Assignment 3, cloud app development

Exercise 1: Managing APIs with Google Cloud Endpoints

Objective: Deploy and manage an API using Google Cloud Endpoints.

Instructions:

1. **Setup**:
    - Ensure you have a Google Cloud account.
    - Install the Google Cloud SDK and gcloud command-line tool.
2. **Create a Project**:
    - Create a new project in the Google Cloud Console.
3. **Prepare the API**:
    - Create a simple REST API using Python Flask.

    Example app.py:

    ```
    from flask import Flask, jsonify

    app = Flask(__name__)

    @app.route('/api/hello', methods=['GET'])
    def hello():
        return jsonify({'message': 'Hello, World!'})

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

    We craet app.py python file with our application
4. **Create an OpenAPI Specification**:
    - Create an openapi.yaml file to define your API.

Example openapi.yaml:

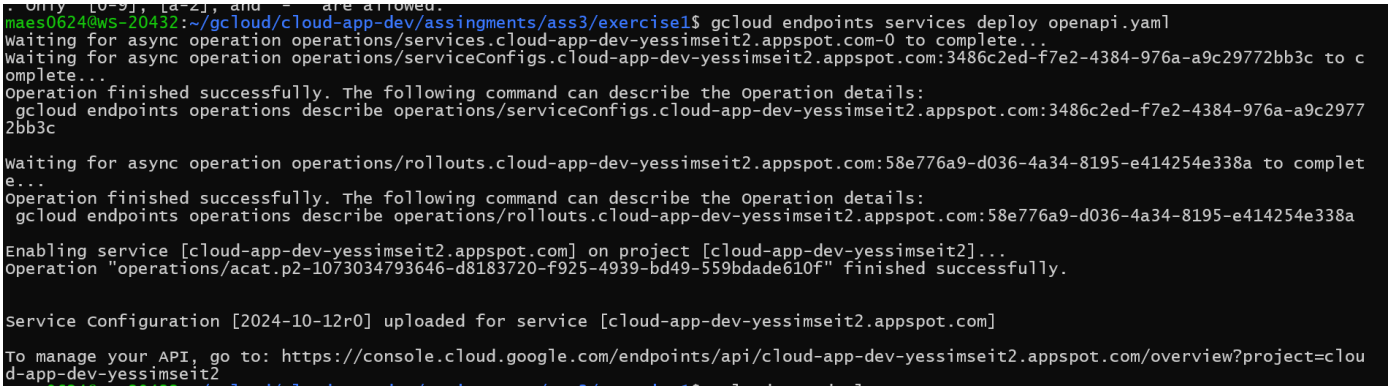
```
openapi: 3.0.0
info:
  title: Hello World API
  description: A simple API to say hello
  version: 1.0.0
paths:
  /api/hello:
    get:
      summary: Returns a hello message
      responses:
        '200':
          description: A hello message
          content:
            application/json:
```

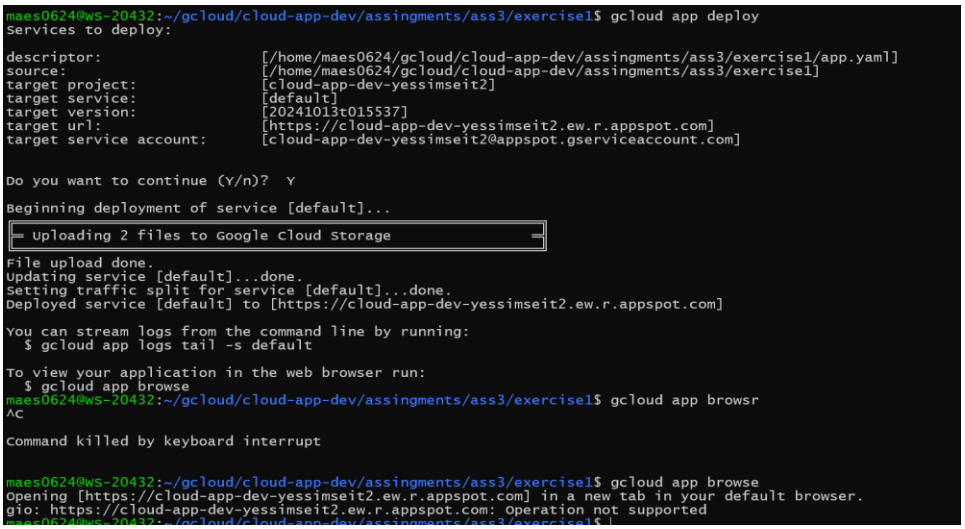
```
schema:
  type: object
  properties:
    message:
      type: string
      example: Hello, World!
```

