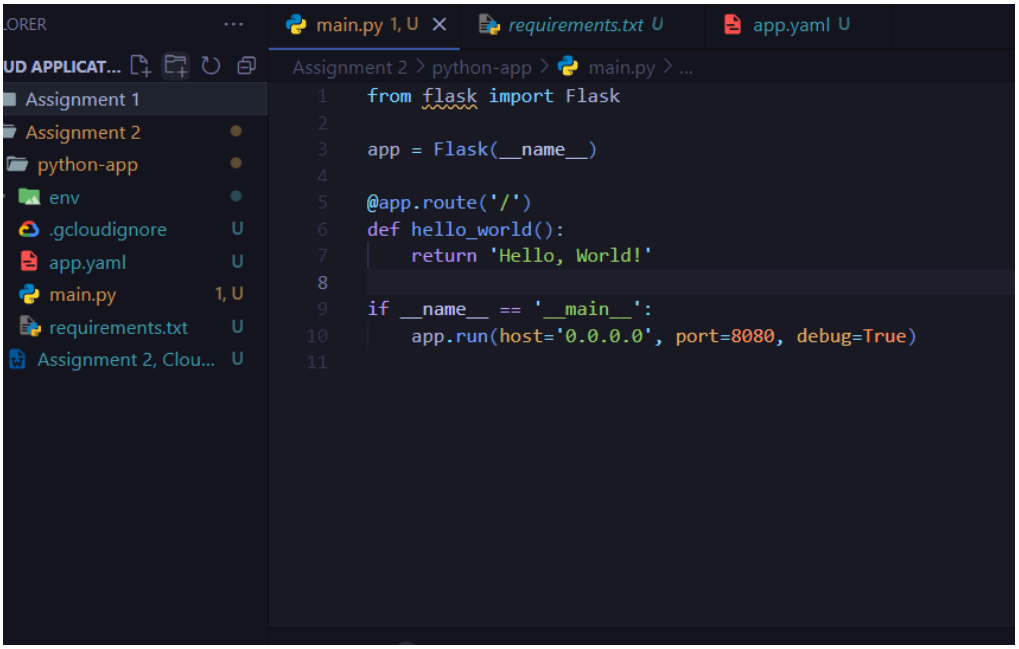
Cloud Application Development. Assignment №2
Amen Azat, 21B030774

