SOFE 4790U: Distributed Systems

Date: September 18, 2022

Lab 1: Introduction to Google Kubernetes Engine (GKE)

Group 14
Members:

Preet Patel	**100708239**
Zachary Carman	**100697844**
Philip Jasionowski	**100751888**
Rodaba Ebadi	**100708585**

Learning Objectives:

In this lab, we learned how to use the Google Cloud Platform (GCP) and Google Kubernetes Engine to deploy cloud applications. We learned how to compose YAML files containing service and deployment parameters including learning about parameters and environmental variables such as ports, replicas, images, etc. to deploy applications on GCP.

Links:

Github: https://github.com/preetpatel87/Distributed-Systems-Group-14
Video Demonstrations:

MySQL Deployment:
https://drive.google.com/file/d/1tJMyuk8-nLJXiT4kdE6j1hzrO2-amMxo/view?usp=sharing
MongoDB Deployment:
https://drive.google.com/file/d/1X8hQhDLB2XyQM_DDV-IaxAmWVk6HL8vW/view?usp=sharing

Discussion

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

Docker:

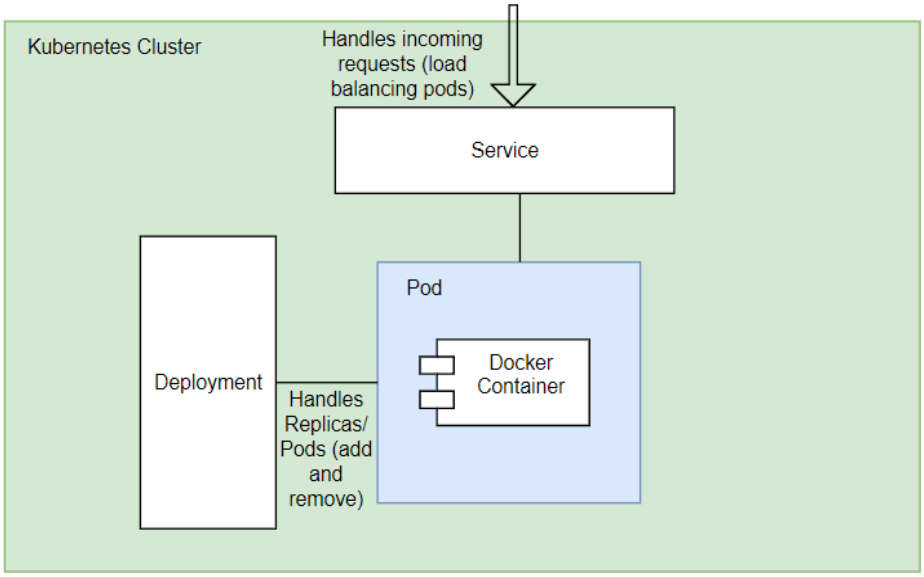
Docker is a lightweight tool that does not require guest OS to share projects across different work environments. Docker automates the deployment of applications for use across different working environments. One example of these different environments could be between developers and testers. Docker aims to make this transition seamless. Docker also allows for frameworks to be running on different applications at once freeing up space.

The Docker engine is split up into four main sections. Client and Server, Docker Images, Docker Containers, Docker registry. The client and server are connected through a rest API which facilitates the communication between the two elements. The docker images are within the Docker server and the registry. A Docker image is a file that consists of a set of instructions which is executable code from which a docker container can be created. Docker container is a standalone executable software package consisting of all the components such as libraries and other dependencies required to run the application. Docker registry is an open source server-side service used for hosting docker images. Docker has its own public registry called Docker Hub.

Kubernetes:

Kubernetes is an open-source platform for managing containerized workloads and services. Kubernetes was created to combat monoliths and reduce deployment time. This resulted in microservices which split the service into smaller sections. These microservices were independent and could be updated separately instead of updating the whole system. With containers, developers can package up the services neatly with all the dependencies and necessary configuration can get delivered together.