5. **Deploy the API to Google Cloud Endpoints**:

   Create a new service and deploy your API.

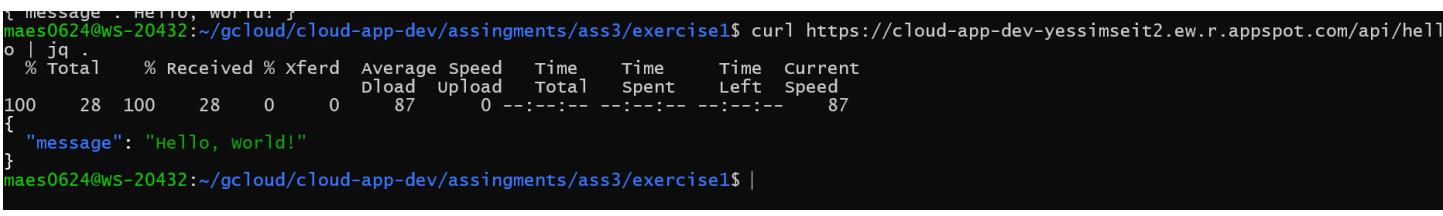
Use the following commands to deploy the API configuration and service:
gcloud endpoints services deploy openapi.yaml use this command we deploy our API





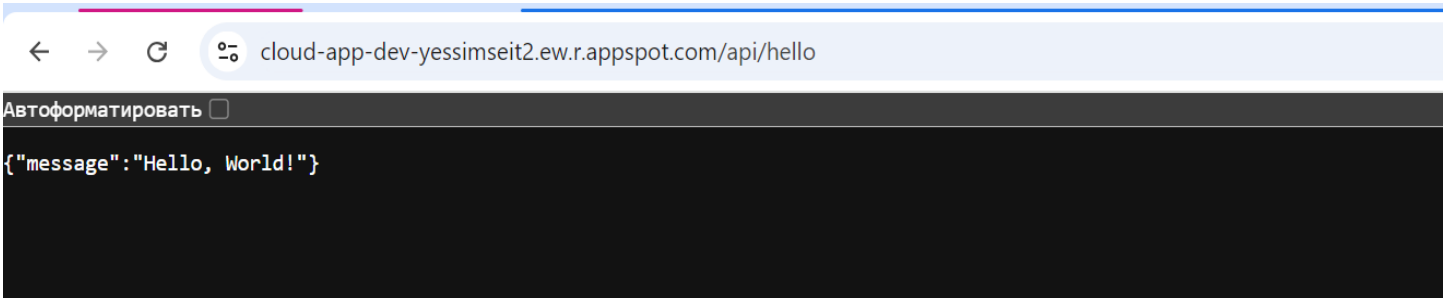
gcloud app deploy command to deploy application.