Exercise 1: Google App Engine

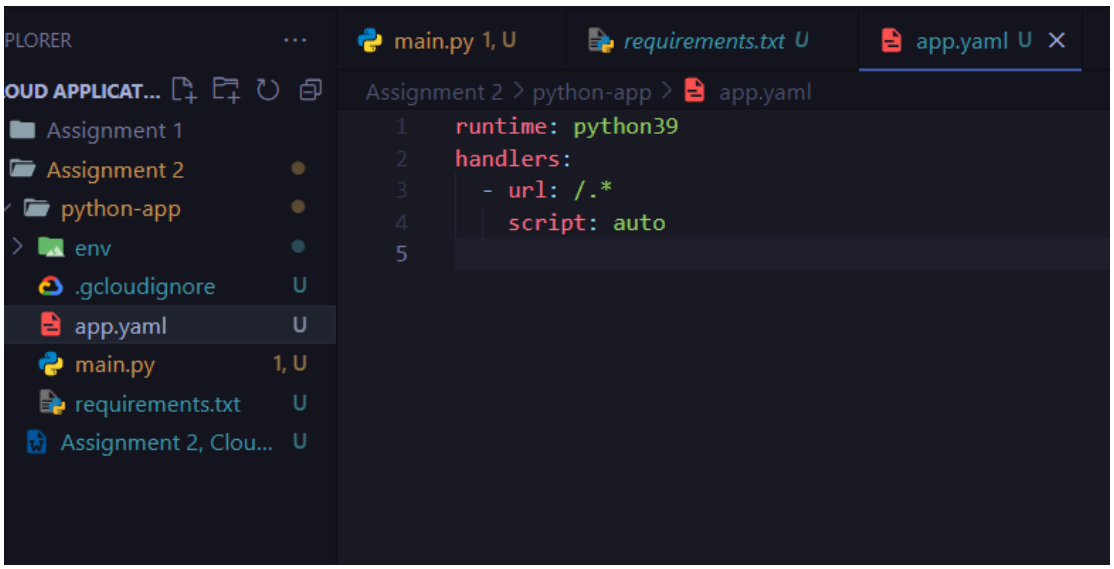
1. **Objective**: Deploy a simple web application on Google App Engine.
2. **Instructions**:
    1. **Setup**:
        - Ensure you have a Google Cloud account.
        - Install the Google Cloud SDK on your local machine.
    2. **Create a Project**:
        - Create a new project in the Google Cloud Console.
    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)
```



4. **Create the App Engine Configuration**:
   Create a `app.yaml` file with the following content:

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



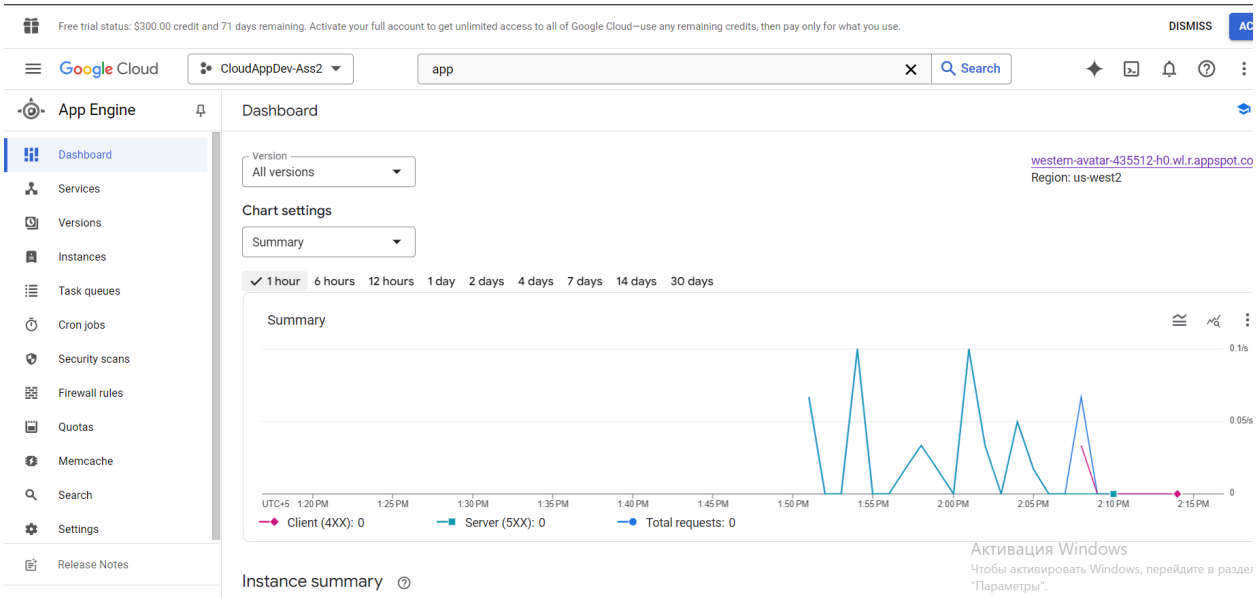
5. **Deploy the Application**:
   - Use the following command to deploy the application to Google App Engine: `gcloud app deploy`
   -

```
E:\Study I KBTU\Semester VII\Cloud Application Development I Serek A\Assignment 2\python-app>gcloud app deploy
You are creating an app for project [western-avatar-435512-h0].
WARNING: Creating an App Engine application for a project is irreversible and the region
cannot be changed. More information about regions is at
<https://cloud.google.com/appengine/docs/locations>.

