# Google Cloud Quickstart - Full Guide

## 1. Create and Manage a VM using Cloud Shell

**Step 1: Open Cloud Shell**

1. Go to https://console.cloud.google.com/

2. Click the Cloud Shell icon ( >_ ) in the top-right.

3. A terminal opens at the bottom.

**Step 2: Initialize gcloud CLI**

1. Run: gcloud init .

2. Follow prompts to authenticate and set project.

**Step 3: Verify gcloud Setup**

Run: gcloud config list.

**Step 4: List Projects**

Run: gcloud projects list.

**Step 5: Set Active Project** → your project ID

Run: gcloud config set project PROJECT_ID.

**Step 6: Check Authentication**

Run: gcloud auth list

**Step 7: Create a VM Instance**

Run: → VM name

gcloud compute instances create my-vm --zone=us-central1-a.

**Step 8: List Running VMs**

Run: gcloud compute instances list .

**Step 9: Delete a VM**

Run: gcloud compute instances delete my-vm .

**Step 10: Enable Compute Engine API**

Run: gcloud services enable compute.googleapis.com

## 2. Cloud Shell and gcloud CLI Setup

**Step 1: Open Cloud Shell**

1. Go to https://console.cloud.google.com
2. Click the Cloud Shell icon ( >_ ) at the top-right.
3. A terminal opens at the bottom.

**Step 2: Initialize gcloud CLI**

Run: gcloud init

- Sign in

- Choose project

- **Step 3:**

**Verify  Setup**

Run: gcloud config list

- Shows current project, account, region

**Step 4: List Projects**

Run: gcloud projects list

**Step 5: Set Active Project**

Run: gcloud config set project PROJECT_ID

**Step 6: Check Authentication**

Run: gcloud auth list

**Step 7: Create a VM (Test)**

```
gcloud compute instances create my-vm --zone=us-central1-a
```

**Step 8: List VMs**

Run: gcloud compute instances list

**Step 9: Delete VM**

Run: gcloud compute instances delete my-vm --zone=us-central1-a

**Step 10: Enable API**

Run: gcloud services enable compute.googleapis.com

## 3. Create a Cloud Function Triggered by Cloud Storage

**Step 1: Enable APIs**

Run:

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

**Step 2: Create a Storage Bucket**

Run: Your Cloud Storage Bucket Name

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

**Step 3: Write Cloud Function Code**

1. mkdir gcs-function && cd gcs-function
2. nano main.py

Paste this code:

```
import functions_framework


@functions_framework.cloud_event
def gcs_trigger(cloud_event):
    data = cloud_event.data
    bucket = data['bucket']
    file_name = data['name']
    print(f"File {file_name} uploaded to {bucket}")
```

**Step 4:**

**requirements.txt**

nano requirements.txt

Add:

```
functions-framework
```

**Step 5: Deploy Cloud**

**Function**

Run:

```
gcloud functions deploy gcs_trigger \
--gen2 \
--runtime=python311 \
--region=us-central1 \
--source=. \

--entry-point=gcs_trigger \

--trigger-event-filters="type=google.cloud.storage.object.v1.finalized" \

--trigger-event-filters="bucket=BUCKET_NAME" \

--allow-unauthenticated
```

**Step 6: Test**

1. Create a test file and upload it:

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

2. View logs:

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

## 4.Deploy a Web App on App Engine (Auto Scaling)

**Step 1:**

**Enable App Engine API**

Run:

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

**Step 2: Create App Engine App**

Run:

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

**Step 3: Create App Files**

- mkdir app-engine-demo && cd app-engine-demo
- nano main.py

Add: // paste in the boilerplate code for a flask app

```
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)
```

**Step 4: Create**

**requirements.txt**

nano requirements.txt

Add:

```
Flask
gunicorn
```

**Step 5: Create app.yaml** → yml

nano app.yaml and add

```
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
```

**Step 6: Deploy**

Run:

```
gcloud app deploy
```

**Step 7: View App**

Run:

```
gcloud app browse
```

**Step 8: Logs and Monitoring**

Logs:

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

Services:

```
gcloud app services list
```

# 5.Cloud Storage Quickstart

**Step 1:**

**Open Cloud Console and Ensure Project is Selected**

**Step 2: Enable Cloud Storage API**

Go to APIs & Services > Library > Search 'Cloud Storage API' > Enable

**Step 3: Create a Bucket**

Navigation Menu > Storage > Buckets > Create

Set name, location, storage class, access control

**Step 4: Upload File**

1. Open bucket > Upload Files > Choose File > Open

**Step 5: Download File**

Open bucket > Click file > Download

**Step 6: Make File Public (Optional)**

1. Click file > Permissions tab > Add principal: allUsers
2. Role: Storage Object Viewer

**Step 7: Delete File or Bucket**

Delete files first, then delete bucket

## 6.Cloud SQL (MySQL) Setup

**Step 1: Enable Cloud SQL Admin API**

**Step 2: Create SQL Instance**

Navigation Menu > SQL > Create Instance > Choose MySQL

Set ID, root password, region, zone, machine type, storage

Click Create

**Step 3: Connect to SQL**

Console: Open SQL > Click Instance > Get IP

CLI:

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

or

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

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

Edit Instance > Enable HA > Choose standby zone > Save

**Step 5: Create Read Replica (Optional)**

Open Instance > Create Read Replica

**Step 6: Backup & Restore**

Enable Auto Backup:

Edit > Enable automatic backups

Manual Backup:

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

Restore:

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

## 7.Cloud Pub/Sub API

**Step 1: Enable Cloud Pub/Sub API**

Go to APIs & Services > Library > Search 'Cloud Pub/Sub API' > Enable

**Step 2: Create Topic**

Navigation Menu > Pub/Sub > Topics > Create Topic

**Step 3: Create Subscription**

Click Topic > Create Subscription

Choose Pull or Push Push → publish, Pull → subscribe

**Step 4: Publish Message**

Run:

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

**Step 5: Pull Message**

Run:

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

**Step 6: Python Publisher**

```
pip install google-cloud-pubsub
from google.cloud import pubsub_v1

project_id = "your-project-id"

topic_id = "my-topic"

publisher = pubsub_v1.PublisherClient()

topic_path = publisher.topic_path(project_id, topic_id)

message = "Hello, Pub/Sub from Python!"

publisher.publish(topic_path, message.encode("utf-8"))
```

Step 6: Python Subscriber

```
from google.cloud import pubsub_v1

subscriber = pubsub_v1.SubscriberClient()

subscription_path = subscriber.subscription_path("your-project-id", "my-subscription")
def callback(message):

    print(f"Received: {message.data.decode('utf-8')}")

    message.ack()

subscriber.subscribe(subscription_path, callback=callback)

while True:

    time.sleep(10)
```