KAZAKH BRITISH TECHNICAL UNIVERSITY

REPORT

Assignment 3, Cloud Computing

Akhmetkan Azhar
24MD0470
15.11.2024

Table of Contents

Introduction

This assignment covers the most important aspects of cloud technologies that are the basis for developing, deploying, and managing modern applications in a cloud environment. In particular, it covers such key components as Identity and Security Management, the use of Google Kubernetes Engine for container orchestration, and application deployment using Google App Engine and Google Cloud Functions. These technologies play a key role in ensuring the security, scalability, and flexibility of cloud solutions.

Let's start with Identity and Security Management, which is the basis for data protection and access control in cloud systems. Proper configuration of user roles and rights is critical to ensuring the security of cloud resources and preventing unauthorized access. These tools allow you to manage access, protect data, and ensure security compliance, making them an integral part of any cloud infrastructure.

Second is Google Kubernetes Engine, in turn, is a powerful solution for automated deployment and management of containerized applications. GKE simplifies the process of scaling and managing many application instances, which is especially important for ensuring high availability and resilience in cloud environments. Using containers in the cloud can significantly increase productivity and simplify the development and testing processes.

Google App Engine allows developers to deploy applications without having to worry about server infrastructure. This solution is ideal for creating and scaling web applications, as it allows you to focus on functionality rather than the technical aspects of setting up and managing servers.

Google Cloud Functions provides the ability to easily create event-driven functions that are automatically executed when certain events occur in the cloud. This simplifies the creation of microservice architectures and the integration of various cloud services, allowing you to effectively respond to changes and events in the system.

All these components together form a powerful tool for developing, deploying and managing applications in the cloud. Their use provides not only security and scalability, but also flexibility in development, which makes them indispensable for modern cloud solutions.

Exercise Descriptions

Exercise 1: Setting Up IAM Roles

Objective is to learn how to create a project in Google Cloud and configure Identity and Access Management (IAM) roles for different users in the project.

To complete this exercise, you will first need to create a new project in Google Cloud using the Google Cloud Console. You need to click on the console icon in the upper right corner (Image 1.1).

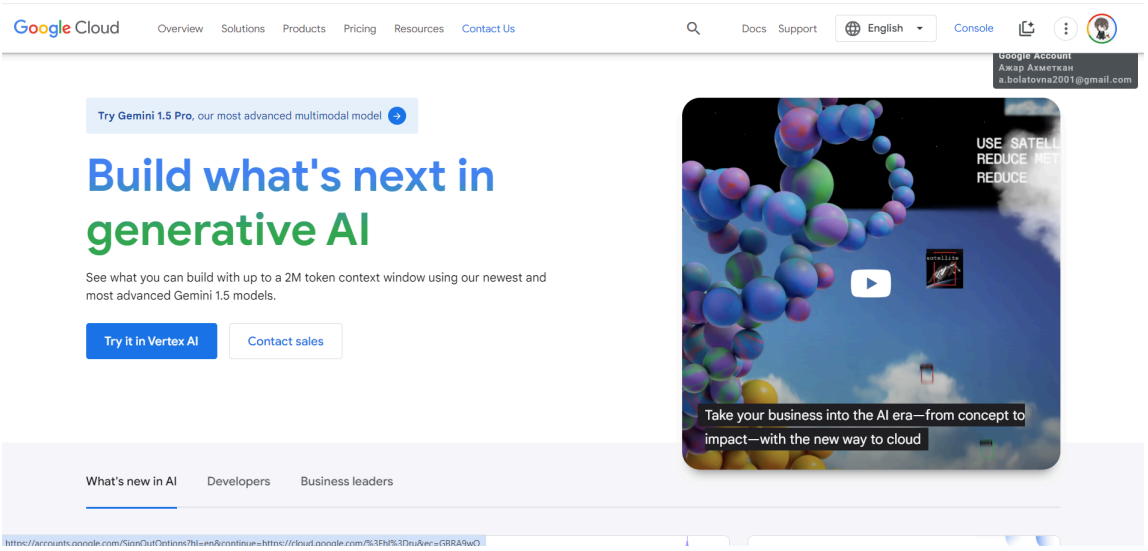

Image 1.1

Go to your projects list and click on the "new project" button in the right corner of the window (Image 1.2).

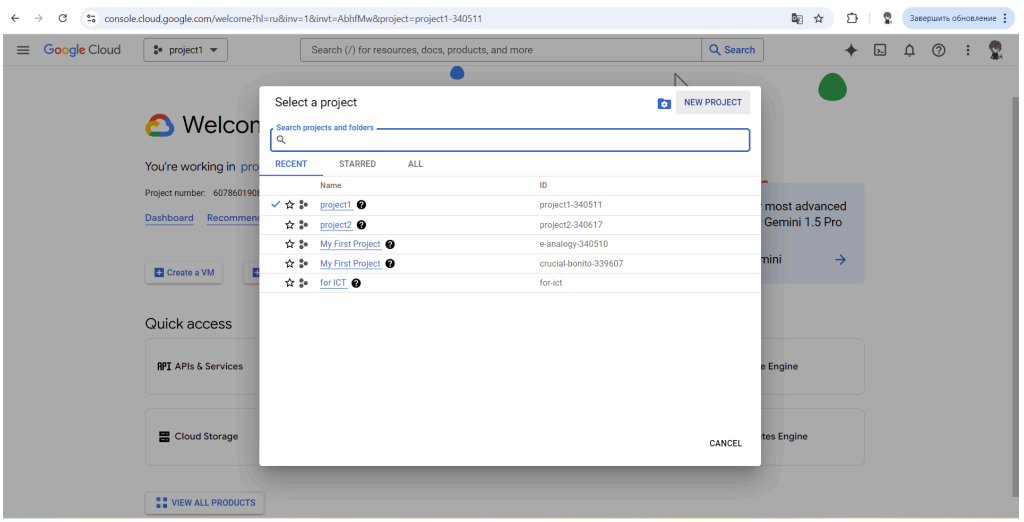

Image 1.2

A window will open where you will need to give a name to the new project and make some settings on how to specify the organization and click “CREATE” button for creation (Image 1.3).

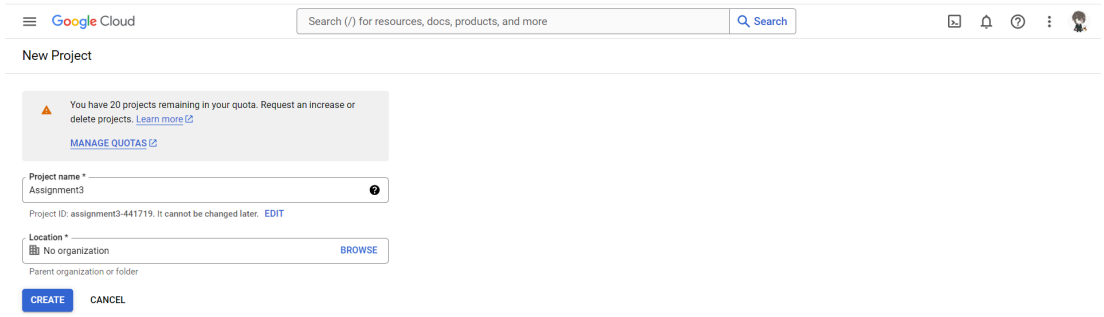

Image 1.3

Once you have done this, you will need to go to the IAM section (Image 1.4)