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)
 [14] europe-west6  (supports standard and flexible and search_api)
 [15] northamerica-northeast1 (supports standard and flexible and search_api)
 [16] southamerica-east1 (supports standard and flexible and search_api)
 [17] us-central    (supports standard and flexible and search_api)
 [18] us-east1      (supports standard and flexible and search_api)
 [19] us-east4      (supports standard and flexible and search_api)
 [20] us-west1      (supports standard and flexible)
 [21] us-west2      (supports standard and flexible and search_api)
 [22] us-west3      (supports standard and flexible and search_api)
 [23] us-west4      (supports standard and flexible and search_api)
 [24] cancel
Please enter your numeric choice:  21
```

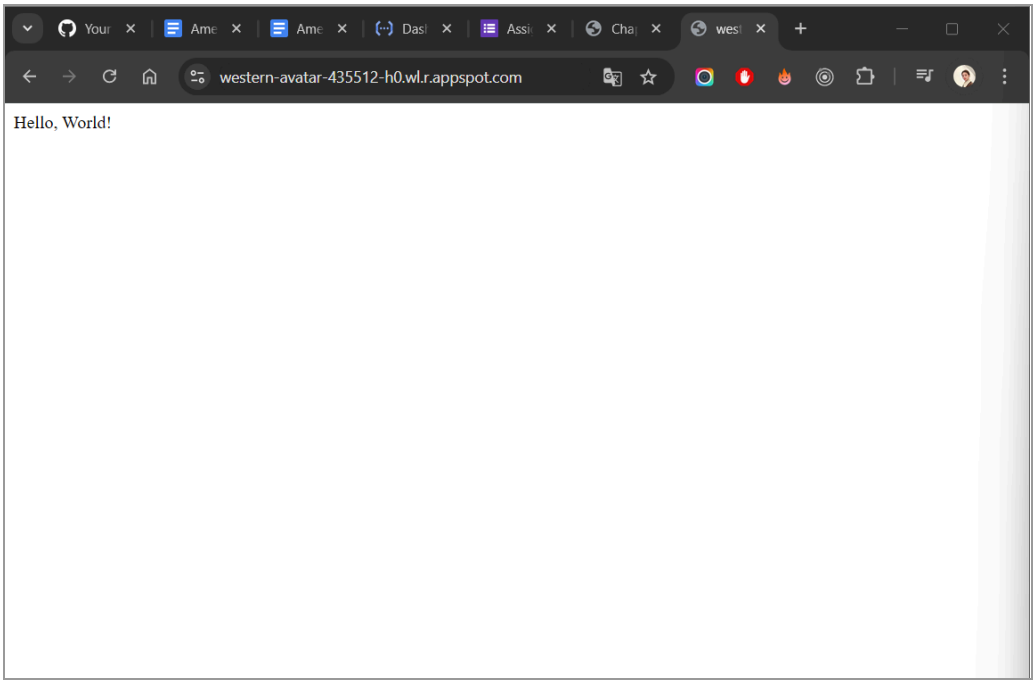
- ```
  Services to deploy:

  descriptor:                  [E:\Study I KBTU\Semester VII\Cloud Application Development I Serek A\Assignment 2\python-app\app.yaml]
  source:                      [E:\Study I KBTU\Semester VII\Cloud Application Development I Serek A\Assignment 2\python-app]
  target project:              [western-avatar-435512-h0]
  target service:              [default]
  target version:              [20241004t133840]
  target url:                  [https://western-avatar-435512-h0.wl.r.appspot.com]
  target service account:      [western-avatar-435512-h0@appspot.gserviceaccount.com]


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

  Beginning deployment of service [default]...
  Created .gcloudignore file. See `gcloud topic gcloudignore` for details.
  ╔═══════════════════════════════════════════════════════════╗
  ═══ Uploading 873 files to Google Cloud Storage ═══
  ╚═══════════════════════════════════════════════════════════╝
  File upload done.
  Updating service [default]...failed.
  ```
- ```
  Updating service [default]...done.
  Setting traffic split for service [default]...done.
  Deployed service [default] to [https://western-avatar-435512-h0.wl.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
  ```

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

3. **Deliverables**:
   - A deployed web application on Google App Engine.

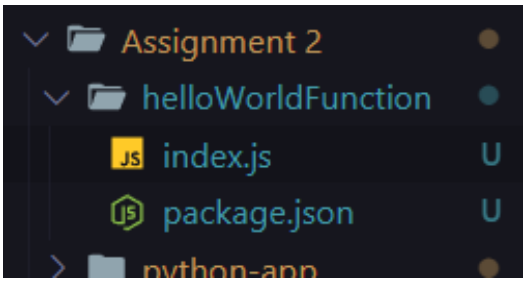


   - A screenshot of the running application.



Exercise 2: Building with Google Cloud Functions

1. **Objective**: Create a Google Cloud Function that processes HTTP requests.
2. **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!');
};
```

