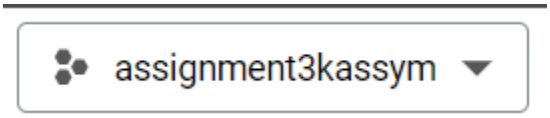
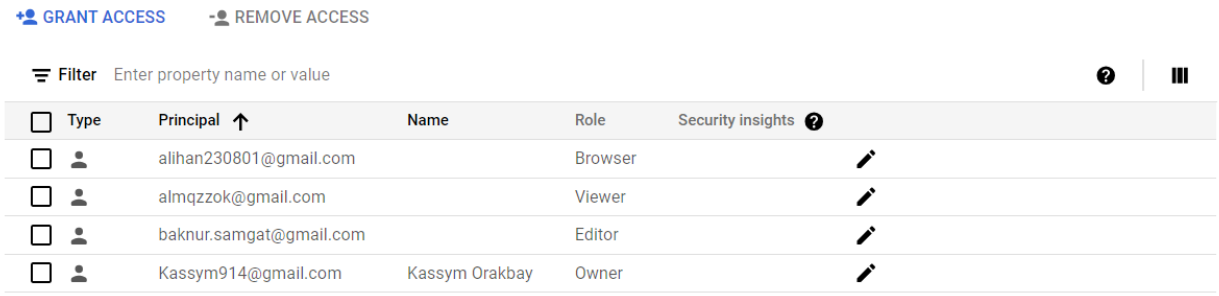
1. Identity and Security Management

Exercise 1: Setting Up IAM Roles

- Create a new Google Cloud project.



- Set up Identity and Access Management (IAM) roles for different team members (e.g., Viewer, Editor, Owner).

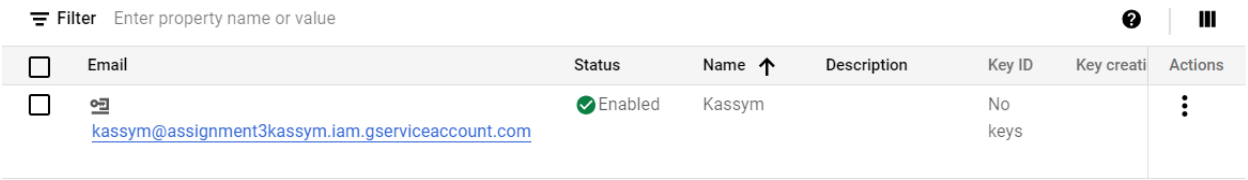


- Assign specific roles to users and document the permissions associated with each role.

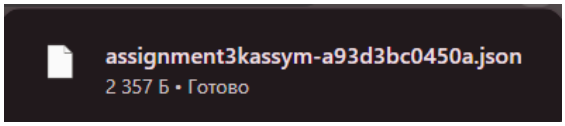
  Browser - Access to browse GCP resources.
  Editor - View, create, update, and delete most Google Cloud resources. See the list of permissions included.
  Owner - Full access to most Google Cloud resources. See the list of permissions included.
  Viewer - View most Google Cloud resources. See the list of included permissions.

Exercise 2: Service Accounts

- Create a service account that can access Google Cloud Storage.



  Generate and download a key for this service account, and use it to authenticate a Python script that uploads a file to a Cloud Storage bucket.




```python
from gcloud import storage
from oauth2client.service_account import ServiceAccountCredentials
import json


1 usage
class Uploader:
    def __init__(self, creds):
        self.creds = creds

    def upload_file(self):
        credentials_dict = {
            'type': self.creds['type'],
            'client_id': self.creds['client_id'],
            'client_email': self.creds['client_email'],
            'private_key_id': self.creds['private_key_id'],
            'private_key': self.creds['private_key'],
        }
        credentials = ServiceAccountCredentials.from_json_keyfile_dict(
            credentials_dict
        )
        client = storage.Client(credentials=credentials, project='assignment3kassym')
        bucket = client.get_bucket('mybucket')
        blob = bucket.blob('myfile.txt')
        blob.upload_from_filename('myfile.txt')


1 usage
def read_json():
    with open('assignment3kassym-a93d3bc0450a.json') as f:
        data = json.load(f)
```

