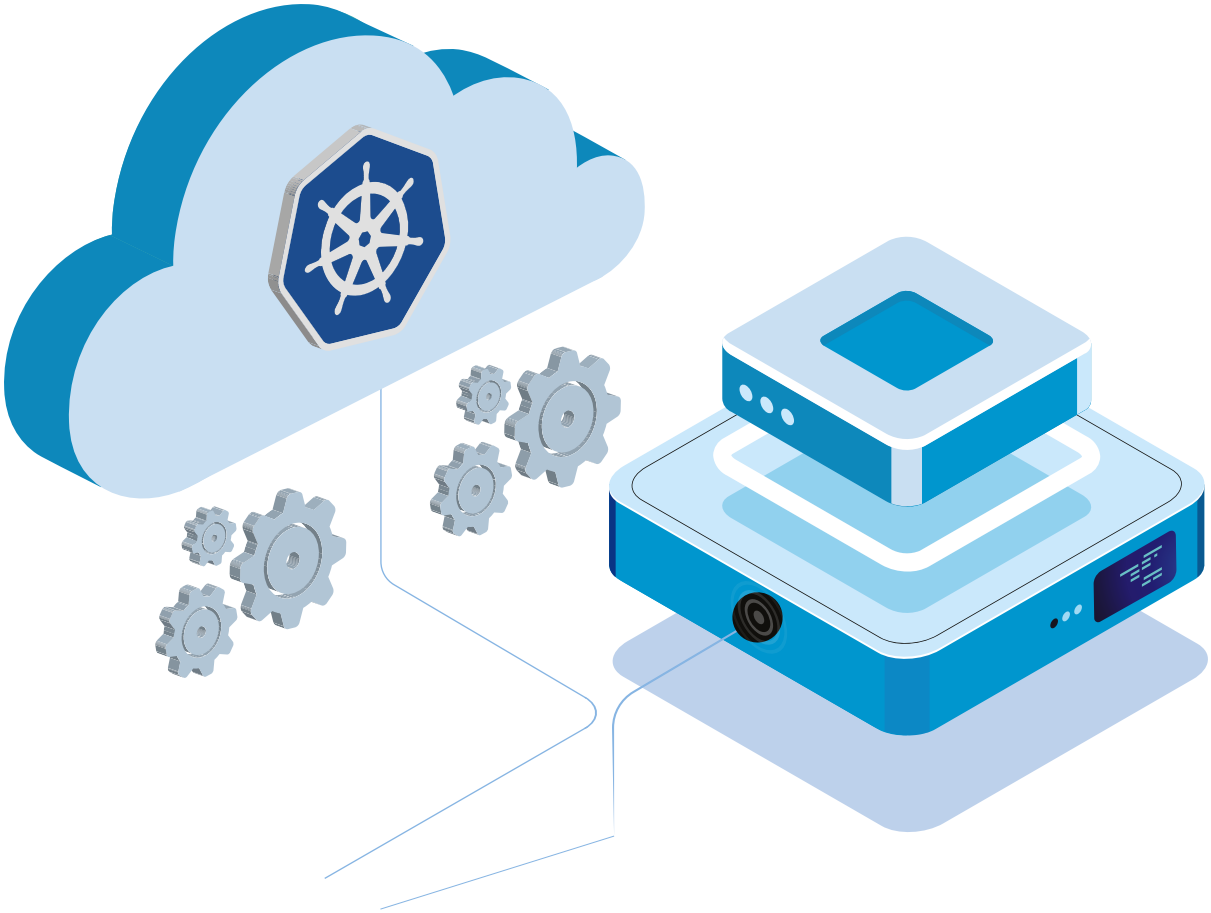


Department of Computer Science & Engineering
CSL7090 - Software & Data Engineering (SDE)

Assignment 2

Prepared by: Aviral Tripathi
Roll no: m22ma012
Instructor: Dr. Sumit Kalra

Contents

1 Task 1: Setting Up a Virtual Machine (VM) on GCP

1.1 Create a GCP account if you don't have one.

I have created a GCP account and loaded my $50 credit in it, and also created a new project by the name of 'Assignment2' under the institution iitj.ac.in, here's my GCP dashboard.

