Assignment 3, cloud app development

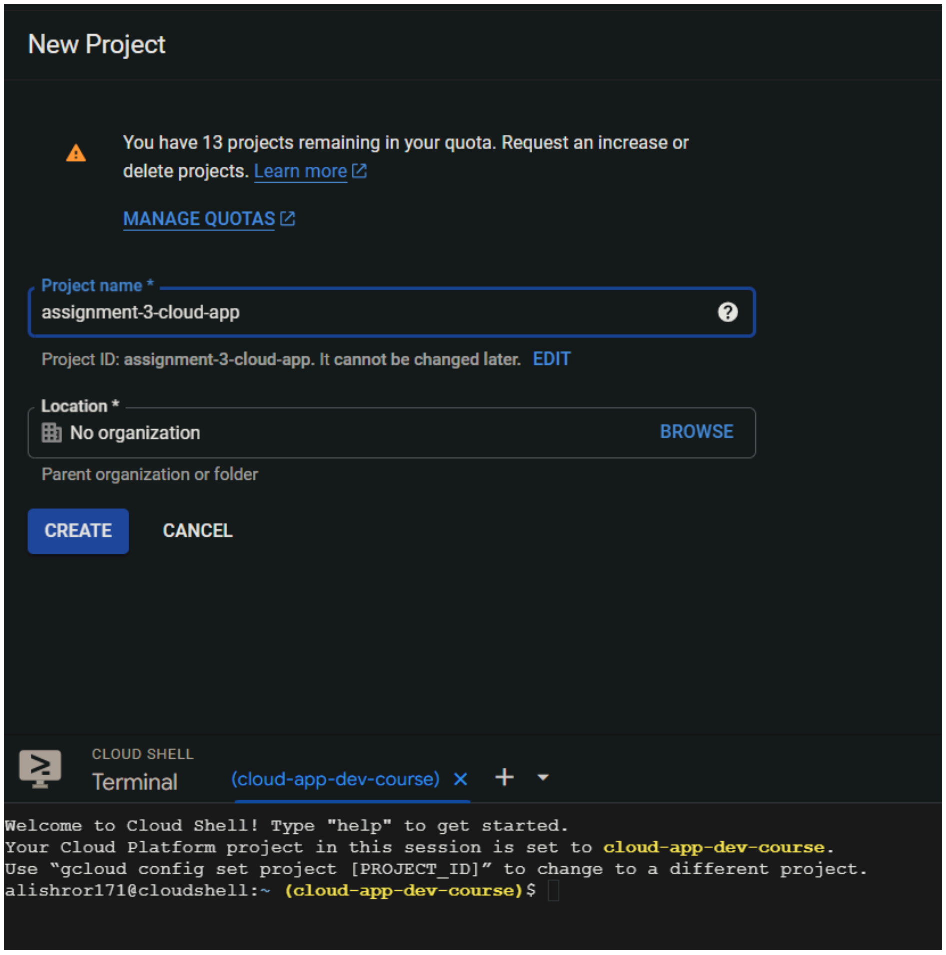
Put all deliverables into github repository in your profile. Defend by explaining deliverables and answering questions.
Deliverables: report (pdf)
Google form (before teams deadline):
https://docs.google.com/forms/d/e/1FAIpQLSe0GyNdOYIvM1tX_I_CtlPod5jBf-ACLGdHYZq1gVZbUeBzIg/viewform?usp=sf_link

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.