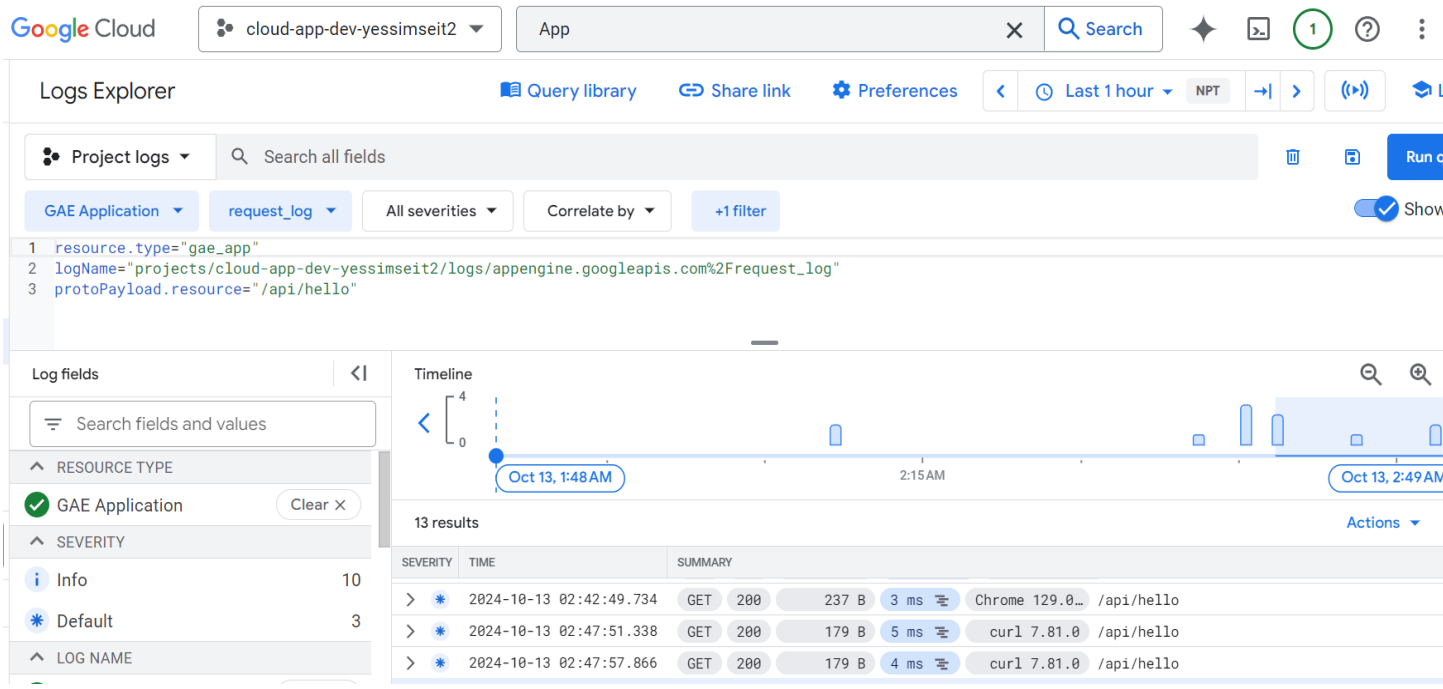
6. **Test the API**:
   - Once deployed, use the provided URL to test the API endpoint via a web browser or curl.



Deliverables:

- A deployed API on Google Cloud Endpoints.
- A screenshot of a successful API call response.

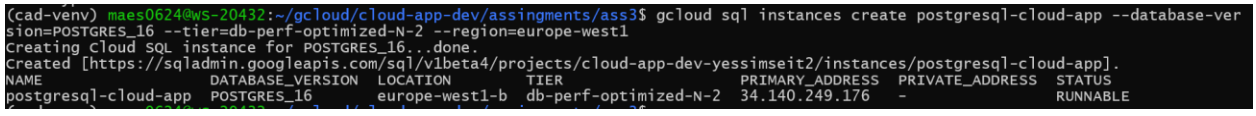




Exercise 2: Google Cloud Databases

Objective: Set up and interact with a Google Cloud SQL database.

