**Build a Docker Application on GCP. You can use Git Repository to build.**

**✅ STEP 1: Enable Required APIs (no billing needed)**

Go to the following links and enable them:

1. Enable Compute Engine API
2. Enable Container Registry API

**✅ STEP 2: Install Prerequisites (on your local system)**

Make sure you have:

* Docker installed
* gcloud CLI installed and authenticated

bash

CopyEdit

gcloud auth login

gcloud config set project YOUR\_PROJECT\_ID

**✅ STEP 3: Write Flask App Code**

Create a file named **app.py**:

python

CopyEdit

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def hello():

return "Hello from Google Compute Engine + Docker Free Tier!"

if \_\_name\_\_ == '\_\_main\_\_':

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

**✅ STEP 4: Create Dockerfile**

Create a file named **Dockerfile** (no file extension):

Dockerfile

CopyEdit

# Use the official lightweight Python image

FROM python:3.9-slim

# Set working directory

WORKDIR /app

# Copy all files to working directory

COPY . .

# Install dependencies

RUN pip install flask

# Run the app

CMD ["python", "app.py"]

**✅ STEP 5: Build Docker Image Locally**

bash

CopyEdit

docker build -t flask-app .

**✅ STEP 6: Tag the Image for Google Container Registry**

bash

CopyEdit

docker tag flask-app gcr.io/YOUR\_PROJECT\_ID/flask-app

Replace YOUR\_PROJECT\_ID with your actual GCP project ID.

Example:

bash

CopyEdit

docker tag flask-app gcr.io/primal-cascade-456415-q3/flask-app

**✅ STEP 7: Authenticate Docker to Push to GCR**

bash

CopyEdit

gcloud auth configure-docker

**✅ STEP 8: Push the Image to GCR**

bash

CopyEdit

docker push gcr.io/YOUR\_PROJECT\_ID/flask-app

✅ This works **without enabling billing** because you're using gcr.io (Container Registry), not Artifact Registry.

**✅ STEP 9: Create a VM on Google Compute Engine**

1. Go to Google Compute Engine > VM instances
2. Click **Create Instance**
3. Choose:
   * Region: us-central1 (for Free Tier)
   * Machine: e2-micro (Free Tier eligible)
4. Under **Container**, check:  
   ✅ "Deploy a container image to this VM instance"
5. In the container image URL field, enter:

bash

CopyEdit

gcr.io/YOUR\_PROJECT\_ID/flask-app

1. Under Firewall, ✅ check both:
   * Allow HTTP
   * Allow HTTPS
2. Click **Create**

**✅ STEP 10: Open Port 8080 in Firewall**

By default, GCP blocks non-standard ports. Open port 8080 like this:

1. Go to Firewall rules
2. Click **“CREATE FIREWALL RULE”**
3. Name: allow-8080
4. Targets: All instances
5. Source IP ranges: 0.0.0.0/0
6. Protocols and ports:  
   ✅ Check "Specified protocols and ports" → TCP: 8080
7. Click **Create**

**✅ STEP 11: Access Your App**

1. Go to VM Instances
2. Copy the **External IP**
3. Open your browser:

cpp

CopyEdit

http://EXTERNAL\_IP:8080

🎉 You should see:

sql

CopyEdit

Hello from Google Compute Engine + Docker Free Tier!

**Develop, Build and Deploy a container Application On Google Compute Engine.**

**🚀 STEP 1: Enable Required Google Cloud Services**

Enable the following APIs (no billing needed):

* Compute Engine API
* Container Registry API

**🧱 STEP 2: Create Your Flask App Locally**

1. In your project folder, create a file called app.py:

python

CopyEdit

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def hello():

return "Hello from Google Compute Engine + Docker (Free Tier)"

if \_\_name\_\_ == '\_\_main\_\_':

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

**🐳 STEP 3: Create Dockerfile**

In the same folder, create a file named Dockerfile (no extension):

dockerfile

CopyEdit

# Use official Python base image

FROM python:3.9-slim

# Set working directory

WORKDIR /app

# Copy project files

COPY . .

# Install Flask

RUN pip install flask

# Run the app

CMD ["python", "app.py"]

**🧪 STEP 4: Build and Tag the Docker Image**

bash

CopyEdit

docker build -t flask-app .

Now tag the image for Google Container Registry:

bash

CopyEdit

docker tag flask-app gcr.io/YOUR\_PROJECT\_ID/flask-app

🔁 Replace YOUR\_PROJECT\_ID with your actual project ID.

**🔐 STEP 5: Authenticate Docker to Push to GCR**

bash

CopyEdit

gcloud auth configure-docker

**📤 STEP 6: Push Docker Image to Google Container Registry**

bash

CopyEdit

docker push gcr.io/YOUR\_PROJECT\_ID/flask-app

✅ You’re now ready to deploy!

