

Lab 1: Introduction to Google Kubernetes Engine (GKE)

Course: SOFE 4790U

Date: 18/09/2022

Lab #: 1 (Individual Report)

Submitted by: Matheeshan Sivalingam(100703887)

Objective:

1. Get familiar with Docker images and containers.
2. Learn various Kubernetes tools.
3. Learn how to use Google Cloud Platform (GCP).
4. Compose YAML files to deploy cloud applications.

Discussion:

Summarize what you have learned about docker and Kubernetes including the used terminologies and descriptions. What are the advantages and disadvantages of using docker images against virtual machines?

Docker is an open-source platform used in DevOps that allows developers to build, deploy, and manage containers. Containers are a virtualized run-time environment that hosts isolated applications. These containers only contain the necessary libraries and dependencies required to run the applications. These also include a Docker image which holds instructions about what is required when creating these containers. Using Docker allows one to run code effectively in any environment. Kubernetes is an open-source system that automates and allows one to manage multiple containers. A good analogy that helped me understand how and what Kubernetes does is to think of Kubernetes as a conductor of an orchestra. Similar to how a conductor acts as a guide to control and manage the music in an orchestra or choir, Kubernetes essentially helps manages a group of containers so that they can work together and perform their intended services when needed. These containers are grouped together into pods where they share the same data and resources which are then deployed and managed autonomously through Kubernetes.

The advantages of using docker containers against virtual machines are the following:

- Docker containers are a lot more lightweight as they only contain the necessary dependencies required to run
- High portability as it can be easily run on different environments
- Better performance as they are hosted in a single Docker environment

The disadvantages of using docker containers against virtual machines are the following:

- Poor security as a cluster can be exploited if an attacker gets access to one container
- Can only emulate an operating system whereas virtual machines can emulate multiple

Design:

MongoDB is another type of database. It's required to deploy it using GKE using a YAML file. If you used any Kubernetes tool in your deployment that is not included in the lab you should describe it and why you used it

To use MongoDB on the Google Kubernetes Engine, I deployed it as a statefulset using a YAML file. The YAML file for the MongoDB deployment followed a similar format to the MySQL file with the exception of the container port being set to the default port for MongoDB instances (port: 27017). No additional Kubernetes tools were used in the deployment that were not included in the lab.

```
1  #Service parameters
2  apiVersion: v1
3  kind: Service
4  metadata:
5    name: mongodb
6    labels:
7      app: mongodb
8  spec:
9    clusterIP: None
10   selector:
11     app: mongodb
12   ports:
13     #Default port for MongoDB instances
14     - port: 27017
15   ---
16   #Deployment Parameters
17   apiVersion: apps/v1
18   kind: StatefulSet
19   metadata:
20     name: mongodb
21   spec:
22     serviceName: mongodb
23     replicas: 1
24     selector:
25       matchLabels:
26         app: mongodb
27     template:
28       metadata:
29         labels:
30           app: mongodb
31           selector: mongodb
32       spec:
33         containers:
34         - name: mongodb
35           image: mongo:4.0.17
36           ports:
37             #Default port for MongoDB instances
38             - containerPort: 27017
```

Figure 1 - Configuration parameters set on the MongoDB deployment

```
mathees64@cloudshell:~ (rapid-stage-362020)$ kubectl apply -f mongodb.yaml
service/mongodb created
statefulset.apps/mongodb created
mathees64@cloudshell:~ (rapid-stage-362020)$
```

Figure 2 - Used apply command to create the MongoDB service and statefulset

```
mathees64@cloudshell:~ (rapid-stage-362020)$ kubectl run -it mongo-shell --image=mongo:4.0.17 /bin/bash
If you don't see a command prompt, try pressing enter.
root@mongo-shell:/#
root@mongo-shell:/#
```

Figure 3 - Used run command to run a MongoContainer inside the cluster

```
root@mongo-shell:/# mongo mongodb-0.mongodb
MongoDB shell version v4.0.17
connecting to: mongodb://mongodb-0.mongodb:27017/test?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("6271e15a-309e-4959-92b2-50067a8bf467") }
MongoDB server version: 4.0.17
Welcome to the MongoDB shell.
```

Figure 4 - Connect and open the MongoShell on the mongodb-0 pod

```
> use person
switched to db person
> db.person.insertMany([{id: 1, age: 30, name: 'tom'}, {id: 2, age: 23, name: 'adam'}, {id: 3, age: 79, name: 'Joe'}])
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("63278bae8a8a399240d7a534"),
    ObjectId("63278bae8a8a399240d7a535"),
    ObjectId("63278bae8a8a399240d7a536")
  ]
}
> db.person.find();
{ "_id" : ObjectId("63278bae8a8a399240d7a534"), "id" : 1, "age" : 30, "name" : "tom" }
{ "_id" : ObjectId("63278bae8a8a399240d7a535"), "id" : 2, "age" : 23, "name" : "adam" }
{ "_id" : ObjectId("63278bae8a8a399240d7a536"), "id" : 3, "age" : 79, "name" : "Joe" }
```

Figure 5 - Successfully inserted document to MongoDB database.