Exercise 1: Understanding Cloud Computing Models

IaaS builds the infrastructure of cloud-based technology. **PaaS** helps developers build custom apps via an API that can be delivered over the cloud. **SaaS** is cloud-based software companies can sell and use.

Feature	IaaS	PaaS	SaaS
Control	High control over infrastructure	Moderate control over application deployment	Low control over software functionality
Flexibility	Very flexible; customize resources	Moderately flexible; focuses on development	Limited flexibility; predefined software
Use Cases	Hosting websites, storage, enterprise applications	Application development, testing, and deployment	Email services, CRM, collaboration tools

GCP services

- **IaaS**: Google Compute Engine, Google Cloud Storage.
- **PaaS**: Google App Engine, Google Cloud Functions.
- **SaaS**: Google Workspace, Google Cloud Identity.

Real-world example

- **IaaS**: building web applications and needs scalable server infrastructure (Amazon Web Services)
- **PaaS**: creating a mobile application and wants to deploy it (Google App Engine)
- **SaaS**: needs productivity tools email services, crm (Microsoft 365)

Exercise 2: Exploring Google Cloud Platform's Core Services

The primary use case of Compute Engine:

- provide scalable virtual machines for running applications
- hosting websites
- processing large datasets

Google Kubernetes Engine (GKE) simplifies the management of containerized applications by automating their deployment, scaling, and overall management through Kubernetes.

Advantages of Cloud Storage offer for data management:

- **Scalability**: Capable of storing nearly unlimited data.
- **Durability and Availability**: Data is redundantly stored in multiple locations to guarantee high availability and protection.
- **Accessibility**: Data can be accessed worldwide via a straightforward API, making it ideal for various applications, including websites and mobile apps.
- **Cost-Effectiveness**: Various storage classes enable optimized pricing based on how often the data is accessed.

Businesses would choose BigQuery because it efficiently handles large datasets and delivers fast query performance, all without the user having to manage any underlying infrastructure.

I don't have access to google cloud since I didn't get a free trial and they are asking for billing.

Exercise 3: Creating and Managing Virtual Machines with Compute Engine

Firstly we create the VM instance then we set the parameters:

- **proper name**
- **region -** *us-east1*
- **zone** - *us-east1-a*
- **machine type** - *f1-micro*
- **operating system** - *windows*

Next is connect to the VM using SSH and install a basic web server. To connect we just need to click the **SSH** next to our instance. To install basic web server, we write the next codes:

- **Apache** - *sudo apt install apache2*
- **Nginx** - *sudo apt install nginx*

To start web server we need next codes:

- **Apache** - *sudo systemctl status apache2*
- **Nginx** - *sudo systemctl status nginx*

Next is Stop, Start, and Delete the VM

- **Stop -** our VM instance in list; three dots on the right; select Stop
- **Start -** our VM instance in list; three dots on the right; select Start
- **Delete -** our VM instance in list; three dots on the right; select Delete; confirm

Exercise 4: Deploying a Containerized Application on Google Kubernetes Engine (GKE)

- **Create a simple Docker container for a web application.**

Firstly we need to create the simple web application, for example in java by following code

```
package com.example.demo;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@SpringBootApplication
public class DemoApplication {
    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
    @RestController
    class HelloController {
        @GetMapping("/")
        public String hello() {
            return "Hello from GKE!";
        }
    }
}
```

Then we need to create the Dockerfile by following code

```
FROM openjdk:17-jdk-slim
WORKDIR /app
COPY target/demo-0.0.1-SNAPSHOT.jar /app/demo.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "demo.jar"]
```

Then build the Docker Image by following code

```
mvn clean package
docker build -t gke-java-app
```

- **Push the container image to Google Container Registry (GCR).**

Authenticate Docker with Google Cloud

```
gcloud auth configure-docker
```

Tag the Docker Image

```
docker tag gke-java-app gcr.io/your-project-id/gke-java-app
```

Push the Image to GCR

```
docker push gcr.io/your-project-id/gke-java-app
```

- **Create a GKE cluster in Google Cloud Console.**

Firstly we start by clicking on **Create Cluster** in section **Kubernetes Engine**, configure settings:

- **Cluster name**: *my-gke-cluster*
- **Zone**: *us-central1-a*
- **Project ID**: *my-gcp-project*

then we authenticate kubectl with GKE

```
gcloud container clusters get-credentials my-gke-cluster --zone us-central1-a --project my-gcp-project
```

