“Kazakh-British Technical University” JSC

School of Information Technology and Engineering

Report : Cloud application development

Assignment 3, cloud app development

Student Eleu Ernur Bakytzhanuly 21B031072

Exercise 1: Managing APIs with Google Cloud Endpoints

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

Step 1:

I activated the Cloud Endpoints API in the APIs & Services area to manage and deploy APIs on Google Cloud.

Step 2: Preparing API

I created a simple Python Flask application that returns "Hello, World!".

The app.py file appeared like this:

```
  GNU nano 6.2
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)
```

Flask Installation: To execute the Flask application, I installed Flask with the command:

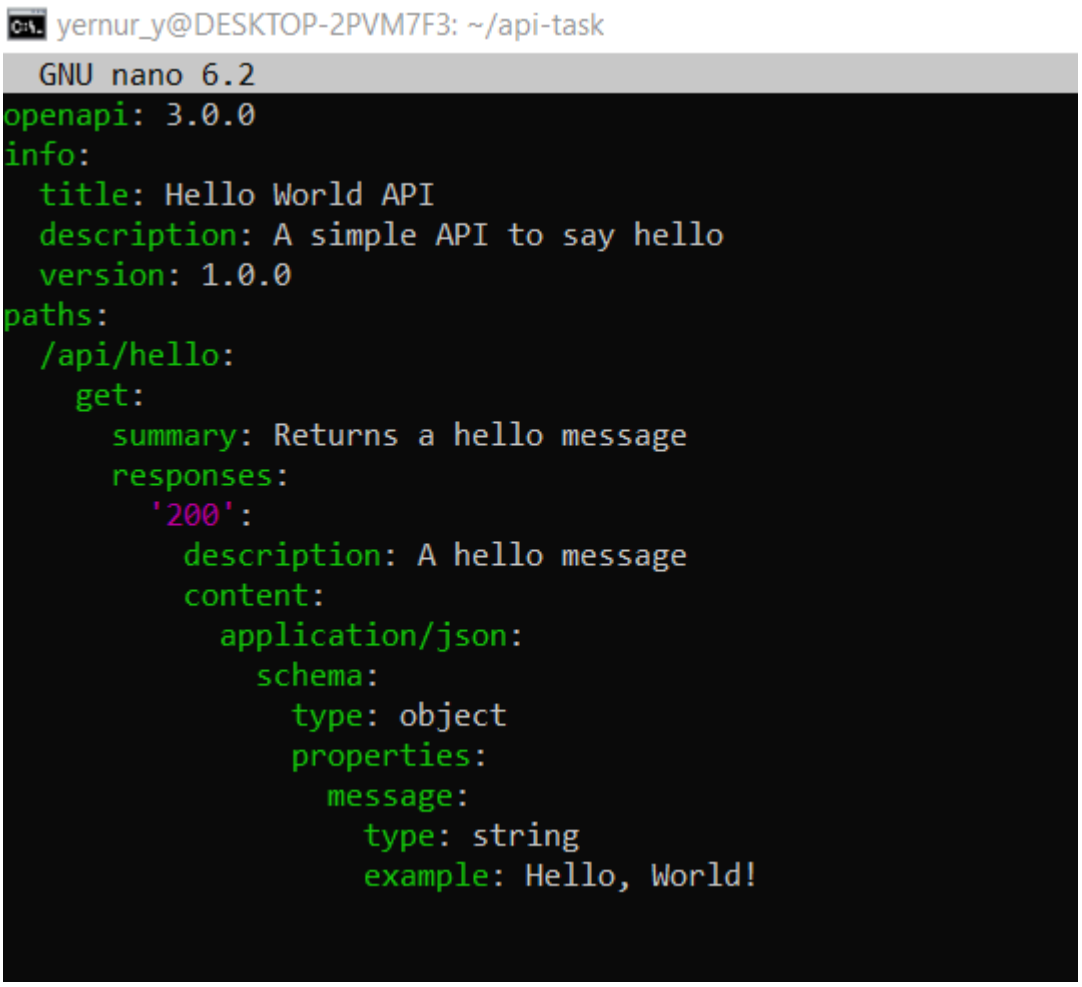
```
pip install Flask
```

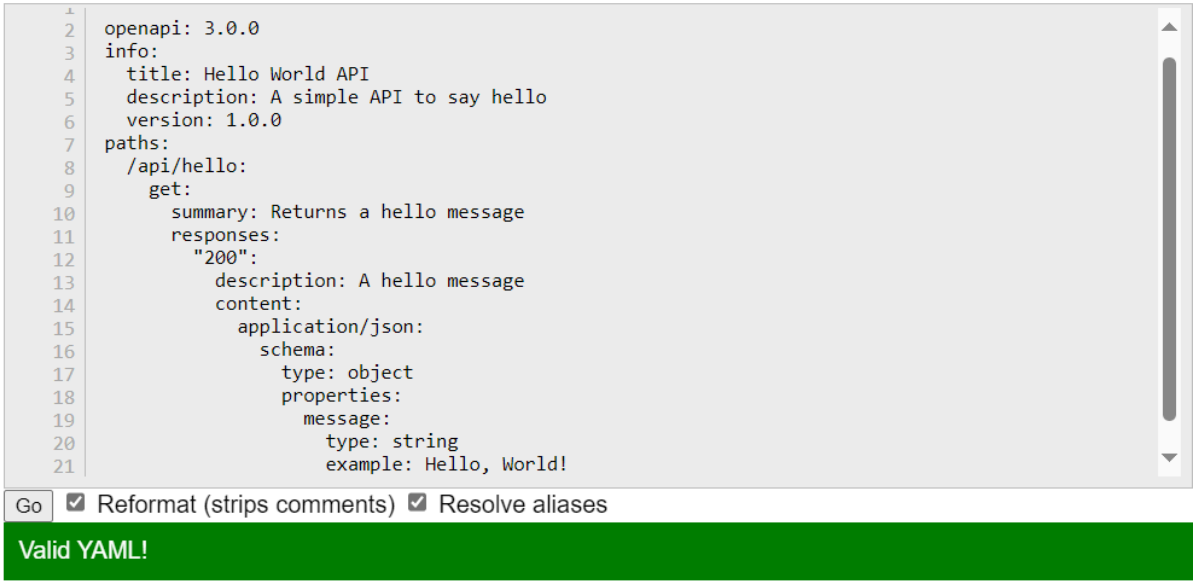
```
yernur_y@DESKTOP-2PVM7F3:~/api-manage$ pip install Flask
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: Flask in /home/yernur_y/.local/lib/python3.10/site-packages (2.0.3)
Requirement already satisfied: click>=7.1.2 in /home/yernur_y/.local/lib/python3.10/site-packages (from Flask) (8.1.7)
Requirement already satisfied: Werkzeug>=2.0 in /home/yernur_y/.local/lib/python3.10/site-packages (from Flask) (2.0.3)
Requirement already satisfied: itsdangerous>=2.0 in /home/yernur_y/.local/lib/python3.10/site-packages (from Flask) (2.2.0)
Requirement already satisfied: Jinja2>=3.0 in /home/yernur_y/.local/lib/python3.10/site-packages (from Flask) (3.1.4)
Requirement already satisfied: MarkupSafe>=2.0 in /home/yernur_y/.local/lib/python3.10/site-packages (from Jinja2>=3.0->Flask) (2.1.5)
yernur_y@DESKTOP-2PVM7F3:~/api-manage$
```

Step 3: Creating an OpenAPI Specification

I created an OpenAPI specification to define the API's structure. The specification was written in YAML style and detailed the API endpoint and response.

