Assignment 3, cloud app development

By Altair Kabdrakhmanov, 21B030829

Exercise 1: Managing APIs with Google Cloud Endpoints

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

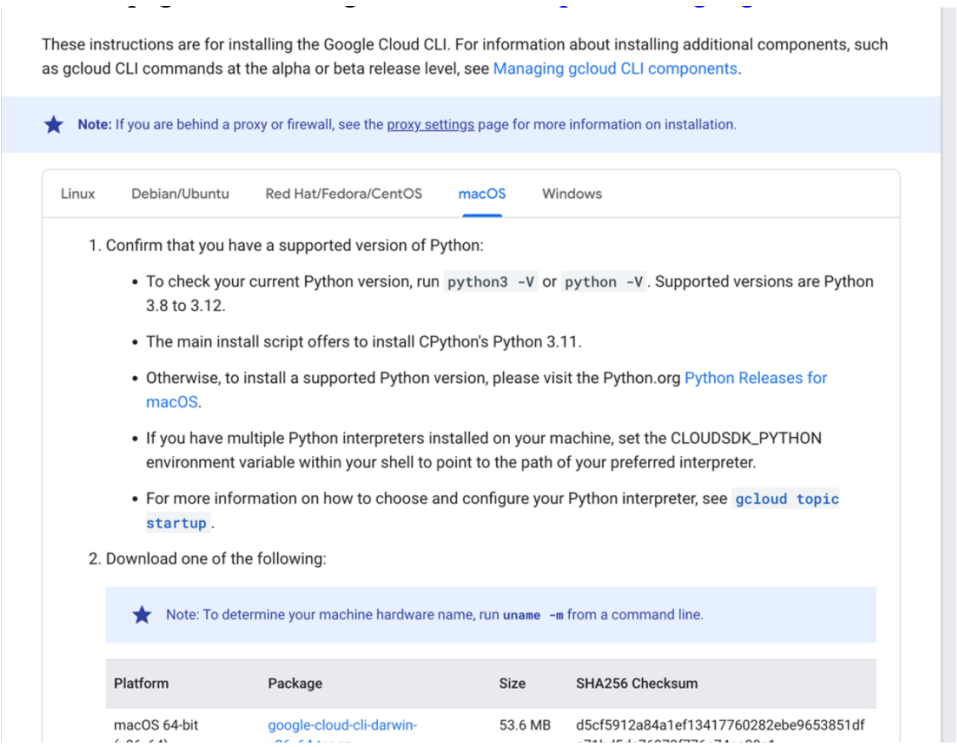
Instructions:

1. **Setup**:
   - Ensure you have a Google Cloud account.

     Answer: To be insure you have a Google Cloud account you need to go https://console.cloud.google.com and **try to authenticate**. After that you need to add you debit card to get Free Trail for 300 dollars **to try and start working** with Google Cloud for the first time.

   - Install the Google Cloud SDK and `gcloud` command-line tool.

     Answer: Go to this page **"Install the gcloud CLI"** - https://cloud.google.com/sdk/docs/install.



These instructions are for installing the Google Cloud CLI. For information about installing additional components, such as gcloud CLI commands at the alpha or beta release level, see Managing gcloud CLI components.

Note: If you are behind a proxy or firewall, see the proxy settings page for more information on installation.

Linux | Debian/Ubuntu | Red Hat/Fedora/CentOS | macOS | Windows

1. Confirm that you have a supported version of Python:
   - To check your current Python version, run `python3 -V` or `python -V`. Supported versions are Python 3.8 to 3.12.
   - The main install script offers to install CPython's Python 3.11.
   - Otherwise, to install a supported Python version, please visit the Python.org Python Releases for macOS.
   - If you have multiple Python interpreters installed on your machine, set the CLOUDSDK_PYTHON environment variable within your shell to point to the path of your preferred interpreter.
   - For more information on how to choose and configure your Python interpreter, see `gcloud topic startup`.
2. Download one of the following:

   Note: To determine your machine hardware name, run `uname -m` from a command line.

Platform	Package	Size	SHA256 Checksum
macOS 64-bit	google-cloud-cli-darwin-	53.6 MB	d5cf5912a84a1ef13417760282ebe9653851df

Select your **operating machine**. For my case it is MacOS. Check you current version of Python. It **must match** with requirements for Google Cloud SDK. Download your package. For my case it is Apple Silicon Extract this archive file on Home directory. To add Google Cloud SDK to your PATH. You need to use this command in root of folder where you extracted the archive. For my case it is HOME directory.
Execute this command: ./google-cloud-sdk/install.sh. And now you can run the gcloud auth login command and do authentication with your google cloud.

```
(base) kabdrakhman@MacBook-Pro-Altair Assignment-3 % gcloud auth login
Your browser has been opened to visit:

    https://accounts.google.com/o/oauth2/auth?response_type=code&client_id=32555940559.apps.googleusercontent.com&redirect_uri=http%3A%2F%2Flocalhost%3A8085%2F&scope=op
enid+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloud-platform+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fappengi
ne.admin+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fsqlservice.login+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcompute+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Faccounts
.reauth&state=Mh6o3LSHZ5WIdtA9nWT15sOVk64A2G&access_type=offline&code_challenge=0RLa4BMjXRTFbbRHi2bInZEW6BMHSNnuy96VpKfeZ2A&code_challenge_method=S256


You are now logged in as [altairkabdrakhmanov@gmail.com].
Your current project is [cloudappdev-midterm-ak-439206].  You can change this setting by running:
  $ gcloud config set project PROJECT_ID


Updates are available for some Google Cloud CLI components.  To install them,
please run:
  $ gcloud components update

(base) kabdrakhman@MacBook-Pro-Altair Assignment-3 %
```

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

     Answer: To create a new project use this command gcloud projects create --name="CloudAppDevAssignment3-AK"

```
PROBLEMS    DEBUG CONSOLE    OUTPUT    TERMINAL    PORTS                                                    zsh

(base) kabdrakhman@MacBook-Pro-Altair Assignment-3 % gcloud projects create --name="CloudAppDevAssignment3-AK"
No project ID provided.

Use [cloudappdevassignment3-ak] as project ID (Y/n)?  y

Create in progress for [https://cloudresourcemanager.googleapis.com/v1/projects/cloudappdevassignment3-ak].
Waiting for [operations/cp.4685267936194296477] to finish...done.
Enabling service [cloudapis.googleapis.com] on project [cloudappdevassignment3-ak]...
Operation "operations/acat.p2-1059330099607-a2926f75-68ee-4cd9-94b9-cd666b9a6027" finished successfully.
(base) kabdrakhman@MacBook-Pro-Altair Assignment-3 %
```

3. **Prepare the API**:
   - Create a simple REST API using Python Flask.

     Answer: For that I used simple Flask app with endpoint /api/hello that

return ***"Hello, World!"*** message

exercise1-endpoints > main.py

```python
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)
```

4. **Create an OpenAPI Specification**:
   - Create an `openapi.yaml` file to define your API.