New Project
You have 13 projects remaining in your quota. Request an increase or delete projects. Learn more
MANAGE QUOTAS
Project name *
assignment-3-cloud-app
Project ID: assignment-3-cloud-app. It cannot be changed later. EDIT
Location *
No organization
BROWSE
Parent organization or folder
CREATE
CANCEL
CLOUD SHELL
Terminal
(cloud-app-dev-course)

```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to cloud-app-dev-course.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
alishror171@cloudshell:~ (cloud-app-dev-course)$
```

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

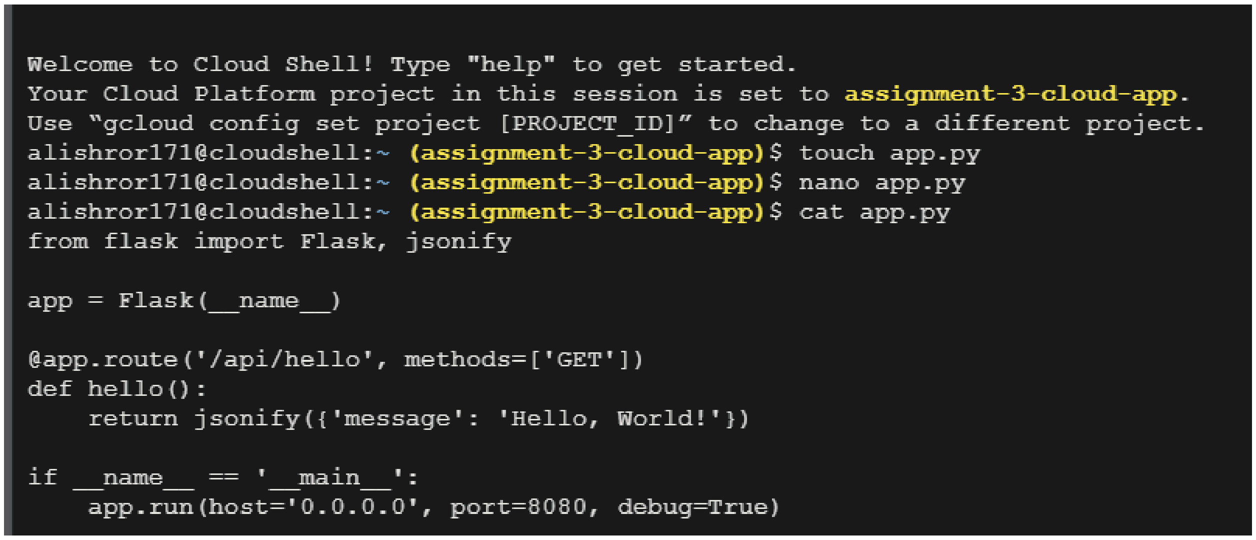
Example `app.py`:

```
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)
```



```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to assignment-3-cloud-app.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
alishror171@cloudshell:~ (assignment-3-cloud-app)$ touch app.py
alishror171@cloudshell:~ (assignment-3-cloud-app)$ nano app.py
alishror171@cloudshell:~ (assignment-3-cloud-app)$ cat 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)
```