The openapi.yaml file appeared like this:

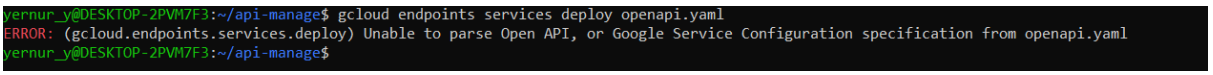




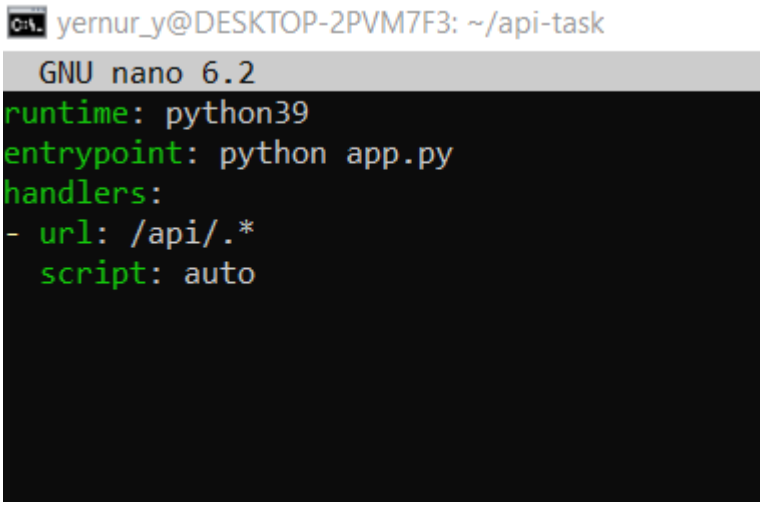
Step 4: Deploy the API to Google Cloud Endpoints

To deploy the API configuration in accordance with the OpenAPI definition, I executed the following command:

```
gcloud endpoints services deploy openapi.yaml
```

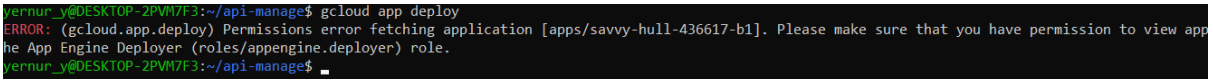


I developed an app.yaml file to specify how the Flask application should be deployed on Google App Engine. The file provided the required configurations for App Engine:



The deployment was done using the following command:

```
gcloud app deploy
```



To deploy, you need a billing account

Step 5: Test the API.

After successful deployment, I needed to obtain the API URL. By accessing this URL or using the curl command, I ensured that the /api/hello endpoint delivered the expected message:

```
curl https://savvy-hull-436617-b1.appspot.com/api/hello
```

The expected response should be as follows:

```
{"message": "Hello, World!"}
```

Exercise 2: Google Cloud Databases

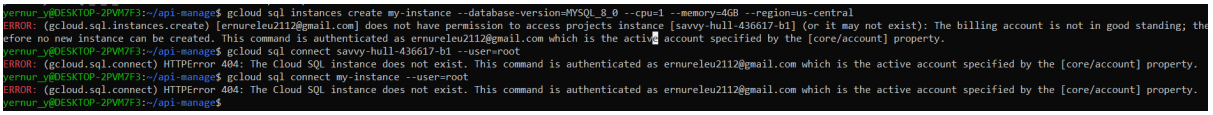
Step 1: Create a Cloud SQL Instance

1. I chose MySQL as the database type for this test. However, PostgreSQL and SQL Server were also offered as options.
2. I set up the SQL instance to allow public IP access, which is required to connect to the database from my local PC or other applications.