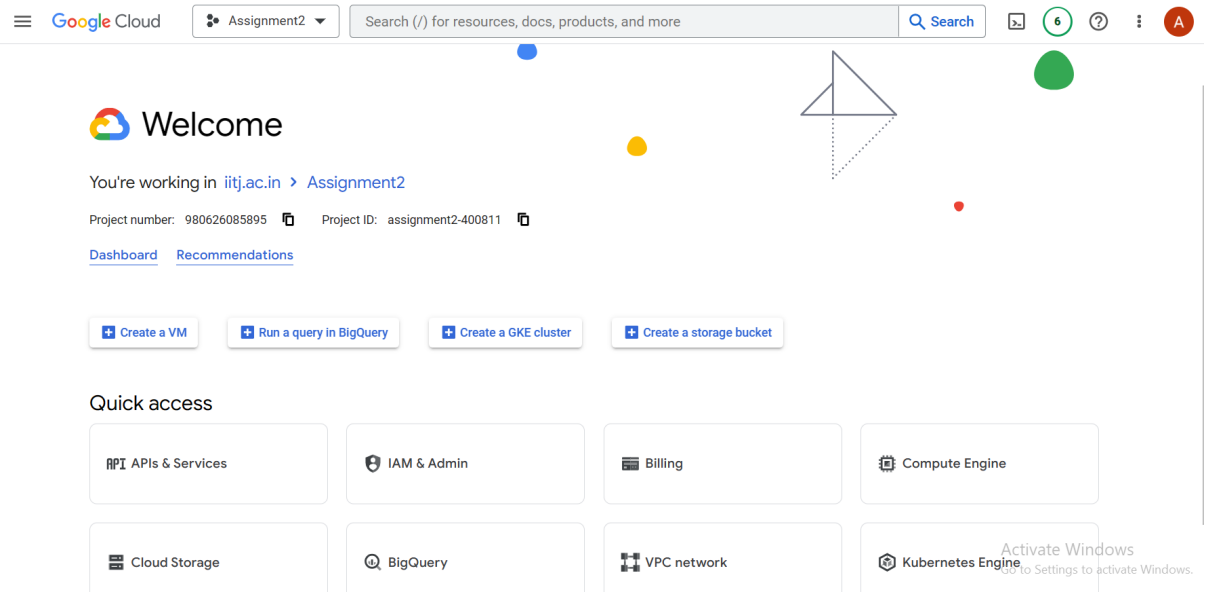


Figure 1.1: My GCP Dashboard

1.2 Set up a Virtual Machine instance using Google Compute Engine.

I have created a Virtual Machine (VM), by the name 'vm', and the following configuration details.

- Location: Default
- Type: E2-micro (0.25-2 vCPU, 1 shared core): chosen to reduce cost
- RAM: 1GB
- Operating System: Ubuntu: chosen because of familiarity with CLI commands
- Storage: 50GB

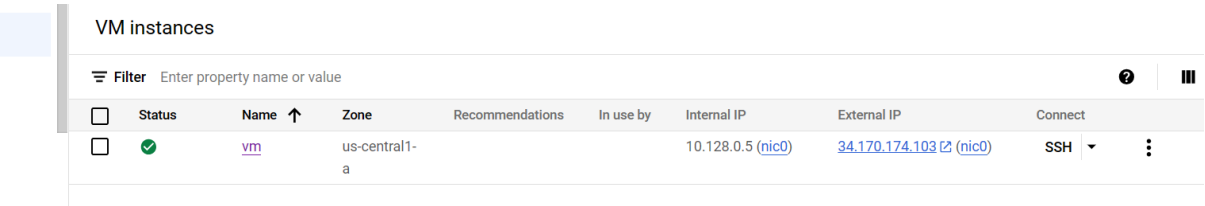


Figure 1.2: Virtual Machine created (vm)

1.3 Access the VM using SSH from your local machine.

We can access the VM by using ssh (secure shell) command from out local machine, by using the external IP address of the VM, this external IP can be obtained from the GCP compute engine dashboard.

```
ssh m22ma012@34.170.174.103
```