Instructions:

1. **Setup**:
   - Ensure you have a Google Cloud account.
   - Install the Google Cloud SDK.
2. **Create a Cloud SQL Instance**:
   - Navigate to the Google Cloud Console and create a new Cloud SQL instance.
   - Choose MySQL, PostgreSQL, or SQL Server as the database type.
   - Configure the instance settings (region, machine type, etc.).
   1) We create postgresql database on GCP on europe-west 1 zone.
      - 

   2) We create new user

```
(cad-venv) maes0624@ws-20432:~/gcloud/cloud-app-dev/assingments/ass3$ gcloud sql users set-password root \
    --host=% \
    --instance=postgresql-cloud-app \
    --password=root-cloud-app
Updating Cloud SQL user...done.
(cad-venv) maes0624@ws-20432:~/gcloud/cloud-app-dev/assingments/ass3$
```

3. **Create a Database and Table**:
   - Connect to your Cloud SQL instance using the Cloud SQL client or mysql command-line tool.
   - Create a new database and a table with sample data.

```
postgres=> USE sample_db;
ERROR:  syntax error at or near "USE"
LINE 1: USE sample_db;
        ^
postgres=> \l
postgres=> \c sample_db
Password:
psql (14.13 (Ubuntu 14.13-0ubuntu0.22.04.1), server 16.4)
WARNING: psql major version 14, server major version 16.
         Some psql features might not work.
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA3
4, bits: 256, compression: off)
You are now connected to database "sample_db" as user "postgres
.
sample_db=>
```

```
postgres=> \c sample_db
Password:
psql (14.13 (Ubuntu 14.13-0ubuntu0.22.04.1), server 16.4)
WARNING: psql major version 14, server major version 16.
         Some psql features might not work.
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
You are now connected to database "sample_db" as user "postgres".
sample_db=> CREATE TABLE users (id SERIAL PRIMARY KEY, name VARCHAR(100) NOT NULL, email VARCHAR(100) NOT NULL);
CREATE TABLE
sample_db=> INSERT INTO users (name, email) VALUES ('Manarbek Yessimseit', 'm_esimseit@kbtu.kz');
INSERT 0 1
sample_db=> INSERT INTO users (name, email) VALUES ('Azat Amen', 'a_amen@kbtu.kz');
INSERT 0 1
```

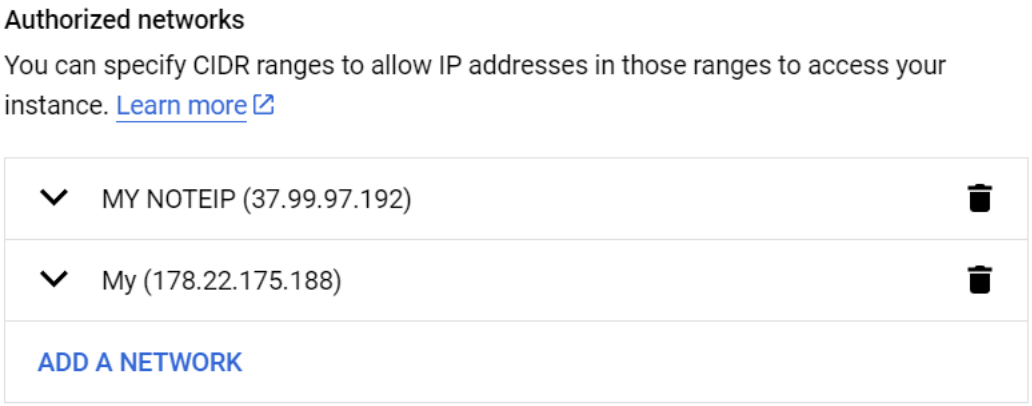
```
(cad-venv) maes0624@ws-20432:~/gcloud/cloud-app-dev/assingments/ass3$ gcloud sql connect postgresql-cloud-app
Allowlisting your IP for incoming connection for 5 minutes...do
ne.
Connecting to database with SQL user [postgres].Password:
psql (14.13 (Ubuntu 14.13-0ubuntu0.22.04.1), server 16.4)
WARNING: psql major version 14, server major version 16.
         Some psql features might not work.
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

postgres=> \c sample_db
Password:
psql (14.13 (Ubuntu 14.13-0ubuntu0.22.04.1), server 16.4)
WARNING: psql major version 14, server major version 16.
         Some psql features might not work.
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
You are now connected to database "sample_db" as user "postgres".
sample_db=> CREATE TABLE users (id SERIAL PRIMARY KEY, name VARCHAR(100) NOT NULL, email VARCHAR(100) NOT NULL);
CREATE TABLE
sample_db=> INSERT INTO users (name, email) VALUES ('Manarbek Yessimseit', 'm_esimseit@kbtu.kz');
INSERT 0 1
sample_db=> INSERT INTO users (name, email) VALUES ('Azat Amen', 'a_amen@kbtu.kz');
INSERT 0 1
sample_db=> SELECT * FROM users;
 id |        name         |       email
----+---------------------+--------------------
  1 | Manarbek Yessimseit | m_esimseit@kbtu.kz
  2 | Azat Amen           | a_amen@kbtu.kz
(2 rows)

sample_db=>
```