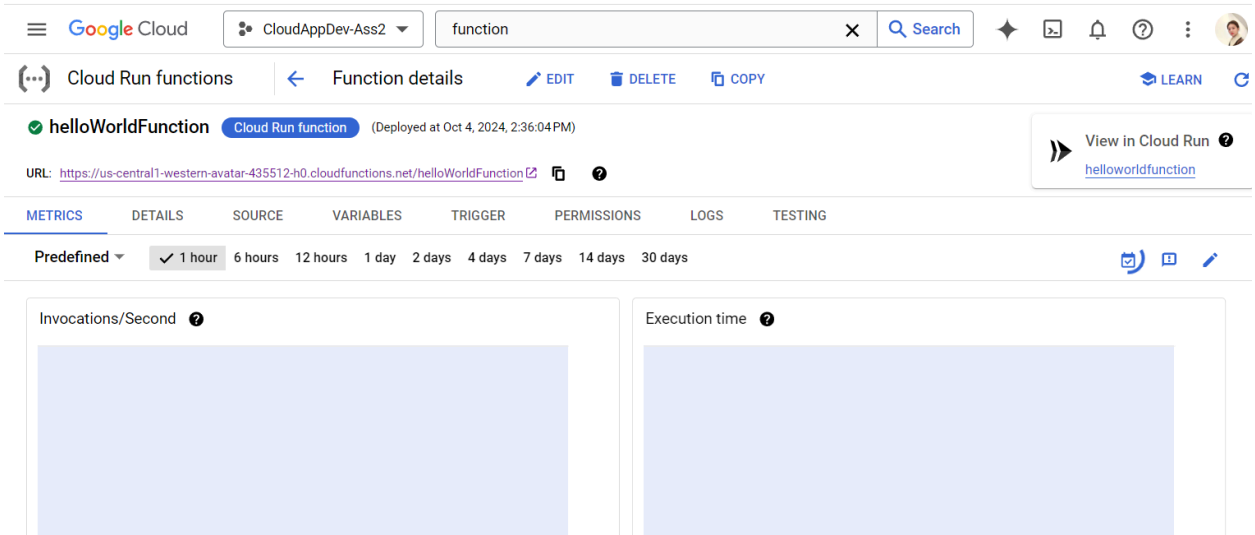
- 

4. **Deploy the Function**:
   - Use the following command to deploy the function:
     `gcloud functions deploy helloWorldFunction --runtime nodejs18 --trigger-http`

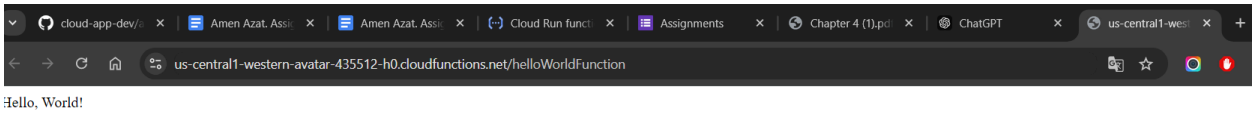
```
buildConfig:
  automaticUpdatePolicy: {}
  build: projects/968084628117/locations/us-central1/builds/a69d585f-b291-47b6-b059-9f8e4bcec6f8
  dockerRegistry: ARTIFACT_REGISTRY
  dockerRepository: projects/western-avatar-435512-h0/locations/us-central1/repositories/gcf-artifacts
  entryPoint: helloWorld
  runtime: nodejs18
  serviceAccount: projects/western-avatar-435512-h0/serviceAccounts/968084628117-compute@developer.gserviceaccount.com
  source:
    storageSource:
      bucket: gcf-v2-sources-968084628117-us-central1
      generation: '1728034508115803'
      object: helloWorldFunction/function-source.zip
  sourceProvenance:
    resolvedStorageSource:
      bucket: gcf-v2-sources-968084628117-us-central1
      generation: '1728034508115803'
      object: helloWorldFunction/function-source.zip
createTime: '2024-10-04T09:31:02.316239952Z'
environment: GEN_2
labels:
  deployment-tool: cli-gcloud
name: projects/western-avatar-435512-h0/locations/us-central1/functions/helloWorldFunction
serviceConfig:
  allTrafficOnLatestRevision: true
  availableCpu: '0.1666'
  availableMemory: 256M
  environmentVariables:
    LOG_EXECUTION_ID: 'true'
  ingressSettings: ALLOW_ALL
  maxInstanceCount: 6
  maxInstanceRequestConcurrency: 1
  revision: helloworldfunction-00001-sew
  service: projects/western-avatar-435512-h0/locations/us-central1/services/helloworldfunction
  serviceAccountEmail: 968084628117-compute@developer.gserviceaccount.com
  timeoutSeconds: 60
  uri: https://helloworldfunction-oznabjbqoa-uc.a.run.app
state: ACTIVE
updateTime: '2024-10-04T09:36:04.398143849Z'
url: https://us-central1-western-avatar-435512-h0.cloudfunctions.net/helloWorldFunction
```

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

3. **Deliverables**:
    - A deployed Google Cloud Function.