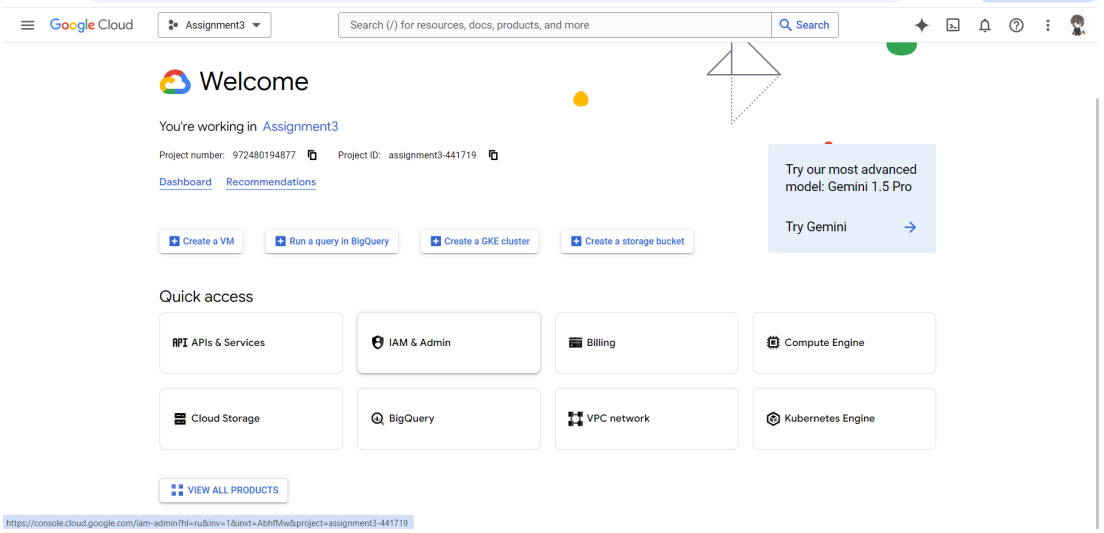

Image 1.4

and set up roles for different users or groups. Basic roles such as Viewer, Editor, and Owner should be assigned to the appropriate users with specific access rights (Image 1.5).

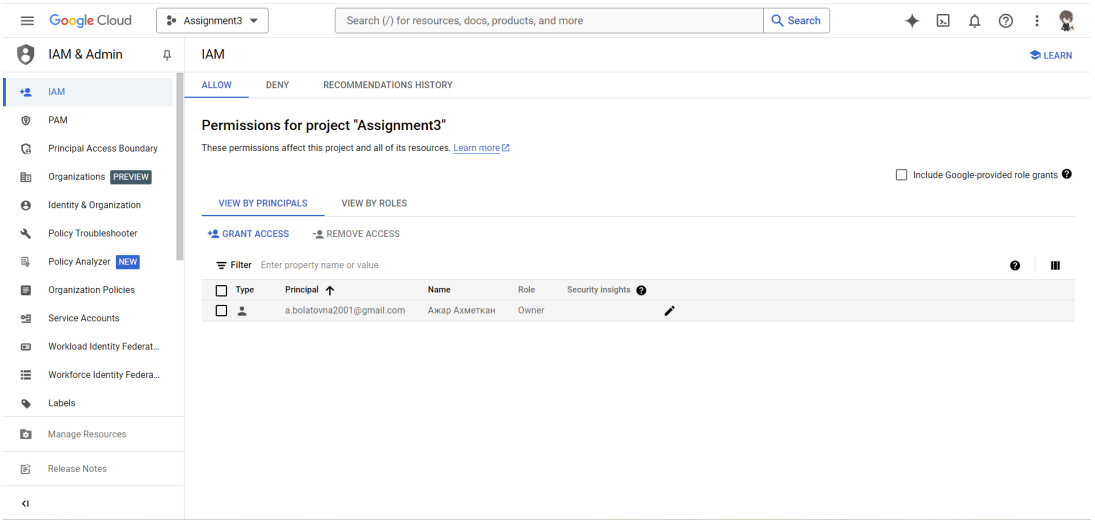

Image 1.5

Once roles are assigned, it is important to document the permissions associated with each role so that you understand what actions users can take within the project.

The end result of this exercise is a new project in Google Cloud that is configured with different IAM roles (e.g. Viewer, Editor, Owner) for multiple users or groups. Each user will be given a specific set of permissions based on the assigned role, and the exact

permissions available to each role will be documented so that you can understand exactly what users can and cannot do within the project.

Exercise 2: Service Accounts

Objective is to understand what service accounts are, how to create them, and how to configure them to authenticate applications in Google Cloud, particularly to access Google Cloud Storage.

In this exercise, you will create a service account that will be used to access Google Cloud Storage. To do this, go to the Google Cloud Console and go to the Service Accounts section (Image 2.1),

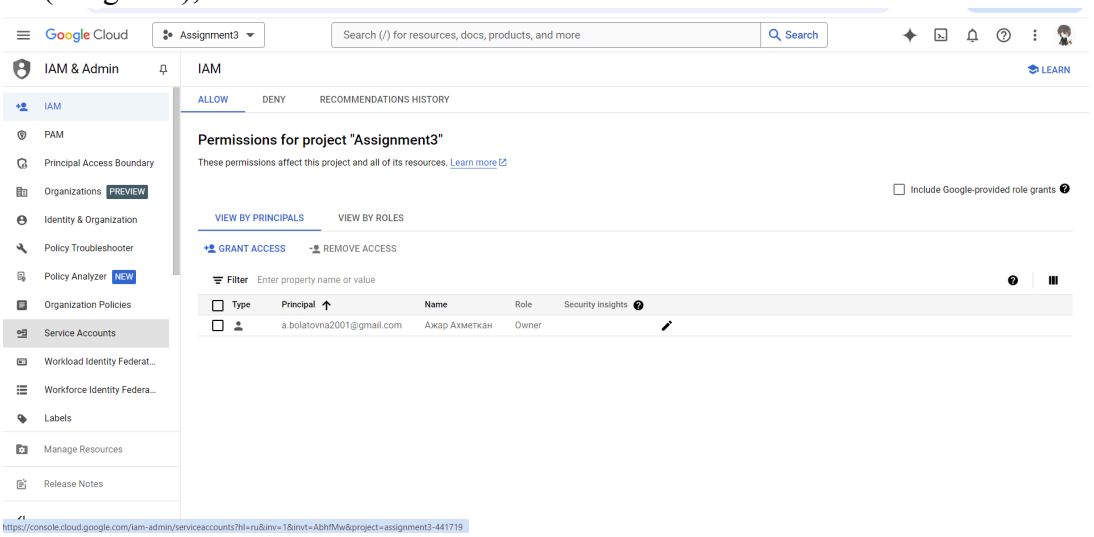

Image 2.1

We can see the “CREATE SERVICE ACCOUNT” button when we enter the section and click it (Image 2.2).

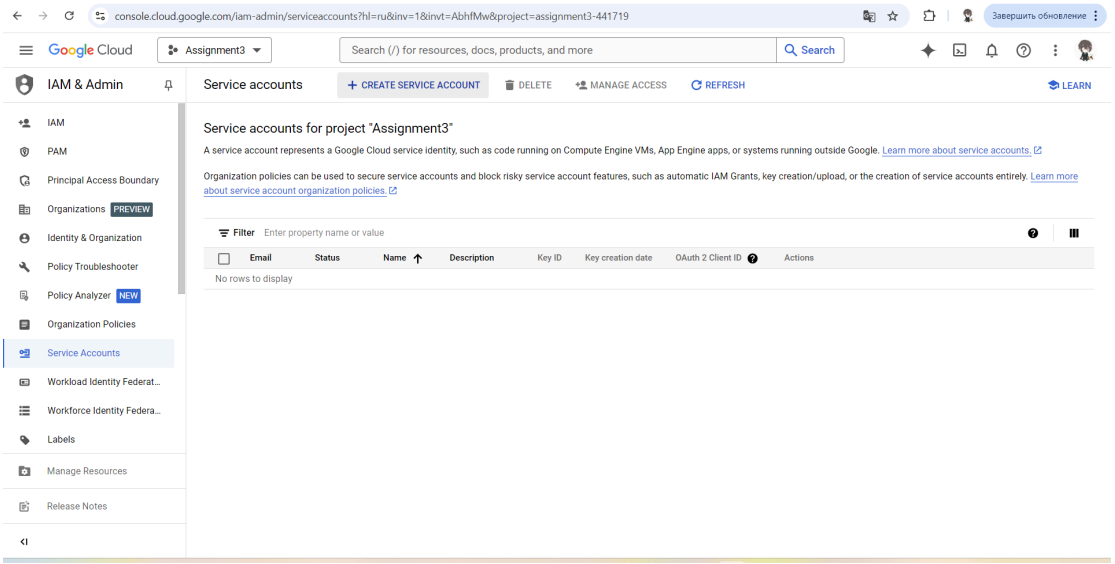

Image 2.2

create a new account, and grant it the necessary rights to work with Cloud Storage (Image 2.3).