How you see out database run, we success create new table and add data.

4. **Connect to the Database**:
    - Create a connection to the Cloud SQL instance from a Python application.

```
import psycopg2

conn = psycopg2.connect(
    user = 'postgres',
    password = 'postgres-cloud-app-dev',
    database = 'sample_db',
    host = '34.140.249.176',
    port = 5432
)

cursor = conn.cursor()
cursor.execute('SELECT * FROM users')

rows = cursor.fetchall()

for row in rows:
    print(row)

cursor.close()
conn.close()
```

Regarding to security we need add IPv4 address to allow access to database. As we remember postgresql have pg_gba.conf file that contains all settings of valid addresses that can connect to the database.

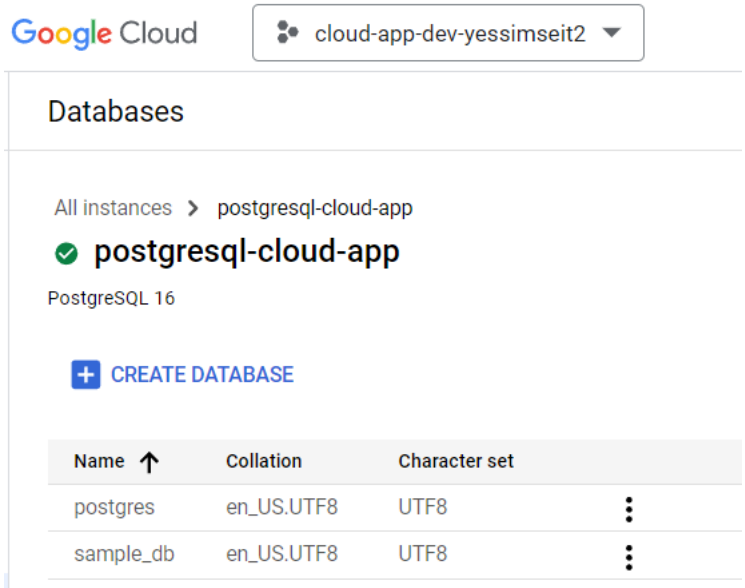