     Answer: The important note is to write in the host in this way: project_id.appspot.com. As it says on official

- If you are using the `cloud.goog` domain, confirm that the value for the `host` field is in the following format, and that the project ID is correct:

```
API_NAME.endpoints.YOUR_PROJECT_ID.cloud.goog
```

- If you are using the `appspot.com` domain (supported for App Engine only), confirm that the `host` field is in the following format, and that the project ID is correct:

```
YOUR_PROJECT_ID.appspot.com
```

documentation. The openapi.yaml itself:

```yaml
exercise1-endpoints > openapi.yaml
swagger: '2.0'
info:
  title: Hello World API
  description: A simple API to say hello
  version: "1.0.0"
host: cloudappdevassignment3-ak.appspot.com
schemes:
  - https
paths:
  /api/hello:
    get:
      operationId: getHelloMessage
      summary: Returns a hello message
      responses:
        200:
          description: A hello message
          schema:
            type: object
            properties:
              message:
                type: string
                example: "Hello, World!"
```

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

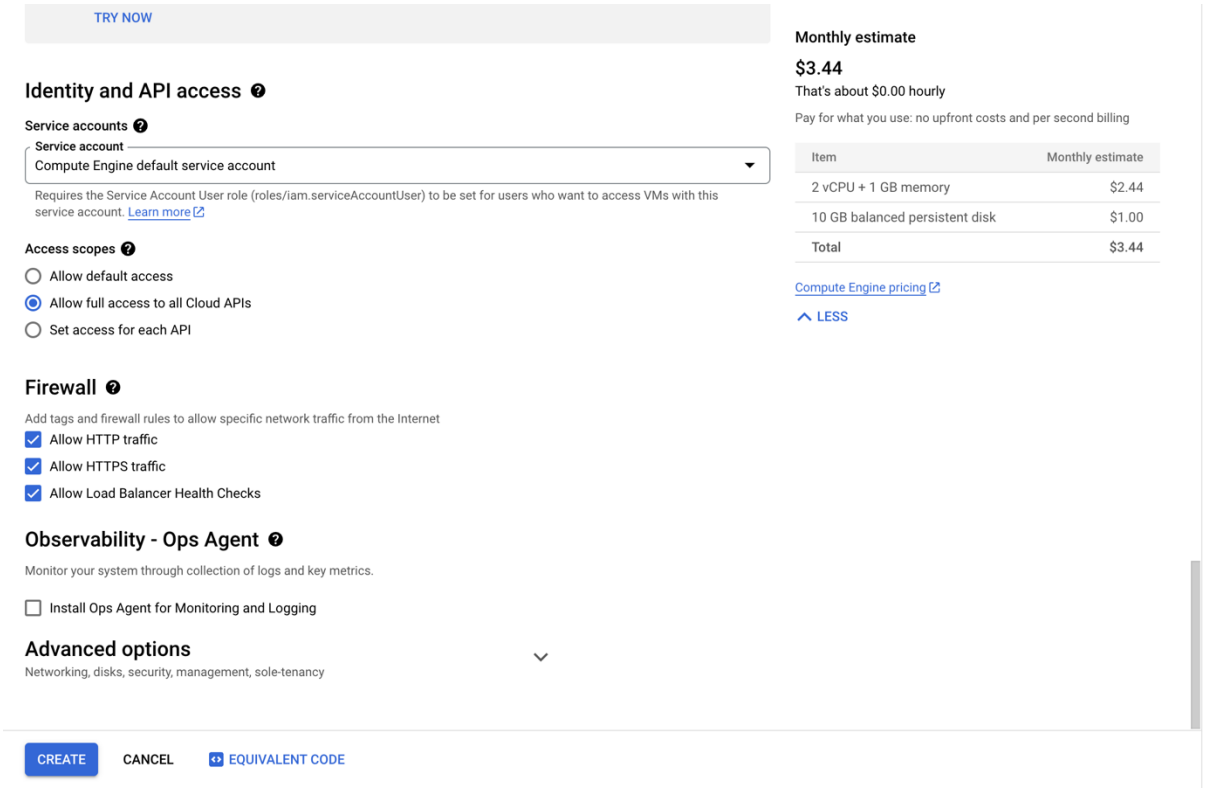
Answer:

Use this command to create new service with described endpoints `gcloud endpoints services deploy openapi.yaml`

```
(base) kabdrakhman@MacBook-Pro-Altair exercise1-endpoints % gcloud endpoints services deploy openapi.yaml
Waiting for async operation operations/services.cloudappdevassignment3-ak.appspot.com-0 to complete...
Waiting for async operation operations/serviceConfigs.cloudappdevassignment3-ak.appspot.com:620dc729-2504-4df3-a810-c4d4af18a320 to complete...
Operation finished successfully. The following command can describe the Operation details:
 gcloud endpoints operations describe operations/serviceConfigs.cloudappdevassignment3-ak.appspot.com:620dc729-2504-4df3-a810-c4d4af18a320

Waiting for async operation operations/rollouts.cloudappdevassignment3-ak.appspot.com:6fa4e660-c2ef-46b9-bebf-49f9bb2f4328 to complete...
Operation finished successfully. The following command can describe the Operation details:
 gcloud endpoints operations describe operations/rollouts.cloudappdevassignment3-ak.appspot.com:6fa4e660-c2ef-46b9-bebf-49f9bb2f4328

Enabling service [cloudappdevassignment3-ak.appspot.com] on project [cloudappdevassignment3-ak]...
Operation "operations/acat.p2-1059330099607-1af812f0-76b5-43fd-9cff-273a6fb96d1b" finished successfully.


Service Configuration [2024-11-03r0] uploaded for service [cloudappdevassignment3-ak.appspot.com]

To manage your API, go to: https://console.cloud.google.com/endpoints/api/cloudappdevassignment3-ak.appspot.com/overview?project=cloudappdevassignment3-ak
(base) kabdrakhman@MacBook-Pro-Altair exercise1-endpoints %
```

To deploy our application, we need to define requirements.txt that containes all dependencies to our application. Do this with this command: `pip freeze > requirements.txt`. Important note is to enable Compute Engine API and create VM that will operate our application.

Here is a good to use e2-micro because it cost cheap and allow http/https traffic to allow external services communicate with our VM and application itself.

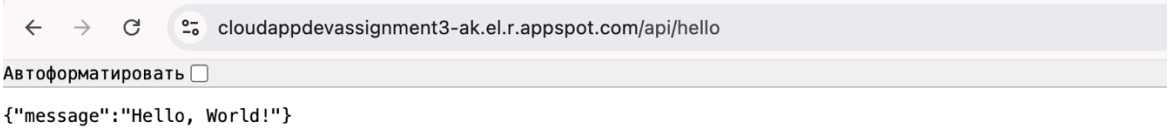


gcloud use service account to our app and it doesn't have permission to add files to cloud bucket. For resolve this problem you need to **Google Cloud Console**: https://console.cloud.google.com. Find **bucket** in search panel, go to the **Permissions** tab. Click **add** and assign the **storage object admin** role with read and write access. After that you can deploy with this command `gcloud app deploy` and the result:

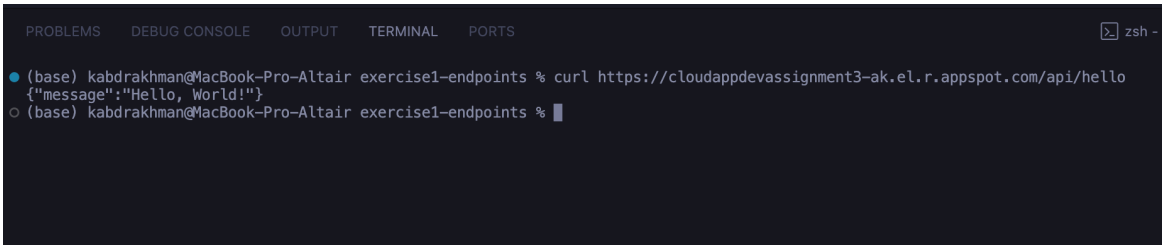
```
descriptor:                  [/Users/kabdrakhman/Documents/KBTU/7th semester/Cloud-Application-Development/Assignment-3/exercise1-endpoints/app.yaml]
source:                      [/Users/kabdrakhman/Documents/KBTU/7th semester/Cloud-Application-Development/Assignment-3/exercise1-endpoints]
target project:              [cloudappdevassignment3-ak]
target service:              [default]
target version:              [20241103t111822]
target url:                  [https://cloudappdevassignment3-ak.el.r.appspot.com]
target service account:      [cloudappdevassignment3-ak@appspot.gserviceaccount.com]


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

Beginning deployment of service [default]...
╔═══════════════════════════════════════════════════════════╗
╠═ Uploading 0 files to Google Cloud Storage                ═╣
╚═══════════════════════════════════════════════════════════╝
File upload done.
Updating service [default]...done.
Setting traffic split for service [default]...done.
Deployed service [default] to [https://cloudappdevassignment3-ak.el.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. **Test the API**:

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

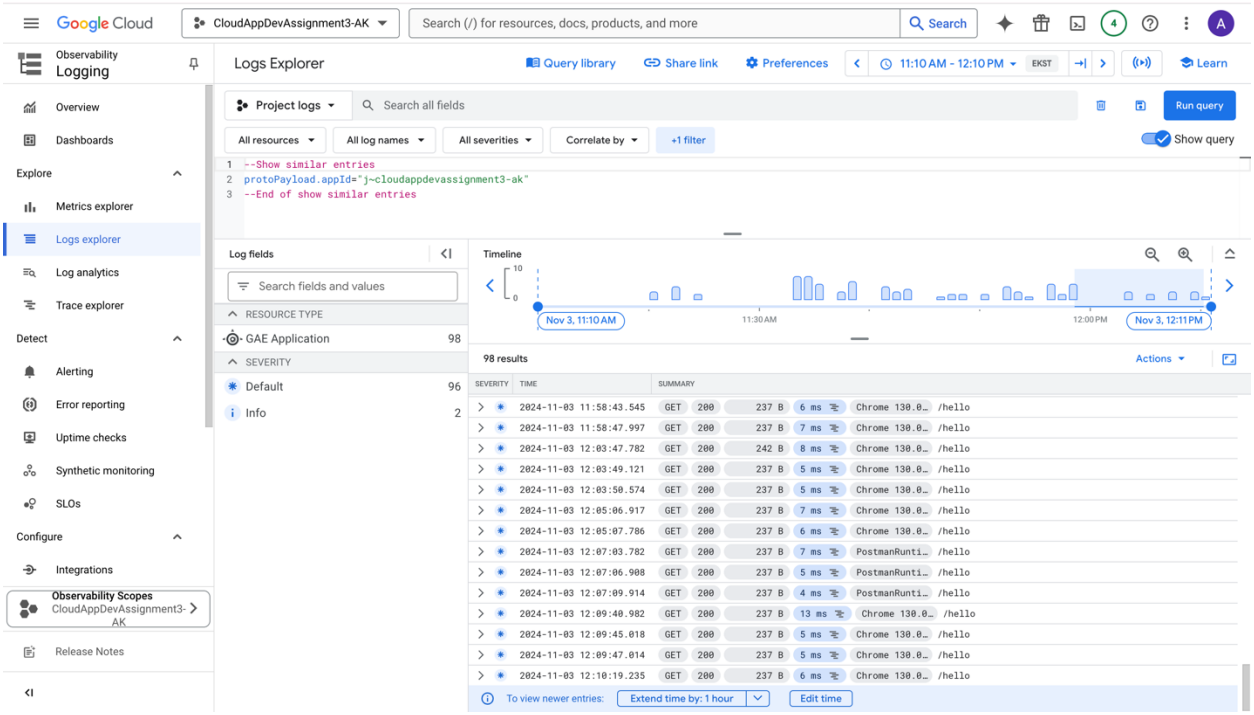


  curl response:



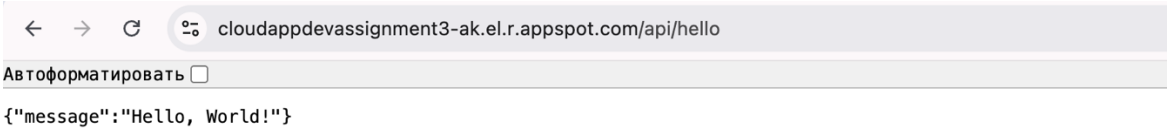
Deliverables:

- A deployed API on Google Cloud Endpoints.
  Answer:



- A screenshot of a successful API call response.

Answer:



```
PROBLEMS    DEBUG CONSOLE    OUTPUT    TERMINAL    PORTS                                                    zsh -

● (base) kabdrakhman@MacBook-Pro-Altair exercise1-endpoints % curl https://cloudappdevassignment3-ak.el.r.appspot.com/api/hello
  {"message":"Hello, World!"}
○ (base) kabdrakhman@MacBook-Pro-Altair exercise1-endpoints %
```

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.

      Answer: To be insure you have a Google Cloud account you need to go https://console.cloud.google.com and **try to authenticate**. After that you need to add you debit card to get Free Trail for 300 dollars **to try and start working** with Google Cloud for the first time.

    - Install the Google Cloud SDK.

      Answer: Go to this page **"Install the gcloud CLI"** - https://cloud.google.com/sdk/docs/install.

These instructions are for installing the Google Cloud CLI. For information about installing additional components, such as gcloud CLI commands at the alpha or beta release level, see Managing gcloud CLI components.

Note: If you are behind a proxy or firewall, see the proxy settings page for more information on installation.

Linux | Debian/Ubuntu | Red Hat/Fedora/CentOS | macOS | Windows

1. Confirm that you have a supported version of Python:
   - To check your current Python version, run `python3 -V` or `python -V`. Supported versions are Python 3.8 to 3.12.
   - The main install script offers to install CPython's Python 3.11.
   - Otherwise, to install a supported Python version, please visit the Python.org Python Releases for macOS.
   - If you have multiple Python interpreters installed on your machine, set the CLOUDSDK_PYTHON environment variable within your shell to point to the path of your preferred interpreter.
   - For more information on how to choose and configure your Python interpreter, see `gcloud topic startup`.
2. Download one of the following:

   Note: To determine your machine hardware name, run `uname -m` from a command line.

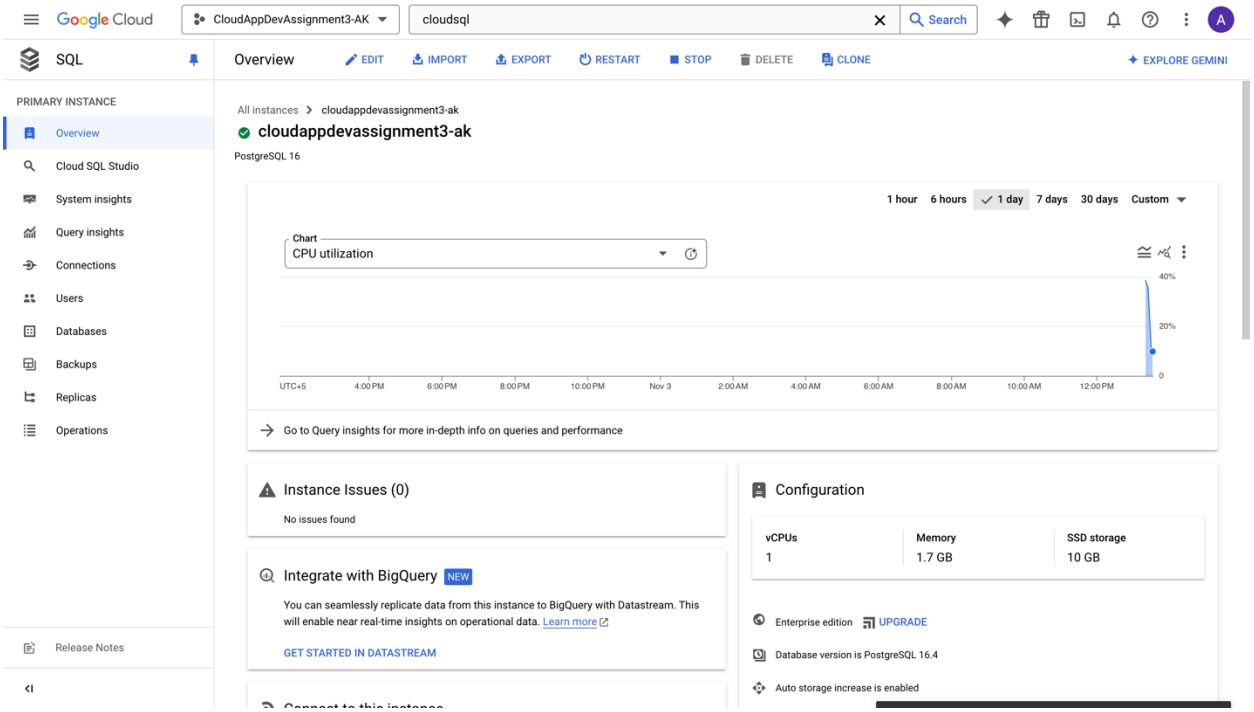
Platform	Package	Size	SHA256 Checksum
macOS 64-bit	google-cloud-cli-darwin-	53.6 MB	d5cf5912a84a1ef13417760282ebe9653851df

Select your **operating machine**. For my case it is MacOS. Check you current version of Python. It **must match** with requirements for Google Cloud SDK. Download your package. For my case it is Apple Silicon Extract this archive file on Home directory. To add Google Cloud SDK to your PATH. You need to use this command in root of folder where you extracted the archive. For my case it is HOME directory.
Execute this command: ./google-cloud-sdk/install.sh. And now you can run the gcloud auth login command and do authentication with your google cloud.

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.).

Answer:

Configured CloudSQL with PostgreSQL16



3. **Create a Database and Table**:
   - Connect to your Cloud SQL instance using the Cloud SQL client or `mysql` command-line tool.

   Answer:

   Before connecting to CloudSQL instance it needs to be connected to CloudSQL Server with the following command: `gcloud sql connect cloudappdevassignment3-ak --user=postgres`
   As result we are in gcloud postgres console

- Create a new database and a table with sample data.

Answer:

Execute this command to create new Database

```
CREATE DATABASE sample_db;
```

Execute this command to select database

```
\c sample_db;
```

Execute this command to create table in PostgreSQL

```
CREATE TABLE users (
    id SERIAL 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');
```

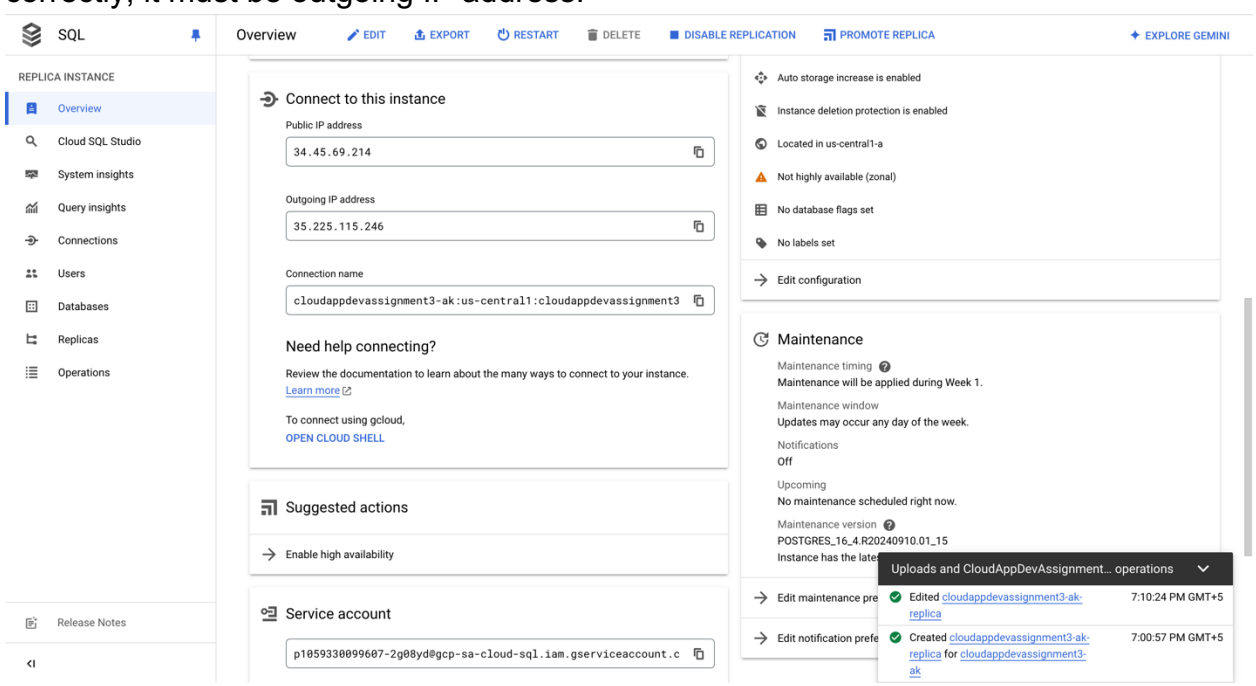
The result:

```
CREATE TABLE
sample_db=> INSERT INTO users (name, email) VALUES ('Alice', 'alice@example.com');
INSERT INTO users (name, email) VALUES ('Bob', 'bob@example.com');
INSERT 0 1
INSERT 0 1
sample_db=> select * from users
sample_db-> ;
 id | name  |       email
----+-------+-------------------
  1 | Alice | alice@example.com
  2 | Bob   | bob@example.com
(2 rows)

sample_db=> select * from users;
 id | name  |       email
----+-------+-------------------
  1 | Alice | alice@example.com
  2 | Bob   | bob@example.com
(2 rows)

sample_db=>
```

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

Answer:

Write this code to connect to CloudSQL. Important note it is to write host correctly, it must be outgoing IP address.



The result:

connect.py × cloud-sql-proxy

Search Assignment-3 — connect.py — Assignment-3

exercise2-databases > connect.py

```python
import psycopg2

# Establish the connection
with psycopg2.connect(
    user='postgres',
    password='12345678',
    database='sample_db',
    host='35.225.115.246',
    port=5432
) as conn:

    # Create a cursor and execute the query
    with conn.cursor() as cursor:
        cursor.execute('SELECT * FROM users')

        # Fetch and print all rows
        rows = cursor.fetchall()
        for row in rows:
            print(row)
```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

zsh - exercise2-databases

```
(venv) (base) kabdrakhman@MacBook-Pro-Altair exercise2-databases % python3 connect.py
(1, 'Alice', 'alice@example.com')
(2, 'Bob', 'bob@example.com')
(venv) (base) kabdrakhman@MacBook-Pro-Altair exercise2-databases %
```

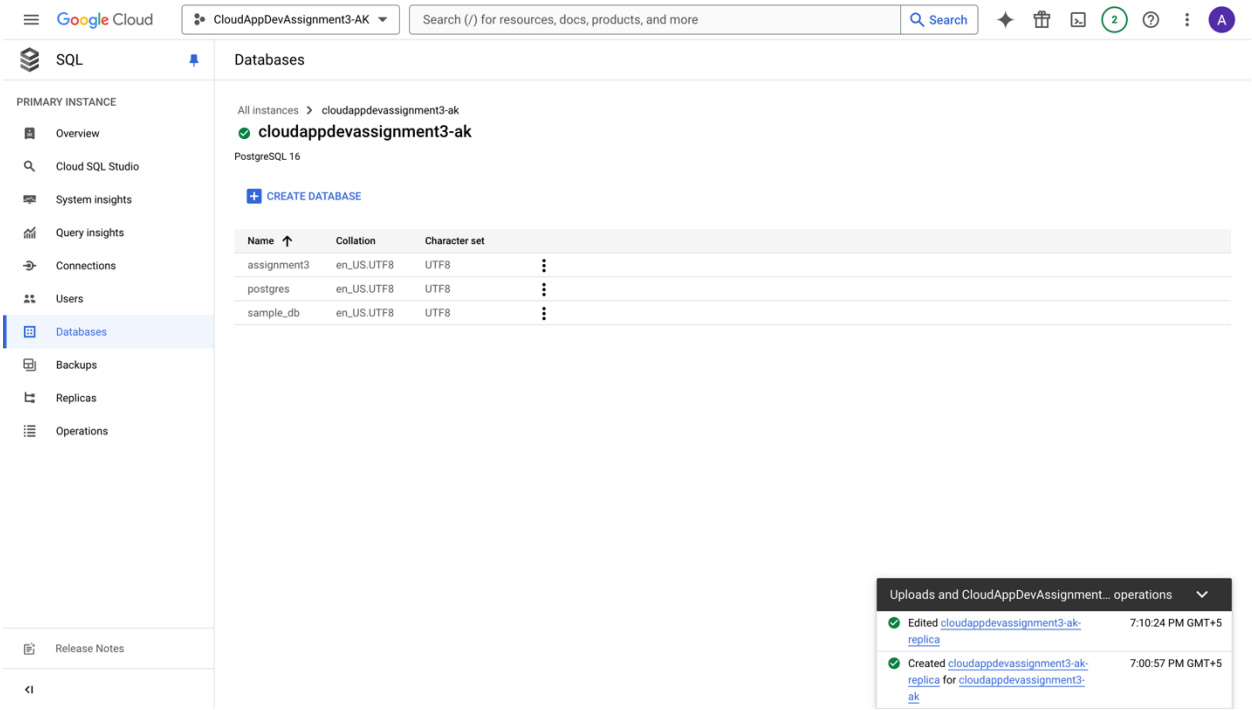
5. **Run the Connection Code**:

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

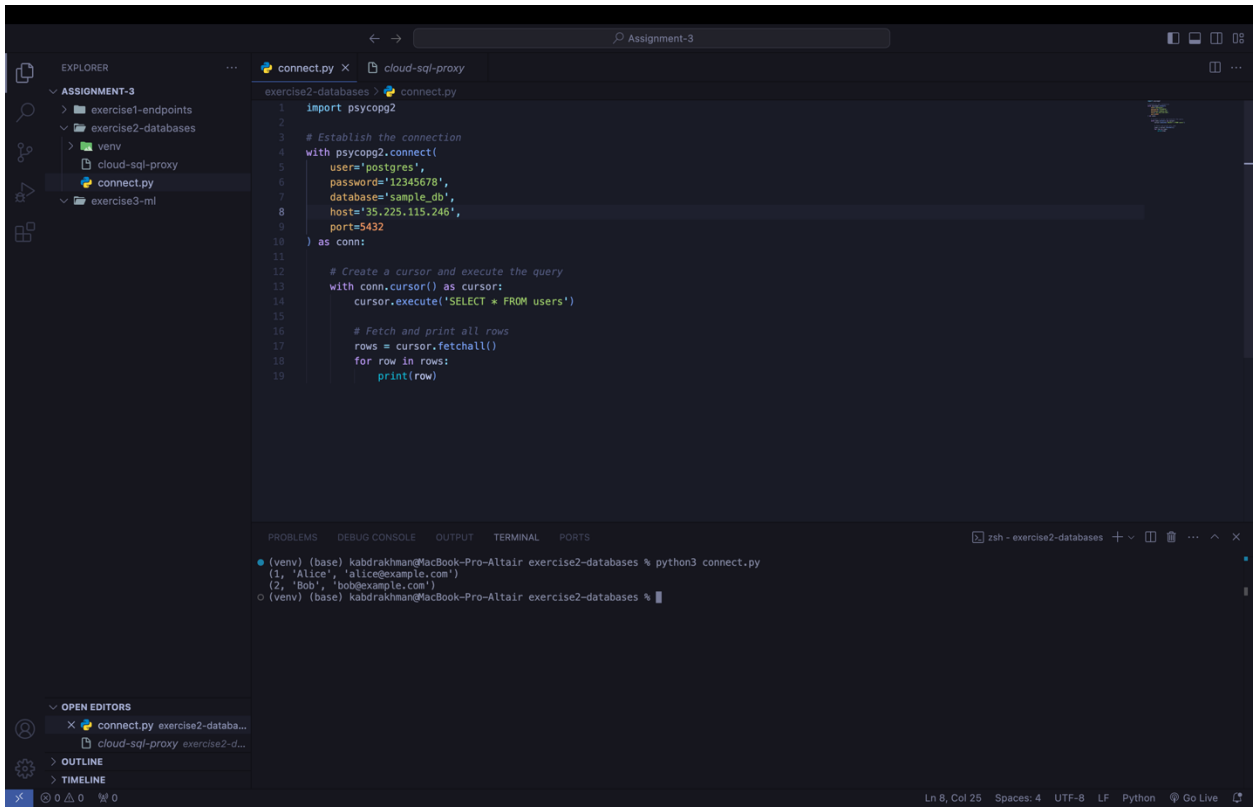
Deliverables:

- A working Cloud SQL database with sample data.

My Google Cloud SQL Databases list



- A Python script that successfully connects to and queries the database.
  The python script that extract data from Google Cloud SQL (PostgreSQL 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.

     Answer: To be insure you have a Google Cloud account you need to go https://console.cloud.google.com and **try to authenticate**. After that you need to add you debit card to get Free Trail for 300 dollars **to try and start working** with Google Cloud for the first time.

   - Install the Google Cloud SDK and TensorFlow.

     Answer:
     Google Cloud SDK installation

Go to this page **“Install the gcloud CLI”** - https://cloud.google.com/sdk/docs/install.

These instructions are for installing the Google Cloud CLI. For information about installing additional components, such as gcloud CLI commands at the alpha or beta release level, see Managing gcloud CLI components.

Note: If you are behind a proxy or firewall, see the proxy settings page for more information on installation.

Linux | Debian/Ubuntu | Red Hat/Fedora/CentOS | macOS | Windows

1. Confirm that you have a supported version of Python:
   - To check your current Python version, run `python3 -V` or `python -V`. Supported versions are Python 3.8 to 3.12.
   - The main install script offers to install CPython's Python 3.11.
   - Otherwise, to install a supported Python version, please visit the Python.org Python Releases for macOS.
   - If you have multiple Python interpreters installed on your machine, set the CLOUDSDK_PYTHON environment variable within your shell to point to the path of your preferred interpreter.
   - For more information on how to choose and configure your Python interpreter, see `gcloud topic startup`.
2. Download one of the following:

Note: To determine your machine hardware name, run `uname -m` from a command line.

Platform	Package	Size	SHA256 Checksum
macOS 64-bit	google-cloud-cli-darwin-	53.6 MB	d5cf5912a84a1ef13417760282ebe9653851df

Select your **operating machine**. For my case it is MacOS. Check you current version of Python. It **must match** with requirements for Google Cloud SDK. Download your package. For my case it is Apple Silicon Extract this archive file on Home directory. To add Google Cloud SDK to your PATH. You need to use this command in root of folder where you extracted the archive. For my case it is HOME directory.
Execute this command: ./google-cloud-sdk/install.sh. And now you can run the gcloud auth login command and do authentication with your google cloud.

Tensorflow installation

To install tensorflow it must align with correct python version. It must be between 3.7 and 3.11 versions. I have python with 3.13 version and for my case I created virtual environment with 3.11 version with this command: `python3.11 -m venv tf_env` and executed this command to activate this environment `source tf_env/bin/activate`. Now you can install

TensorFlow with this command `pip install tensorflow`.