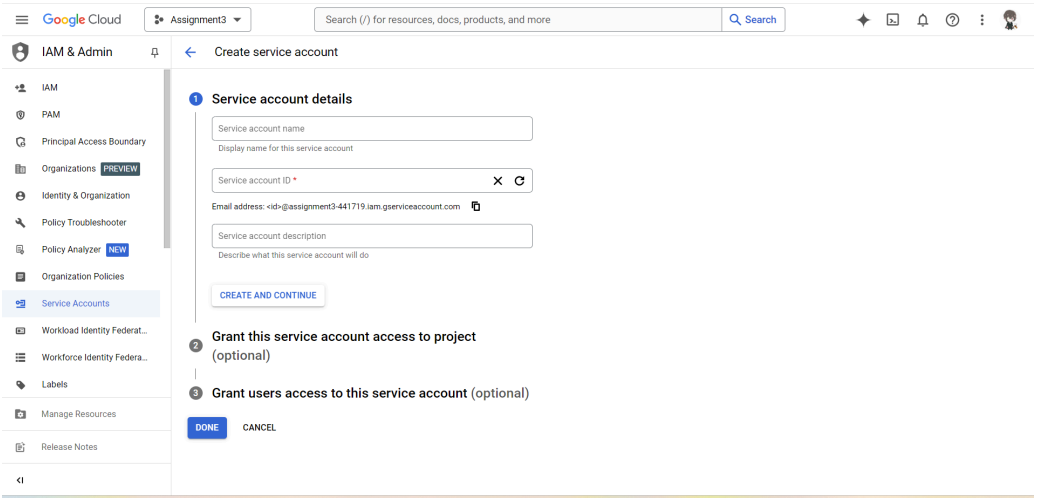

Image 2.3

Next, you will need to generate a key for the service account in JSON format and download it. To do this, select the desired service account, open its settings, and in the Keys section (Image 2.4),

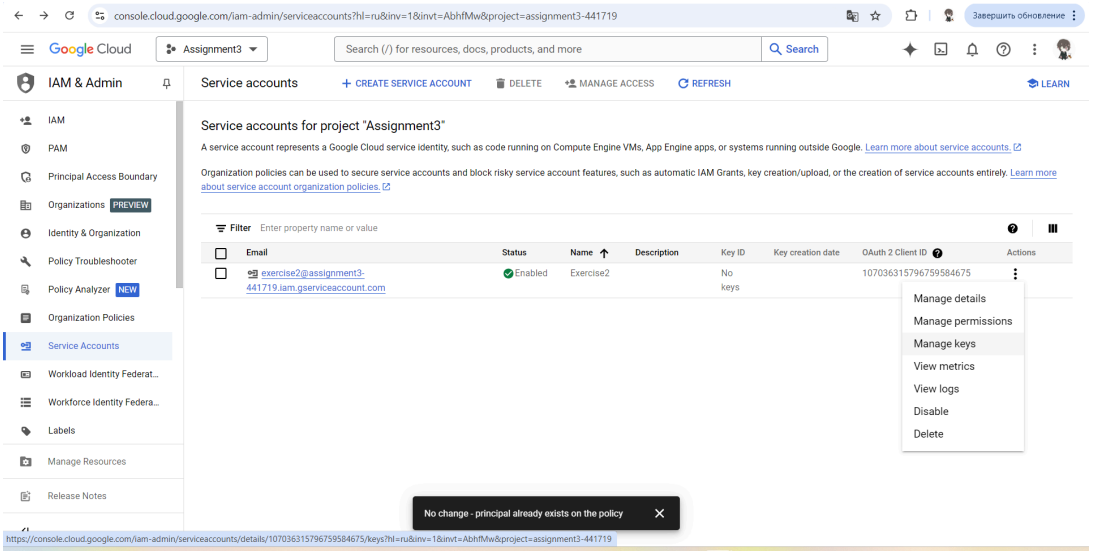

Image 2.4

click Add Key → Create New Key (Image 2.5)

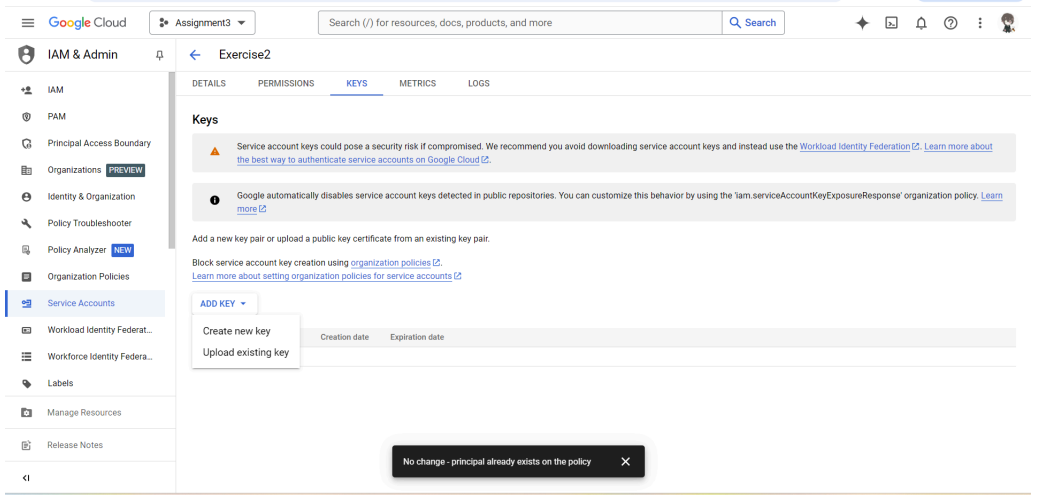

Image 2.5

Select the JSON format (Image 2.6)

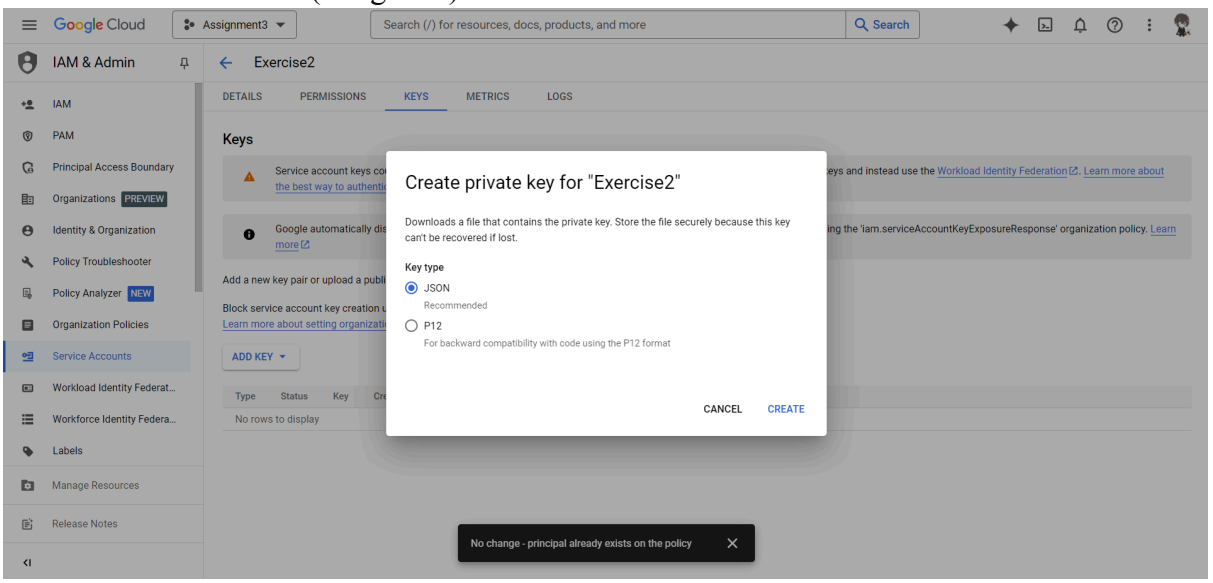

Image 2.6

and download the key (Image 2.7).