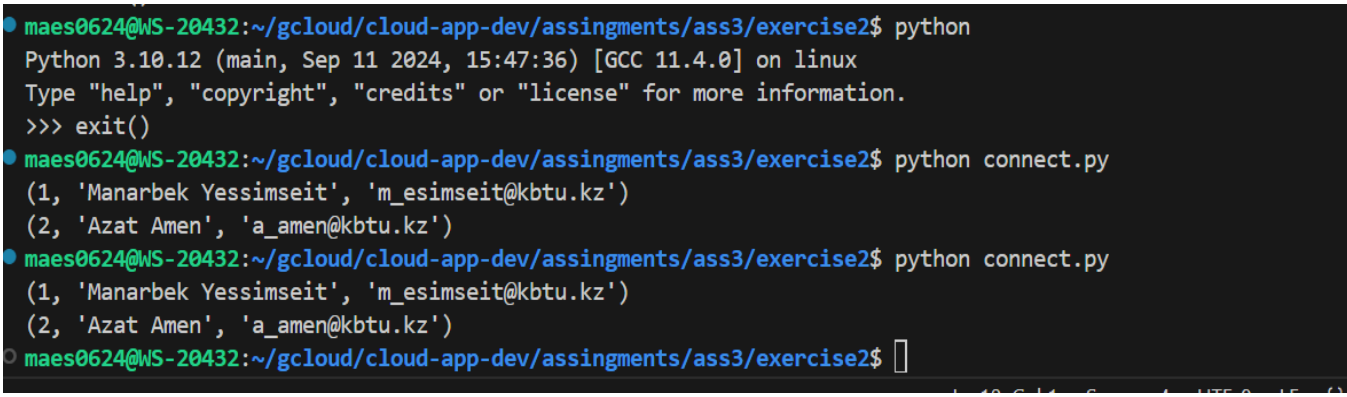
25. **Run the Connection Code**:

Execute the Python script to verify that you can retrieve data from the Cloud SQL instance.

Deliverables:

- A working Cloud SQL database with sample data.
- A Python script that successfully connects to and queries the database.



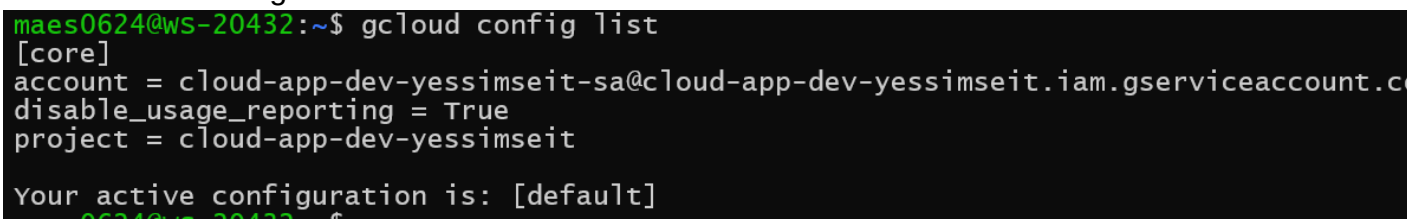
- 

Exercise 3: Integrating Machine Learning with Google Cloud

Objective: Train and deploy a machine learning model using Google Cloud AI Platform.

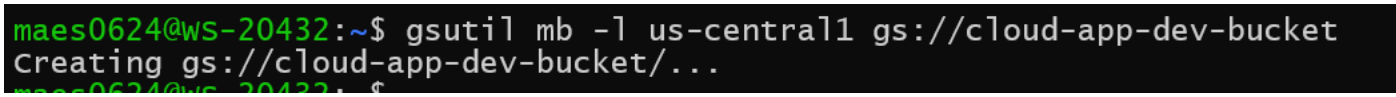
Instructions:

1. **Setup**:
   - Ensure you have a Google Cloud account.
   - Install the Google Cloud SDK and TensorFlow.



2. **Create a Cloud Storage Bucket**:
   - Create a new Cloud Storage bucket to store your training data and model.