2. Google Kubernetes Engine (GKE)

Exercise 4: Deploying a Simple Application

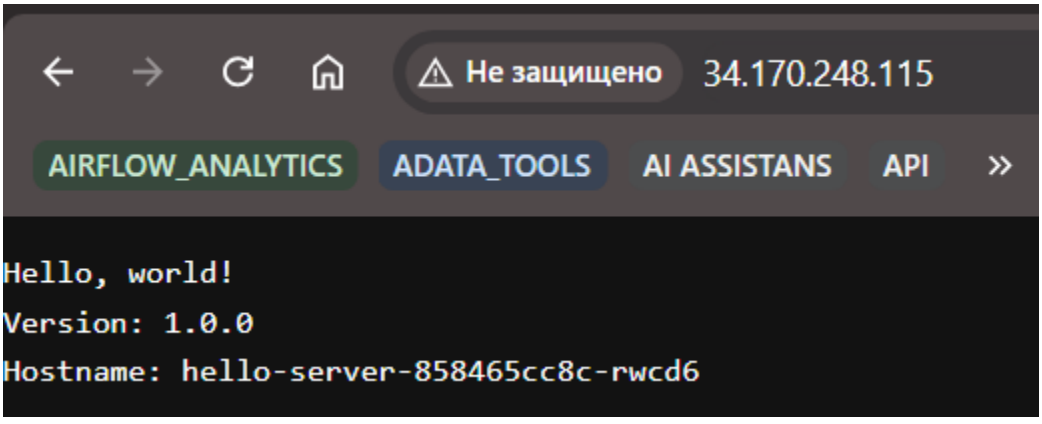
- Set up a GKE cluster using the Google Cloud Console or gcloud command line.

```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to assignment3kassym.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
kassym914@cloudshell:~ (assignment3kassym)$ gcloud config set project assignment3kassym
Updated property [core/project].
kassym914@cloudshell:~ (assignment3kassym)$ gcloud container clusters create-auto hello-clus
    --location=us-central1
Note: The Kubelet readonly port (10255) is now deprecated. Please update your workloads to u
kubelet-readonly-port for ways to check usage and for migration instructions.
Creating cluster hello-cluster in us-central1...working.
```

- Deploy a simple containerized application (e.g., a Hello World app) to the cluster and expose it via a LoadBalancer service.

```
STATUS: RUNNING
kassym914@cloudshell:~ (assignment3kassym)$ gcloud container clusters get-credentials hello-cluster \
    --location us-central1
Fetching cluster endpoint and auth data.
kubeconfig entry generated for hello-cluster.
kassym914@cloudshell:~ (assignment3kassym)$ kubectl create deployment hello-server \
    --image=us-docker.pkg.dev/google-samples/containers/gke/hello-app:1.0
Warning: autopilot-default-resources-mutator:Autopilot updated Deployment default/hello-server: defaulted unsp
efaults).
deployment.apps/hello-server created
```

```
deployment.apps/hello-server created
kassym914@cloudshell:~ (assignment3kassym)$ kubectl expose deployment hello-server \
    --type LoadBalancer \
    --port 80 \
    --target-port 8080
service/hello-server exposed
kassym914@cloudshell:~ (assignment3kassym)$
```



Exercise 5: Managing Pods and Deployments

- Create a Deployment for a multi-container application using Kubernetes YAML files.

```
kassym914@cloudshell:~ (assignment3kassym)$ touch multi-container-deployment.yaml
echo "
apiVersion: apps/v1
kind: Deployment
metadata:
```

```
kassym914@cloudshell:~ (assignment3kassym)$ kubectl apply -f multi-container-deployment.yaml
```

```
kassym914@cloudshell:~ (assignment3kassym)$ touch multi-container-service.yaml
echo "
apiVersion: v1
kind: Service
metadata:
  name: multi-container-service
spec:
  selector:
    app: multi-container-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: NodePort
" > multi-container-service.yaml
```

```
kassym914@cloudshell:~ (assignment3kassym)$ kubectl apply -f multi-container-service.yaml
Warning: resource services/multi-container-service is missing the kubectl.kubernetes.io/last-appli
sed on resources created declaratively by either kubectl create --save-config or kubectl apply. Th
service/multi-container-service configured
kassym914@cloudshell:~ (assignment3kassym)$ kubectl get services
NAME                    TYPE           CLUSTER-IP       EXTERNAL-IP      PORT(S)        AGE
hello-server            LoadBalancer   34.118.225.179   34.170.248.115   80:30405/TCP   14m
kubernetes              ClusterIP      34.118.224.1     <none>           443/TCP        19m
multi-container-service NodePort       34.118.225.50    <none>           80:30215/TCP   2m59s
```

