Lab 1: Introduction to Google Kubernetes Engine (GKE)

Course: SOFE 4790U
Date: 18/09/2022
Lab #: 1 (Group Report)
Submitted by: Matheeshan Sivalingam(100703887)
Hemshikha Sultoo (100670616)
Aaditya Rajput (100622434)

Objective:

1. Get familiar with Docker images and containers.
2. Learns 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?

Matheeshan:

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.

Kubernetes is an open-source system that automates and allows one to manage multiple containers. Kubernetes 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

Hemshikha:

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:

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

Matheeshan:

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 was not included in the lab.

```yaml
#Service parameters
apiVersion: v1
kind: Service
metadata:
  name: mongodb
  labels:
    app: mongodb
spec:
  clusterIP: None
  selector:
    app: mongodb
  ports:
    #Default port for MongoDB instances
    - port: 27017
---
#Deployment Parameters
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mongodb
spec:
  serviceName: mongodb
  replicas: 1
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
        selector: mongodb
    spec:
      containers:
        - name: mongodb
          image: mongo:4.0.17
          ports:
            #Default port for MongoDB instances
            - 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.

Hemshikha:

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
kind: Service
```

```yaml
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
```

```
hem_distributedsystems@cloudshell:~ (cohesive-cell-362020)$ kubectl run -it mongo2 --image=mongo:4.0.17 --rm -- /bin/bash
If you don't see a command prompt, try pressing enter.
root@mongo2:/#
root@mongo2:/# 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("56f28212-602c-4e46-acc2-f709b753612b") }
MongoDB server version: 4.0.17
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2022-09-19T02:04:06.277+0000 I STORAGE  [initandlisten]
2022-09-19T02:04:06.277+0000 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage en
2022-09-19T02:04:06.277+0000 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes-filesystem
2022-09-19T02:04:06.994+0000 I CONTROL  [initandlisten]
2022-09-19T02:04:06.994+0000 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2022-09-19T02:04:06.994+0000 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2022-09-19T02:04:06.994+0000 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
```

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

Part Two: MongoDB