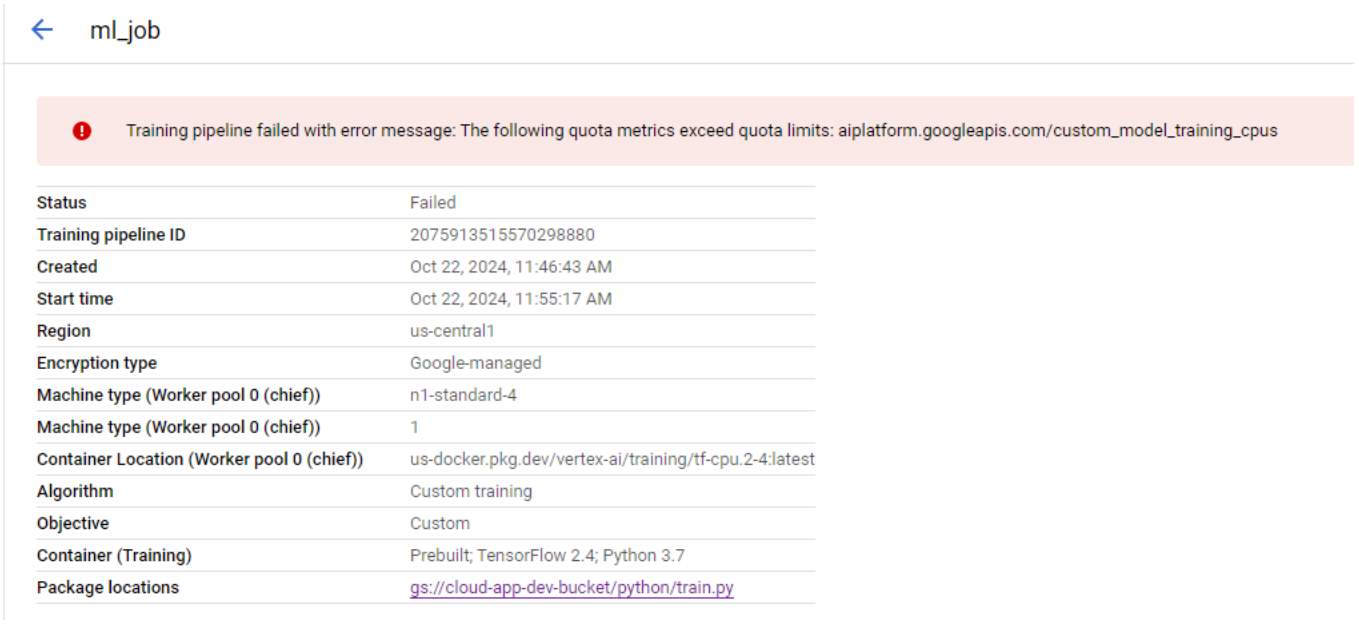
Creating cloud storage bucket on us-central1 zone



3. **Prepare Training Data**:
   - Upload sample training data to your Cloud Storage bucket. For example, use a dataset for classification or regression.

```
maes0624@WS-20432:~/gcloud/cloud-app-dev/assingments/ass3/exercise3$ gsutil cp data.csv gs://cloud-app-dev-bucket/dataset/data.csv
Copying file://data.csv [Content-Type=text/csv]...
/ [1 files][ 36.0 MiB/ 36.0 MiB]  695.0 KiB/s
Operation completed over 1 objects/36.0 MiB.
maes0624@WS-20432:~/gcloud/cloud-app-dev/assingments/ass3/exercise3$
```

4. **Create a Training Script**:
   - Write a simple TensorFlow training script.



We have quota limit so reason I can't use google machine power to trian my model. So I just train on my laptop

```python
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
import pandas as pd

# Load the dataset
data = pd.read_csv('./data.csv')

# Preprocessing the data
features = data[['Model Year', 'Make', 'Model', 'E.V_Type', 'Base MSRP']]
target = data['Electric Range']

# Handle categorical data
features['Make'] = LabelEncoder().fit_transform(features['Make'])
features['Model'] = LabelEncoder().fit_transform(features['Model'])
features['E.V_Type'] = LabelEncoder().fit_transform(features['E.V_Type'])

# Handle missing values
features.fillna(features.mean(), inplace=True)
target.fillna(target.mean(), inplace=True)

# Split the data
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)

# Normalize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Create a TensorFlow model
def create_model():
    model = tf.keras.Sequential([
        tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
        tf.keras.layers.Dense(32, activation='relu'),
        tf.keras.layers.Dense(1)
    ])
```