Step 2: Create a Database and Table

Using the MySQL command-line client, I connected to the Cloud SQL instance:

```
gcloud sql connect savvy-hull-436617-b1 --user=root
```



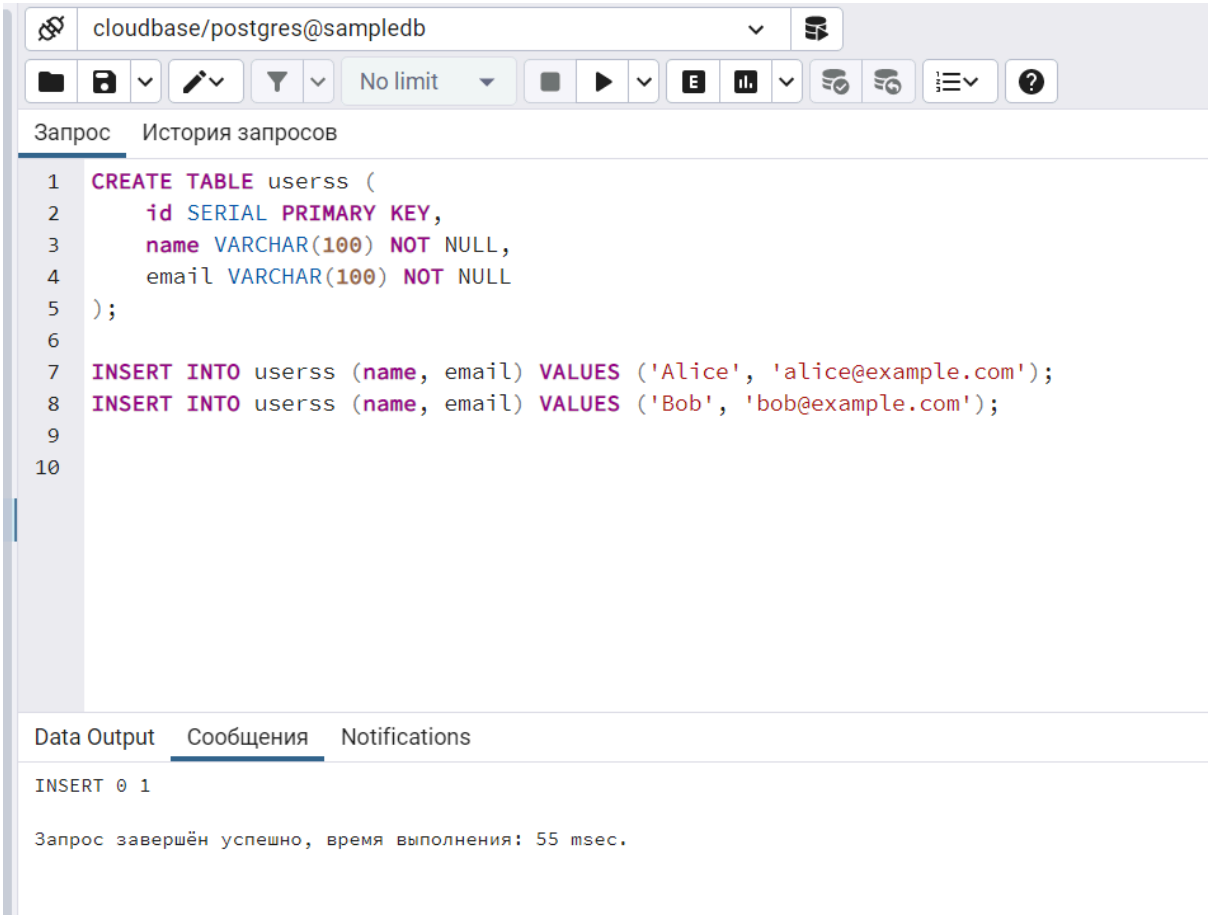
Once connected, I created a new database called `sample_db`:

```
To create a database, i need a billing account
CREATE DATABASE sample_db;
USE sample_db;
```

I then created a `users` table and inserted sample data:

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL
);

INSERT INTO users (name, email) VALUES ('Alice',
'alice@example.com');
INSERT INTO users (name, email) VALUES ('Bob', 'bob@example.com');
```



Step 3: Connect to the Database from a Python Application

I wrote a Python script to connect to the Cloud SQL instance using the `mysql-connector-python` library. Here's the content of the script:

yernur_y@DESKTOP-2PVM7F3: ~/api-manage

