Introduction to Google Kubernetes Engine (GKE)

Lab 1 Report

Hemshikha Sultoo
100670616

September 18, 2022

Discussion

a) Summary on Docker

What is Docker?
Docker can be described as a tool that uses the containerization method to facilitate the automation of the deployment of applications. Containers generated are lightweight software packages that are equipped with all the necessary dependencies required to run the application. Thus, Docker allows applications to be efficient, compatible and portable across different platforms.

Components of Docker

1) Docker Client and Server
   Docker uses a Client-Server architecture where communication between the two is facilitated through the REST API. Any command by the Docker client is first translated to by the REST API, then sent through to the Docker Daemon which in turn checks for the request and interacts with the OS to make necessary changes to the containers.

2) Docker Images
   Docker images can be described as a template of instructions required to generate Docker containers. To build a docker image, a docker file is required. The docker image can be stored under the docker registry.

3) Docker Containers
   A docker container is an isolated executable software package that is equipped with all required applications and dependencies. Each application within the container is isolated. A docker host can have multiple containers running simultaneously.

4) Docker Registry/Hub
   The registry stores and distributes the docker images depending on whether they are set to be public or private. The docker client can use the pull command to get and image or the push command to store the image into the registry.

b) Summary on Kubernetes

In simple terms, Kubernetes orchestrates, controls and manages services just like an App Developer would. The containers created by Docker are managed and manipulated by Kubernetes API. It allows containers to communicate with each other and allows replacement or upgrades to be performed without much downtime.

c) Advantages and disadvantages of using docker images against using virtual machines.

Topic	Docker Image	Virtual Machine
Scaling	Scaling up with Docker is way easier.	It's harder to scale up with virtual machines.
Boot-up time	Takes less time to boot-up.	Takes more time to boot-up.
Efficiency	More efficient.	Less efficient.
Operating System	Takes less space.	Demands a large amount of space.
Performance	Use of containers hosted by a single Docker engine supports better performance.	Running numerous virtual machines simultaneously eventually takes a toll on the overall performance of the system.
Space/memory allocation and management	Different containers are able to share and reuse data as well as space.	Space and data occupied cannot be shared.
Portability	Compatible with various platforms and therefore, portable.	Not compatible with all platforms.

Design

To deploy MongoDB, a file called mongo.yaml was used. The format is pretty similar to the one used in the first part of the lab except that StatefulSet configuration was added. The Service file was created as well. Without the service file, the connection to MongoDB kept failing. Other than the Kubernetes API, no other tools were used.

mongo.yaml

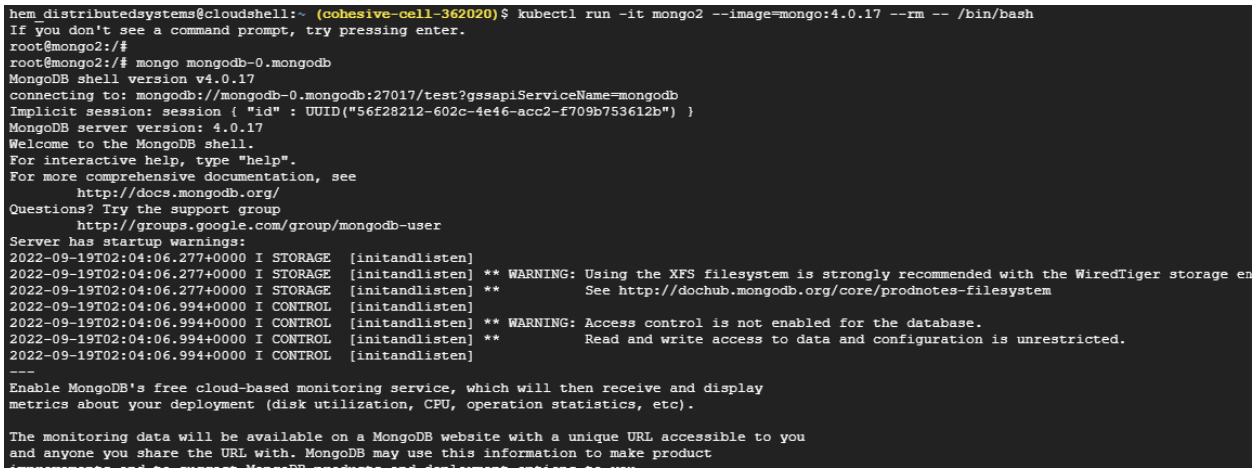
```
apiVersion: v1
```

```yaml
kind: Service
metadata:
  name: mongodb-service
spec:
  type: LoadBalancer
  ports:
    - port: 27017
  selector:
    app: mongodb
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mongodb
spec:
  serviceName: mongodb
  replicas: 1
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
        - image: mongo:4.0.17
          name: mongodb
          ports:
            - containerPort: 27017
              name: mongodb
          volumeMounts:
            - mountPath: /data/db
              name: pvc
  volumeClaimTemplates:
    - metadata:
        name: pvc
      spec:
        accessModes:
          - ReadWriteOnce
```

```
        resources:
          requests:
            storage: 1Gi
```

mongo-svc.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: mongodb
  labels:
    app: mongobd
spec:
  clusterIP: None
  selector:
    app: mongodb
  ports:
  - port: 27017
    targetPort: 27017
```



```
> use myMongoDB;
switched to db myMongoDB
> create table fruits(id int, name varchar(50), price int);
2022-09-19T02:08:06.603+0000 E QUERY    [js] SyntaxError: missing ; before statement @(shell):1:7
> create table fruits(id int, name varchar(50), price int)
2022-09-19T02:09:07.329+0000 E QUERY    [js] SyntaxError: missing ; before statement @(shell):1:7
> db.myMongoDB.insert([{id: 0045, fruit: 'apple', size: 'small', color: 'green'}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 1,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
> db.myMongoDB.find()
{ "_id" : ObjectId("6327d001c83e98c3a7311c12"), "id" : 37, "fruit" : "apple", "size" : "small", "color" : "green" }
> db.myMongoDB.insert({id: 0045, fruit: 'apple', size: 'small', color: 'green'})
WriteResult({ "nInserted" : 1 })
> db.myMongoDB.find()
{ "_id" : ObjectId("6327d001c83e98c3a7311c12"), "id" : 37, "fruit" : "apple", "size" : "small", "color" : "green" }
{ "_id" : ObjectId("6327d099c83e98c3a7311c13"), "id" : 37, "fruit" : "apple", "size" : "small", "color" : "green" }
>
```

Video Links:

Part 1 MYSQL:

https://drive.google.com/file/d/1cEluSehbIWMWJ1IJqR5-1k3cMHiGIsNO/view?usp=sharing

Part 2 MongoDB:

https://drive.google.com/file/d/1hQlY7pk8ASAHoJPkGrWaET1Jhyy6a6OS/view?usp=sharing