5. **Train the Model**:
    - Submit a training job to Google Cloud AI Platform.

```
gcloud ai models upload \
 --region=us-central1 \
1. --display-name=saved_model \
2. --artifact-uri=gs://cloud-app-dev-bucket/models/saved_model.pb \
3. --container-image-uri=us-docker.pkg.dev/vertex-ai/prediction/tf2-cpu.2-8:latest
4.
```

5. **Deploy the Model**:
    - Deploy the trained model to an AI Platform endpoint.

```
6.    gcloud ai models versions create v1 \
7.    --model=model0id \
8.    --region=us-central1 \
9.    --runtime-version=2.8 \
10.   --python-version=3.8 \
11.   --origin=gs://your-bucket-name/model/ \
12.   --machine-type=n1-standard-4
```

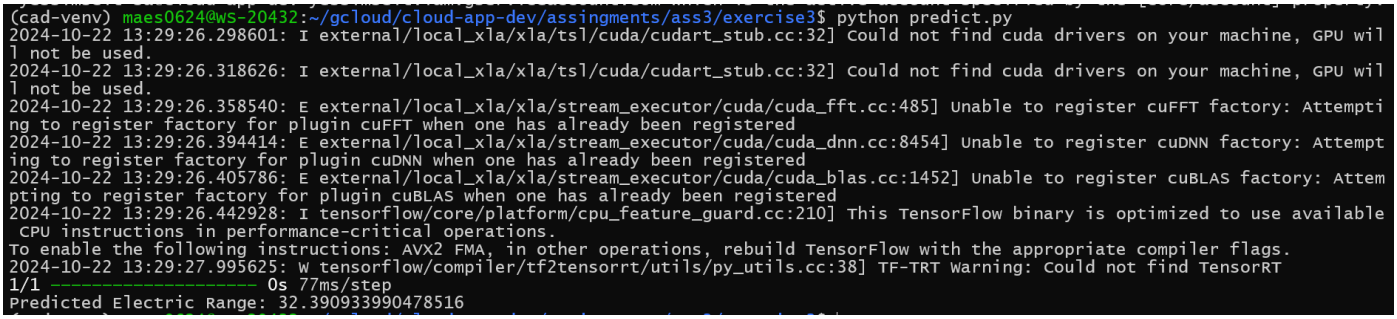
13. **Test the Model**:
    - Use the deployed model endpoint to make predictions.

predict.py > ...

```python
import tensorflow as tf
import pandas as pd
from sklearn.preprocessing import StandardScaler, LabelEncoder


loaded_model = tf.keras.models.load_model('my_model.keras')

new_data = pd.DataFrame({
    'Model Year': [2022],
    'Make': ['Tesla'],
    'Model': ['Model 3'],
    'E.V_Type': ['Battery Electric Vehicle (BEV)'],
    'Base MSRP': [39990]
})

new_data['Make'] = LabelEncoder().fit_transform(new_data['Make'])
new_data['Model'] = LabelEncoder().fit_transform(new_data['Model'])
new_data['E.V_Type'] = LabelEncoder().fit_transform(new_data['E.V_Type'])

scaler = StandardScaler()
new_data_scaled = scaler.fit_transform(new_data)

prediction = loaded_model.predict(new_data_scaled)
print(f"Predicted Electric Range: {prediction[0][0]}")

```

Deliverables:

- A trained machine learning model deployed on Google Cloud AI Platform.
- A script that makes predictions using the deployed model.
- Report



I really tried to deploy this model into gcloud. But experience on ML not enough for save.