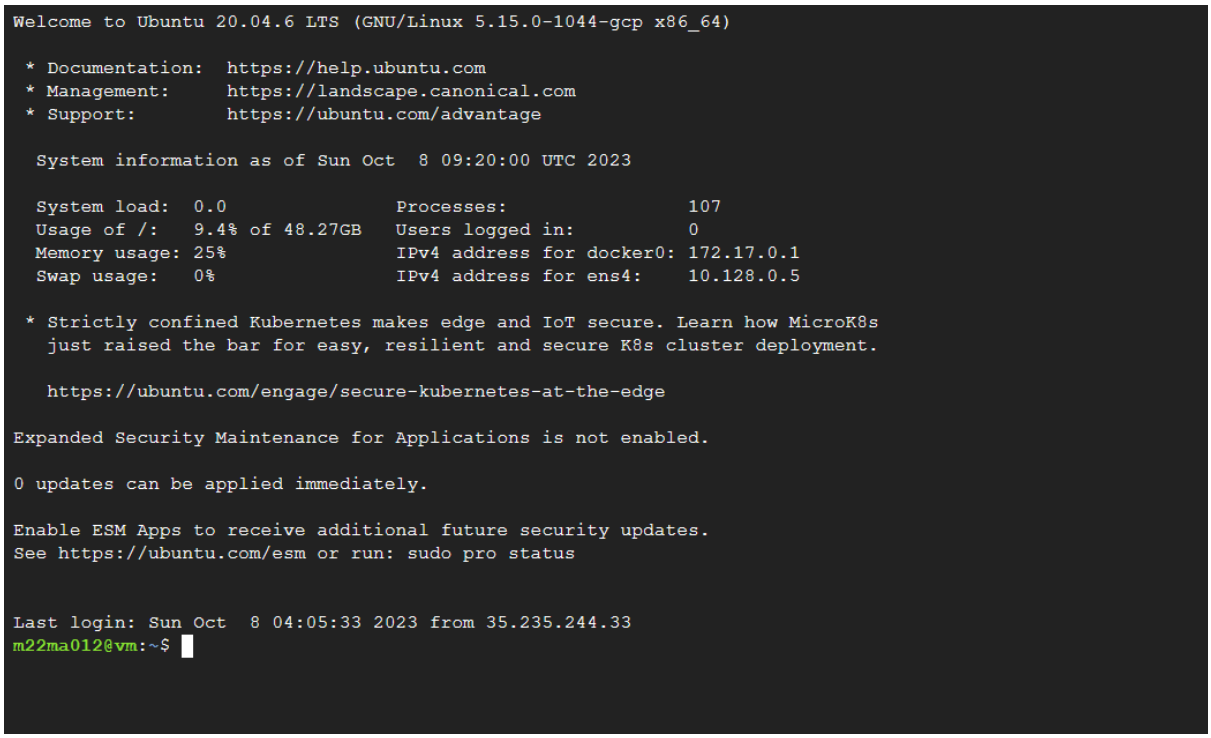


Figure 1.3: Accessing the Virtual Machine using SSH

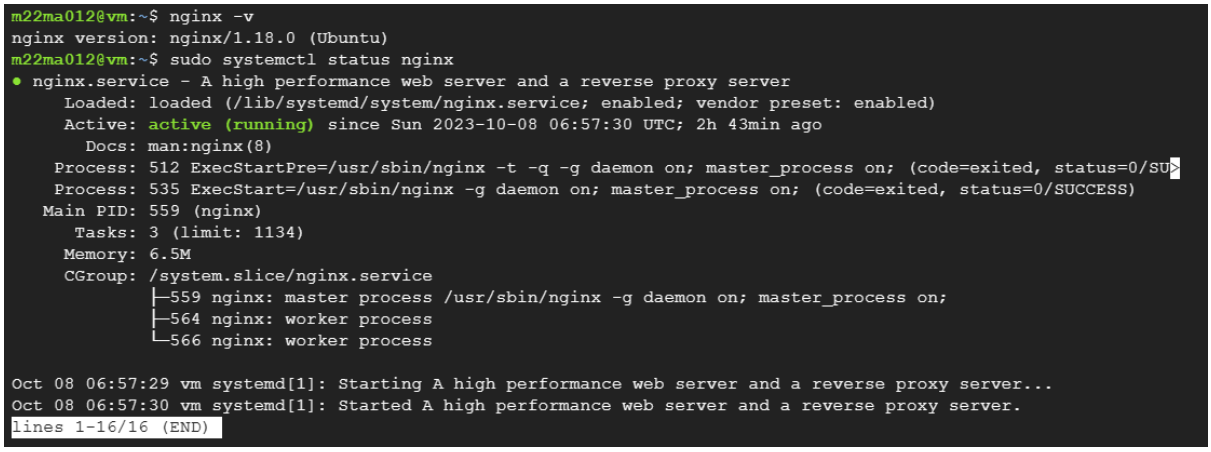
1.4 Install Nginx web server on the VM.

We can install Nginx on our VM by using the following commands:

```
sudo apt update
sudo apt install nginx
```

These commands will install Nginx, and we can now start Nginx and check the status using the commands:

```
sudo systemctl start nginx
sudo systemctl status nginx
```



```
m22ma012@vm:~$ nginx -v
nginx version: nginx/1.18.0 (Ubuntu)
m22ma012@vm:~$ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2023-10-08 06:57:30 UTC; 2h 43min ago
       Docs: man:nginx(8)
    Process: 512 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SU>
    Process: 535 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Main PID: 559 (nginx)
      Tasks: 3 (limit: 1134)
     Memory: 6.5M
     CGroup: /system.slice/nginx.service
             ├─559 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
             ├─564 nginx: worker process
             └─566 nginx: worker process

Oct 08 06:57:29 vm systemd[1]: Starting A high performance web server and a reverse proxy server...
Oct 08 06:57:30 vm systemd[1]: Started A high performance web server and a reverse proxy server.
lines 1-16/16 (END)
```

Figure 1.4: Nginx is Active and running on the VM.

1.5 Display a custom webpage through the web server to demonstrate successful setup.

To display a custom webpage, I have modified the index.html file, which is the default file displayed, when we enter the external IP of VM in a browser, I have modified that file with a simple web page having "Assignment 2" as title and my name under it.