```
  GNU nano 6.2
import mysql.connector

cnx = mysql.connector.connect(
    user='yernur_y',
    password='qwerty1234',
    host='95.56.211.36',
    database='sample_db'
)
cursor = cnx.cursor()
cursor.execute('SELECT * FROM users')
for row in cursor:
    print(row)
cursor.close()
cnx.close()
```

Before running the script, I installed the required MySQL connector library:

```
pip install mysql-connector-python
```

Running the Python Script, I executed the Python script to verify the connection and retrieve data from the Cloud SQL instance:

```
python connect.py
```

The expected output was:

```
(1, 'Alice', 'alice@example.com')
(2, 'Bob', 'bob@example.com')
```

	id [PK] integer	name character varying (100)	email character varying (100)
1	1	Alice	alice@example.com
2	2	Bob	bob@example.com

Exercise 3: Integrating Machine Learning with Google Cloud

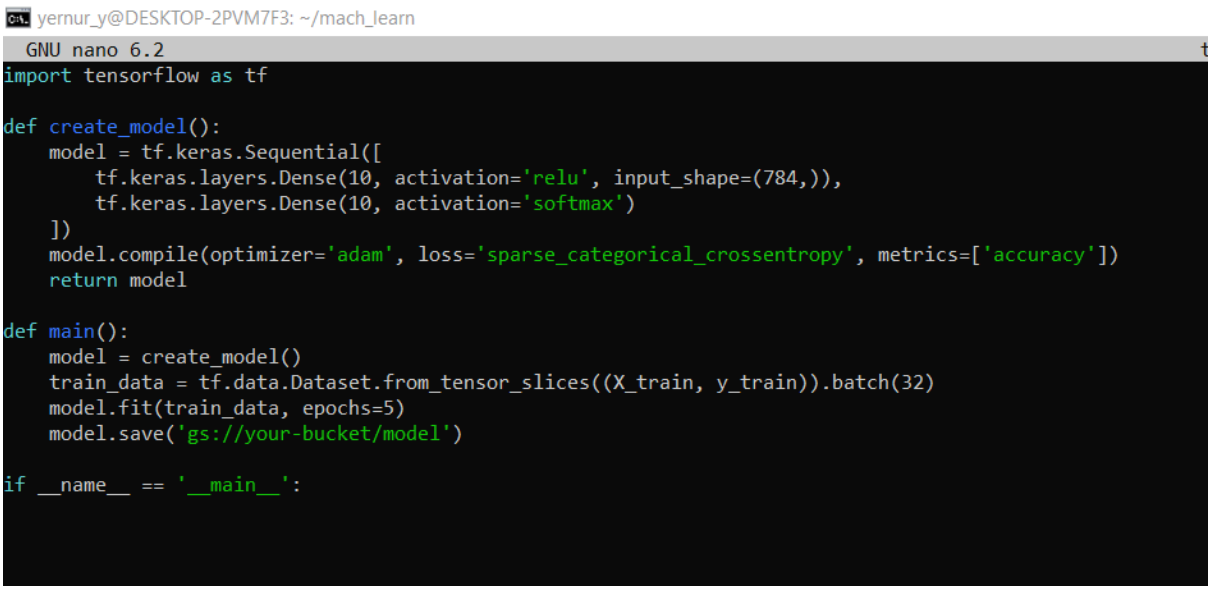
The objective of this experiment was to build and deploy a TensorFlow machine learning model on Google Cloud AI platform. This report describes the procedures followed to complete the work, the problems faced, and the outcomes obtained.

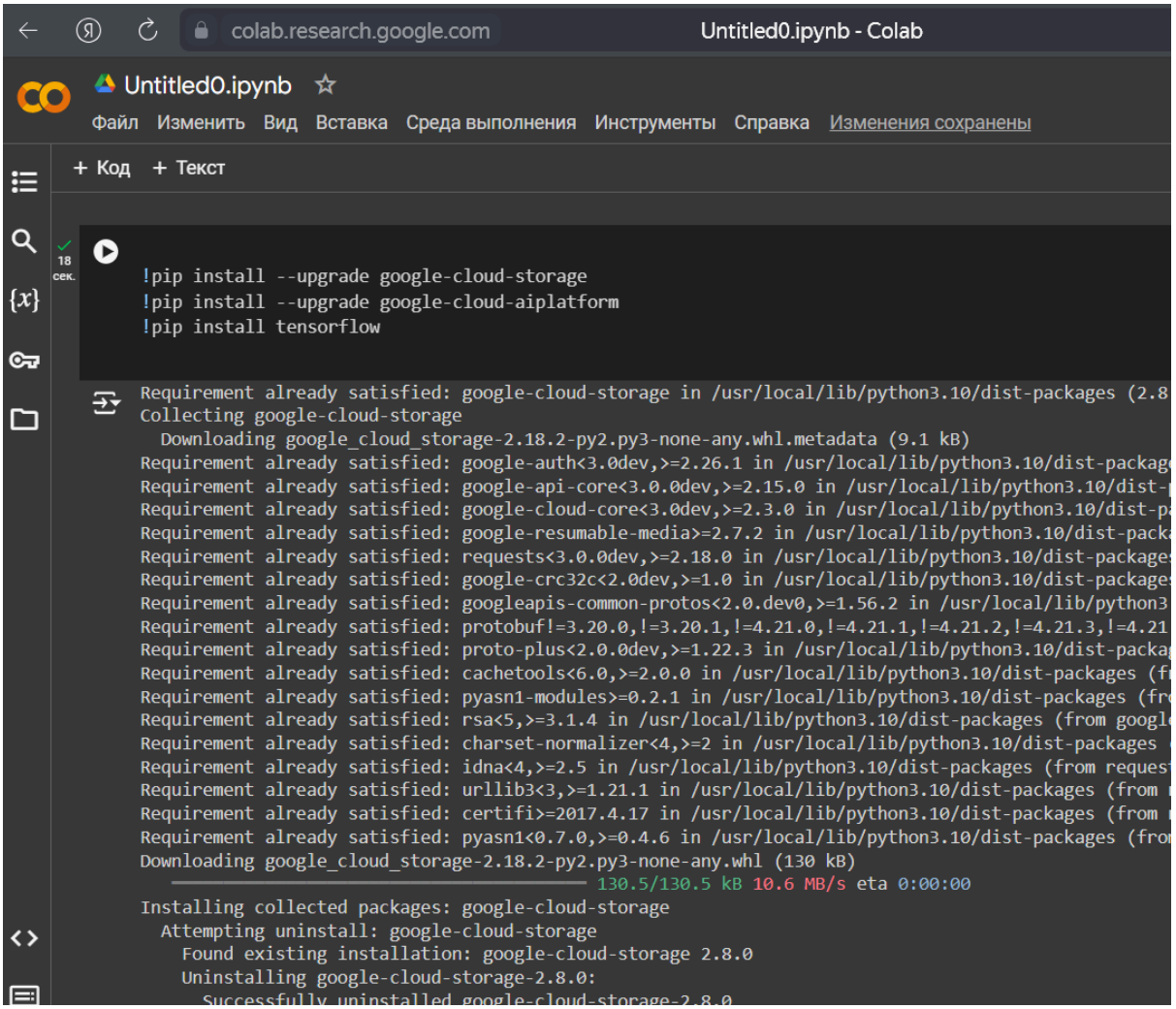
The setup procedure:

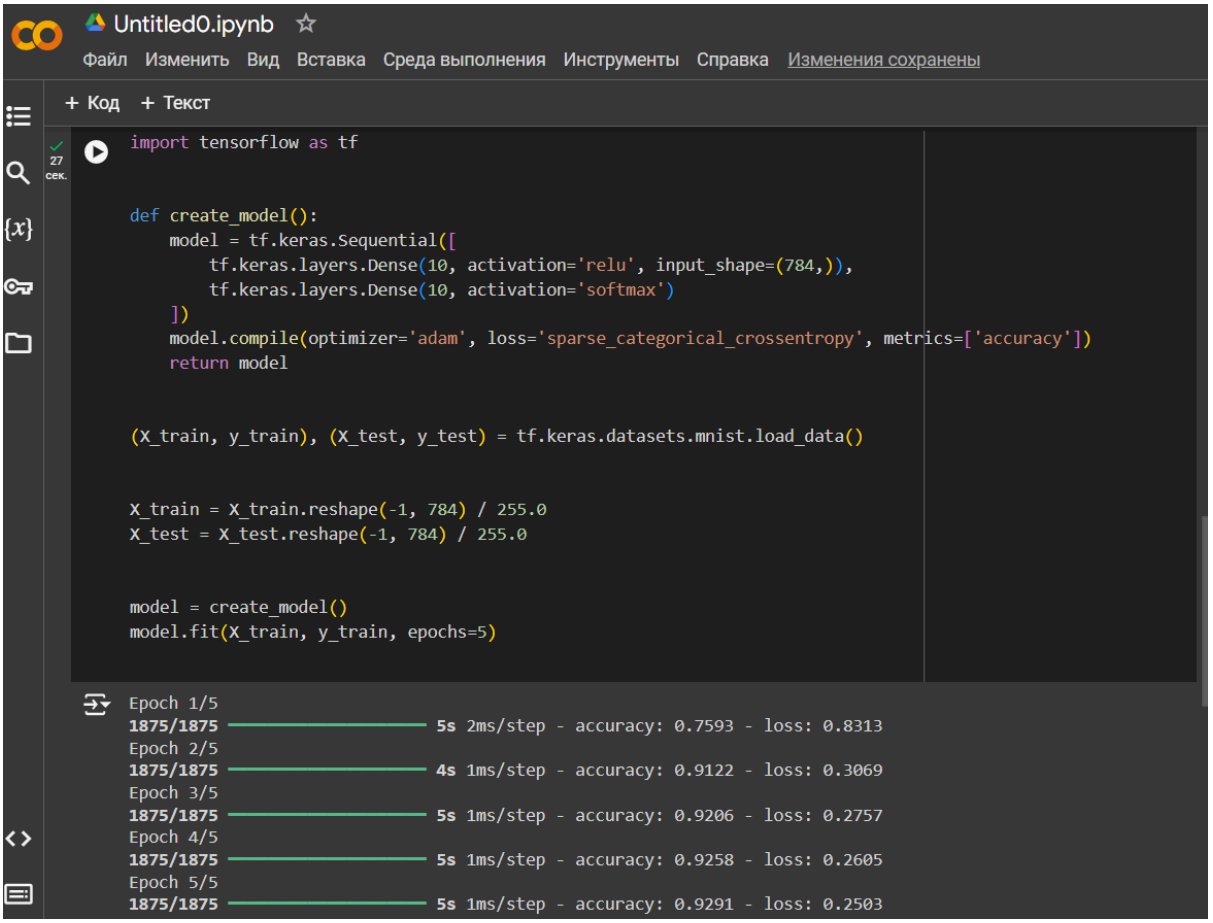
