apply the deployment to the cluster:

kubectl apply -f deployment.yaml

7. Expose the Application via a Service:

- Define a service.yaml to expose the application (e.g., as a LoadBalancer for external access).
- Apply the service:

kubectl apply -f service.yaml

- Get the external IP address for access:

kubectl get svc

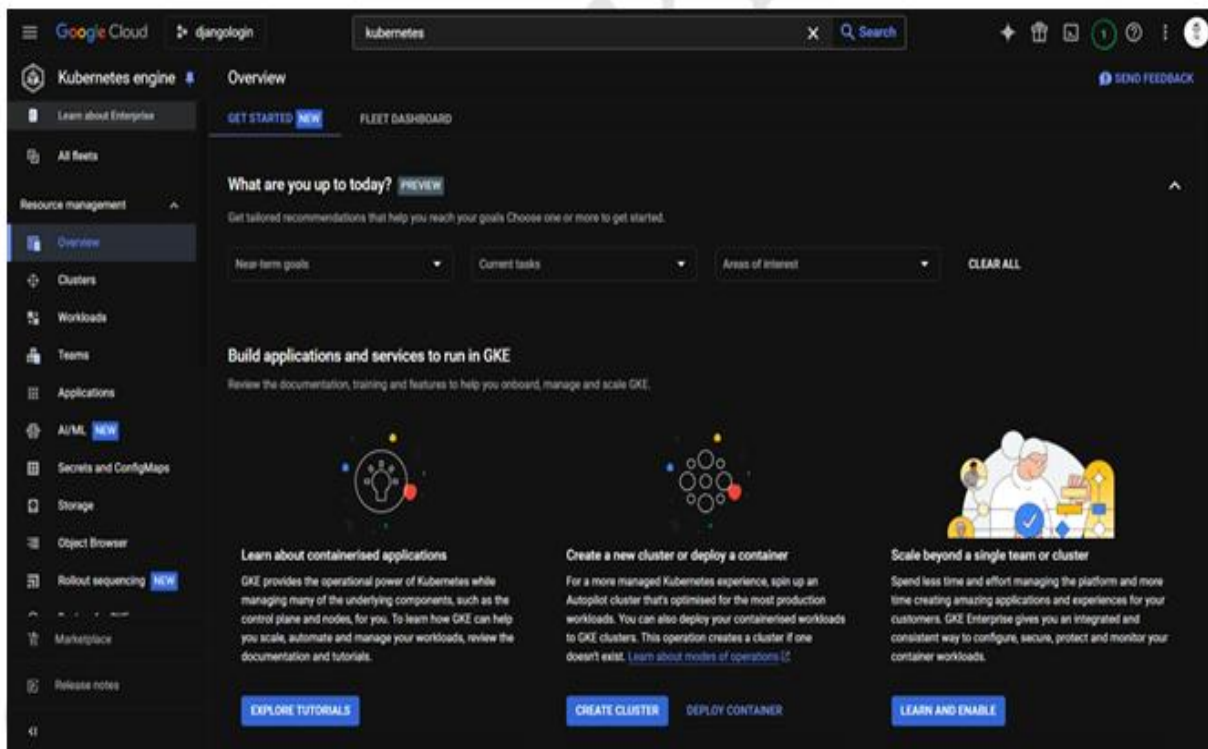
8. Verify the Deployment:

- Open a browser and navigate to the external IP to check the application.
- Use kubectl to monitor the status of your pods and services:

kubectl get pods

kubectl get svc

Output



Google Cloud

djangojin

kubernetes

Search

LEARN Tutorial

Create a deployment

1 Deployment configuration

Define how Kubernetes deploys, manages and scales container image

2 Container details

3 Expose (optional)

Define how your deployment is exposed

Deployment configuration

A deployment is a configuration which defines how Kubernetes deploys, manages and scales your container image. Kubernetes will ensure that your system matches this configuration. These replicas will be created by default.

Deployment name *

deployment-1

Namespace *

default

Labels

Key 1 *

app

Value 1

deployment-1

+ ADD KUBERNETES LABEL

Cluster

New cluster will be in Autopilot mode (see [here](#)). GKE will automatically provision, configure and manage the resources and hardware.

Your deployment will use compute instances managed in a logical grouping called a 'cluster', which will be configured in a way that's great for getting started with Kubernetes.

This cluster will be created automatically if cluster

DEPLOY

CANCEL

VIEW YAML

Deployments

Kubernetes Engine works with [containerized applications](#), applications packaged into hardware-independent, isolated user-space instances, for example by using [Docker](#).

In Kubernetes Engine and Kubernetes, these containers are collectively called workloads.

Continuous delivery

Looking for fully managed continuous delivery with multi-environment progressions, scaling and security built-in? [Try Google Cloud Deploy](#)

Recommended for you

Overview of deploying workloads

[Help document](#)

Create Kubernetes controller objects to deploy and manage containerized applications and other workloads on a cluster.

Stateless applications

[Help document](#)

Deploy a stateless Linux application using GKE.

Deploying a stateful application

[Help document](#)

Deploy Stateful applications in GKE, with stored data accessible to the server, clients, and other applications.

Scaling applications

[Help document](#)

Scale a deployed application in GKE.

Performing rolling updates

[Help document](#)

Perform rolling updates for applications in GKE.

Exposing applications using services