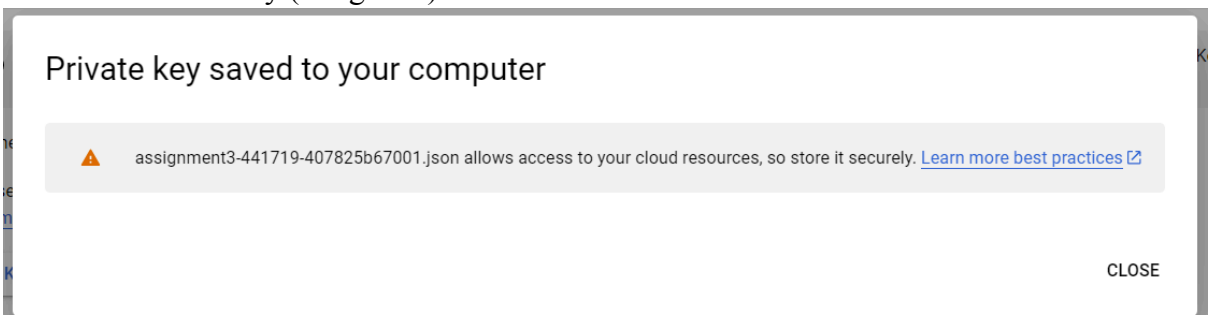

Image 2.7

This key will then be used to authenticate a Python script that will upload a file to Cloud Storage using the google-cloud-storage library.

After completing the exercise, a service account with the necessary rights to work with Google Cloud Storage will be created. The generated key for the service account will be used to authenticate the Python script that will upload the file to Cloud Storage. This will demonstrate how to use service accounts securely and efficiently to access cloud resources without having to manually enter credentials.

Exercise 3: Organization Policies

Objective is to learn how to use organizational policies to restrict certain actions within an organization or project in Google Cloud.

To set up an organizational policy, go to the Organizational Policies section of the Google Cloud Console (Image 3.1) This will open a page where you can set up organizational policies for your organization or project.

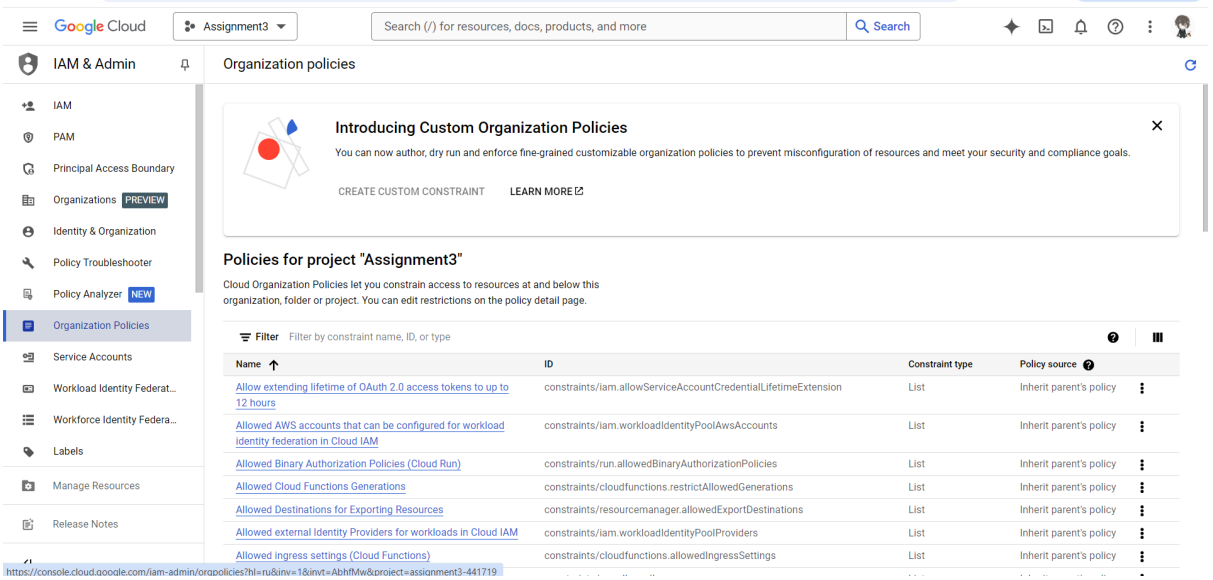

Image 3.1

On the Organization Policies page, you'll see a list of available policies. Each policy restricts or allows specific actions in the organization. apply the desired policy (for example, disallow the creation of public IP addresses). For example, for this exercise we will use a policy that restricts the creation of public IP addresses. Let's take the Compute/RestrictPublicIP policy. This is a policy that prohibits the creation of public IP addresses and click on it to open the settings (Image 3.2).

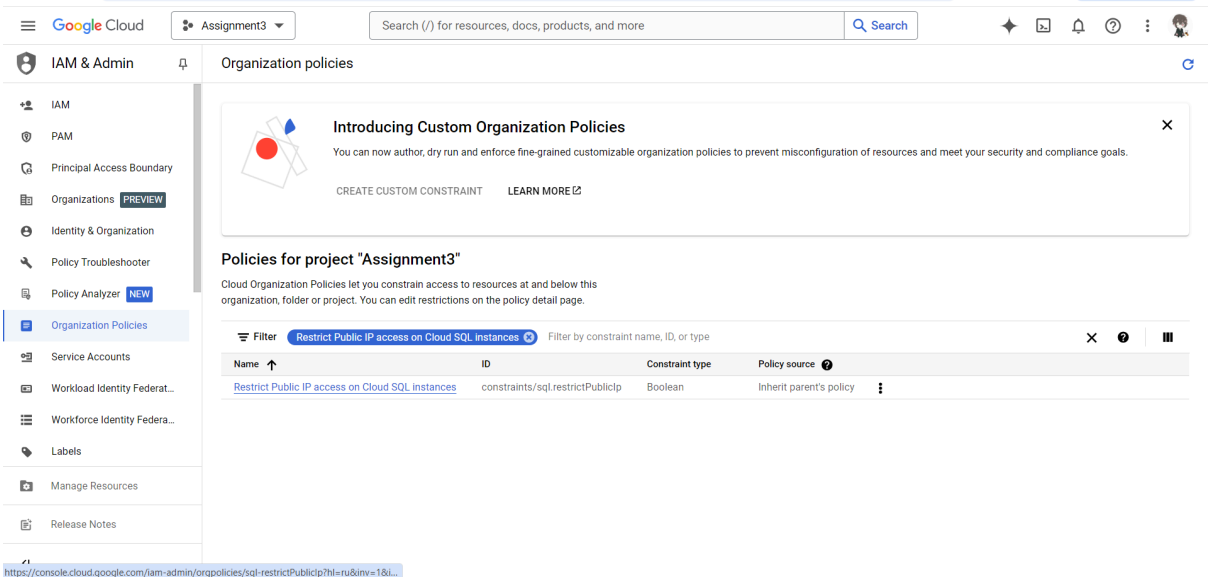

Image 3.2

In the policy settings, you will see two main states: Not Configured and Enforced. Click the Modify button and select Enforced to enable the policy and prohibit the creation of public IP addresses. Click Save to apply the changes. This action will restrict the ability to create public IP addresses in your project or organization. This policy can be applied at either the project level or the entire organization. After applying the policy, it is important to test that it is valid using the gcloud command or the console to ensure that it is applied correctly. To do this, go to the VPC network → External IP addresses section. Try to create a new external IP address. If the policy was applied correctly, you will receive an error that creating public IP addresses is prohibited.

During the exercise, you will set up an organizational policy that restricts the creation of public IP addresses at the project or organization level. You will be given a detailed description of how to apply this policy, as well as how to verify that it is in effect. After completing the exercise, you are expected to be able to confidently manage organizational policies and ensure security and compliance across your organization.

Exercise 4: Deploying a Simple Application on GKE

Objective is to get hands-on experience with Google Kubernetes Engine (GKE) and deploying a simple containerized application.

At this point, a GKE cluster is created, which can be created either through the Google Cloud Console or using the gcloud command. Let's go to Kubernetes Engine (Image 4.1).

