Assignment 2, Cloud Application Development

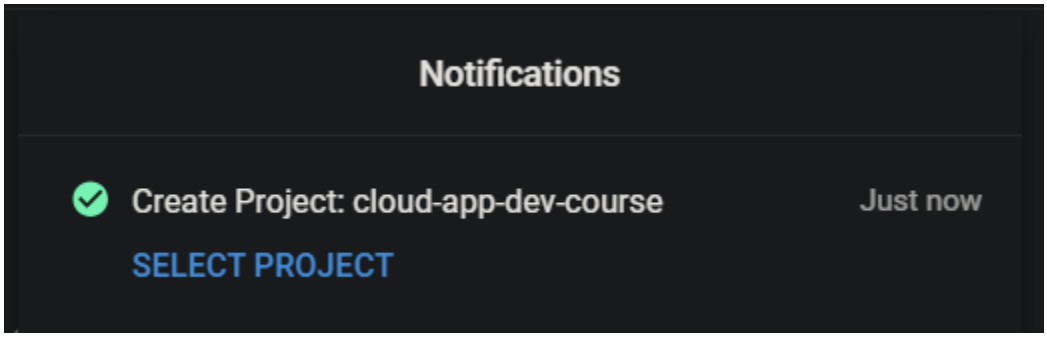
Put all deliverables into github repository in your profile. Defend by explaining deliverables and answering questions.
Deliverables: report in pdf
Google form:
https://docs.google.com/forms/d/e/1FAIpQLSe0GyNdOYlvM1tX_I_CtlPod5jBf-ACLGdHYZq1gVZbUeBzIg/viewform?usp=sf_link

Exercise 1: Google App Engine

Objective: Deploy a simple web application on Google App Engine.

Instructions:

1. **Setup**:
    - Ensure you have a Google Cloud account.
    - Install the Google Cloud SDK on your local machine.
2. **Create a Project**:
    - Create a new project in the Google Cloud Console.



Project itself was created using UI of https://console.cloud.google.com/

3. **Prepare the Application**:
    - Write a simple "Hello, World!" web application using Python (Flask).

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)
```

```
alishror171@cloudshell:~ (cloud-app-dev-course)$ mkdir task1
alishror171@cloudshell:~ (cloud-app-dev-course)$ cd task1
alishror171@cloudshell:~/task1 (cloud-app-dev-course)$ touch app.py
alishror171@cloudshell:~/task1 (cloud-app-dev-course)$ nano app.py
alishror171@cloudshell:~/task1 (cloud-app-dev-course)$ cat 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)
```

I used a console commands for creating and editing the files, namely mkdir, touch, nano and cat

4. **Create the App Engine Configuration**:

Create a `app.yaml` file with the following content:

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

```
alishror171@cloudshell:~/task1 (cloud-app-dev-course)$ touch app.yaml
alishror171@cloudshell:~/task1 (cloud-app-dev-course)$ nano app.yaml
alishror171@cloudshell:~/task1 (cloud-app-dev-course)$ cat app.yaml
runtime: python39
handlers:
  - url: /.*
    script: auto
```

Then the configuration file was created with the same way

5. **Deploy the Application**:

Use the following command to deploy the application to Google App Engine:

```
gcloud app deploy
```

```
alishror171@cloudshell:~/task1 (cloud-app-dev-course)$ nano requirements.txt
alishror171@cloudshell:~/task1 (cloud-app-dev-course)$ gcloud app deploy
Services to deploy:

descriptor:                  [/home/alishror171/task1/app.yaml]
source:                      [/home/alishror171/task1]
target project:              [cloud-app-dev-course]
target service:              [default]
target version:              [20241006t155451]
target url:                  [https://cloud-app-dev-course.an.r.appspot.com]
target service account:      [cloud-app-dev-course@appspot.gserviceaccount.com]