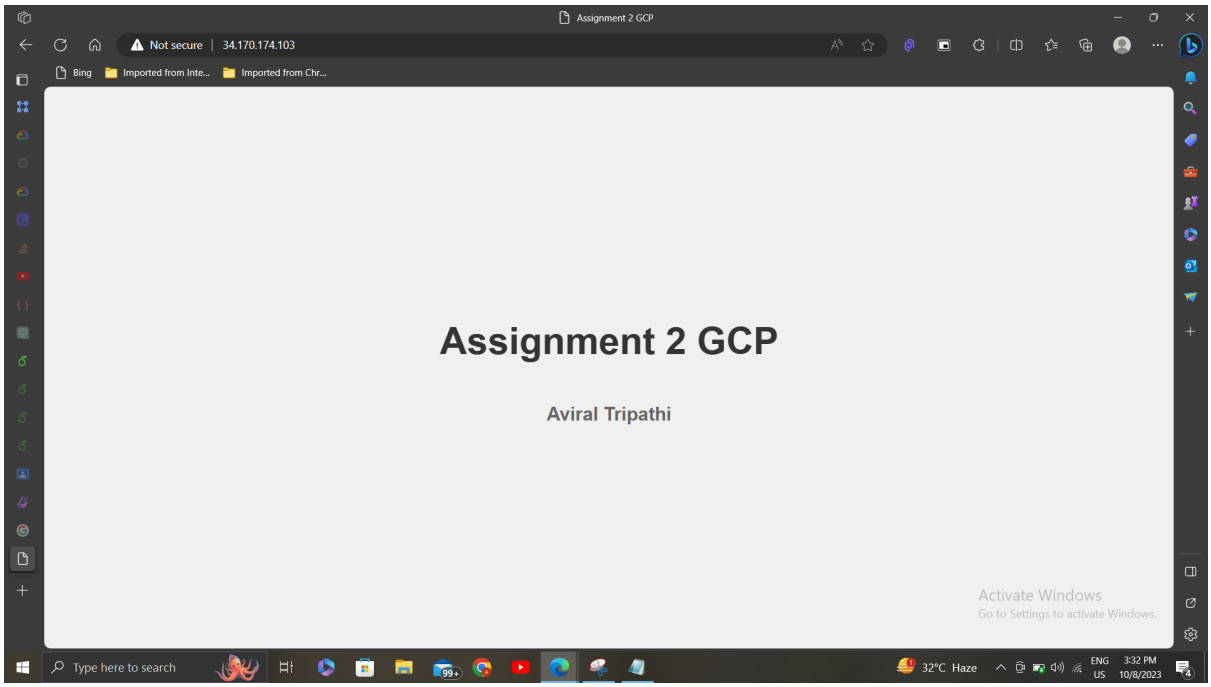


Figure 1.5: Custom webpage for my VM.

2 Task 2: Docker Containerization

2.1 Installing docker on VM

For installing docker I used the snap package management system, I used the following commands for installing docker:

```
sudo apt update
sudo snap install docker
```

to verify, we can check the docker version

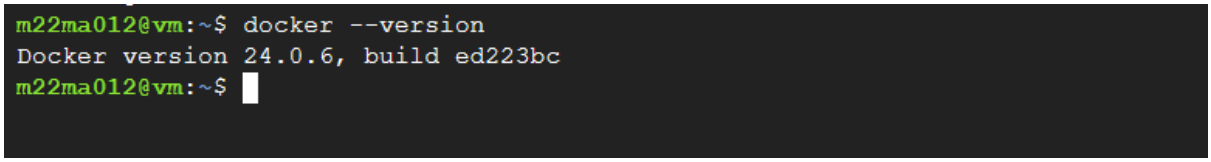


Figure 2.1: Docker installed successfully on VM.

to ensure smoother operations of next tasks, it is important ro run Docker without 'sudo', for that we can use these commands:

```
sudo groupadd docker
sudo usermod -aG docker $USER
newgrp docker
```

Now we can use docker without 'sudo'.

2.2 Build a Docker image for a simple application

I have created a directory called "docker-app", and placed a python file app.py in it, this file has a simple print statement that prints "hello-docker"

For creating an image of this app, i have made a file 'Dockerfile' (without extension) in the same directory (docker-app), this file has a blueprint about how to create a docker image.

Run this command to build docker image of the directory 'docker-app' (as Dockerfile is in the same directory, we use a dot to get the adress of the file):

```
docker build -t docker-app .
```

we can now finf all the created images in the 'docker images'

```
m22ma012@vm:~/docker-app$ docker images
REPOSITORY                                                                    TAG       IMAGE ID       CREATED        SIZE
docker-app-image                                                              latest    9f91320be12f   5 days ago     139MB
docker-app                                                                    latest    9f91320be12f   5 days ago     139MB
gcr.io/assignment2-400811/docker-app                                          latest    9f91320be12f   5 days ago     139MB
us-central1-docker.pkg.dev/assignment2-400811/docker-app/image-docker-app     latest    9f91320be12f   5 days ago     139MB
nginx                                                                         latest    61395b4c586d   2 weeks ago    187MB
busybox                                                                       latest    a416a98b71e2   2 months ago   4.26MB
hello-world                                                                   latest    9c7a54a9a43c   5 months ago   13.3kB
m22ma012@vm:~/docker-app$
```

Figure 2.2: Docker images.