```
(base) kabdrakhman@MacBook-Pro-Altair exercise3-ml % python3.11 -m venv tf_env
(base) kabdrakhman@MacBook-Pro-Altair exercise3-ml % source tf_env/bin/activate
(tf_env) (base) kabdrakhman@MacBook-Pro-Altair exercise3-ml % pip install tensorflow
Collecting tensorflow
  Downloading tensorflow-2.18.0-cp311-cp311-macosx_12_0_arm64.whl.metadata (4.0 kB)
Collecting absl-py>=1.0.0 (from tensorflow)
  Using cached absl_py-2.1.0-py3-none-any.whl.metadata (2.3 kB)
Collecting astunparse>=1.6.0 (from tensorflow)
  Downloading astunparse-1.6.3-py2.py3-none-any.whl.metadata (4.4 kB)
Collecting flatbuffers>=24.3.25 (from tensorflow)
  Downloading flatbuffers-24.3.25-py2.py3-none-any.whl.metadata (850 bytes)
Collecting gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 (from tensorflow)
  Downloading gast-0.6.0-py3-none-any.whl.metadata (1.3 kB)
Collecting google-pasta>=0.1.1 (from tensorflow)
  Downloading google_pasta-0.2.0-py3-none-any.whl.metadata (814 bytes)
Collecting libclang>=13.0.0 (from tensorflow)
  Downloading libclang-18.1.1-1-py2.py3-none-macosx_11_0_arm64.whl.metadata (5.2 kB)
Collecting opt-einsum>=2.3.2 (from tensorflow)
  Downloading opt_einsum-3.4.0-py3-none-any.whl.metadata (6.3 kB)
Collecting packaging (from tensorflow)
Ln 1, Col 1   Spaces: 4   UTF-8   LF   Plain Text
```

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

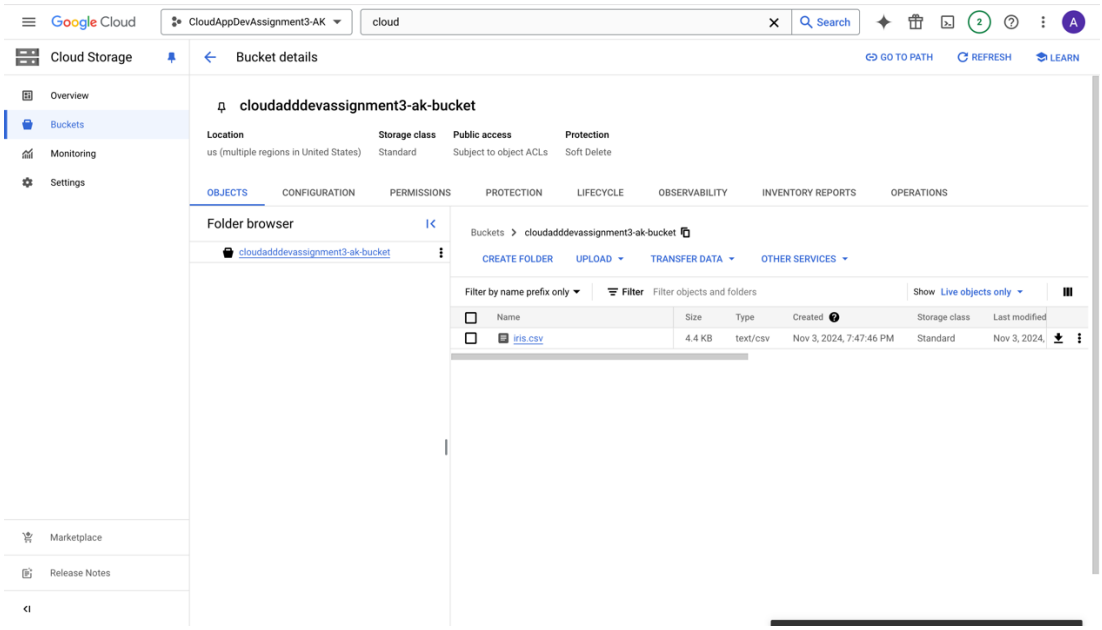
     Answer: executed this command to crate Cloud Storage bucket `gsutil mb gs://cloudadddevassignment3-ak-bucket`.

