Cloud development

Assignment 2

Exercise 1:

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

1. **Setup**:
   - Ensure you have a Google Cloud account.
   - Install the Google Cloud SDK on your local machine.

```
aipislam03@cloudshell:~$ gcloud --version
Google Cloud SDK 495.0.0
alpha 2024.09.27
app-engine-go 1.9.76
app-engine-java 2.0.30
app-engine-python 1.9.114
app-engine-python-extras 1.9.107
beta 2024.09.27
bigtable
```

2. **Create a Project**:
   - Create a new project in the Google Cloud Console.

```
aipislam03@cloudshell:~$ gcloud projects create z1x2c3 --name="Simple app"
```

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

```
  GNU nano 7.2                                                    hello.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)
^G Help      ^O Write Out   ^W Where Is   ^K Cut       ^T Execute   ^C Location  M-U Undo      M-A Set Mark  M-] To Bracket  M-Q Previous  ^B Back
^X Exit      ^R Read File   ^\ Replace     ^U Paste     ^J Justify   ^/ Go To Line M-E Redo      M-6 Copy      ^Q Where Was    M-W Next      ^F Forward
```

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

Create a app.yaml file with the following content:

runtime: python39

handlers:

- url: /.*

script: auto

```
  GNU nano 7.2
runtime: python39
handlers:
  - url: /.*
    script: auto

^G Help        ^O Write Out   ^W Where Is    ^K Cut
^X Exit        ^R Read File   ^\ Replace     ^U Paste
```

1. **Deploy the Application**:

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

gcloud app deploy

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

For this task to be completed billing account is needed.

After running `gcloud app deploy app.yaml` you will be asked to choose zone, to view available list of zones you can run: gcloud compute zones list. As the result your app will be deployed, and you will be able to view it through given URL.

```
Please choose the region where you want your App Engine application located:

 [1] asia-east1     (supports standard and flexible)
 [2] asia-east2     (supports standard and flexible and search_api)
 [3] asia-northeast1 (supports standard and flexible and search_api)
 [4] asia-northeast2 (supports standard and flexible and search_api)
 [5] asia-northeast3 (supports standard and flexible and search_api)
 [6] asia-south1    (supports standard and flexible and search_api)
 [7] asia-southeast1 (supports standard and flexible)
 [8] asia-southeast2 (supports standard and flexible and search_api)
 [9] australia-southeast1 (supports standard and flexible and search_api)
 [10] europe-central2 (supports standard and flexible)
 [11] europe-west    (supports standard and flexible and search_api)
 [12] europe-west2   (supports standard and flexible and search_api)
 [13] europe-west3   (supports standard and flexible and search_api)
```

Exercise 2: Building with Google Cloud Functions

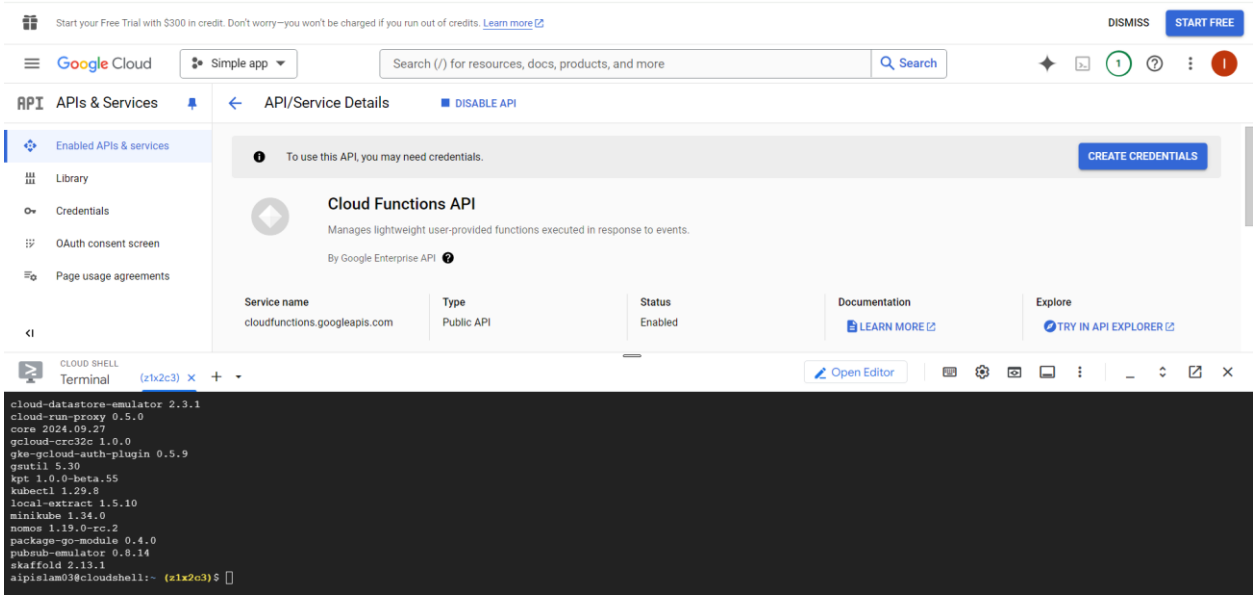
Objective: Create a Google Cloud Function that processes HTTP requests.