2.3 Push the Docker image to Google Artifact Registry.

to push the desired docker images in GCR (google container registory), we first need to perform google cloud authentication:

```
gcloud auth login
gcloud auth configure-docker
```

before pushing the image, we need to tag it, and the tagged image will be seen in the 'docker images'

```
docker tag docker-app:latest gcr.io/assignment2-400811/docker
app:latest
```

After authorizing we need to run this command to push the image in GCR, in a repository by the name of 'docker-app', within which we will pushing the image

```
docker push gcr.io/assignment2-400811/docker-app:latest
```

now we can find the image in the GCR:

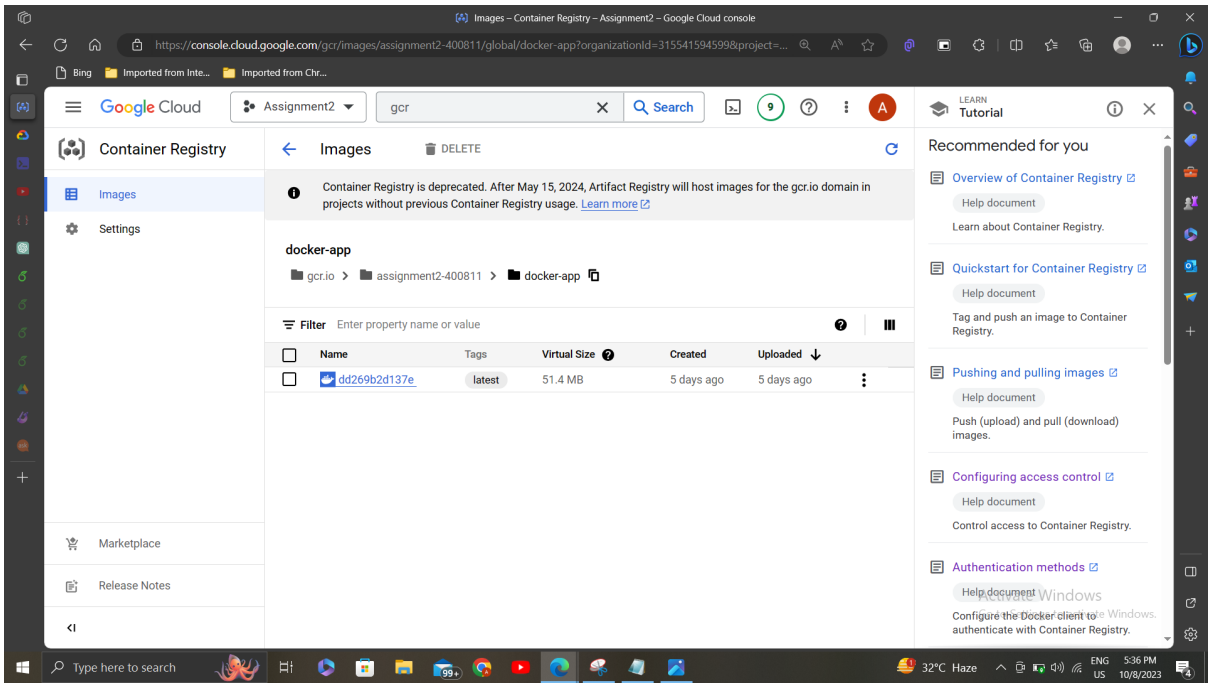


Figure 2.3: Docker Image pushed in GCR.

2.4 Create a Docker Compose file that defines a multi-container application (e.g., web server with Nginx + backend application).

We can create a docker-compose file using 'touch' command:

```
touch docher-compose.yml
```

the docker-compose.yml file has the following:

services in this docker-compose file are:

- **web** is the first service. It uses the official Nginx Docker image (nginx:latest) as its base image. It also maps port 8080 on the host to port 80 in the container.
- **backend** is the second service. It uses a custom Docker image (gcr.io/assignment2-400811/docker-app:latest).
- **my-database** is the third service. It uses the BusyBox Docker image (busybox:latest), this container is a *simple placeholder and not actually be a functioning database container.*

```
  GNU nano 4.8
  ↪  docker-compose.yml
version: '3'

services:
  web:
    image: nginx:latest
    ports:
      - "8080:80"
    networks:
      - myapp-net

  backend:
    image: gcr.io/assignment2-400811/docker-app:latest
    environment:
      - DB_HOST=my-database
    networks:
      - myapp-net
  my-database:
    image: busybox:latest
    networks:
      - myapp-net

networks:
  myapp-net:
```

2.5 Deploy the multi-container application on your VM (from Task 1) using Docker Compose.

first we need to install docker-compose

```
sudo apt install docker-compose
```

Now we run the docker-compose.yml file (automatically detected by docker-compose), docker-compose will create containers for this multi-container app

```
docker-compose up -d
```

(-d flag runs the containers in background)

```
m22ma012@vm:~/docker-app$ docker-compose up -d
Creating network "docker-app_myapp-net" with the default driver
Creating docker-app_web_1         ... done
Creating docker-app_backend_1     ... done
Creating docker-app_my-database_1 ... done
m22ma012@vm:~/docker-app$ docker-compose ps
          Name                        Command               State                    Ports
-----------------------------------------------------------------------------------------------------------
docker-app_backend_1       python app.py                    Exit 0
docker-app_my-database_1   sh                               Exit 0
docker-app_web_1           /docker-entrypoint.sh ngin ...   Up       0.0.0.0:8080->80/tcp,:::8080->80/tcp
m22ma012@vm:~/docker-app$
```

Figure 2.4: Containers of the multi-container app.