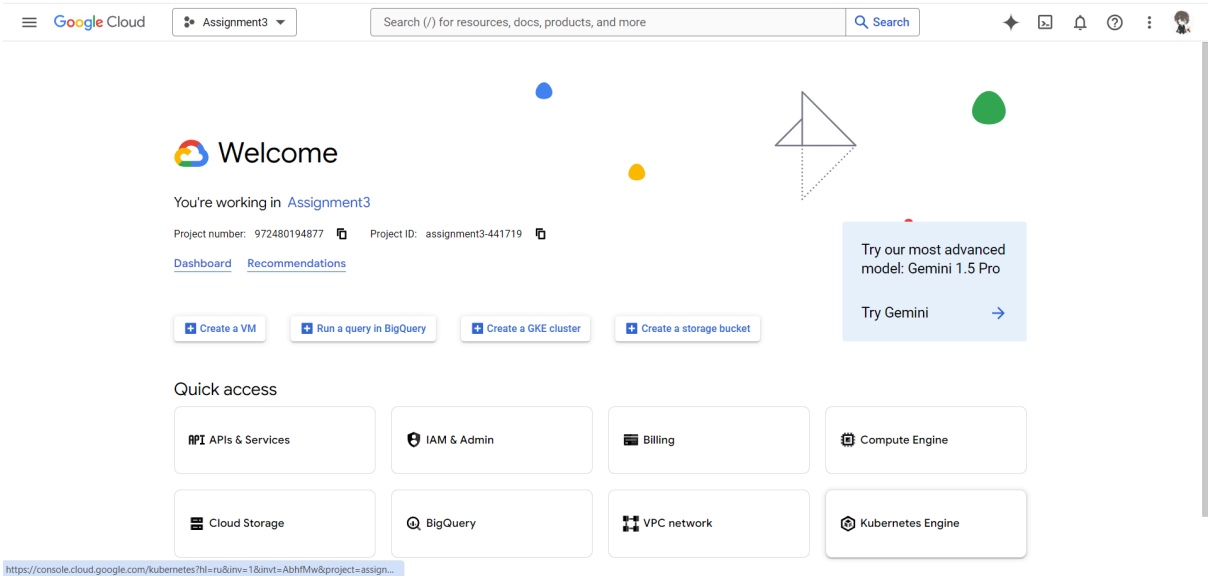

Image 4.1

And click Clusters in Google Cloud Console (Image 4.2).

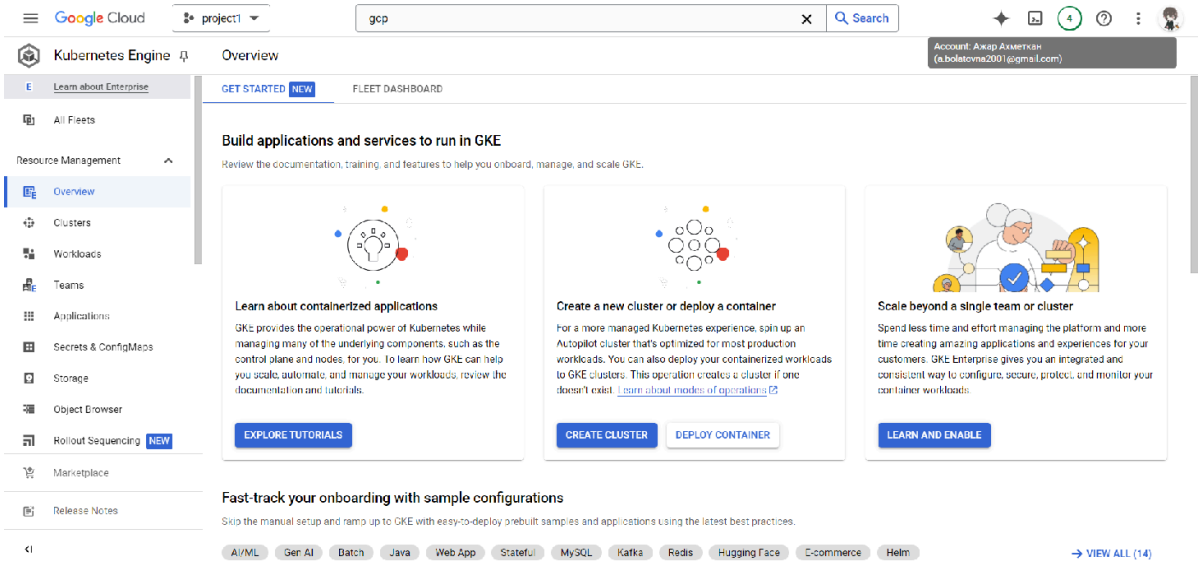

Image 4.2

configured the settings, such as choosing the cluster type, specifying the number of nodes, and selecting the region (Image 4.3).

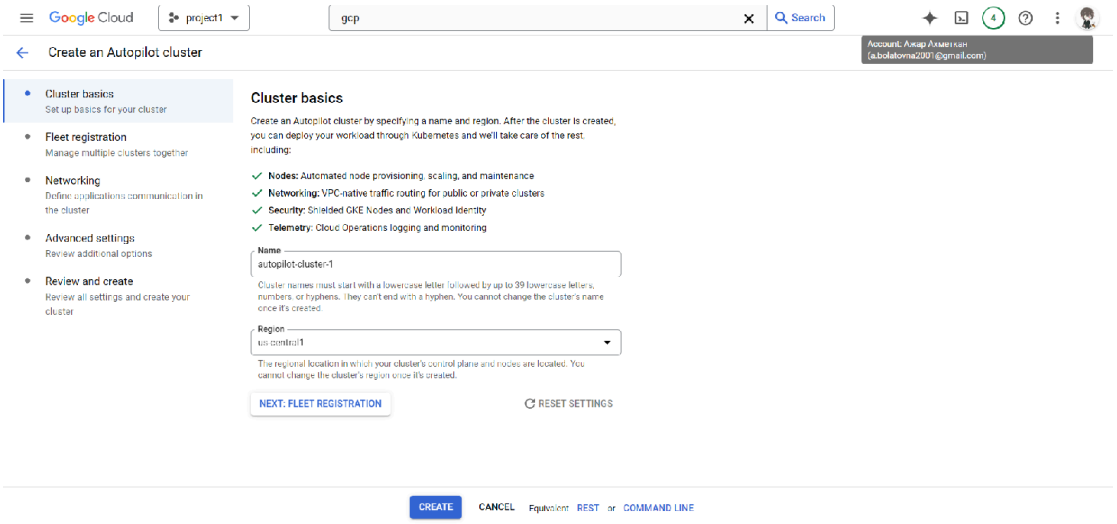

Image 4.3

After reviewing the settings, I clicked “Create” to launch the cluster (Image 4.4).

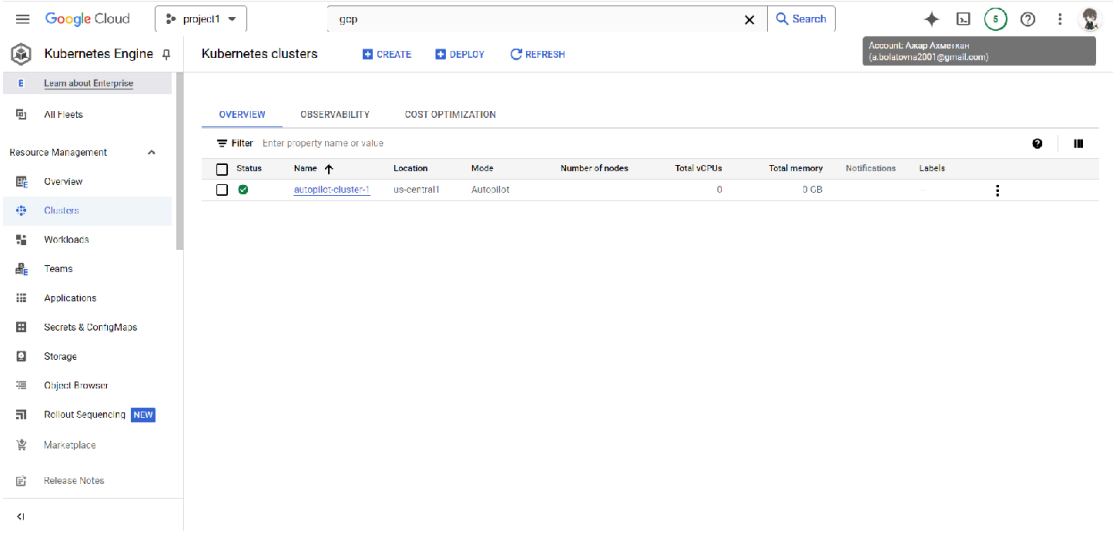

Image 4.4

Next, a Docker image for the containerized application must be created and uploaded to the Google Container Registry (GCR). I opened the Terminal, using Google Cloud Shell, opened it from the Google Cloud Console and used a text editor nano in the terminal to create a Dockerfile (Image 4.5).