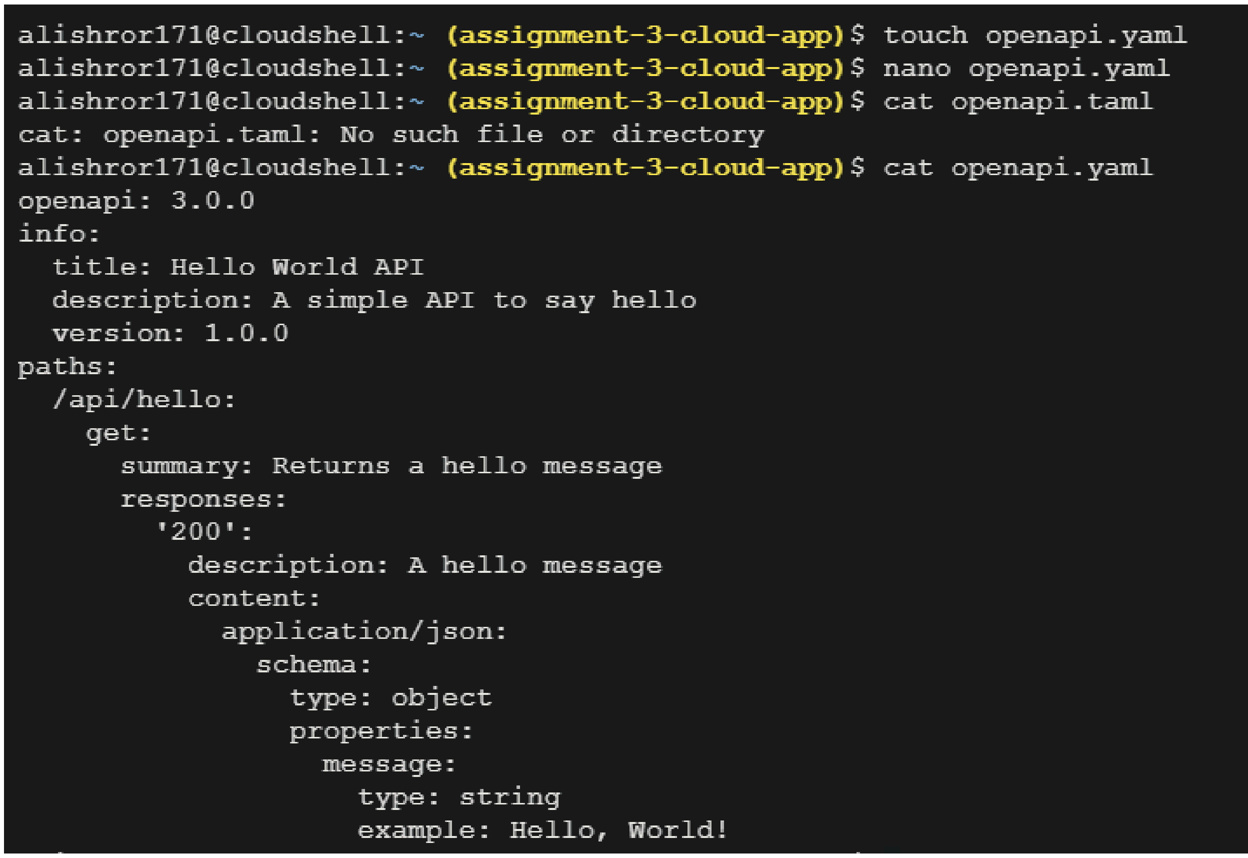
I used cloud console for this task

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!
```



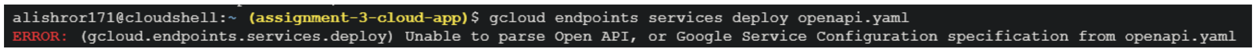
```
alishror171@cloudshell:~ (assignment-3-cloud-app)$ touch openapi.yaml
alishror171@cloudshell:~ (assignment-3-cloud-app)$ nano openapi.yaml
alishror171@cloudshell:~ (assignment-3-cloud-app)$ cat openapi.taml
cat: openapi.taml: No such file or directory
alishror171@cloudshell:~ (assignment-3-cloud-app)$ cat 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
gcloud app deploy
```



