

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 create `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

```
maes0624@ws-20432:~/gcloud/cloud-app-dev/assingments/ass3/exercise1$ gcloud endpoints services deploy openapi.yaml
Waiting for async operation operations/services.cloud-app-dev-yessimseit2.appspot.com-0 to complete...
Waiting for async operation operations/serviceconfigs.cloud-app-dev-yessimseit2.appspot.com:3486c2ed-f7e2-4384-976a-a9c29772bb3c to c
omplete...
Operation finished successfully. The following command can describe the operation details:
gcloud endpoints operations describe operations/serviceconfigs.cloud-app-dev-yessimseit2.appspot.com:3486c2ed-f7e2-4384-976a-a9c2977
2bb3c

Waiting for async operation operations/rollouts.cloud-app-dev-yessimseit2.appspot.com:58e776a9-d036-4a34-8195-e414254e338a to complet
e...
Operation finished successfully. The following command can describe the operation details:
gcloud endpoints operations describe operations/rollouts.cloud-app-dev-yessimseit2.appspot.com:58e776a9-d036-4a34-8195-e414254e338a

Enabling service [cloud-app-dev-yessimseit2.appspot.com] on project [cloud-app-dev-yessimseit2]...
Operation "operations/acat.p2-1073034793646-d8183720-f925-4939-bd49-559bdade610f" finished successfully.

Service Configuration [2024-10-12r0] uploaded for service [cloud-app-dev-yessimseit2.appspot.com]

To manage your API, go to: https://console.cloud.google.com/endpoints/api/cloud-app-dev-yessimseit2.appspot.com/overview?project=clou
d-app-dev-yessimseit2
```

```
maes0624@ws-20432:~/gcloud/cloud-app-dev/assingments/ass3/exercise1$ gcloud app deploy
Services to deploy:

descriptor:      [/home/maes0624/gcloud/cloud-app-dev/assingments/ass3/exercise1/app.yaml]
source:          [/home/maes0624/gcloud/cloud-app-dev/assingments/ass3/exercise1]
target project:  [cloud-app-dev-yessimseit2]
target service:  [default]
target version:  [20241013t015537]
target url:      [https://cloud-app-dev-yessimseit2.ew.r.appspot.com]
target service account: [cloud-app-dev-yessimseit2@appspot.gserviceaccount.com]

Do you want to continue (Y/n)? Y
Beginning deployment of service [default]...

Uploading 2 files to Google Cloud Storage
File upload done.
Updating service [default]...done.
Setting traffic split for service [default]...done.
Deployed service [default] to [https://cloud-app-dev-yessimseit2.ew.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
maes0624@ws-20432:~/gcloud/cloud-app-dev/assingments/ass3/exercise1$ gcloud app browsr
AC
command killed by keyboard interrupt

maes0624@ws-20432:~/gcloud/cloud-app-dev/assingments/ass3/exercise1$ gcloud app browse
Opening [https://cloud-app-dev-yessimseit2.ew.r.appspot.com] in a new tab in your default browser.
gio: https://cloud-app-dev-yessimseit2.ew.r.appspot.com: operation not supported
maes0624@ws-20432:~/gcloud/cloud-app-dev/assingments/ass3/exercise1$
```

`gcloud app deploy` command to deploy application.

