Python Flask API to Google Cloud

https://cobalt-bond-367303.ue.r.appspot.com/

These steps are to deploy a simple Python web app written with the Flask web framework.

1. **Setup and Requirements**

    - Sign into your Google Cloud Console and create a new project or reuse an existing one. If you do not already have a Gmail or Google Workspace account, you must create one.
    - The **Project name** is the display name for this project's participants. It is a character string not used by Google APIs. You can update it at any time.
    - The **Project ID** must be unique across all Google Cloud projects and is immutable (it cannot be changed after it has been set). The Cloud Console auto-generates a unique string; you will need to reference the Project ID (it is typically identified as PROJECT_ID).
    - Next, you must enable billing in the Cloud Console to use Cloud resources/APIs. To shut down resources so you do not incur billing beyond this tutorial, you can delete the resources you created or delete the whole project. New users of Google Cloud are eligible for the $300 Free Trial program.

2. **Start Cloud Shell**

While Google Cloud can be operated remotely from your laptop, you will be using Cloud Shell, a command line environment running in the Cloud. Once connected to Cloud Shell, you should see that you are already authenticated and that the project is already set to your project ID.

- Run the following command in Cloud Shell to confirm that you are authenticated:

  cloudy auth list

- Run the following command in Cloud Shell to confirm that the gcloud command knows about your project:

  gcloud config list project

3. **Write the web app**

After Cloud Shell launches, you can use the command line to invoke the Cloud SDK gcloud command or other tools available on the virtual machine instance. In addition, you can use your $HOME directory in persistent disk storage to store files across projects and between Cloud Shell sessions.

- Let us get started by creating a new folder in your $HOME directory for the application:

mkdir ~/helloworld

cd ~/helloworld

- Create a file named main.py:

  touch main.py

- Edit the file with your preferred command line editor (nano, vim, or emacs) or click the Cloud Shell Editor button.
- To directly edit the file with Cloud Shell Editor, use this command:

  cloudshell edit main.py

  main.py

```
import flask

# If `entrypoint` is not defined in app.yaml, App Engine will look for an app
# called `app` in `main.py`.
app = flask.Flask(__name__)


@app.get("/")
def hello():
    """Return a friendly HTTP greeting."""
    return "Hello World!\n"


if __name__ == "__main__":
    # Used when running locally only. When deploying to Google App
    # Engine, a webserver process such as Gunicorn, will serve the app. This
    # can be configured by adding an `entrypoint` to app.yaml.
    app.run(host="localhost", port=8080, debug=True)
```

4. Define the dependencies

- To specify the dependencies of your web app, go back to the terminal and create a requirements.txt file in the root directory of your project with the exact version of Flask to use:

  touch requirements.txt

- To edit the file with Cloud Shell Editor, use this command:

  cloudshell edit requirements.txt

requirements.txt
# https://pypi.org/project/Flask
Flask==2.2.2

5. **Configure the deployment**

- To deploy your web app to App Engine, you need an app.yaml file. This configuration file defines your web app's settings for App Engine.
- From the terminal, create and edit the app.yaml file in the root directory of your project:

  touch app.yaml

- To edit the file with Cloud Shell Editor, use this command:

  cloudshell edit app.yaml

  app.yaml

  runtime: python39

6. **Deploy the web app**

- From the terminal, check the content of your directory:

  ls

- You must have the three following files:

  app.yaml main.py requirements.txt

- Deploy your web app with the following command:

  gcloud app deploy

- You may need to choose a deployment region by selecting the region where you want your App Engine application to be located.
- Confirm to launch the deployment.

Your web app is now ready to respond to HTTP requests on
https://PROJECT_ID.REGION_ID.r.appspot.com.

7. **Test the web app**

- First, retrieve your web app hostname with the gcloud app describe command:

APPENGINE_HOSTNAME=$(gcloud app describe --format "value(defaultHostname)")

- Test your web app with this simple HTTP GET request:

  curl https://$APPENGINE_HOSTNAME

- You should get the following answer:

  Hello World!

Summary

In the previous steps, you set up a simple Python web app, ran, and deployed the application on App Engine.

References:

https://codelabs.developers.google.com/codelabs/cloud-app-engine-python3#0