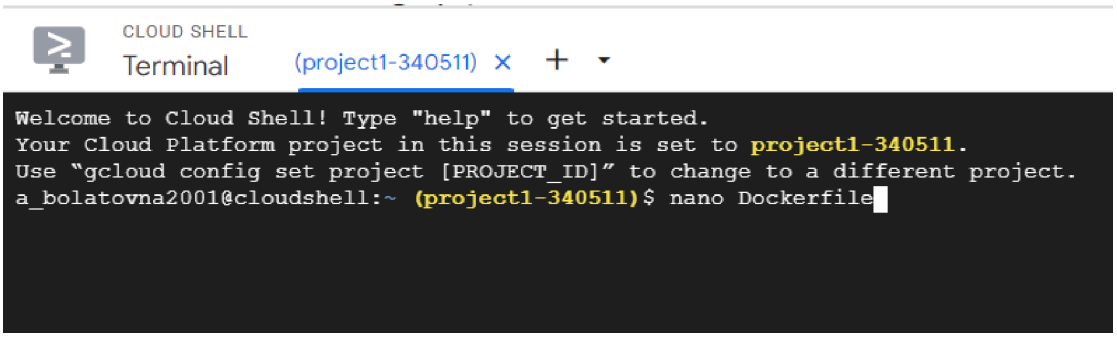

Image 4.5

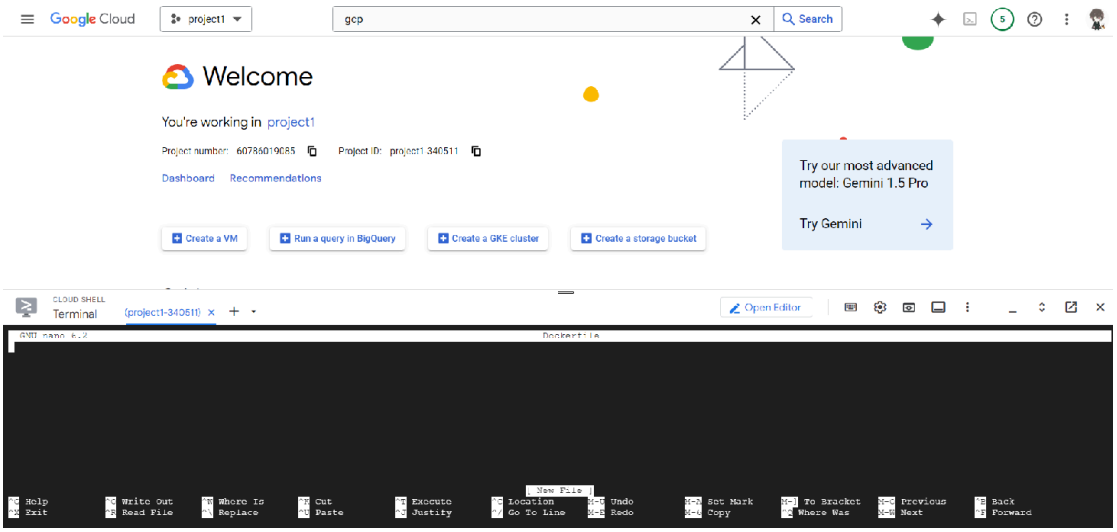

Image 4.6

Write the necessary instructions simple web application using Nginx in the Dockerfile and save it with pressing Ctrl + O (Image 4.7).

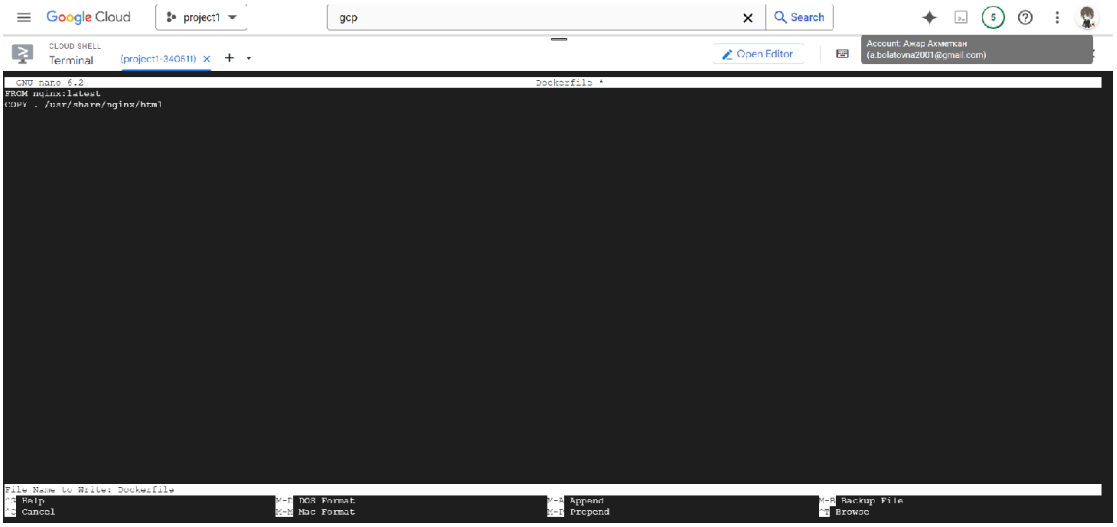

Image 4.7

After that, a Kubernetes Deployment can be created using a YAML file that will deploy the application to the GKE cluster. LoadBalancer is used as a service to ensure the application is accessible from the Internet.

After completing the exercise, you will have created and configured a working GKE cluster on which a simple containerized application, such as the "Hello World" application, is deployed. The application will be exposed via a LoadBalancer service, making it accessible from the Internet. This process will give you an understanding of how to create a cluster, deploy a container, and configure the application to be accessible via the Kubernetes cloud infrastructure.

Exercise 5: Managing Pods and Deployments

Objective is to learn how to manage deployments in Kubernetes, scale applications, and update them.

In this exercise, you create a multi-container application by preparing a Kubernetes Deployment YAML file. This file specifies the containers that will run in a single Pod. Once

the application is deployed, you can scale the number of replicas using the kubectl scale command.

```
kubectl scale deployment multi-container-app --replicas=5
```

and can update with command

```
kubectl set image deployment/multi-container-app container-1=nginx:latest
```

You can also change the container image and perform the update through Kubernetes if you need to update the application.

After completing the exercise, a multi-container application will be deployed using Kubernetes YAML files to configure resources. The participant will learn how to scale the number of replicas in a Deployment and update the application by replacing the old container with a new image. This experience will help deepen the understanding of how to manage the lifecycle of applications in Kubernetes, as well as how to adapt scalable solutions to changing requirements.

Exercise 6: ConfigMaps and Secrets

Objective is to understand how to manage configuration data and sensitive information in GKE using ConfigMaps and Secrets.

To work with ConfigMaps and Secrets, you need to create these objects using the kubectl create configmap

```
kubectl create configmap app-config --from-literal=env=production
--from-literal=db-host=localhost
```

and kubectl create secret commands.

```
kubectl create secret generic app-secret --from-literal=api-key=your-api-key
```

ConfigMap is used to store configuration data such as environment settings, and Secret is used to store sensitive information such as API keys. Next, the Deployment YAML file is updated so that the application can use these objects by including them as environment variables.