2.6 Adjust the Docker Compose configuration to scale the application horizontally.

this command creates 3 replicas of 'backend' service, we can scale any service like this, by creating its replicas.

```
m22ma012@vm:~/docker-app$ docker-compose up -d --scale backend=3
Creating network "docker-app_myapp-net" with the default driver
Creating docker-app_web_1         ... done
Creating docker-app_backend_1     ... done
Creating docker-app_backend_2     ... done
Creating docker-app_backend_3     ... done
Creating docker-app_my-database_1 ... done
m22ma012@vm:~/docker-app$ docker-compose ps
          Name                        Command               State                    Ports
-----------------------------------------------------------------------------------------------------------
docker-app_backend_1       python app.py                    Exit 0
docker-app_backend_2       python app.py                    Exit 0
docker-app_backend_3       python app.py                    Exit 0
docker-app_my-database_1   sh                               Exit 0
docker-app_web_1           /docker-entrypoint.sh ngin ...   Up       0.0.0.0:8080->80/tcp,:::8080->80/tcp
m22ma012@vm:~/docker-app$
```

Figure 2.5: Scaling the 'backend' service of the multi-container app.

3 Task 3: Container Deployment on GCP

3.1 Create a Google Kubernetes Engine (GKE) cluster

To create a cluster we use the following set of commands:

```
gcloud auth login
gcloud config set project assignment2-400811
gcloud container clusters create kubernetes-cluster --num-nodes=1
↪  --zone=us-central1-a
```

A cluster will be created in the Google Kubernetes Engine (GKE):

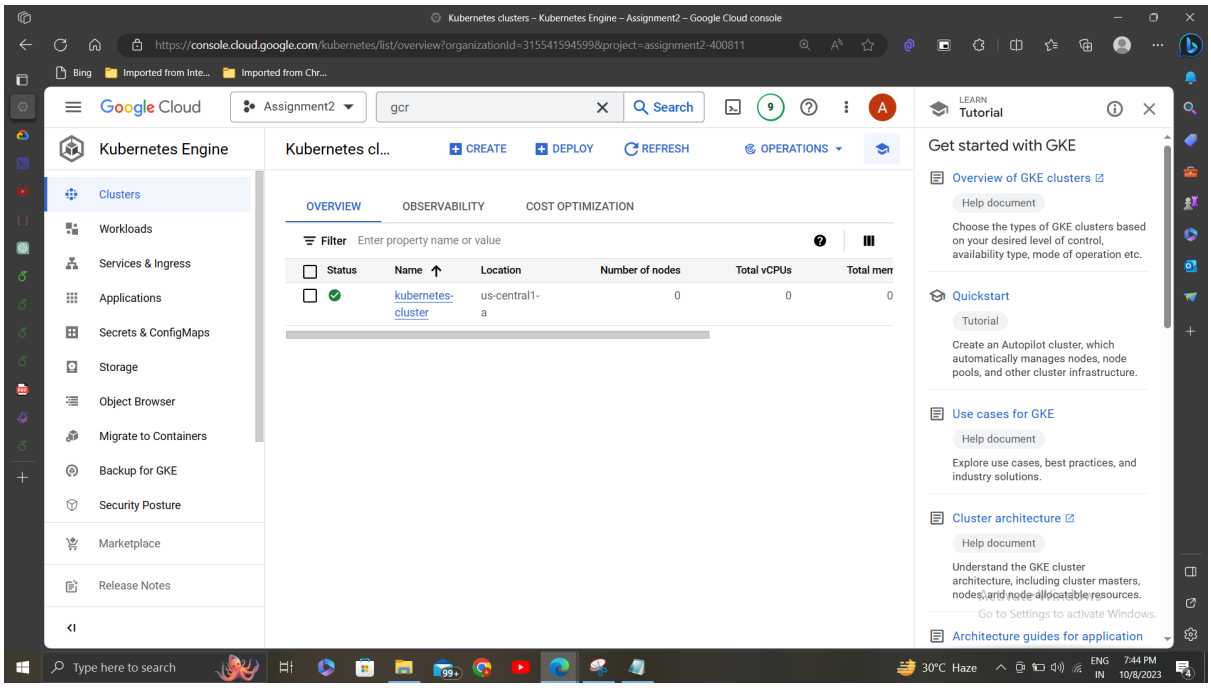


Figure 3.1: GKE cluster created.