- Scale the Deployment to manage the number of replicas and update the application with a new container image.

```
kassym914@cloudshell:~ (assignment3kassym)$ kubectl scale deployment multi-container-app --replicas=5
deployment.apps/multi-container-app scaled
```

```
kassym914@cloudshell:~ (assignment3kassym)$ touch multi-container-deployment.yaml
echo "
apiVersion: apps/v1
kind: Deployment
metadata:
  name: multi-container-app
spec:
  replicas: 5  # Update the replicas count here
  selector:
```

```
kassym914@cloudshell:~ (assignment3kassym)$ kubectl get deployments
kubectl get pods -l app=multi-container-app
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
hello-server          1/1     1            1           19m
multi-container-app   2/5     5            2           9m41s
NAME                                   READY   STATUS    RESTARTS   AGE
multi-container-app-b94d76fdb-2j2ds    0/2     Pending   0          105s
multi-container-app-b94d76fdb-2qqqp    0/2     Pending   0          105s
multi-container-app-b94d76fdb-5fmjs    2/2     Running   0          9m42s
multi-container-app-b94d76fdb-bk5pn    0/2     Pending   0          105s
multi-container-app-b94d76fdb-kl7b5    2/2     Running   0          9m42s
kassym914@cloudshell:~ (assignment3kassym)$
```

Exercise 6: ConfigMaps and Secrets

- Implement ConfigMaps and Secrets in your GKE application.

```
kassym914@cloudshell:~ (assignment3kassym)$ kubectl apply -f configmap.yaml
Warning: resource configmaps/app-config is missing the kubectl.kubernetes.io/last-applied-confi
ources created declaratively by either kubectl create --save-config or kubectl apply. The missi
configmap/app-config configured
kassym914@cloudshell:~ (assignment3kassym)$ kubectl get configmap app-config -o yaml
apiVersion: v1
data:
  app_mode: production
```

```
kassym914@cloudshell:~ (assignment3kassym)$ touch secret.yaml
echo "
apiVersion: v1
kind: Secret
metadata:
  name: app-secret
type: Opaque
data:
  db_password: $(echo -n 'mysecretpassword' | base64)
" > secret.yaml
kassym914@cloudshell:~ (assignment3kassym)$ kubectl apply -f secret.yaml
Warning: resource secrets/app-secret is missing the kubectl.kubernetes.io/las
ces created declaratively by either kubectl create --save-config or kubectl a
secret/app-secret configured
```

- Use a ConfigMap to pass configuration data and a Secret to manage sensitive information (e.g., API keys).

```
kassym914@cloudshell:~ (assignment3kassym)$ touch secret.yaml
echo "
apiVersion: v1
kind: Secret
metadata:
  name: app-secret
type: Opaque
data:
  api_key: $(echo -n 'my-sensitive-api-key' | base64)
" > secret.yaml
kassym914@cloudshell:~ (assignment3kassym)$ kubectl apply -f secret.yaml
secret/app-secret configured
```

```
kassym914@cloudshell:~ (assignment3kassym)$ kubectl exec -it multi-container-app-6c46c4dd98-sjbq7 -- printenv
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=multi-container-app-6c46c4dd98-sjbq7
NGINX_VERSION=1.21.6
NJS_VERSION=0.7.3
PKG_RELEASE=1~bullseye
DATABASE_HOST=db.example.com
DATABASE_PORT=5432
APP_MODE=production
API_KEY=my-sensitive-api-key
```

3. App Engine and Cloud Functions

Exercise 7: Deploying an App on App Engine

- Create a simple web application (e.g., a Flask or Node.js app) and deploy it to Google App Engine.

```
C:\Users\kassy\AppData\Local\Google\Cloud SDK>gcloud services enable cloudbuild.googleapis.com
Operation "operations/acf.p2-238610034257-37d2407e-88f3-47e0-951f-5f119efa394c" finished successfully.

C:\Users\kassy\AppData\Local\Google\Cloud SDK>gcloud app create --project=assignment3kassym
You are creating an app for project [assignment3kassym].
WARNING: Creating an App Engine application for a project is irreversible and the region
cannot be changed. More information about regions is at
<https://cloud.google.com/appengine/docs/locations>.
```

```
C:\Users\kassy\Desktop\masters\CloudComputing\assignment3\flask>gcloud app deploy
Services to deploy:

descriptor:                  [C:\Users\kassy\Desktop\masters\CloudComputing\assignment3\flask\app.yaml]
source:                      [C:\Users\kassy\Desktop\masters\CloudComputing\assignment3\flask]
target project:              [assignment3kassym]
```