Do you want to continue (Y/n)?  y

Beginning deployment of service [default]...
Uploading 1 file to Google Cloud Storage
100%
100%
File upload done.
Updating service [default]...done.
Setting traffic split for service [default]...done.
Deployed service [default] to [https://cloud-app-dev-course.an.r.appspot.com]

You can stream logs from the command line by running:
  $ gcloud app logs tail -s default

To view your application in the web browser run:
  $ gcloud app browse
alishror171@cloudshell:~/task1 (cloud-app-dev-course)$ cat requirements.txt
Flask==3.0.3
```

Also, requirements.txt was created because the deployer couldn't recognise this module

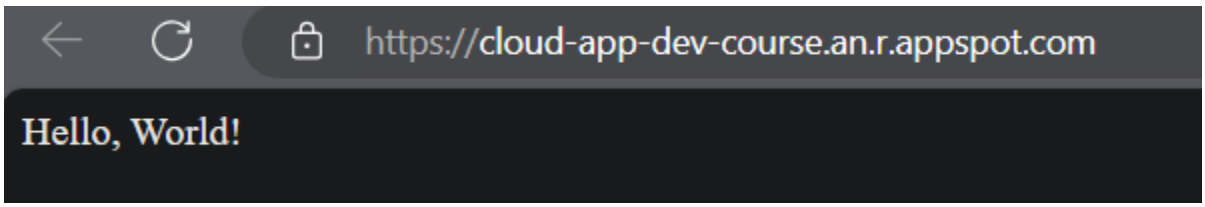
```
  File "<frozen importlib._bootstrap_external>", line 850, in exec_module    File "<frozen importlib._bootstrap>", line 228, in _call_with_frames_removed    File "/srv/main.py", line 1, in
<module>      from flask import Flask  ModuleNotFoundError: No module named 'flask'
2024-10-06 15:46:15 default[20241006t154342]  [2024-10-06 15:46:15 +0000] [18] [INFO] Worker exiting (pid: 18)
2024-10-06 15:46:15 default[20241006t154342]  [2024-10-06 15:46:15 +0000] [11] [ERROR] Worker (pid:18) exited with code 3