    - A screenshot showing the response from the function.



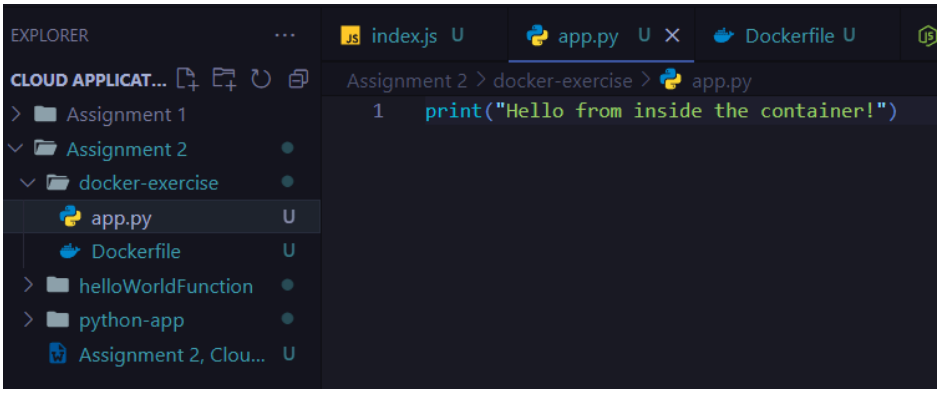

```
url: https://us-central1-western-avatar-435512-h0.cloudfunctions.net/helloWorldFunction

E:\Study I KBTU\Semester VII\Cloud Application Development I Serek A\Assignment 2\helloWorldFunction>curl https://us-central1-western-avatar-435512-h0.cl
oudfunctions.net/helloWorldFunction
Hello, World!
E:\Study I KBTU\Semester VII\Cloud Application Development I Serek A\Assignment 2\helloWorldFunction>
```

Exercise 3: Containerizing Applications

1. **Objective**: Containerize a simple application using Docker.
2. **Instructions**:
    1. **Setup**:

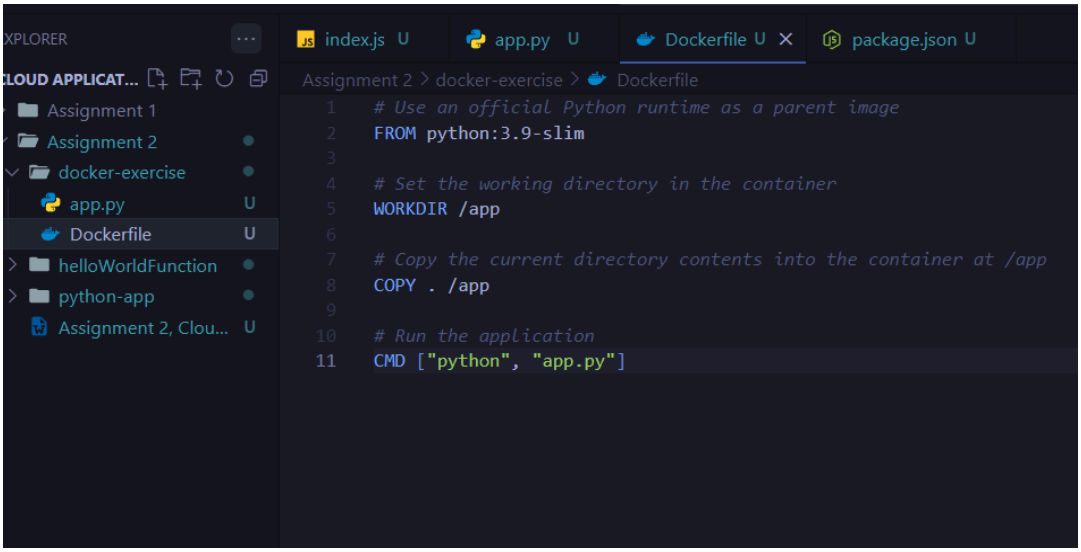
- Ensure Docker is installed on your local machine.

2. **Create a Simple Application**:
   - Write a simple Python application.
     Example `app.py`:
     ```
     print("Hello from inside the container!")
     ```



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`

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