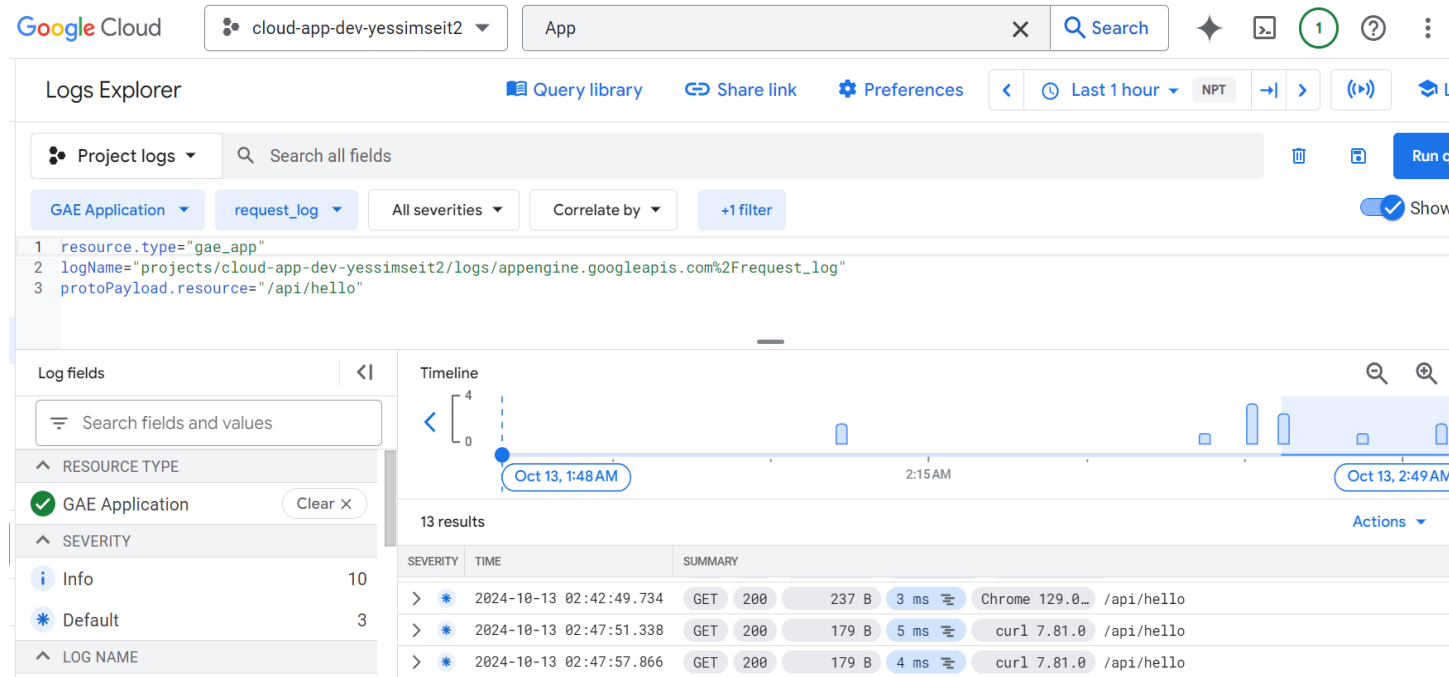
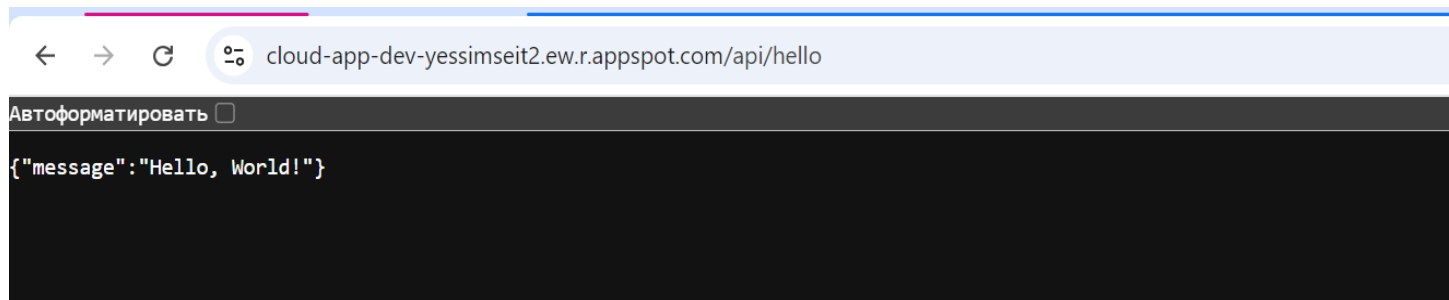
6. Test the API:

- Once deployed, use the provided URL to test the API endpoint via a web browser or `curl`.

```
{ message: "Hello, world!" }
maes0624@ws-20432:~/gcloud/cloud-app-dev/assingments/ass3/exercise1$ curl https://cloud-app-dev-yessimseit2.ew.r.appspot.com/api/he11
o | jq
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload    Dload  Upload    Total   Spent    Left   Speed
100    28    100    28    0    0    87      0  --:--:-- --:--:-- --:--:--    87
{
  "message": "Hello, world!"
}
maes0624@ws-20432:~/gcloud/cloud-app-dev/assingments/ass3/exercise1$ |
```

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:

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

```
(cad-venv) maes0624@ws-20432:~/gcloud/cloud-app-dev/assigments/ass3$ gcloud sql instances create postgresql-cloud-app --database-ver
sion=POSTGRES_16 --tier=db-perf-optimized-N-2 --region=europe-west1
Creating Cloud SQL instance for POSTGRES_16...done.
Created [https://sqladmin.googleapis.com/sql/v1beta4/projects/cloud-app-dev-yessimseit2/instances/postgresql-cloud-app].
NAME          DATABASE_VERSION  LOCATION    TIER          PRIMARY_ADDRESS  PRIVATE_ADDRESS  STATUS
-----
postgresql-cloud-app  POSTGRES_16      europe-west1-b  db-perf-optimized-N-2  34.140.249.176    -                RUNNABLE
```

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_SHA384, 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...done.
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.

```
5. import psycopg2
6.
7. conn = psycopg2.connect(
8.     user = 'postgres',
9.     password = 'postgres-cloud-app-dev',
10.    database = 'sample_db',
11.    host = '34.140.249.176',
12.    port = 5432
13.)
14.
15.cursor = conn.cursor()
16.cursor.execute('SELECT * FROM users')
17.
18.rows = cursor.fetchall()
19.
20.for row in rows:
21.    print(row)
22.
23.cursor.close()
24.conn.close()
```

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

Authorized networks

You can specify CIDR ranges to allow IP addresses in those ranges to access your instance. [Learn more](#)

▼ MY NOTEIP (37.99.97.192)	
▼ My (178.22.175.188)	
ADD A NETWORK	

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.

Databases

All instances > postgresql-cloud-app

✓ postgresql-cloud-app

PostgreSQL 16

+ CREATE DATABASE

Name ↑	Collation	Character set	
postgres	en_US.UTF8	UTF8	⋮
sample_db	en_US.UTF8	UTF8	⋮

```
maes0624@WS-20432:~/gcloud/cloud-app-dev/assingments/ass3/exercise2$ python
Python 3.10.12 (main, Sep 11 2024, 15:47:36) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
maes0624@WS-20432:~/gcloud/cloud-app-dev/assingments/ass3/exercise2$ python connect.py
(1, 'Manarbek Yessimseit', 'm_esimseit@kbtu.kz')
(2, 'Azat Amen', 'a_amen@kbtu.kz')
maes0624@WS-20432:~/gcloud/cloud-app-dev/assingments/ass3/exercise2$ python connect.py
(1, 'Manarbek Yessimseit', 'm_esimseit@kbtu.kz')
(2, 'Azat Amen', 'a_amen@kbtu.kz')
maes0624@WS-20432:~/gcloud/cloud-app-dev/assingments/ass3/exercise2$
```

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.

```
maes0624@WS-20432:~$ gcloud config list
[core]
account = cloud-app-dev-yessimseit-sa@cloud-app-dev-yessimseit.iam.gserviceaccount.c
disable_usage_reporting = True
project = cloud-app-dev-yessimseit

Your active configuration is: [default]
```

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

```
maes0624@WS-20432:~$ gsutil mb -l us-central1 gs://cloud-app-dev-bucket
Creating gs://cloud-app-dev-bucket/...
```

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.