2024-10-06 15:46:15 default[20241006t154342]  [2024-10-06 15:46:15 +0000] [11] [ERROR] Shutting down: Master
2024-10-06 15:46:15 default[20241006t154342]  [2024-10-06 15:46:15 +0000] [11] [ERROR] Reason: Worker failed to boot.
2024-10-06 15:46:16 default[20241006t154342]  "GET /favicon.ico HTTP/1.1" 500
```

Then the application booted as expected
Also, I renamed app.py to main.py because flask expected main.py as an entry point

6. **Access the Application**:
    - Once deployed, access your application using the URL provided by Google App Engine.

Deliverables:

- A deployed web application on Google App Engine.
- A screenshot of the running application.



Exercise 2: Building with Google Cloud Functions

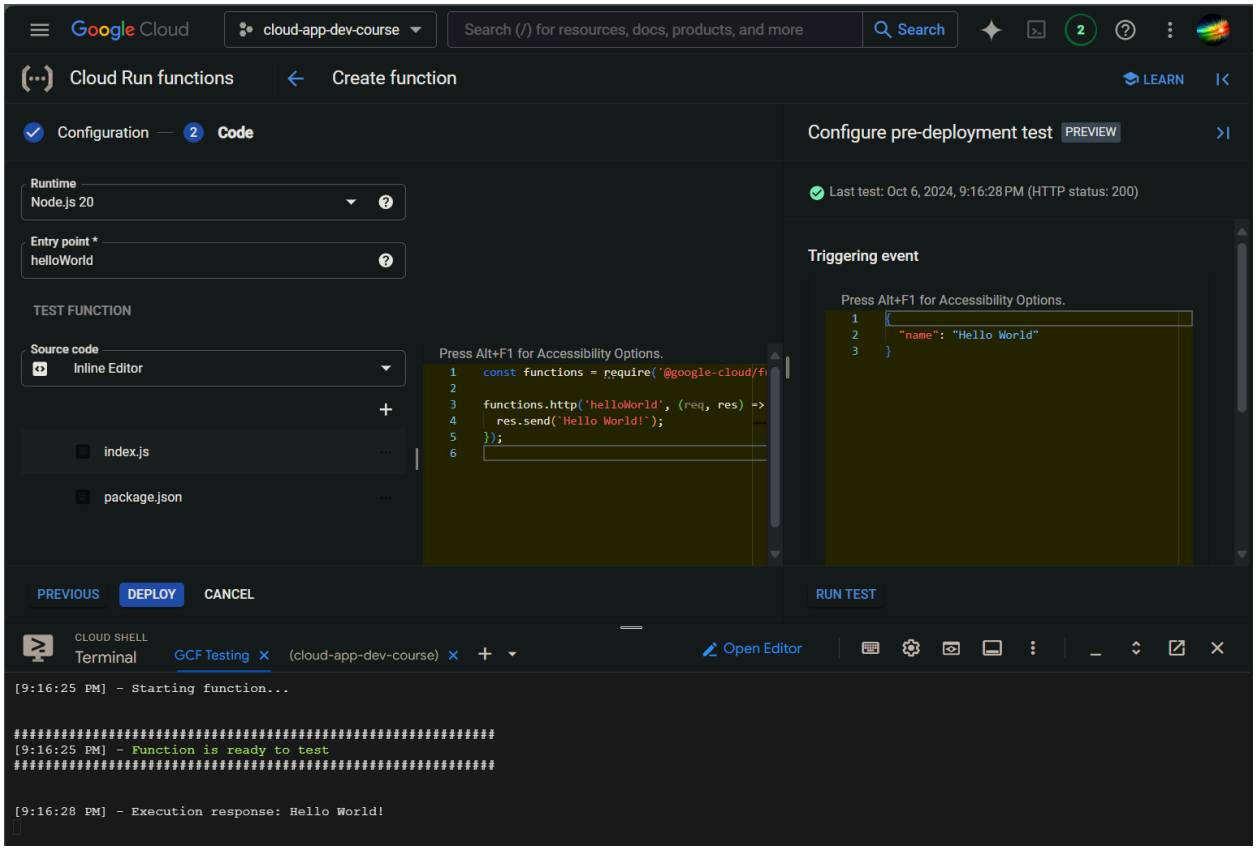
Objective: Create a Google Cloud Function that processes HTTP requests.

Instructions:

1. **Setup**:
   - Ensure you have a Google Cloud account.
   - Install the Google Cloud SDK on your local machine.
2. **Create a Function**:
   - Create a new Google Cloud Function using the following configuration:
     - **Name**: `helloWorldFunction`
     - **Trigger**: HTTP