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

Beginning deployment of service [default]...
Created .gcloudignore file. See `gcloud topic gcloudignore` for details.
#============================================================#
#= Uploading 692 files to Google Cloud Storage              =#
#============================================================#
File upload done.
Updating service [default]...done.
Setting traffic split for service [default]...done.
Deployed service [default] to [https://assignment3kassym.de.r.appspot.com]
```

- Configure app.yaml for the deployment, specifying runtime and service settings.

Version	Status	Traffic Allocation	Instances	Runtime	Runtime Lifecycle
20241117t174514	Serving	100%	0	python310	General Availability

-

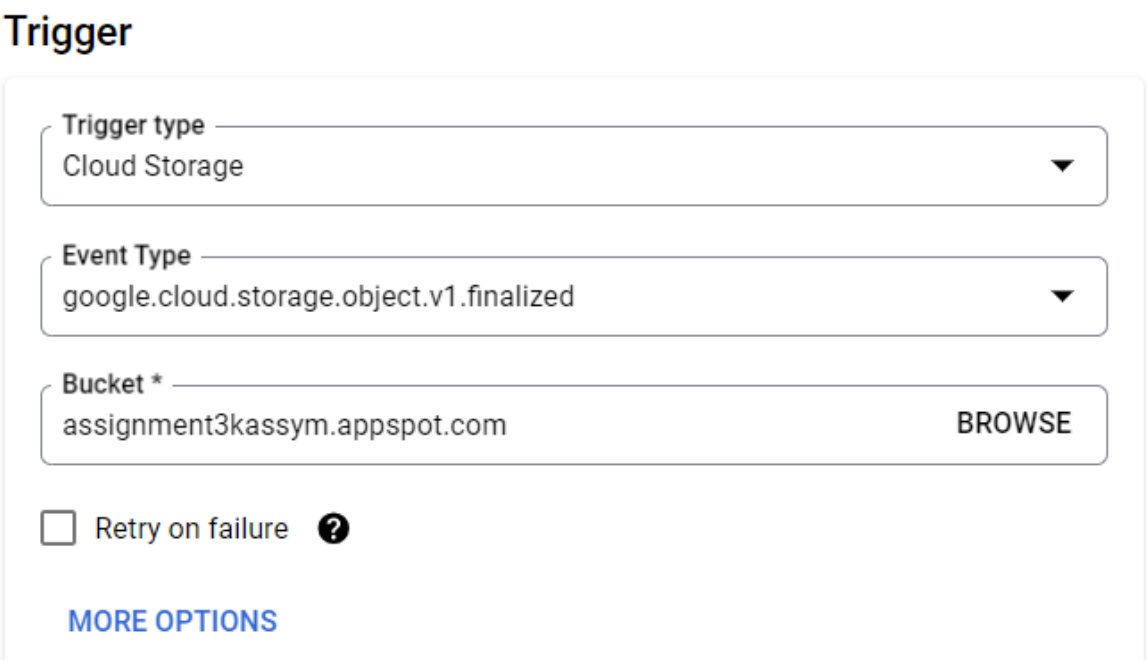
```yaml
runtime: python310
service: flask-service
entrypoint: gunicorn -w 2 -b :$PORT main:app

