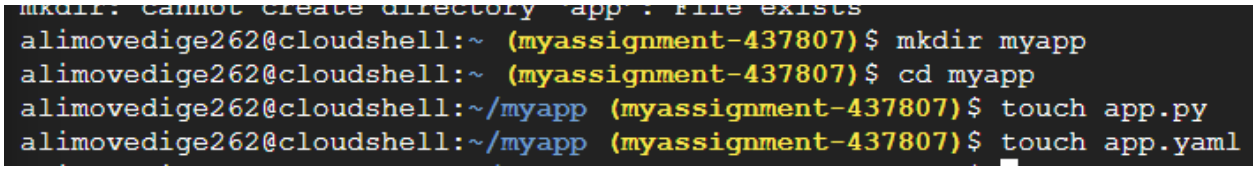
Assignment 2

EXERCISE 1: Google App Engine

- Ensure you have a Google Cloud account. – alimovedige262@gmail.com

1) create directory myapp and add files app.py and app.yaml
2) use command “mkdir” in order to create myapp
3) use command “touch” to create files app.py and app.yaml

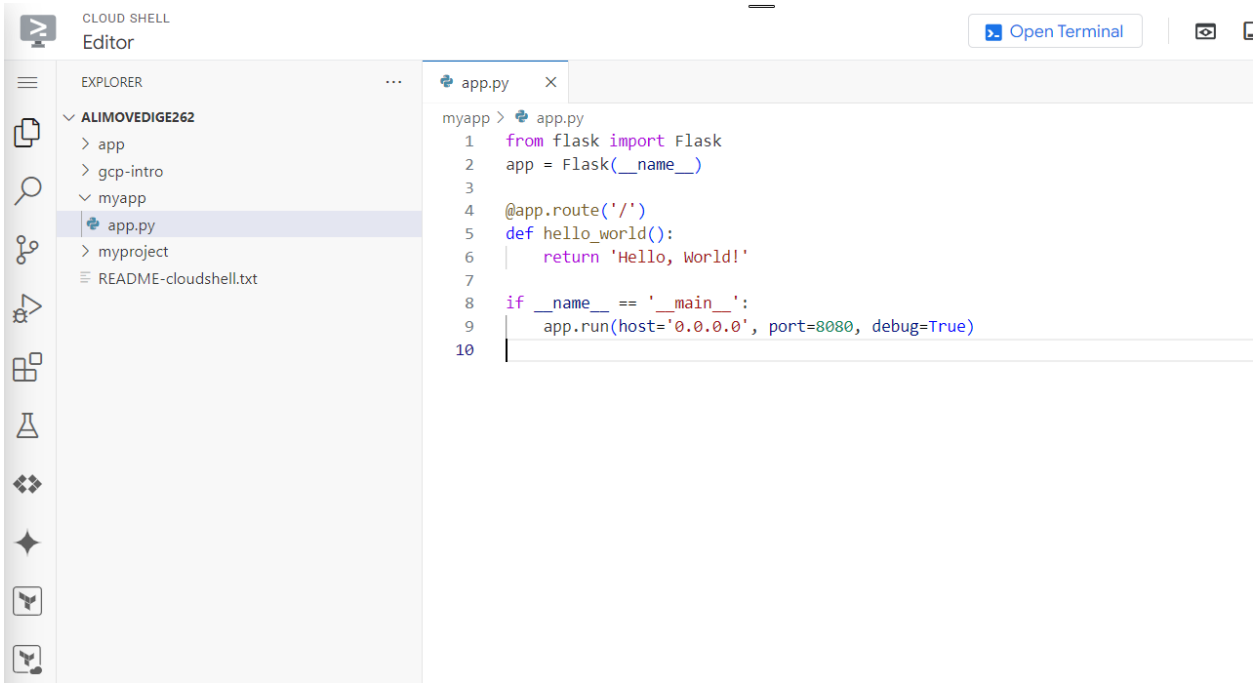


4) Click on button “Open editor” and add code of app.py and app.yaml
   Example app.py:

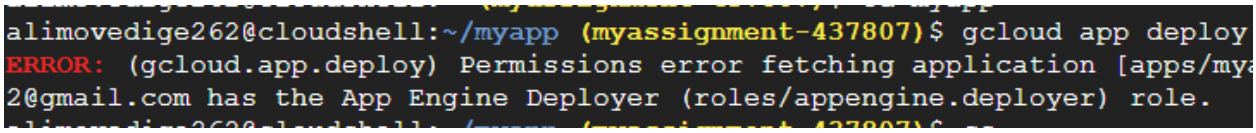
```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080, debug=True)
```

app.yaml

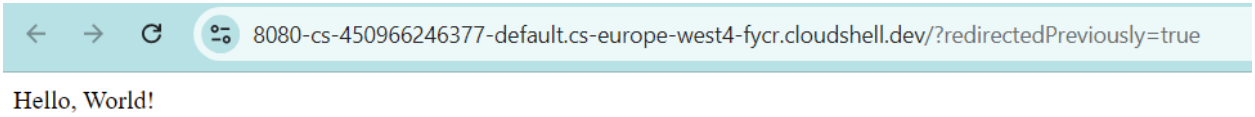
```
runtime: python39
handlers:
  - url: /.*
    script: auto
```



5) Deploy the application

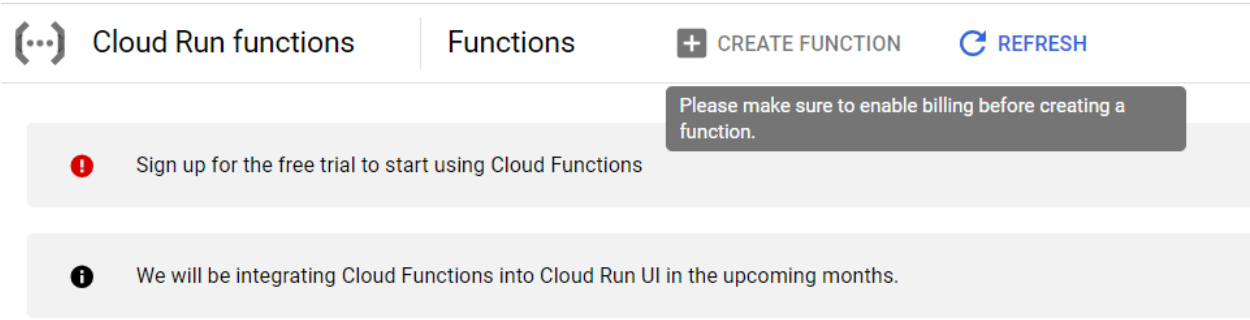


6) After the deployment is complete, the URL of the application will be received. It can be opened in a browser.
7) There is no permission on my account, because it needs billing account to have App Engine Deployer role
8) Expected A deployed web application based on the Google App Engine.



Exercise 2: Building with Google Cloud Functions

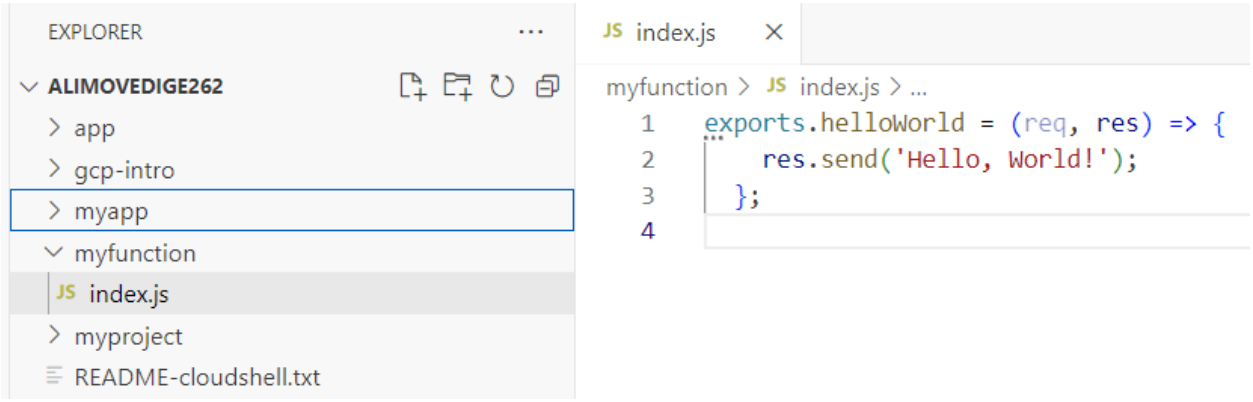
1) Create a Google Cloud feature that handles HTTP requests.
2) In order to create functions billing account is needed.
3) While creating configure the function with the following parameters:
   Name: hello World Function
   Trigger: HTTP
   Runtime environment: Node.js 18
   Entry point: hello World