The initial step was to create the environment required to set up and develop the model. Because the Google Cloud AI platform required the activation of a paying account to utilize, I opted to use Google Colab, which provides free resources for completing a machine learning work. Google Ecolab provides access to the GPU and CPU, making it a perfect platform for such projects without the need to pay for resources.

What was done at the setup stage:

I made sure that all the necessary libraries are installed in Google Colab, such as tenzorflow, google-cloud-storage and google-cloud-aiplatform.



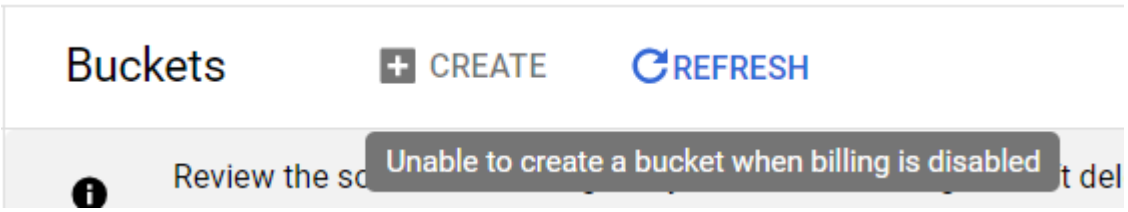

colab.research.google.com
Untitled0.ipynb - Colab
Untitled0.ipynb
Файл Изменить Вид Вставка Среда выполнения Инструменты Справка Изменения сохранены
+ Код + Текст
18 сек.
!pip install --upgrade google-cloud-storage
!pip install --upgrade google-cloud-aiplatform
!pip install tensorflow
Requirement already satisfied: google-cloud-storage in /usr/local/lib/python3.10/dist-packages (2.8
Collecting google-cloud-storage
Downloading google_cloud_storage-2.18.2-py2.py3-none-any.whl.metadata (9.1 kB)
Requirement already satisfied: google-auth<3.0dev,>=2.26.1 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: google-api-core<3.0.0dev,>=2.15.0 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: google-cloud-core<3.0dev,>=2.3.0 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: google-resumable-media>=2.7.2 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: requests<3.0.0dev,>=2.18.0 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: google-crc32c<2.0dev,>=1.0 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: googleapis-common-protos<2.0.dev0,>=1.56.2 in /usr/local/lib/python3
Requirement already satisfied: protobuf!=3.20.0,!=3.20.1,!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21
Requirement already satisfied: proto-plus<2.0.0dev,>=1.22.3 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (f
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (fr
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from googl
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from reques
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (fro
Downloading google_cloud_storage-2.18.2-py2.py3-none-any.whl (130 kB)
130.5/130.5 kB 10.6 MB/s eta 0:00:00
Installing collected packages: google-cloud-storage
Attempting uninstall: google-cloud-storage
Found existing installation: google-cloud-storage 2.8.0
Uninstalling google-cloud-storage-2.8.0:
Successfully uninstalled google-cloud-storage-2.8.0



Train the Model:

Model Training on Google Cloud AI Platform

The training script is then submitted to Google Cloud AI Platform, which automates the training process using cloud resources. This step is essential for handling larger datasets and more complex models that require significant computational power.



mand