← ml_job

 Training pipeline failed with error message: The following quota metrics exceed quota limits: aiplatform.googleapis.com/custom_model_training_cpus

Status	Failed
Training pipeline ID	2075913515570298880
Created	Oct 22, 2024, 11:46:43 AM
Start time	Oct 22, 2024, 11:55:17 AM
Region	us-central1
Encryption type	Google-managed
Machine type (Worker pool 0 (chief))	n1-standard-4
Machine type (Worker pool 0 (chief))	1
Container Location (Worker pool 0 (chief))	us-docker.pkg.dev/vertex-ai/training/tf-cpu.2-4:latest
Algorithm	Custom training
Objective	Custom
Container (Training)	Prebuilt; TensorFlow 2.4; Python 3.7
Package locations	gs://cloud-app-dev-bucket/python/train.py

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

```
train.py > ...
1 import tensorflow as tf
2 from sklearn.model_selection import train_test_split
3 from sklearn.preprocessing import StandardScaler, LabelEncoder
4 import pandas as pd
5
6 # Load the dataset
7 data = pd.read_csv('./data.csv')
8
9 # Preprocessing the data
10 features = data[['Model Year', 'Make', 'Model', 'E.V_Type', 'Base MSRP']]
11 target = data['Electric Range']
12
13 # Handle categorical data
14 features['Make'] = LabelEncoder().fit_transform(features['Make'])
15 features['Model'] = LabelEncoder().fit_transform(features['Model'])
16 features['E.V_Type'] = LabelEncoder().fit_transform(features['E.V_Type'])
17
18 # Handle missing values
19 features.fillna(features.mean(), inplace=True)
20 target.fillna(target.mean(), inplace=True)
21
22 # Split the data
23 X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)
24
25 # Normalize the features
26 scaler = StandardScaler()
27 X_train = scaler.fit_transform(X_train)
28 X_test = scaler.transform(X_test)
29
30 # Create a TensorFlow model
31 def create_model():
32     model = tf.keras.Sequential([
33         tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
34         tf.keras.layers.Dense(32, activation='relu'),
35         tf.keras.layers.Dense(1)
36     ])
```

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 > ...  
1 import tensorflow as tf  
2 import pandas as pd  
3 from sklearn.preprocessing import StandardScaler, LabelEncoder  
4  
5  
6 loaded_model = tf.keras.models.load_model('my_model.keras')  
7  
8 new_data = pd.DataFrame({  
9     'Model Year': [2022],  
10    'Make': ['Tesla'],  
11    'Model': ['Model 3'],  
12    'E.V_Type': ['Battery Electric Vehicle (BEV)],  
13    'Base MSRP': [39990]  
14 })  
15  
16 new_data['Make'] = LabelEncoder().fit_transform(new_data['Make'])  
17 new_data['Model'] = LabelEncoder().fit_transform(new_data['Model'])  
18 new_data['E.V_Type'] = LabelEncoder().fit_transform(new_data['E.V_Type'])  
19  
20 scaler = StandardScaler()  
21 new_data_scaled = scaler.fit_transform(new_data)  
22  
23 prediction = loaded_model.predict(new_data_scaled)  
24 print(f"Predicted Electric Range: {prediction[0][0]}")  
25
```

Deliverables:

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

```
(cad-venv) maes0624@ws-20432:~/gcloud/cloud-app-dev/assingments/ass3/exercise3$ python predict.py
2024-10-22 13:29:26.298601: I external/local_xla/xla/tsl/cuda/cudart_stub.cc:32] Could not find cuda drivers on your machine, GPU will not be used.
2024-10-22 13:29:26.318626: I external/local_xla/xla/tsl/cuda/cudart_stub.cc:32] Could not find cuda drivers on your machine, GPU will not be used.
2024-10-22 13:29:26.358540: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:485] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when one has already been registered
2024-10-22 13:29:26.394414: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:8454] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when one has already been registered
2024-10-22 13:29:26.405786: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1452] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been registered
2024-10-22 13:29:26.442928: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-10-22 13:29:27.995625: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT warning: Could not find TensorRT
1/1 ----- 0s 77ms/step
Predicted Electric Range: 32.390933990478516
```

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