**Primary Content Source**: <https://www.youtube.com/watch?v=7-s5ugThckY>

1. Virtual environment creation and activation on Visual Studio Application:

* Folder (as an example “Flask\_App\_GCP\_Deployment”) creation in local desired location (as an example “Downloads”)
* On Visual Studio: **File** – **Open Folder** – (open the created folder)
* Creation of virtual environment with name “venv”: **python –m venv venv**
* Activation of virtual environment: **venv/Scripts/Activate**

1. Installation of required libraries on virtual environment:

* **pip install Flask**
* **pip install python-dotenv**
* **pip install Flask-CORS**
* **pip install gunicorn**

1. Creation of environment file (**.env**):

FLASK\_DEBUG=1

PORT=8080

1. Creation of **app.py** file

# Importing necessary libraries

from flask import Flask  # Flask is the web framework for building the web application

from dotenv import load\_dotenv  # load\_dotenv is used to load environment variables from a .env file

import os  # os provides a way of interacting with the operating system, including reading environment variables

from flask\_cors import CORS  # CORS is used to handle Cross-Origin Resource Sharing (CORS) for the Flask app

# Load environment variables from the .env file

load\_dotenv()  # This loads the environment variables defined in a .env file into the environment

# Initialize the Flask application

app = Flask(\_\_name\_\_)  # Create a new Flask application instance

# Enable Cross-Origin Resource Sharing (CORS) for the app

CORS(app)  # This allows the app to accept requests from different domains, which is important for front-end and back-end separation

# Get the environment variables from the loaded .env file

# The FLASK\_DEBUG environment variable will be loaded if it's present in the .env file

app.config['DEBUG'] = os.environ.get('FLASK\_DEBUG')  # Set the 'DEBUG' configuration option using the environment variable

# Define a route for the root URL ("/")

@app.route('/')

def hello\_world():

    # This function will be executed when a user accesses the root URL of the application

    return 'Hello, World!'  # The server will respond with the string 'Hello, World!'

# Run the application if the script is executed directly

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

    app.run()  # Start the Flask development server. By default, it runs on http://127.0.0.1:5000

1. Creation of **app.yaml** file:

runtime: python312

entrypoint: gunicorn -b :$PORT app:app

1. Creation of **requirements.txt** file:

Using command: **pip freeze > requirements.txt**

Below is the post-creation of **requirements.txt**:

blinker==1.9.0

click==8.1.8

colorama==0.4.6

Flask==3.1.0

flask-cors==5.0.1

gunicorn==23.0.0

itsdangerous==2.2.0

Jinja2==3.1.6

MarkupSafe==3.0.2

packaging==24.2

python-dotenv==1.1.0

Werkzeug==3.1.3

1. Creation of **Dockerfile**:

FROM python:3.12.9-bookworm

ENV PYTHONUNBUFFERED True

ENV APP\_HOME /back-end

WORKDIR $APP\_HOME

COPY . ./

RUN pip install --no-cache-dir --upgrade pip

RUN pip install --no-cache-dir -r requirements.txt

CMD exec gunicorn --bind :$PORT --workers 1 --threads 8 --timeout 0 app:app

1. Creation of **.dockerignore**:

Dockerfile

README.md

\*.pyc

\*.pyo

\*.pyd

\_\_pycache\_\_

.pytest\_cache

1. Entire code run on virtual environment: **python app.py** (ctrl + left click on the link <http://127.0.0.1:5000>)
2. Installation of Google Cloud CLI (Google Cloud SDK):

* Install from: <https://cloud.google.com/sdk/docs/install> (search with "[Google Cloud CLI installer](https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe)”)

1. Update system environment variables with Google Cloud SDK installation path  
   - Windows search (**Edit the system environment variables**) -> **Environment Variables** -> Double click on **Path** -> Click on **New** -> Paste the installation path as **C:\Program Files (x86)\Google\Cloud SDK\google-cloud-sdk\bin** (as per specific installation path)
2. Google Cloud SDK path initiation on virtual environment  
   - From left panel on visual studio: **venv** -> **Scripts** -> **activate**  
   At the very first update path with the below command:

$gcloudPath = "C:\Program Files (x86)\Google\Cloud SDK\google-cloud-sdk\bin"

$env:PATH = "$env:PATH;$gcloudPath"

1. Test Google Cloud access on virtual environment:

**gcloud –version**

**gcloud help**

1. Google Cloud initialization on virtual environment: **gcloud init**
2. Google Cloud deployment from virtual environment: **gcloud run deploy --source .**
3. In the process of deployment, on the fly asks:  
   - Service name (as an example “test1”)  
   - Region specification (as an example with option 2 -> asia-east1)  
   - Allowing unauthenticated invocations or not (as an example “y”)
4. Post successful deployment, it will generate Service URL like below for ctrl + left click   
   [**https://test1-389793337085.asia-east1.run.app**](https://test1-389793337085.asia-east1.run.app)
5. From **Google Cloud Console -> Cloud Run** to see the App on cloud