3.2 Deploy a sample application container into the GKE cluster.

First we need to install 'kubectl' to interact with the kubernetes cluster:

```
curl -LO
↪  https://storage.googleapis.com/kubernetes-release/release/$(curl
↪  -s https://storage.googleapis.com/kubernetes-release/release/sta
↪  ble.txt)/bin/linux/amd64/kubectl
chmod +x ./kubectl
sudo mv ./kubectl /usr/local/bin/kubectl
sudo apt-get install apt-transport-https ca-certificates gnupg
echo "deb [signed-by=/usr/share/keyrings/cloud.google.gpg]
↪  https://packages.cloud.google.com/apt cloud-sdk main" | sudo tee
↪  -a /etc/apt/sources.list.d/google-cloud-sdk.list
deb [signed-by=/usr/share/keyrings/cloud.google.gpg]
↪  https://packages.cloud.google.com/apt cloud-sdk main
curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo
↪  apt-key --keyring /usr/share/keyrings/cloud.google.gpg add -

sudo apt-get install kubectl

sudo apt-get install google-cloud-sdk-gke-gcloud-auth-plugin
```

Then I created a deployment.yaml file, to deploy the app

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sample-app-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sample-app
  template:
    metadata:
      labels:
        app: sample-app
    spec:
      containers:
      - name: sample-app
        image: gcr.io/assignment2-400811/docker-app
        ports:
        - containerPort: 80
```

This app is using the same image as present in my GCR (docker-app image)

to deploy this image run the command

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

```
m22ma012@vm:~$ ls
deployment.yaml  docker-app  service.yaml  snap
m22ma012@vm:~$ nano deployment.yaml
m22ma012@vm:~$ kubectl apply -f deployment.yaml
deployment.apps/sample-app-deployment unchanged
m22ma012@vm:~$ kubectl get pods
NAME                                    READY   STATUS             RESTARTS        AGE
sample-app-deployment-5ffcf874-vxjwc    0/1     CrashLoopBackOff   13 (114s ago)   9h
m22ma012@vm:~$ kubectl logs sample-app-deployment-5ffcf874-vxjwc
Hello-Docker
m22ma012@vm:~$
```

Figure 3.2: running deployment.yaml to print "hello-docker"

3.3 Ensure the application, including Nginx as a reverse proxy, is accessible over the internet.

create a file service.yaml, and run it to get the external IP of the cluster

```
  GNU nano 4.8
  ↪  service.yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: sample-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
  type: LoadBalancer
```

run this file using:

```
kubectl apply -f service.yaml
kubectl get svc nginx-service
```

```
m22ma012@vm:~$ kubectl apply -f service.yaml
service/nginx-service unchanged
m22ma012@vm:~$ kubectl get svc nginx-service
NAME            TYPE           CLUSTER-IP    EXTERNAL-IP     PORT(S)        AGE
nginx-service   LoadBalancer   10.48.6.235   34.16.122.128   80:30442/TCP   9h
m22ma012@vm:~$
```

Figure 3.3: Obtaining the external IP of app, to access it over the internet"

3.4 Scale the application by adjusting the number of replicas in the deployment.

to scale the app we need to change the number of replicas in the deployment.yaml file, suppose we change the number of replicas to 7:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sample-app-deployment
spec:
  replicas: 7 # changed to 7
  selector:
    matchLabels:
      app: sample-app
  template:
    metadata:
      labels:
        app: sample-app
    spec:
      containers:
      - name: sample-app
        image: gcr.io/assignment2-400811/docker-app
        ports:
        - containerPort: 80
```

```
m22ma012@vm:~$ nano deployment.yaml
m22ma012@vm:~$ kubectl apply -f deployment.yaml
deployment.apps/sample-app-deployment configured
m22ma012@vm:~$ kubectl get pods
NAME                                    READY   STATUS             RESTARTS        AGE
sample-app-deployment-5ffcf874-2x2sd    0/1     CrashLoopBackOff   1 (2s ago)      6s
sample-app-deployment-5ffcf874-4nszr    0/1     Completed          1 (3s ago)      6s
sample-app-deployment-5ffcf874-fndc2    0/1     CrashLoopBackOff   1 (2s ago)      6s
sample-app-deployment-5ffcf874-vw656    0/1     Completed          1 (3s ago)      6s
sample-app-deployment-5ffcf874-vxjwc    0/1     CrashLoopBackOff   16 (103s ago)   9h
sample-app-deployment-5ffcf874-xmm84    0/1     CrashLoopBackOff   1 (3s ago)      6s
sample-app-deployment-5ffcf874-zddl9    0/1     CrashLoopBackOff   1 (2s ago)      6s
```

Figure 3.4: Scaling the app"

3.5 Monitor the application's performance and resource utilization using GCP tools.

We can monitor the app's performance on GKE's Observability section:

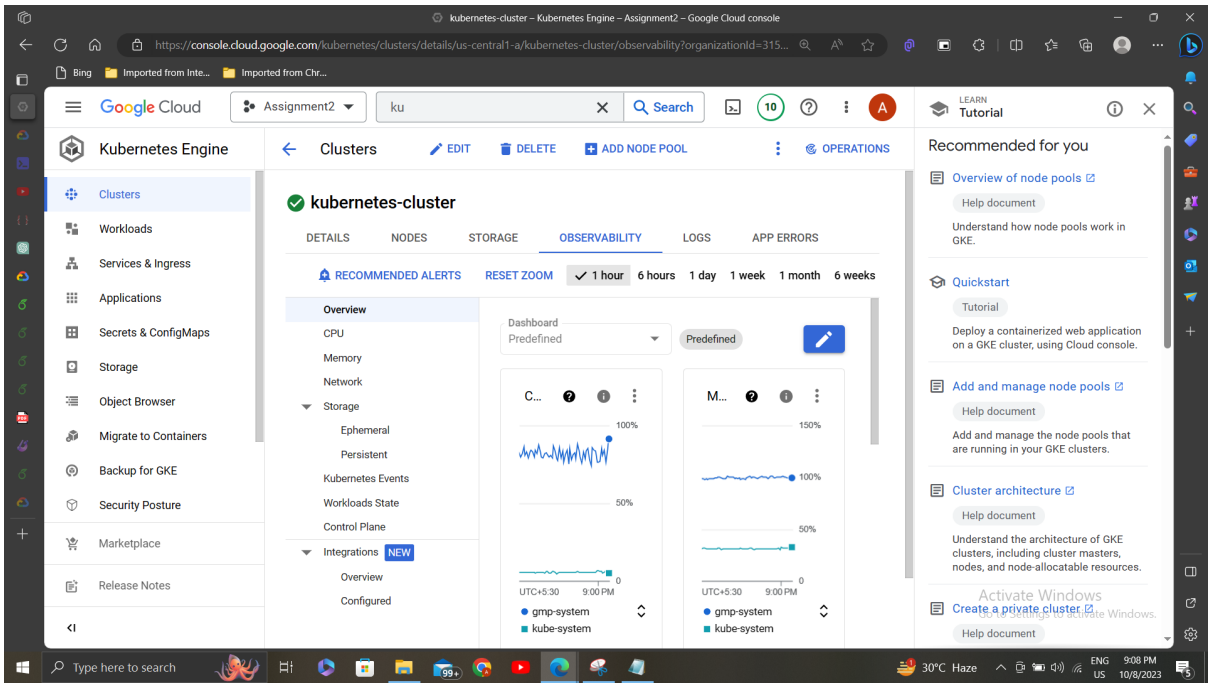


Figure 3.5: Monitoring the app's performance"

4 References

1. https://pwittrock.github.io/docs/tasks/tools/install-kubectl/
2. https://cloud.google.com/kubernetes-engine/docs/how-to/cluster-access-for-kubectl
3. https://cloud.google.com/sdk/?hl=en
4. https://www.educative.io/answers/how-to-install-google-cloud-cli-on-debian-ubuntu
5. https://stackoverflow.com/questions/42697026/install-google-cloud-components-error-from-gcloud-command
6. https://www.youtube.com/watch?v=PUH4Y5yuw0Y
7. https://www.youtube.com/results?search_query=install+kubectl+on+gcp
8. https://www.youtube.com/watch?v=OMquGlhrGgw
9. https://www.youtube.com/results?search_query=gke-gcloud-auth-plugin+
10. https://www.youtube.com/results?search_query=gke-gcloud-auth-plugin+kubectl
11. https://www.youtube.com/results?search_query=how+to+create+and+use+a+gke+cluster&sp=CAI%253D
12. https://www.youtube.com/watch?v=oZUSKoJ73NA
13. https://github.com/actions/runner-images/issues/6778
14. https://cloud.google.com/kubernetes-engine/docs/how-to/cluster-access-for-kubectl
15. https://cloud.google.com/sdk/docs/components#external_package_managers
16. https://stackoverflow.com/questions/48038969/an-image-does-not-exist-locally-with-the-tag-while-pushing-image-to-local-regis
17. https://www.youtube.com/results?search_query=how+to+create+and+use+a+gke+cluster
18. https://www.youtube.com/watch?v=cQeCi2hT3is
19. https://cloud.google.com/blog/products/containers-kubernetes/kubectl-auth-changes-in-gke
20. https://stackoverflow.com/questions/40710526/error-message-service-cloudbuilt-googleapis-com-is-not-for-consumer-when-de
21. https://www.youtube.com/results?search_query=how+to+install+docker+on+gcp+vm
22. https://www.youtube.com/watch?v=FVnA1d_TGw0
23. https://download.docker.com/linux/debian/dists/bullseye/pool/stable/
24. https://forums.docker.com/t/installing-docker-on-buster-e-package-docker-ce-has-no-installation-candidate/108397/11
25. https://medium.com/@kyle.powers103/installing-docker-on-gcloud-vms-1479dd9dde30
26. https://www.youtube.com/watch?v=VNTG9IBnwAI&t=58s
27. https://www.youtube.com/watch?v=VNTG9IBnwAI&t=58s
28. https://download.docker.com/linux/debian/dists/bullseye/pool/stable/amd64/

List of Figures