```
gcloud ai custom-jobs create \
    --region=your-region \
    --display-name=ml-job \
    --python-package-uris=gs://your-bucket/train.py \
    --python-module=train \
```

```
--container-image-uri=gcr.io/cloud-aiplatform/training/tf-cpu.2-4:la
test
```

6. Deploy the Model:

Once the model has been trained, it is deployed on the Google Cloud AI Platform using an endpoint. This makes the model accessible for serving predictions through a REST API. Deployment on Google Cloud also allows version control, making it easier to manage updates or improvements to the model over time.

```
# Create a model resource
gcloud ai models create your-model --region=your-region

# Deploy a version of the model
gcloud ai versions create v1 \
    --model=your-model \
    --origin=gs://your-bucket/model \
    --runtime-version=2.7 \
    --python-version=3.8
```

---

7. Test the Model:

To verify that the model works as expected, a simple Python script (predict.py) was used to send data to the deployed model and retrieve predictions. The model's endpoint was accessed via Google Cloud's AI Platform Prediction Service. This script tests the deployed model by passing an instance of data and printing out the prediction result.

```
from google.cloud import aiplatform

def predict():
    client = aiplatform.gapic.PredictionServiceClient()
    endpoint = client.endpoint_path(project='your-project',
location='your-region', endpoint='your-endpoint-id')

    # Replace this with your input data
    instance = {'input': [/* your data here */]}

    # Make a prediction
```

```
    response = client.predict(endpoint=endpoint,
instances=[instance])
    print(response.predictions)

if __name__ == '__main__':
    predict()
```