```
PROBLEMS   DEBUG CONSOLE   OUTPUT   TERMINAL   PORTS

(tf_env) (base) kabdrakhman@MacBook-Pro-Altair exercise3-ml % gsutil mb gs://cloudadddevassignment3-ak-bucket
Creating gs://cloudadddevassignment3-ak-bucket/...
(tf_env) (base) kabdrakhman@MacBook-Pro-Altair exercise3-ml %
```

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

     Answer:
     I used iris dataset from this site: https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data and uploaded to gcloud bucket with the following command: `gsutil cp iris.csv gs://cloudadddevassignment3-ak-bucket`.

The data in gcloud bucket:



4. **Create a Training Script**:

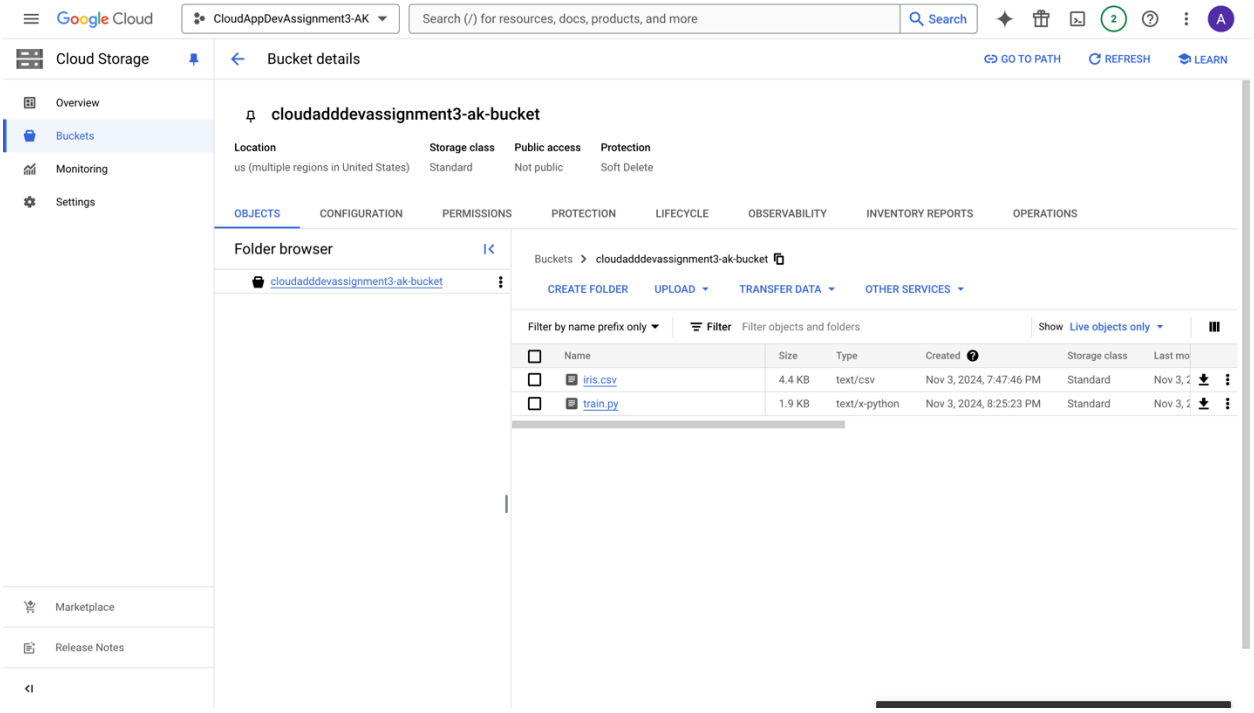
My training script `train.py`

```
import pandas as pd
from google.cloud import storage
from io import StringIO
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Initialize a client for Google Cloud Storage
client = storage.Client()

# Specify the bucket and blob (file) you want to access
bucket_name = 'cloudadddevassignment3-ak-bucket'
blob_name = 'iris.csv'
bucket = client.bucket(bucket_name)
blob = bucket.blob(blob_name)

# Download the CSV file from GCS as a string
data = blob.download_as_text()

# Load the data into a DataFrame with correct column names
column_names = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
'species']
df = pd.read_csv(StringIO(data), header=None, names=column_names)
```

```python
# Print the column names to check them
print("Columns in the DataFrame:", df.columns.tolist())

# Preview the dataset
print(df.head())

# Separate features and target variable
X = df.drop(columns=['species'])  # Drop the species column
y = df['species']  # This is the target variable

# Encode the target labels
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2,
random_state=42)

# Build a simple neural network model
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(10, activation='relu', input_shape=(X_train.shape[1],)),
    tf.keras.layers.Dense(10, activation='relu'),
    tf.keras.layers.Dense(len(label_encoder.classes_), activation='softmax')
])

# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=100, batch_size=5, verbose=1)

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Test Accuracy: {accuracy:.2f}')
```

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

Before submitting trainig job we need to add train.py to gcloud bucket with this command: `gsutil cp train.py gs://cloudadddevassignment3-ak-bucket`



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

Use the following command to deploy our training model:

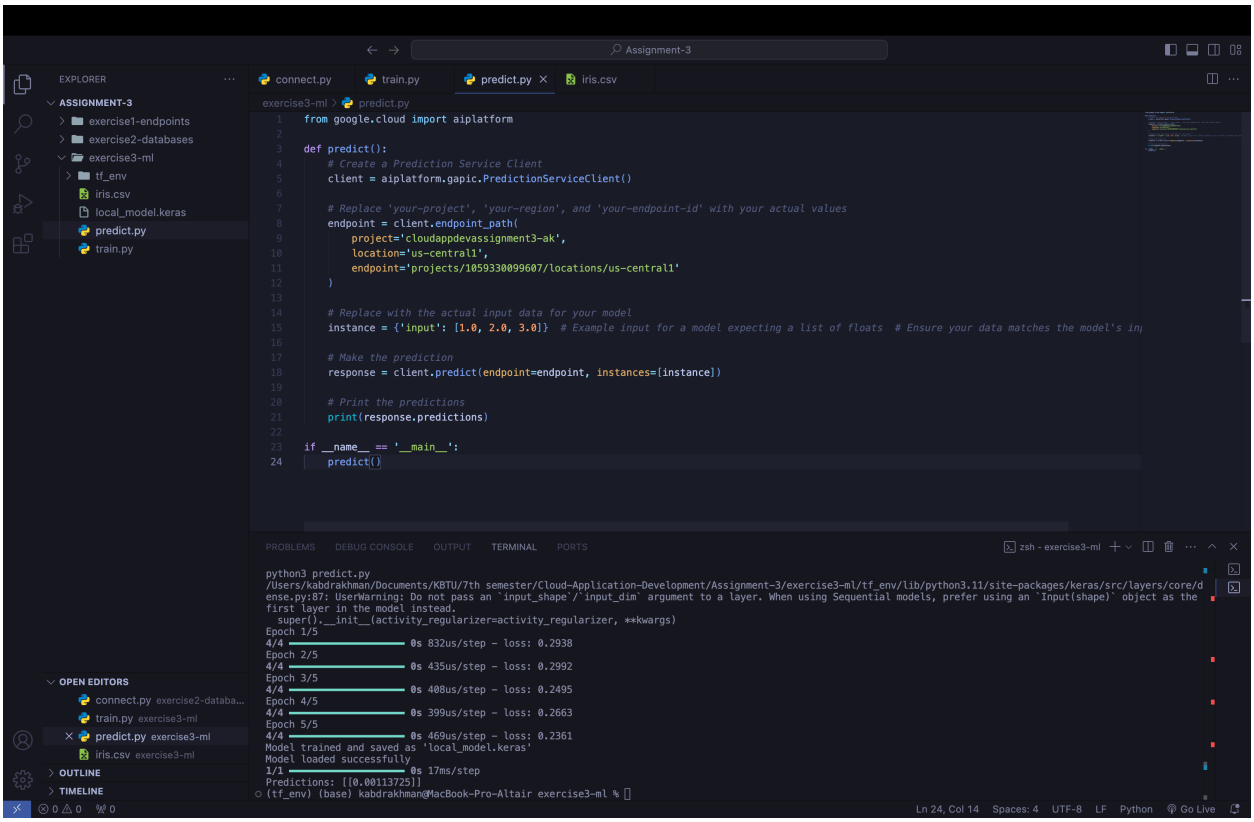
```
gcloud ai custom-jobs create \
  --region=us-central1 \
  --display-name=my-ml-job \
  --python-package-uris=gs://cloudadddevassignment3-ak-bucket/train.py \
  --worker-pool-spec=machine-type=n1-standard-4,replica-count=1,executor-image-uri=gcr.io/cloud-aiplatform/training/tf-cpu.2-4:latest,python-module=train
```