     - **Runtime**: Node.js 18 (or another supported runtime)
     - **Entry Point**: `helloWorld`
3. **Write the Code**:
   - Write a simple function that returns "Hello, World!" when accessed via HTTP.

Example `index.js`:

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



Here you can see that I tested it as well, the result can be seen in the console and input is on the left side from the original code

4. **Deploy the Function**:

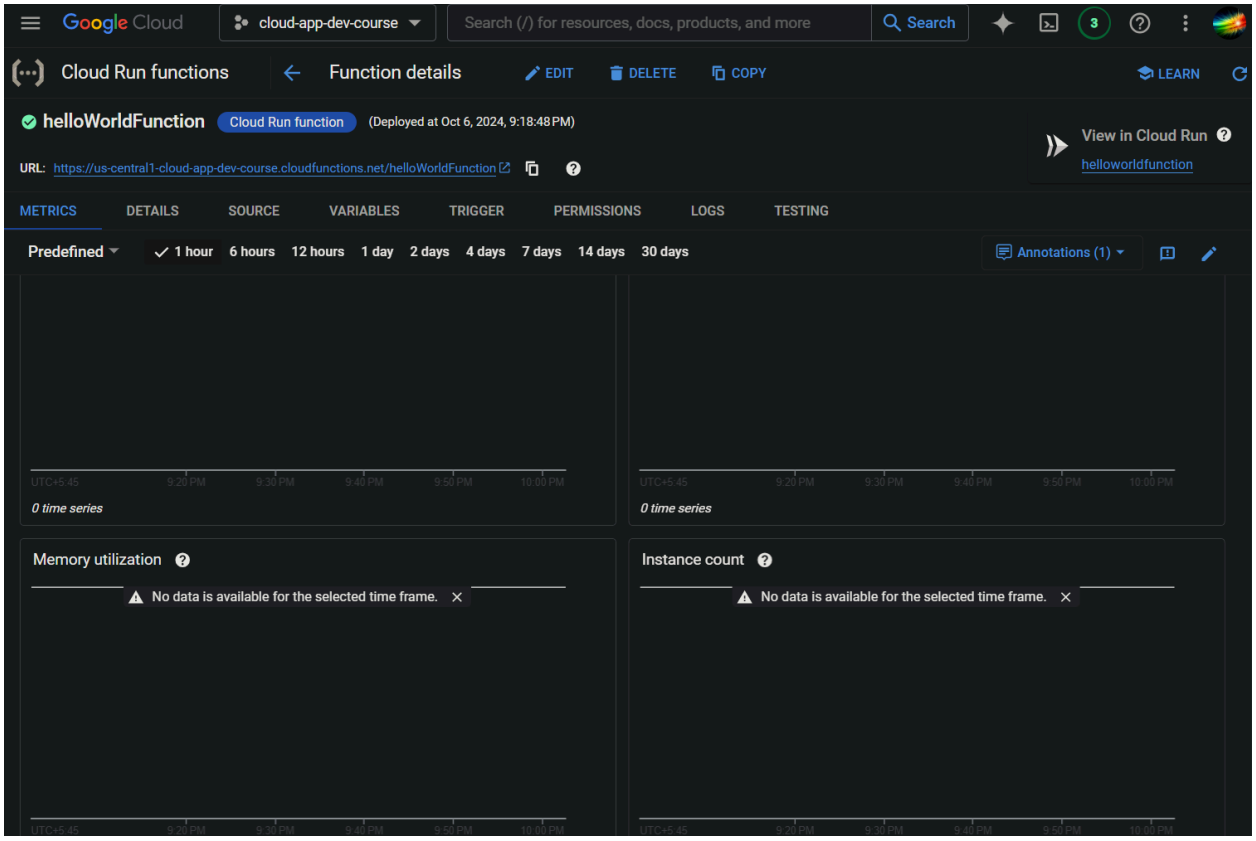
Use the following command to deploy the function:

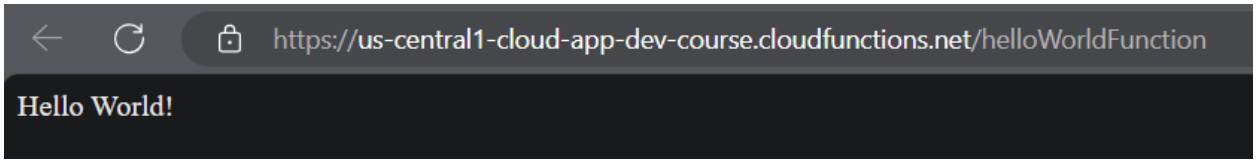
```
gcloud functions deploy helloWorldFunction --runtime nodejs18
--trigger-http
```

5. **Invoke the Function**:
    - Once deployed, use the provided URL to test the function by accessing it via a web browser or `curl`.

Deliverables:

- A deployed Google Cloud Function.
- A screenshot showing the response from the function.





Exercise 3: Containerizing Applications

Objective: Containerize a simple application using Docker.

Instructions:

1. **Setup**:
    - Ensure Docker is installed on your local machine.
2. **Create a Simple Application**:
    - Write a simple Python application.

Example `app.py`:

```
print("Hello from inside the container!")
```

```
alishror171@cloudshell:~ (cloud-app-dev-course)$ mkdir hello-from-docker
alishror171@cloudshell:~ (cloud-app-dev-course)$ cd hello-from-docker
alishror171@cloudshell:~/hello-from-docker (cloud-app-dev-course)$ touch app.py
alishror171@cloudshell:~/hello-from-docker (cloud-app-dev-course)$ nano app.py
alishror171@cloudshell:~/hello-from-docker (cloud-app-dev-course)$ cat app.py
print("Hello from inside the container!")
```

Here I used the console to create a dir and necessary files for the deployment of the docker

3. **Create a Dockerfile**:
    - Write a `Dockerfile` to containerize the application.

Example `Dockerfile`:

```
# Use an official Python runtime as a parent image
FROM python:3.9-slim

# Set the working directory in the container
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Run the application
CMD ["python", "app.py"]
```

4. **Build the Docker Image**:

Build the Docker image using the following command:

```
docker build -t hello-world-app .
```

5. **Run the Docker Container**:

Run the container using the following command:

```
docker run --rm hello-world-app
```

Deliverables:

- A Docker image that runs a simple application.
- A screenshot of the container output showing "Hello from inside the container!"

```
alishror171@cloudshell:~/hello-from-docker (cloud-app-dev-course)$ touch Dockerfile
alishror171@cloudshell:~/hello-from-docker (cloud-app-dev-course)$ nano Dockerfile
alishror171@cloudshell:~/hello-from-docker (cloud-app-dev-course)$ docker build -t hello-from-docker .
[+] Building 7.2s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 296B
 => [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
 => => sha256:9d8cb7037cd8e90893e5f430ce4c048a872511e414580c7641675f2dad0a0351 5.20kB / 5.20kB
 => => sha256:302e3ee498053a7b5332ac79e8efebec16e900289fc1ecd1c754ce8fa047fcab 29.13MB / 29.13MB
 => => sha256:4c0965d3919510b506d8856ebc050a96e996c7dae96e4fb420882dbe7e037e67 3.51MB / 3.51MB
 => => sha256:fdeeec85abbad3878f2008f9445f15a19a5a224d1b7e7715ac6b923072333e57 14.74MB / 14.74MB
 => => sha256:49f94609e5a997dc16086a66ac9664591854031d48e375945a9dbf4d1d53abbc 10.41kB / 10.41kB
 => => sha256:93ab151da4e5310ea79c4ecf306ece628262b86a4d7a49cc601664f19fe44e36 1.75kB / 1.75kB
 => => sha256:62a08b8dd4f53ad5493dabf2af00ccde91abb3771fb2187040bcf2fe94a7ced7 248B / 248B
 => => extracting sha256:302e3ee498053a7b5332ac79e8efebec16e900289fc1ecd1c754ce8fa047fcab
 => => extracting sha256:4c0965d3919510b506d8856ebc050a96e996c7dae96e4fb420882dbe7e037e67
 => => extracting sha256:fdeeec85abbad3878f2008f9445f15a19a5a224d1b7e7715ac6b923072333e57
 => => extracting sha256:62a08b8dd4f53ad5493dabf2af00ccde91abb3771fb2187040bcf2fe94a7ced7
 => [internal] load build context
 => => transferring context: 374B
 => [2/3] WORKDIR /app
 => [3/3] COPY . /app
 => exporting to image
 => => exporting layers
 => => writing image sha256:533190b60949419d7bf28c5d7809b0b898ec531424e70c6c1416d59a1e74f509
```

```
alishror171@cloudshell:~/hello-from-docker (cloud-app-dev-course)$ docker run --rm hello-from-docker
Hello from inside the container!
```