- **Deploy the containerized application to the GKE cluster.**

Then we create a Kubernetes Deployment YAML File

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: java-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: java-app
  template:
    metadata:
      labels:
        app: java-app
    spec:
      containers:
      - name: java-app
        image: gcr.io/your-project-id/gke-java-app
        ports:
        - containerPort: 8080
```

Then we deploy the Application to GKE by following code

```
kubectl apply -f deployment.yaml
```

- **Expose the application to the internet and verify its accessibility.**

By following the code we expose the Deployment Using a Load Balancer and verify

```
kubectl expose deployment java-app --type=LoadBalancer --port 80 --target-port 8080
```

```
kubectl get services java-app
```

Exercise 5: Storing and Accessing Data in Google Cloud Storage

- **Create a new Cloud Storage bucket in the Google Cloud Console.**

By following the path Storage - Cloud Storage - Buckets we can create the new bucket. There is a configuration

- **name** - our bucket
- **storage class** - *Standard, Nearline, Coldline, Archive*
- **location** - *multi-regional or regional*
- **access control** - *Uniform or Fine-grained*

Then by clicking Create we finish

- **Upload various types of files (e.g., text, images, videos) to the bucket.**

In Bucket we choose bucket named “our bucket”, in our page at the top we click upload file button and choose file, for example images and it will be listed

- **Set access permissions for the bucket and test public and private access to the files.**

Private - in this mode files are private, only authorized users can access, if users not logged try to open then it will return error