The exercise will result in the creation of a ConfigMap and Secret, which will be used to store configuration data and sensitive information such as API keys. These resources will be injected into an application deployed to GKE, allowing the use of configuration data and secrets in a secure and controlled manner. The participant will learn how to work with two important Kubernetes tools for managing configurations and secrets.

Exercise 7: Deploying an App on App Engine

Objective is to learn how to deploy a simple web application to Google App Engine.

To complete this exercise, you need to create a simple web application, for example, on Flask or Node.js. An important step is to create an app.yaml file,

```
runtime: python39
entrypoint: gunicorn -b :$PORT main:app
```

which specifies the environment settings and other parameters for deployment to App Engine.

```
gcloud app deploy
```

After that, the application is deployed using the gcloud app deploy command, and it becomes accessible over the Internet.

Upon completion of the exercise, a web application, such as Flask or Node.js, will be deployed to Google App Engine. The app.yaml file configuration will provide the correct settings for deploying and scaling the application. The application will be accessible over the Internet, and the participant will be able to understand how to use the App Engine platform for quickly deploying web applications without having to worry about server infrastructure.

Exercise 8: Using Cloud Functions

Objective is to understand how to create and deploy server-side functions using Google Cloud Functions and trigger them when specific events occur.

Here you need to create a function that will respond to an event, such as uploading a file to Cloud Storage. To do this, write the function code that will perform the required action.

```
def file_upload_handler(event, context):
    print(f"File {event['exercise8']} uploaded.")
```

After writing the code, the function must be deployed via the Google Cloud Console or the gcloud functions deploy command.

```
gcloud functions deploy file_upload_handler --runtime python39 --trigger-resource
bucket-name --trigger-event google.storage.object.finalize
```

To test the function, you can upload a file to Cloud Storage and make sure it worked.

After completing the exercise, a Cloud Function will be written and deployed that will automatically run when a specific event occurs, such as uploading a file to Cloud Storage. The function will perform a pre-defined action, such as sending a notification. This will give the participant an understanding of how to use serverless functions to automate processes and handle events in the cloud.

Exercise 9: Monitoring and Logging

Objective is to learn how to set up monitoring and logging for applications deployed on App Engine and Cloud Functions.

To set up monitoring and logging, you need to enable the relevant tools in the Google Cloud Console. To do this, you need to set up Google Cloud Monitoring to collect metrics such as application performance and load. It is also important to set up Google Cloud Logging to collect and analyze application logs deployed on App Engine or Cloud Functions. This will allow you to analyze performance and find problems in your applications.

After completing the exercise, the participant will set up monitoring and logging tools for applications deployed on App Engine and Cloud Functions. Performance metrics and service logs will be collected, allowing the analysis of the application health and operation. The participant will learn how to use Google Cloud capabilities to monitor, troubleshoot, and optimize the operation of deployed services.

Code Snippets

This section provides key code snippets that can be used to complete various exercises.

Sample Python code for uploading a file to Cloud Storage for exercise 2 - Service Accounts:

```
from google.cloud import storage
from google.oauth2 import service_account

key_path = 'service_account_key_path.json'
credentials = service_account.Credentials.from_service_account_file(key_path)
client = storage.Client(credentials=credentials, project='project-id')
bucket_name = 'exercise2'
file_name = 'local_file_path.txt'
bucket = client.get_bucket(bucket_name)
blob = bucket.blob(file_name)
blob.upload_from_filename(file_name)
```

Example YAML file for Deployment of exercise 4 - Deploying a Simple Application:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: exercise4
spec:
  replicas: 3
  selector:
    matchLabels:
      app: exercise4
  template:
    metadata:
      labels:
        app: exercise4
    spec:
      containers:
      - name: exercise4
        image: gcr.io/project-id/hello-world:latest
        ports:
        - containerPort: 8080
```

Example YAML file for Service (LoadBalancer) of exercise 4 - Deploying a Simple Application:

```
apiVersion: v1
kind: Service
metadata:
  name: exercise4-service
spec:
  type: LoadBalancer
  selector:
    app: exercise4
```

```
ports:
  - protocol: TCP
    port: 80
    targetPort: 8080
```

Example app.yaml file for a Flask application of exercise 7 - Deploying an App on App Engine:

```
runtime: python39
entrypoint: gunicorn -b :$PORT main:app
env_variables:
  VAR_NAME: "value"
instance_class: F2
```

Example code for a Cloud Function that sends a notification when an object is created in Cloud Storage of exercise 8 -Using Cloud Functions:

```
import os
from google.cloud import storage
from google.cloud import pubsub_v1
def notify_on_upload(event, context):
    file_name = event['exercise8']
    publisher = pubsub_v1.PublisherClient()
    topic_name = os.environ['TOPIC_EXERCISE8']
    message_data = f"Файл {file_name} был загружен в Cloud Storage.".encode('utf-8')
    publisher.publish(topic_name, message_data)
```

Results

Exercise 1: Setting Up IAM Roles

In the first exercise, a new project was successfully created in Google Cloud and configured IAM roles for various users, including Viewer, Editor, and Owner. Roles were assigned to users correctly and documented the appropriate permissions for each. This allows to gain a deeper understanding of the importance of access control and how you can securely manage user permissions to cloud resources using IAM. The Viewer role provides read-only access, the Editor role allows you to modify resources, and the Owner role provides full access, including the ability to manage IAM.

I encountered a difficulty in correctly assigning permissions to each role. I resolved the issue by carefully reviewing the documentation and assigning roles according to user responsibilities.

Exercise 2: Service Accounts

A service account with the necessary rights to access Google Cloud Storage was created and an authentication key was successfully generated. Using this key, the Python script was able

to upload files to the Cloud Storage bucket, which confirmed that the task was completed correctly. The problem arose when setting up permissions for the service account. The documentation was examined and it was confirmed that the service account has the Storage Object Admin role, which allowed us to solve the problem.

Exercise 3: Organization Policies

The third exercise examined Google Cloud organizational policies and the application of restrictions to prohibit the creation of public IP addresses in the project. The policy was successfully activated at the project level, and it began to work as expected.

I encountered difficulty in distinguishing between the application of policies at the organizational and project levels. After further study of the documentation, it became clear that the policy should be configured at the project level, which was successfully done.

Exercise 4: Deploying a Simple Application on GKE

A GKE cluster was set up and an application was deployed using an Nginx container. The application was accessible via the LoadBalancer service and I was able to confirm that it was working by accessing it via the external IP.

The main issue was the delay in getting an external IP address for the LoadBalancer service. The issue was resolved after waiting a few minutes for the IP address to be assigned.

Exercise 5: Managing Pods and Deployments

A multi-container deployment was created using Kubernetes YAML files, the application was scaled and upgraded using the new container. The scaling and upgrade were successful and the application continued to run without errors.

The difficulty was in working with YAML files for multi-container pods. After understanding the syntax and structure of these files, the problem was solved.

Exercise 6: ConfigMaps and Secrets

ConfigMaps and Secrets were created to pass configuration data and securely store sensitive information such as API keys in GKE. The configurations were successfully used in containers, which confirmed the correct setup. The difficulty was in the correct reference to Secrets and ConfigMaps in the pod manifests.

It was necessary to understand how to correctly use these references via envFrom and env in pod specifications, the task was successfully solved.

Exercise 7: Deploying an App on App Engine

A simple Flask web application is created and deployed to Google App Engine. The application is accessible via the App Engine URL and the deployment process is successful.

There was a difficulty setting up the app.yaml file and Flask application configuration to work on App Engine. After making the necessary changes to the file and specifying the correct settings, the deployment was successful.

Exercise 8: Using Cloud Functions

A Cloud Function was written that is triggered when an object is uploaded to Cloud Storage. After uploading a test file to the bucket, the function was successfully called and performed its task.

Initially, there was an issue with assigning permissions to the Cloud Function so that it could work with Cloud Storage. After correctly setting permissions for the function's service account, the issue was resolved.

Exercise 9: Monitoring and Logging

Monitoring and logging was set up for both App Engine and Cloud Function applications. Google Cloud Monitoring was used to create dashboards and track service performance, and Google Cloud Logging was used to view logs and diagnose problems.

There were difficulties setting up filters for logging and routing logs in Google Cloud Logging. The issue was resolved by checking that the settings were correct and making sure that the logs were being transferred and displayed correctly in Cloud Logging.

Conclusion

During the assignment, you will gain useful hands-on experience with key Google Cloud services, allowing you to gain a deeper understanding of their role in modern cloud architectures. You will master working with identity and security management, which will demonstrate the importance of properly configuring access and protecting cloud resources. Working with Google Kubernetes Engine (GKE) will help you understand how container orchestration simplifies the deployment and scaling of applications. Learning about Google App Engine will show you how you can deploy applications without having to manage infrastructure, focusing on developing functionality. Finally, working with Google Cloud Functions will help you understand how easy it is to create functions that respond to events and integrate various cloud services to automate processes.

These components play a critical role in building modern cloud architectures. Identity and security management ensures data protection and effective access control, GKE and App Engine offer flexibility and scalability in application deployment, and Cloud Functions allow you to automate responses to events in the cloud. All these technologies together form a powerful platform for developing, deploying and managing cloud applications, providing security, scalability and flexibility in cloud solutions.

References

- Google Cloud. Google Cloud IAM documentation.
  Retrieved from https://cloud.google.com/iam/docs
- Google Cloud. Google Cloud Service Accounts documentation.
  Retrieved from https://cloud.google.com/iam/docs/service-accounts
- Google Cloud. Google Cloud Organization Policies documentation.
  Retrieved from
  https://cloud.google.com/resource-manager/docs/organization-policy/overview
- Google Cloud. Google Kubernetes Engine documentation.
  Retrieved from https://cloud.google.com/kubernetes-engine/docs
- Google Cloud. Kubernetes documentation.
  Retrieved from
  https://kubernetes.io/docs/concepts/workloads/controllers/deployment/
- Google Cloud. Kubernetes ConfigMaps and Secrets documentation.
  Retrieved from https://kubernetes.io/docs/concepts/configuration/configmap/
  Retrieved from https://kubernetes.io/docs/concepts/configuration/secret/
- Google Cloud. Google Cloud App Engine documentation.
  Retrieved from https://cloud.google.com/appengine/docs
- Google Cloud. Google Cloud Functions documentation.
  Retrieved from https://cloud.google.com/functions/docs
- Google Cloud. Google Cloud Monitoring and Logging documentation.
  Retrieved from https://cloud.google.com/monitoring/docs
  Retrieved from https://cloud.google.com/logging/docs

Appendices

Appendix A: Objectives and expected results of the exercises

№	Exercise	Objective	Expected outcome
1	Setting Up IAM Roles	Configure IAM roles for users in Google Cloud.	Creating a project, assigning IAM roles, documenting permissions.
2	Service Accounts	Create a service account to access Cloud Storage.	Create a service account and upload a file using a Python script.
3	Organization Policies	Apply organizational policies to restrict actions.	Apply and verify organizational policies (e.g., ban public IPs).
4	Deploying a Simple Application	Deploy a containerized application to GKE.	Create a GKE cluster, deploy the application, and configure access via LoadBalancer.
5	Managing Pods and Deployments	Manage scaling and updating applications in Kubernetes.	Scaling replicas, updating containers via Kubernetes.
6	ConfigMaps and Secrets	Use ConfigMaps and Secrets to manage configurations.	Create and deploy a ConfigMap and Secret for an application in GKE.
7	Deploying an App on App Engine	Deploy a web application to Google App Engine.	Develop and deploy an application with configuration via app.yaml.
8	Using Cloud Functions	Create and deploy a Cloud Function to handle events.	Write and deploy a Cloud Function that responds to an event (such as a file being uploaded).
9	Monitoring and Logging	Set up monitoring and logging for App Engine and Cloud Functions.	Collect metrics and logs, analyze application status and performance.

Appendix B: Downloaded key in json format