Example index.js:

```
exports.helloWorld = (req, res) => {
  res.send('Hello, World!');
};
```

4) Creating index.js and enter the code which returns "Hello, World!" when accessed via HTTP.



5) Deploy the function
6) After deployment, use the provided URL to test the feature by opening it in a browser or using curl.
   Results:
   Deployed Google Cloud feature

Exercise 3: Containerizing Applications

1) Create a directory and navigate to it.
2) Creating a file app.py with content.
3) Creating a Dockerfile with content.
4) Building a Docker image.
5) Launching the Docker Container

```
alimovedige262@cloudshell:~ (myassignment-437807)$ mkdir mydocker
alimovedige262@cloudshell:~ (myassignment-437807)$ cd mydocker
alimovedige262@cloudshell:~/mydocker (myassignment-437807)$ touch app.py
alimovedige262@cloudshell:~/mydocker (myassignment-437807)$ docker build -t hello-world-app .
[+] Building 6.7s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 465B
 => [internal] load metadata for docker.io/library/python:3.9-slim
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/3] FROM docker.io/library/python:3.9-slim@sha256:49f94609e5a997dc16086a66ac9664591854031d48e375945a9dbf4d1d53abbc
 => => resolve docker.io/library/python:3.9-slim@sha256:49f94609e5a997dc16086a66ac9664591854031d48e375945a9dbf4d1d53abbc
 => => sha256:fdeeec85abbad3878f2008f9445f15a19a5a224d1b7e7715ac6b923072333e57 14.74MB / 14.74MB
 => => sha256:49f94609e5a997dc16086a66ac9664591854031d48e375945a9dbf4d1d53abbc 10.41kB / 10.41kB
 => => sha256:93ab151da4e5310ea79c4ecf306ece628262b86a4d7a49cc601664f19fe44e36 1.75kB / 1.75kB
 => => sha256:9d8cb7037cd8e90893e5f430ce4c048a872511e414580c7641675f2dad0a0351 5.20kB / 5.20kB
 => => sha256:302e3ee498053a7b5332ac79e8efebec16e900289fc1ecd1c754ce8fa047fcab 29.13MB / 29.13MB
 => => sha256:4c0965d3919510b506d8856ebc050a96e996c7dae96e4fb420882dbe7e037e67 3.51MB / 3.51MB
 => => sha256:62a08b8dd4f53ad5493dabf2af00ccde91abb3771fb2187040bcf2fe94a7ced7 248B / 248B
 => => extracting sha256:302e3ee498053a7b5332ac79e8efebec16e900289fc1ecd1c754ce8fa047fcab
 => => extracting sha256:4c0965d3919510b506d8856ebc050a96e996c7dae96e4fb420882dbe7e037e67
 => => extracting sha256:fdeeec85abbad3878f2008f9445f15a19a5a224d1b7e7715ac6b923072333e57
 => => extracting sha256:62a08b8dd4f53ad5493dabf2af00ccde91abb3771fb2187040bcf2fe94a7ced7
 => [internal] load build context
 => => transferring context: 542B
 => [2/3] WORKDIR /app
 => [3/3] COPY . /app
 => exporting to image
 => => exporting layers
 => => writing image sha256:3f8c7ac9d00287b93487b74fd24467da6de5c195e08a6c26646d0e455c54c9b9
 => => naming to docker.io/library/hello-world-app
alimovedige262@cloudshell:~/mydocker (myassignment-437807)$ docker run --rm hello-world-app
Hello from inside the container!
alimovedige262@cloudshell:~/mydocker (myassignment-437807)$
```

6) The screen displays a container that says the message "Hello from inside the container!".