Public - in our file by choosing Edit permissions - Add entry - add allUsers with role Storage Object Viewer we gave public permission and it can be open without login.

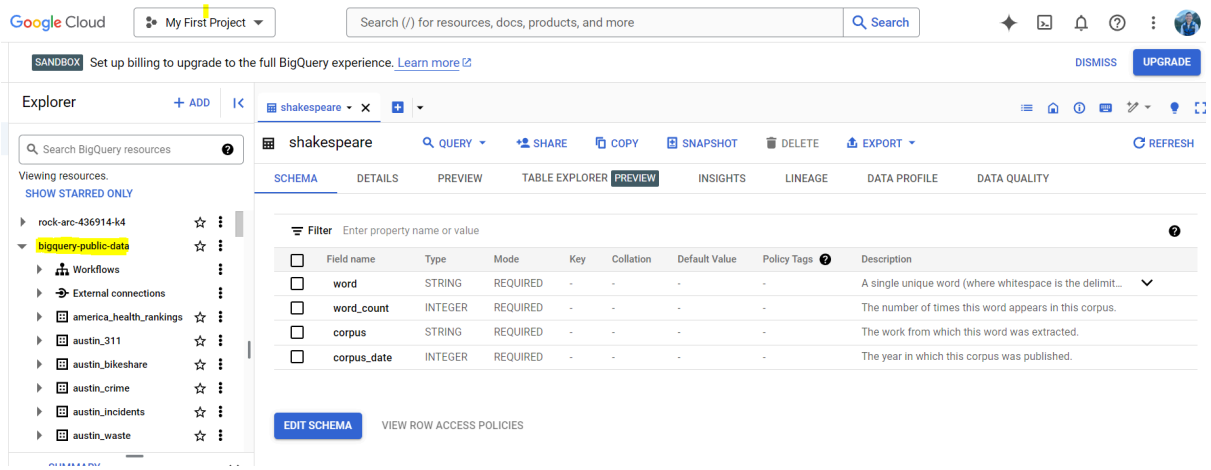
- **Use the Cloud Console to download, move, and delete files in the bucket.**

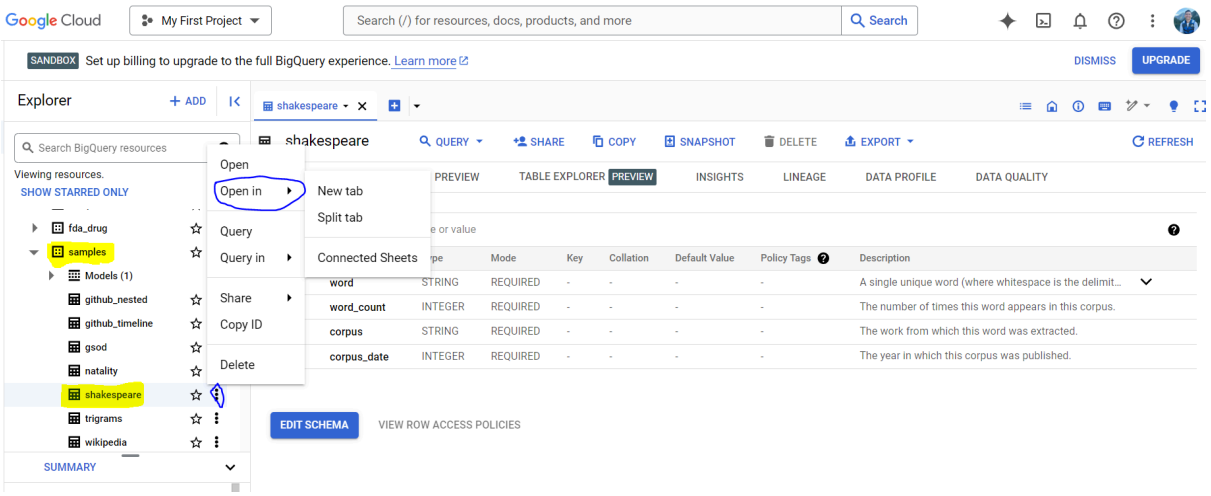
Firstly we choose our file, click on the file, there is the button Download; Move; Delete and by clicking we can manage files.

Exercise 6: Analyzing Data with BigQuery

- **Create a dataset and table by importing a sample dataset provided by Google.**

In **bigquery-public-data.samples** we choose the **samples** and now we can choose the public datasets, one way is by clicking three dots we can open in a new tab and get information and also by 3 dots > Query in > New Tab immediately get query for this dataset





- **Write and execute SQL queries to perform basic data analysis, such as filtering, aggregation, and sorting.**

Now we can play with the dataset, following queries perform basic data analysis:

shows words and how many times they used and sorted ascending

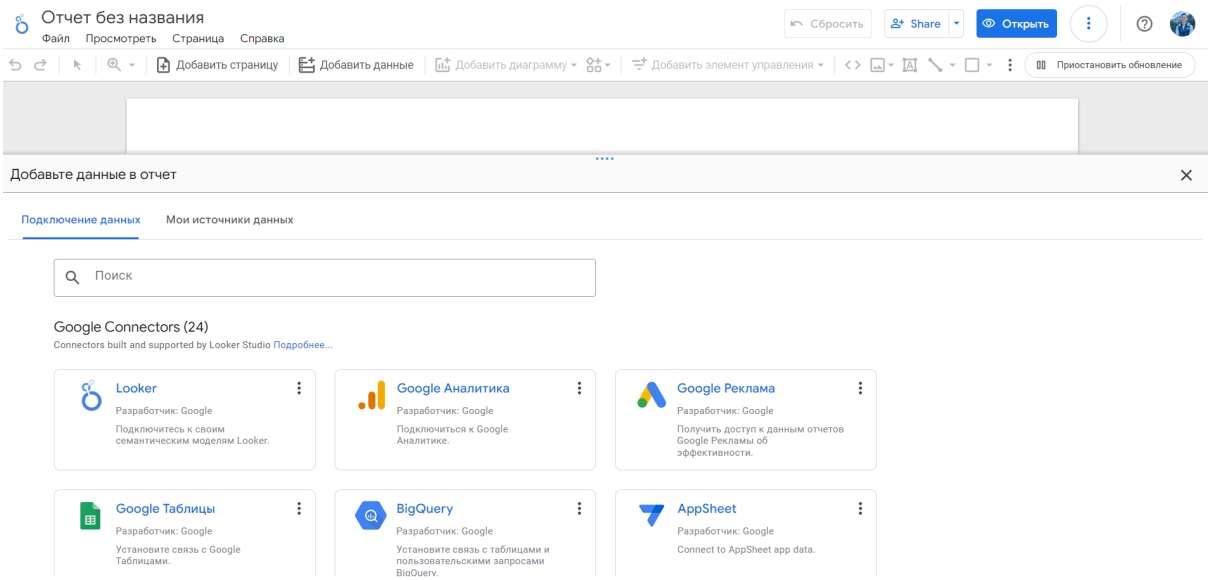
```
SELECT word, SUM(word_count) as count
FROM `bigquery-public-data.samples.shakespeare`
GROUP BY word
ORDER BY count
```

we can just filter the dataset by specific criteria

```
SELECT *
FROM `bigquery-public-data.samples.shakespeare`
WHERE corpus = 'hamlet'
```

- **Visualize the results using Google Data Studio or another visualization tool.**

In Looker Studio interface, we can connect it with BigQuery



After connecting we need next steps to finish the visualization:

- We can add diagram for example pie chart
- In Data tab we set parameters of chart and it automatically show the visualization
  Dimension - *word*
  Metric - *sum(word_count)*
- We can customize the pie chart in Style tab
- In Filter tab, we can add filter like corpus = ‘hamlet’