1. **Setup**:
   - Ensure you have a Google Cloud account.
   - Install the Google Cloud SDK on your local machine.

```
aipislam03@cloudshell:~ (z1x2c3)$ gcloud --version
Google Cloud SDK 495.0.0
alpha 2024.09.27
app-engine-go 1.9.76
app-engine-java 2.0.30
app-engine-python 1.9.114
app-engine-python-extras 1.9.107
beta 2024.09.27
bigtable
bq 2.1.8
```

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

We need to enable Cloud Functions API



Secondly, we run touch index.js in our cloud shell to create index.js file and by using nano index.js we can write this code:

3. **Write the Code**:
    - Write a simple function that returns "Hello, World!" when accessed via HTTP.

-

4. **Deploy the Function**:

Use the following command to deploy the function:

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

To be able to run this command billing account is required:

```
aipislam03@cloudshell:~ (z1x2c3)$ gcloud functions deploy helloWorldFunction --runtime nodejs18 --trigger-http --entry-point=helloWorld
ERROR: (gcloud.functions.deploy) ResponseError: status=[403], code=[Ok], message=[Read access to project 'z1x2c3' was denied: please check billing account associated and retry]
```

After deploying this function, we should be able to view it via provided URL. When clicked it should navigate to our website with ‘Hello, World!’.

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

To test this function we can use curl:
curl https://REGION-PROJECT.cloudfunctions.net/helloWorldFunction

Exercise 3: Containerizing Applications

Objective: Containerize a simple application using Docker.

Instructions:

1. **Setup**:
    - Ensure Docker is installed on your local machine.

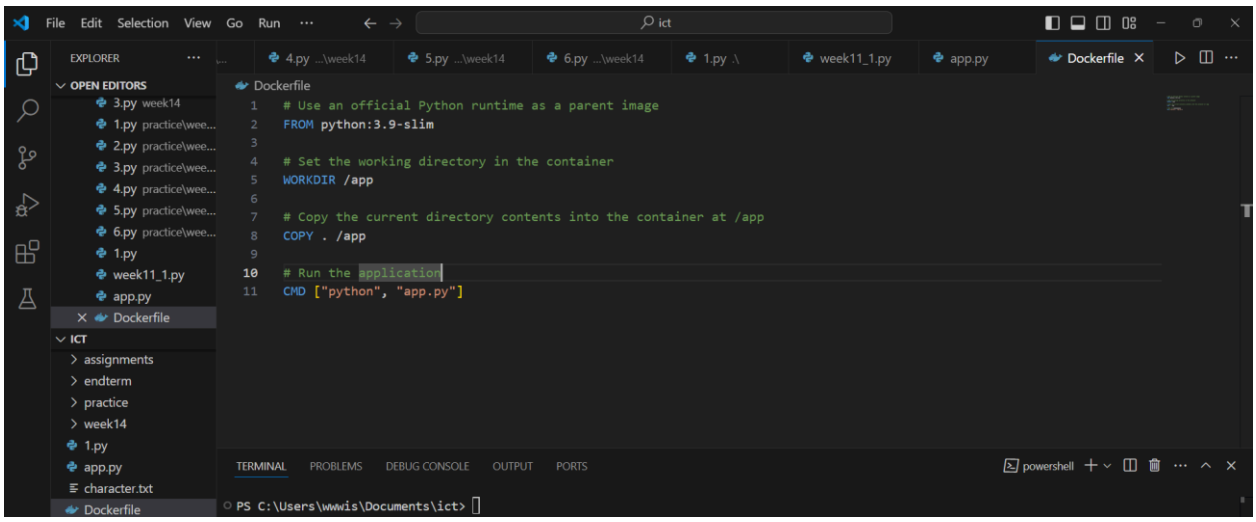
```
PS C:\Users\wwwis> docker --version
Docker version 25.0.3, build 4debf41
```

2. **Create a Simple Application**:
    - Write a simple Python application.

Example app.py:

print("Hello from inside the container!")

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



4. **Build the Docker Image**:

Build the Docker image using the following command:

docker build -t hello-world-app .

```
PS C:\Users\wwwis\Documents\ict> docker build -t hello-world-app .
[+] Building 34.4s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 305B
 => [internal] load metadata for docker.io/library/python:3.9-slim
 => [internal] load .dockerignore
```

5. **Run the Docker Container**:

Run the container using the following command:
docker run --rm hello-world-app

```
PS C:\Users\wwwis\Documents\ict> docker run --rm hello-world-app
Hello from inside the container!
```

```
PS C:\Users\wwwis\Documents\ict> docker image ls
REPOSITORY        TAG       IMAGE ID       CREATED          SIZE
hello-world-app   latest    1e1784d6566b   2 minutes ago    125MB
```