```
exercise3-ml — gcloud.py ai custom-jobs stream-logs projects/1059330099607/locations/us-central1/customJobs/949618884053303296 — 150×40
(tf_env) (base) kabdrakhman@MacBook-Pro-Altair exercise3-ml % gcloud ai custom-jobs create \
  --region=us-central1 \
  --display-name=my-ml-job \
  --python-package-uris=gs://cloudadddevassignment3-ak-bucket/train.py \
  --worker-pool-spec=machine-type=n1-standard-4,replica-count=1,executor-image-uri=gcr.io/cloud-aiplatform/training/tf-cpu.2-4:latest,python-module=tr
ain
Using endpoint [https://us-central1-aiplatform.googleapis.com/]
API [aiplatform.googleapis.com] not enabled on project [cloudappdevassignment3-ak]. Would you like to enable and retry (this will take a few minutes)?
 (y/N)?  y

Enabling service [aiplatform.googleapis.com] on project [cloudappdevassignment3-ak]...
Operation "operations/acat.p2-1059330099607-a940f12e-a928-40a6-8d5f-f3dcb1c6fa39" finished successfully.
CustomJob [projects/1059330099607/locations/us-central1/customJobs/949618884053303296] is submitted successfully.

Your job is still active. You may view the status of your job with the command

  $ gcloud ai custom-jobs describe projects/1059330099607/locations/us-central1/customJobs/949618884053303296

or continue streaming the logs with the command

  $ gcloud ai custom-jobs stream-logs projects/1059330099607/locations/us-central1/customJobs/949618884053303296
(tf_env) (base) kabdrakhman@MacBook-Pro-Altair exercise3-ml %  gcloud ai custom-jobs describe projects/1059330099607/locations/us-central1/customJobs/
949618884053303296
Using endpoint [https://us-central1-aiplatform.googleapis.com/]
createTime: '2024-11-03T15:27:48.890406Z'
displayName: my-ml-job
jobSpec:
  workerPoolSpecs:
  - diskSpec:
      bootDiskSizeGb: 100
      bootDiskType: pd-ssd
    machineSpec:
      machineType: n1-standard-4
    pythonPackageSpec:
      executorImageUri: gcr.io/cloud-aiplatform/training/tf-cpu.2-4:latest
      packageUris:
      - gs://cloudadddevassignment3-ak-bucket/train.py
      pythonModule: train
    replicaCount: '1'
name: projects/1059330099607/locations/us-central1/customJobs/949618884053303296
```

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

```python
from google.cloud import aiplatform

def predict():
    # Create a Prediction Service Client
    client = aiplatform.gapic.PredictionServiceClient()

    # Replace 'your-project', 'your-region', and 'your-endpoint-id' with your actual
values
    endpoint = client.endpoint_path(
        project='cloudappdevassignment3-ak',
        location='us-central1',
        endpoint='projects/1059330099607/locations/us-central1'
    )

    # Replace with the actual input data for your model
    instance = {'input': [1.0, 2.0, 3.0]}  # Example input for a model expecting a
list of floats  # Ensure your data matches the model's input format

    # Make the prediction
    response = client.predict(endpoint=endpoint, instances=[instance])

    # Print the predictions
```

```python
    print(response.predictions)

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



Deliverables:

- A trained machine learning model deployed on Google Cloud AI Platform.

exercise3-ml — gcloud.py ai custom-jobs stream-logs projects/1059330099607/locations/us-central1/customJobs/949618884053303296 — 150×40

```
(tf_env) (base) kabdrakhman@MacBook-Pro-Altair exercise3-ml % gcloud ai custom-jobs create \
  --region=us-central1 \
  --display-name=my-ml-job \
  --python-package-uris=gs://cloudadddevassignment3-ak-bucket/train.py \
  --worker-pool-spec=machine-type=n1-standard-4,replica-count=1,executor-image-uri=gcr.io/cloud-aiplatform/training/tf-cpu.2-4:latest,python-module=tr
ain
Using endpoint [https://us-central1-aiplatform.googleapis.com/]
API [aiplatform.googleapis.com] not enabled on project [cloudappdevassignment3-ak]. Would you like to enable and retry (this will take a few minutes)?
 (y/N)?  y

Enabling service [aiplatform.googleapis.com] on project [cloudappdevassignment3-ak]...
Operation "operations/acat.p2-1059330099607-a940f12e-a928-40a6-8d5f-f3dcb1c6fa39" finished successfully.
CustomJob [projects/1059330099607/locations/us-central1/customJobs/949618884053303296] is submitted successfully.

Your job is still active. You may view the status of your job with the command

  $ gcloud ai custom-jobs describe projects/1059330099607/locations/us-central1/customJobs/949618884053303296

or continue streaming the logs with the command

  $ gcloud ai custom-jobs stream-logs projects/1059330099607/locations/us-central1/customJobs/949618884053303296
(tf_env) (base) kabdrakhman@MacBook-Pro-Altair exercise3-ml %  gcloud ai custom-jobs describe projects/1059330099607/locations/us-central1/customJobs/
949618884053303296
Using endpoint [https://us-central1-aiplatform.googleapis.com/]
createTime: '2024-11-03T15:27:48.890406Z'
displayName: my-ml-job
jobSpec:
  workerPoolSpecs:
  - diskSpec:
      bootDiskSizeGb: 100
      bootDiskType: pd-ssd
    machineSpec:
      machineType: n1-standard-4
    pythonPackageSpec:
      executorImageUri: gcr.io/cloud-aiplatform/training/tf-cpu.2-4:latest
      packageUris:
      - gs://cloudadddevassignment3-ak-bucket/train.py
      pythonModule: train
    replicaCount: '1'
name: projects/1059330099607/locations/us-central1/customJobs/949618884053303296
```

- A script that makes predictions using the deployed model.