```
alishror171@cloudshell:~ (assignment-3-cloud-app)$ gcloud endpoints services deploy openapi.yaml
ERROR: (gcloud.endpoints.services.deploy) Unable to parse Open API, or Google Service Configuration specification from openapi.yaml
```

As it turned out google cloud doesn’t support third version of openapi

   ○

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.

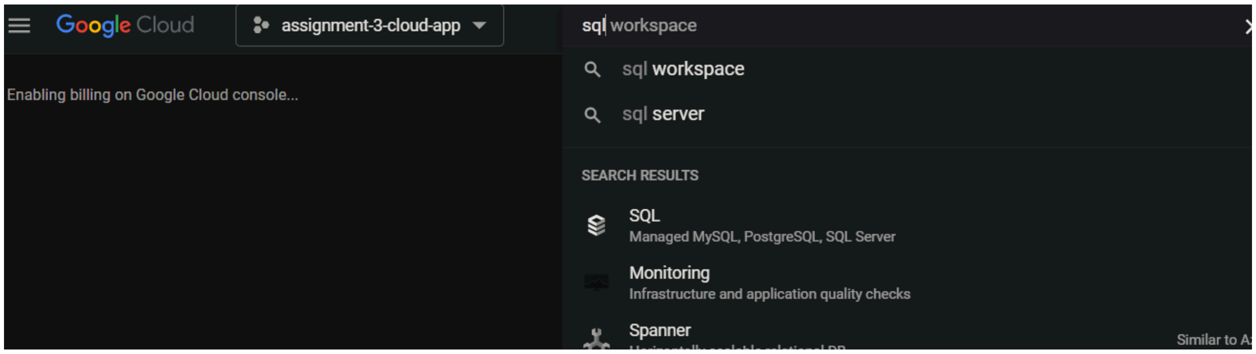
Exercise 2: Google Cloud Databases

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

Instructions:

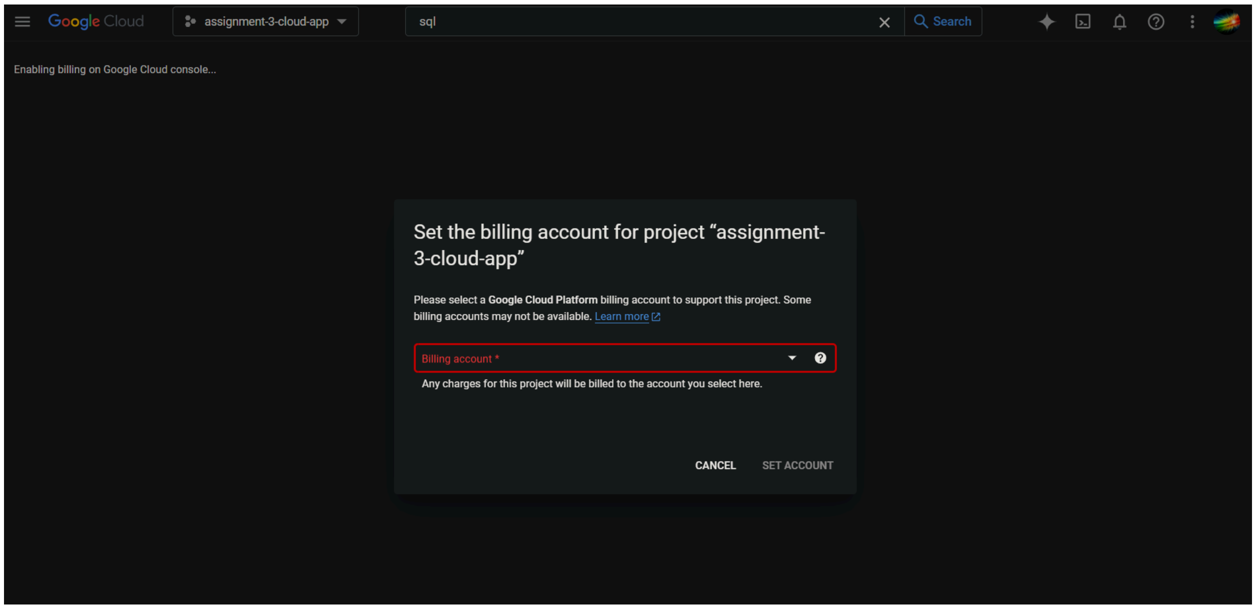
1. **Setup**:
    - Ensure you have a Google Cloud account.
    - Install the Google Cloud SDK.
2. **Create a Cloud SQL Instance**:
    - Navigate to the Google Cloud Console and create a new Cloud SQL instance.