instance_class: F2
automatic_scaling:
  target_cpu_utilization: 0.65
  target_throughput_utilization: 0.75
  max_instances: 5
  min_instances: 1
```

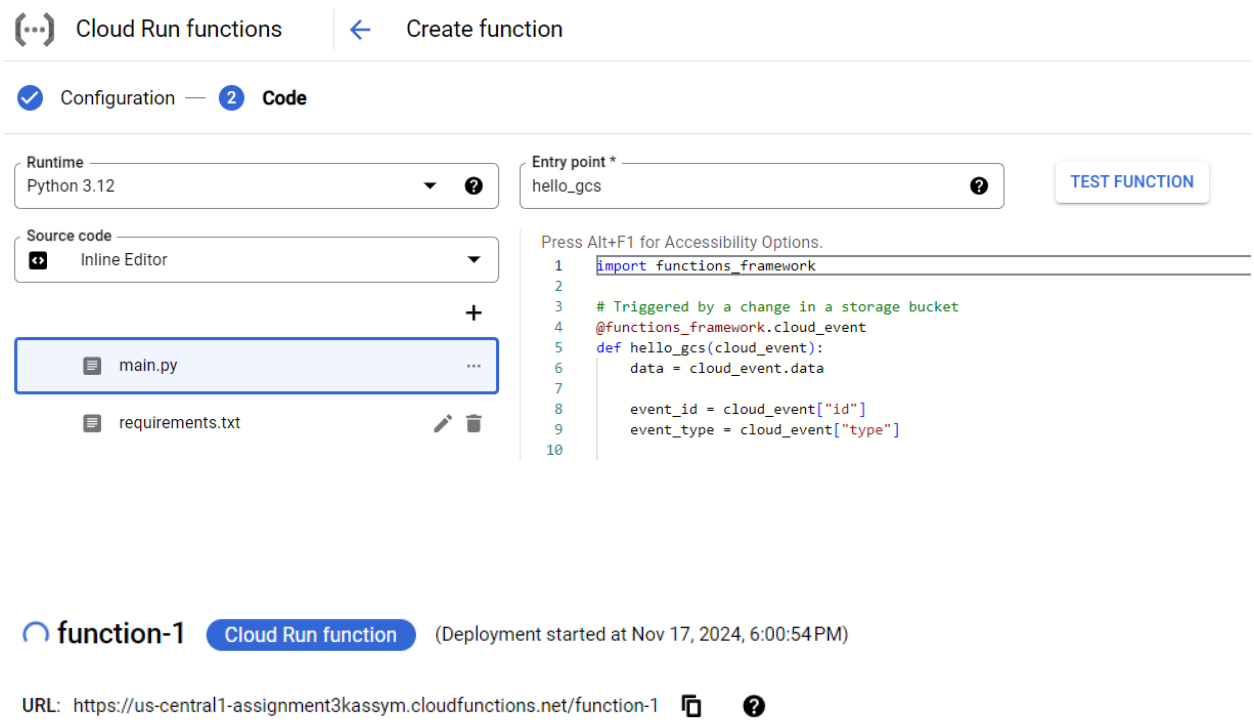
-

Exercise 8: Using Cloud Functions

- Write a Cloud Function that triggers on a specific event (e.g., an object creation in Cloud Storage) and performs a task (e.g., sending a notification).

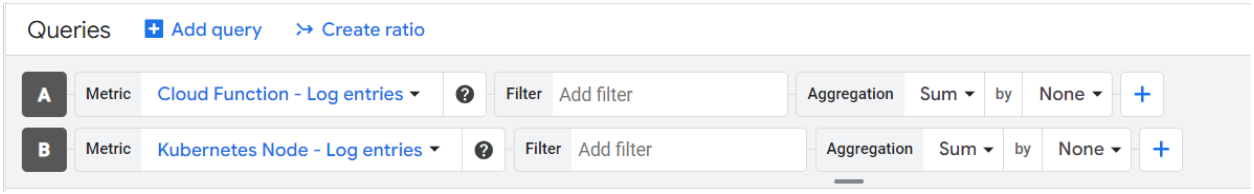


- Deploy the function and test it by uploading a file to the designated Cloud Storage bucket.



Exercise 9: Monitoring and Logging

- Set up monitoring and logging for your App Engine application and Cloud Functions.



1. **Results**
    - All exercises done successfully
    - Given tasks sometimes not understandable, it's complicated to figure out what to do.
2. **Conclusion**
    - Summarize the learning experience.
    - It's important to keep your projects secure by assigning roles and permissions. Especially when working with GKE, as it has high functionality and stands as a base for deploy.
3. **References**
    - - https://cloud.google.com/appengine/docs/standard/monitoring-and-alerting-latency
4. **Appendix**
    - - https://medium.com/@dmahugh_70618/deploying-a-flask-app-to-google-app-engine-faa883b5ffab