**☁️ STEP 7: Create a Google Compute Engine VM with Container**

1. Go to: Compute Engine → VM Instances
2. Click **“Create Instance”**
3. Set:
   * Name: flask-vm
   * Region: us-central1 (Free Tier eligible)
   * Machine: e2-micro (Free Tier eligible)
4. Scroll to the **Container** section:
   * ✅ Check **"Deploy a container image to this VM instance"**
   * Image:

bash

CopyEdit

gcr.io/YOUR\_PROJECT\_ID/flask-app

1. Scroll to **Firewall**
   * ✅ Allow HTTP traffic
   * ✅ Allow HTTPS traffic
2. Click **Create**

**🔓 STEP 8: Open Port 8080 in Firewall**

1. Go to VPC > Firewall rules
2. Click **“Create Firewall Rule”**
3. Fill out:
   * **Name**: allow-8080
   * **Targets**: All instances
   * **Source IP Ranges**: 0.0.0.0/0
   * **Protocols and ports**: TCP 8080
4. Click **Create**

**🌐 STEP 9: Access Your Flask App**

1. Go to VM Instances
2. Copy your **VM’s External IP**
3. Open your browser:

plaintext

CopyEdit

http://EXTERNAL\_IP:8080

**Cloud Models: Create or Sign Up for PaaS, Saas and IaaS services on platform of your choice and upload at least one of your resource or application on each platform.**

**✅ PAAS**

**🔐 Step 1: Open Cloud Shell**

1. Go to https://console.cloud.google.com
2. On the top-right corner, click the **Cloud Shell icon** (a terminal icon >\_)
3. Wait for it to initialize

**📁 Step 2: Create Your Flask App in Cloud Shell**

bash

CopyEdit

mkdir my-flask-app

cd my-flask-app

Now create main.py:

bash

CopyEdit

nano main.py

Paste this code in the nano editor:

python

CopyEdit

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def hello():

return "Hello from Google App Engine in Cloud Shell!"

Press Ctrl+O to save and Ctrl+X to exit.

**📝 Step 3: Create requirements.txt**

bash

CopyEdit

echo "flask" > requirements.txt

**⚙️ Step 4: Create app.yaml**

bash

CopyEdit

nano app.yaml

Paste this:

yaml

CopyEdit

runtime: python39

entrypoint: gunicorn -b :$PORT main:app

handlers:

- url: /.\*

script: auto

Save and exit with Ctrl+O, then Ctrl+X.

**🌀 Step 5: Deploy Your App**

If this is your first time:

bash

CopyEdit

gcloud app create

Then deploy:

bash

CopyEdit

gcloud app deploy

📌 Wait for a few minutes for deployment...

**🌐 Step 6: Visit Your App**

bash

CopyEdit

gcloud app browse

Or manually go to:

arduino

CopyEdit

<https://your-project-id.uc.r.appspot.com/>

**1. IaaS – Google Compute Engine (GCE)**

**Use Case**: Launch a virtual machine and deploy an app or script.

**✅ Steps:**

1. Go to GCP Console > VM Instances
2. Click **"Create Instance"**
3. Set:
   * Name: iaas-vm
   * Machine type: e2-micro (Free Tier)
   * Region: us-central1
4. Under Boot Disk → Choose **Debian/Ubuntu**
5. Click **Create**

**📤 Upload Resource:**

* SSH into VM → gcloud compute ssh iaas-vm
* Upload a sample file:

bash

CopyEdit

echo "Hello from IaaS" > hello.txt

✅ Done! You’ve uploaded a file to IaaS.

**🚀 2. PaaS – Google App Engine**

**Use Case**: Deploy your Flask or Node app without managing infrastructure.

**✅ Steps:**

1. Go to GCP Console > App Engine
2. Click **Create Application**
   * Choose region: us-central
3. Install App Engine SDK:

bash

CopyEdit

gcloud app create

**👨‍💻 Sample App:**

Create files locally:

app.yaml:

yaml

CopyEdit

runtime: python39

entrypoint: gunicorn -b :$PORT main:app

main.py:

python

CopyEdit

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def hello():

return "Hello from PaaS - App Engine!"

Install Flask & deploy:

bash

CopyEdit

pip install flask gunicorn

gcloud app deploy

Open app:

bash

CopyEdit

gcloud app browse

✅ You’ve deployed an app on PaaS.

**🚀 3. SaaS – Google BigQuery**

**Use Case**: Upload data and run queries using GCP's managed SQL engine.

**✅ Steps:**

1. Go to BigQuery Console
2. Create a dataset (e.g., my\_dataset)
3. Click **Create Table**
   * Source: Upload a CSV (e.g., students.csv)
   * Table Name: students
   * Schema: Auto-detect or define manually
4. Click **Create Table**

**🧪 Run Query:**

sql

CopyEdit

SELECT \* FROM `your\_project\_id.my\_dataset.students`;