    - Choose MySQL, PostgreSQL, or SQL Server as the database type.
    - Configure the instance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

It would be nice to do that by myself, but:



Enabling billing on Google Cloud console...

Google is too greedy to allow me to use SQL:



Enabling billing on Google Cloud console...

Set the billing account for project “assignment-3-cloud-app”

Please select a **Google Cloud Platform** billing account to support this project. Some billing accounts may not be available. Learn more

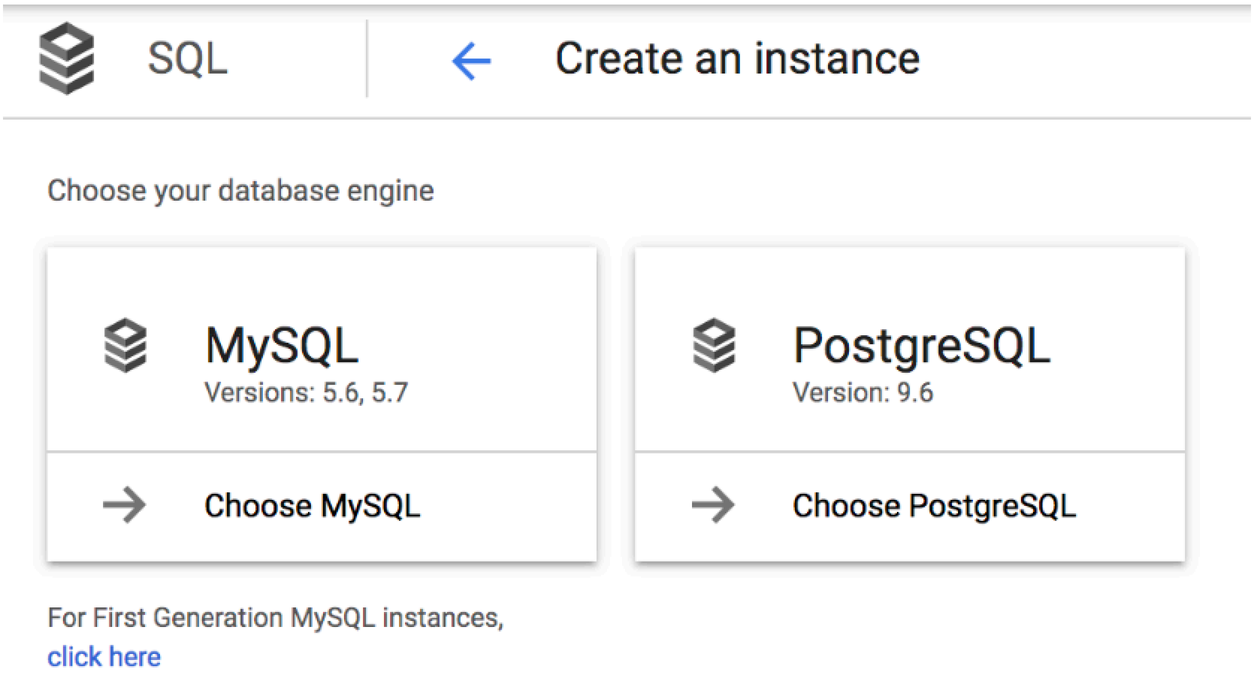
Billing account *

Any charges for this project will be billed to the account you select here.

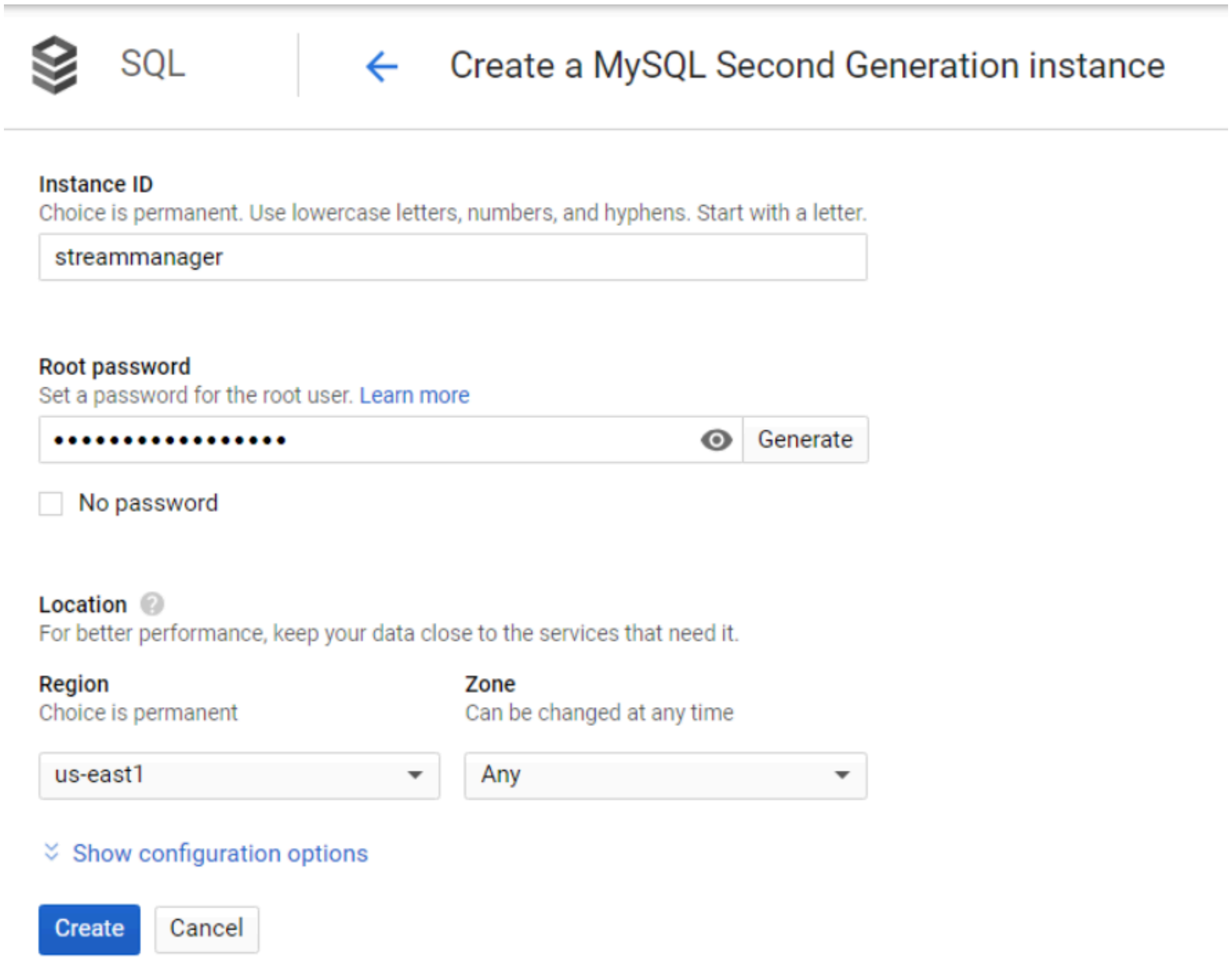
CANCEL SET ACCOUNT