```
{
  "type": "service_account",
  "project_id": "assignment3-441719",
```

```
  "private_key_id": "407825b670015066e51ea14a5782dc1a31c0c6d0",
  "private_key": "-----BEGIN PRIVATE
KEY-----\nMIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQCh/3aoyH2
z86Zm\nCMMmpn00X3oyT4fVS3em9K1PgLiO8pl5YXzDKFH1KBf3waCLLPXAwGBpzxQ
W9krJ\npHio1JAcUJQK4j3uMa7QfbFk3g1lwIrlxEENWkSbynoJwSjP7WGeVT/zSyp15jHa\
nKVUkODtMBqk9pA0j7WsN6EwsugpEzDCvniDCFkiFMEPZxW+XecXIhGHVyuT/joNE\n
HHtqCsDQWnyPX9CeFJPqCVfVmo8UYP6VUM5QhXcWZ+dY+ohwyUDNqcJO3Z+9ZtU
Q\nXnprcgZMHmd8QHUacrXpwLswzU2OyVqawsRVBsXFMLk0O29RTQ9bbwz6es0EIKD
c\nWZoOyAXPAgMBAAECggEACN2Enl7+dhMe55WSLHLPBkc1VjyA9/y9yw63z1KBhx0i\
npgBeRgXWWl671QytHwCyYD6G6DNza4BF/nv4OHENg+cq+t9/uSHR6cn/I8B9JjTf\nxQ
w07Sij+YyemhsOgzZ/8ZzDt6pE5BfZYZGCfc1v/UyeBWgNagproeK1DHuP5Uzb\nB49OSi
+/OxdURYcfYYBR4Q92b9W3F4gnrR9Ct8mohSiz5i/tv7XBTEE7G1MNPgdO\nJGhyyjcloB
QqabhdIiHbCthWsPsDPGZIuLTbhQIQK03jB8DnNqpgCvp7XUkRzxl8\nGkk7piKwWyzOy
q4o4wItKM5+7bvZ08sY1jMntLZ0mQKBgQDX0L56lh5fMCwIlKFL\nMStzX1kG+rpE5Gd
EVp5GrVJ7FfzEEbOEEtJFgfzjTn40IDsQpIUv7QTi/aBC+IZD\nygdHdjaTfLAOqSt2zUjTD
IhbhP3Ljo/2lJ+uvNzSqUREK9XLbfDxDoM/PhNCOu97\ncErIOQX79Osp+OxfIXvDZfXnt
wKBgQDAKWei7LBAyFxAWSKBvjjPS+pUO2E3oCZo\n/FgjCCPs39wXt9LG13SgtJmwIA
dLNX32hsqa82gJGy+bGw3iwXREBnGJ4KdAiMrC\ncV462VJLtfTv5UV/8gc6Cw4Rlr5itH
sOtaobjkv1LLPBUM9/u9DoTxUbtk+IzUCz\n/HbYfTJiqQKBgQC5uUNi3XOPuTX7wGS5/
7uD0wcbcYQBK4oPDnexHRFBKZ/X39/c\nBhJ8jKvjtbX4B9CsttiDcPbmnRtg5t0s01zGS94
VuJUOWNk1qN5F+aATGnUKy0nr\nCuYMiy2CAPblr8+R/K//0txx/Wq3cV6MnQtOGXyj4
pbKDRTlnTwT4NfpKwKBgFuC\nHCk4/4IKJ+w4xjb6fu0woKl2EYi0yjXKnYopMTp3LE9b
KZOpPI/fESUPS4nkMlT4\nwXjimtNjT9tUGdu29lACpYTcJ6mSusO3ywDJRuhpAPhQvTT9
o7VfWFbF62oDumo5\n1oUC0Wcy33MasYWJgiVuq4Wj3uItpDdR7aKQlDLpAoGBAJYA3
4+TGGd1YgFVwIMy\n+O84RLQtVEDapQFBTN7tfxzosq5x+pGW2LSd2TweswKDswX43
4RjuN36deaho3Zt\n5JFeRcqkzplucm6S9GWQSvpuJQNqpvE8paRU1rbztEY9TsgkArw2loM
ws5qlizNf\np2kb3hNv1XXyZzdeDQqF4uYf\n-----END PRIVATE KEY-----\n",
  "client_email": "exercise2@assignment3-441719.iam.gserviceaccount.com",
  "client_id": "107036315796759584675",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/exercise2%40assignment3-441719.i
am.gserviceaccount.com",
  "universe_domain": "googleapis.com"
}
```