So I will just describe how would I do it from now on

I found the next image from the internet (not mine, because it requires billing as was said above):



So firstly we need to click on “Choose MySQL”, the following page should open up:



Create a MySQL Second Generation instance

Instance ID
Choice is permanent. Use lowercase letters, numbers, and hyphens. Start with a letter.
streammanager

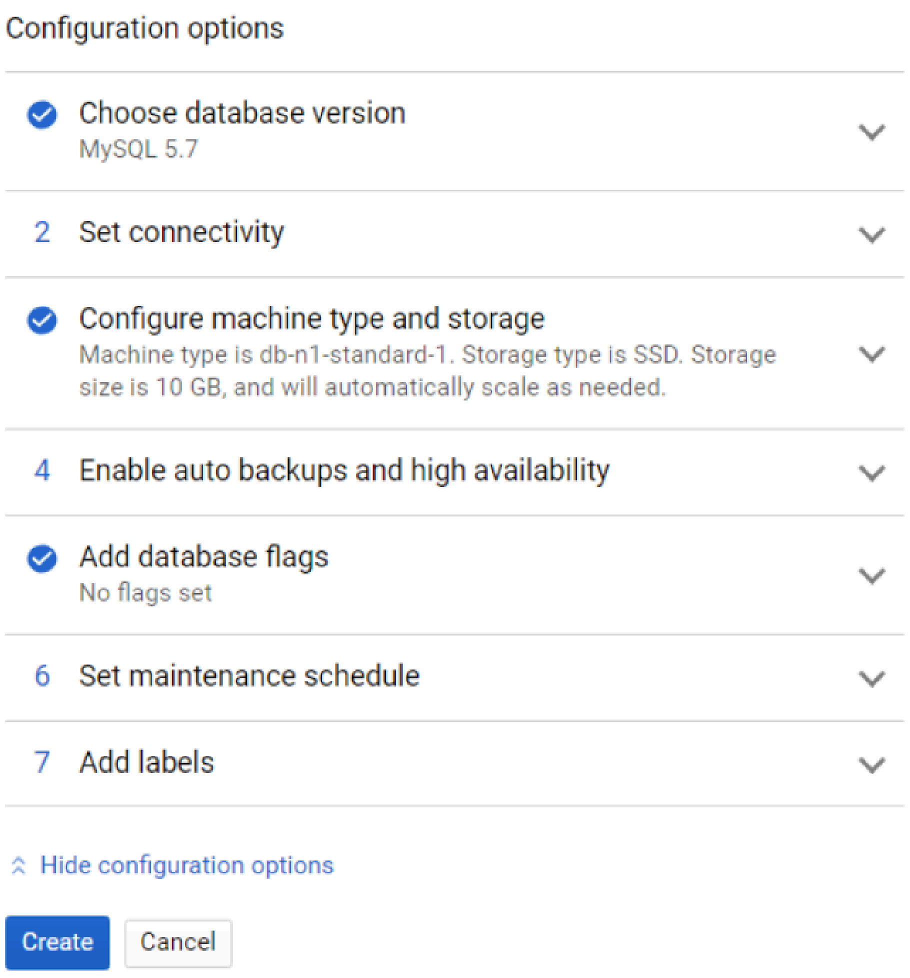
Root password
Set a password for the root user. Learn more
•••••••••••••••••

[ ] No password

Location
For better performance, keep your data close to the services that need it.

Region	Zone
Choice is permanent	Can be changed at any time
us-east1	Any

Where we need to specify all fields, like instance ID, root password which can be generated



Configuration options

✓ Choose database version
MySQL 5.7

2 Set connectivity

✓ Configure machine type and storage
Machine type is db-n1-standard-1. Storage type is SSD. Storage size is 10 GB, and will automatically scale as needed.

4 Enable auto backups and high availability

✓ Add database flags
No flags set

6 Set maintenance schedule

7 Add labels

Make sure that everything is alright and click 'Create'

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.

After the creation, you can connect to the instance using following command:

mysql -u your-username -p -h your-cloud-sql-instance-ip

Then enter the password and afterwards enter these commands to create a database and a table:

```
CREATE DATABASE sample_db;
```

```
USE sample_db;
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');
```

-

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

First you need to install required packages:

pip install mysql-connector-python

Then create a connect.py file using console or SDK:

Example `connect.py`:

```
import mysql.connector

cnx = mysql.connector.connect(
    user='your-username',
    password='your-password',
    host='your-cloud-sql-instance-ip',
    database='sample_db'
)
cursor = cnx.cursor()
cursor.execute('SELECT * FROM users')
for row in cursor:
    print(row)
cursor.close()
cnx.close()
```

5. **Run the Connection Code**:

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

Deliverables:

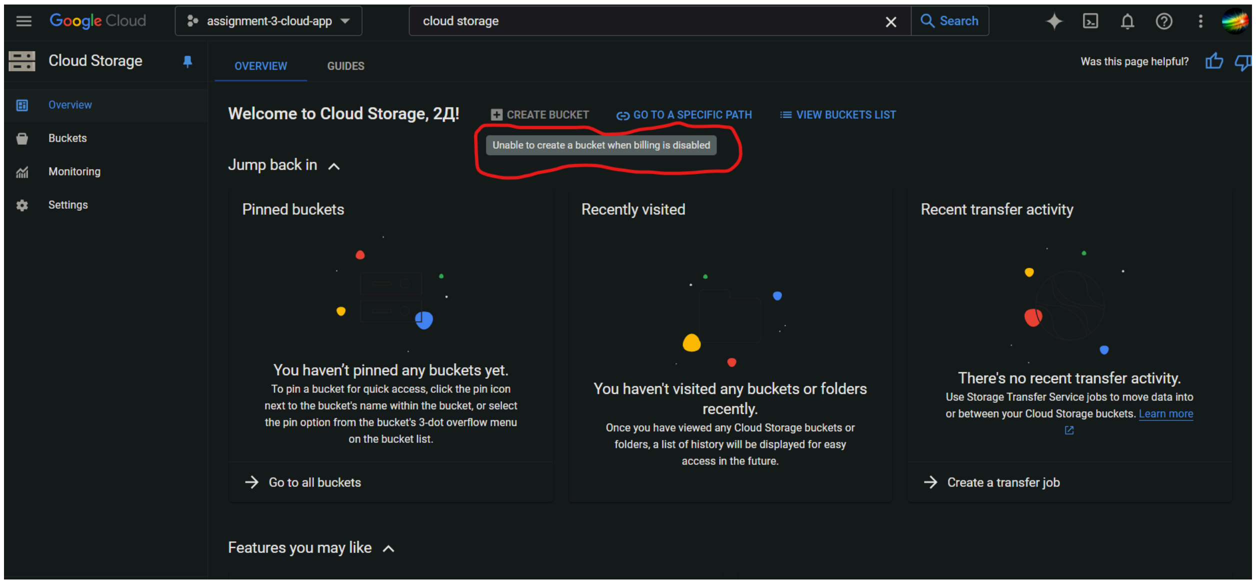
- A working Cloud SQL database with sample data.
- A Python script that successfully connects to and queries the database.

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.



Welcome to Cloud Storage, 2Д!

Unable to create a bucket when billing is disabled

You haven't pinned any buckets yet.
To pin a bucket for quick access, click the pin icon next to the bucket's name within the bucket, or select the pin option from the bucket's 3-dot overflow menu on the bucket list.

You haven't visited any buckets or folders recently.
Once you have viewed any Cloud Storage buckets or folders, a list of history will be displayed for easy access in the future.

There's no recent transfer activity.
Use Storage Transfer Service jobs to move data into or between your Cloud Storage buckets. Learn more

Nice I guess

So it just creates a bucket for the files we are gonna store, there we should have chosen an unique name and a region

Then go to your Cloud Storage bucket, click Upload Files, and add your dataset files.

3. **Prepare Training Data**:
   - Upload sample training data to your Cloud Storage bucket. For example, use a dataset for classification or regression.
4. **Create a Training Script**:
   - Write a simple TensorFlow training script.

After that create a model training script, for example the4 following code for classification

Example `train.py`:
python
Копировать код
```
import tensorflow as tf

def create_model():
    model = tf.keras.Sequential([
        tf.keras.layers.Dense(10, activation='relu',
input_shape=(784,)),
        tf.keras.layers.Dense(10, activation='softmax')
    ])
    model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    return model

def main():
    model = create_model()
    train_data = tf.data.Dataset.from_tensor_slices((X_train,
y_train)).batch(32)
    model.fit(train_data, epochs=5)
    model.save('gs://your-bucket/model')

if __name__ == '__main__':
```

```
main()
```

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

After that usign the code below, we can strat the training of our model (which will use train.py for it):

Use the following command to start training:
bash
Копировать код

```
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:latest
```

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

Use the following command:
bash
Копировать код

```
gcloud ai models create your-model --region=your-region
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**:
   - Use the deployed model endpoint to make predictions.

Just create a new file for the predictions as below:

Example `predict.py`:
python

Копировать код
```
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')
    instance = {'input': [/* your data */]}
    response = client.predict(endpoint=endpoint,
instances=[instance])
    print(response.predictions)

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

○

Deliverables:

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