Kubernetes handles how containers communicate with each other, and how containers can be upgraded with any downtime. A kubernetes cluster is a set of worker nodes with their master node. Worker nodes are containerized applications and the master node is responsible for managing these worker nodes.A pod is the containerized application, a kubernetes service manages the connectivity to the pods, and a kubernetes deployment manages replicas of a pod to provide desired state management.



Docker Advantages and Disadvantages compared to Virtual Machines:

Docker	**Virtual Machines**
Docker requires less memory space	VM requires more memory space
Shorter startup time	Longer startup time
Much higher efficiency	Lower efficiency
Greater Scalability	Lesser Scalability
Easily portable across different platforms	Compatibility issues when porting between different platforms
Can share data/libraries between containers	Cannot share data/libraries between application instances

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

For the deployment of the MongoDB database application, we first created a single YAML file in which we pass the parameters of the deployment and service which includes the port number,

number of replicas, environment variables such as username and password, and the MySQL image. For MongoDB we used the port number 3308, we created 1 replica, with the environment variables username set to ‘user’ and password set to ‘password’. To find an existing public mongoDB image, we used the docker hub registry to find the bitnami/mongoDB image which we used for this lab.

- **Problem:** Deploy a MongoDB application using the Google kubernetes engine (GKE).
- **Solution:** For the deployment of the MongoDB database application, we first created a single YAML file in which we pass the parameters of the deployment and service which includes the port number, number of replicas, environment variables such as username and password, and the MySQL image. For MongoDB we used the port number 3308, we created 1 replica, with the environment variables username set to ‘user’ and password set to ‘password’. To find an existing public mongoDB image, we used the docker hub registry to find the bitnami/mongoDB image which we used for this lab.
- **Video Demonstration:** https://drive.google.com/file/d/1X8hQhDLB2XyQM_DDV-IaxAmWVk6HL8vW/view?usp=sharing
- **YAML File:**

```
apiVersion: v1
kind: Service
metadata:
  name: mongodb
spec:
  type: LoadBalancer
  ports:
    - port: 3308
  selector:
    app: mongodb
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongodb
  template:
```

```
  metadata:
    labels:
      app: mongodb
  spec:
    containers:
      - image: bitnami/mongodb
        name: mongodb
        env:
          - name: MONGODB_ROOT_PASSWORD
            value: password
          - name: MONGODB_ROOT_USERNAME
            value: user
        ports:
          - containerPort: 3308
            name: mongodb
```

Deliverables

Demonstration Links:

MySQL Deployment:
https://drive.google.com/file/d/1tJMyuk8-nLJXiT4kdE6j1hzrO2-amMxo/view?usp=sharing
MongoDB Deployment:
https://drive.google.com/file/d/1X8hQhDLB2XyQM_DDV-IaxAmWVk6HL8vW/view?usp=sharing

MySQL Deployment:

- **Problem:** Deploy a MySQL database application using the Google kubernetes engine (GKE).
- **Solution:** For the deployment of the MySQL database application, we needed to first create a three node cluster on GKE. After this, we had to create a deployment with a public MySQL image. Once this image is deployed, we expose this deployment by creating a load-balancer service. Once the service is running, it can be accessed from any machine by using the provided ip address. We can then connect with the MySQL application and create tables and insert rows into the tables. All of these steps can also be packaged into a single YAML file in which we pass the parameters of the deployment and service which includes the port number, number of replicas, environment variables such as username and password, and the MySQL image.

- **Video Demonstration:** https://drive.google.com/file/d/1tJMyuk8-nLJXiT4kdE6j1hzrO2-amMxo/view?usp=sharing
- **Execution Images:**

```
preetppatel87@cloudshell:~ (nth-agent-362522)$ kubectl apply -f mysql.yaml
service/mysql-service created
deployment.apps/mysql-deployment created
preetppatel87@cloudshell:~ (nth-agent-362522)$ get pods
-bash: get: command not found
preetppatel87@cloudshell:~ (nth-agent-362522)$ kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
mysql-deployment   1/1     1            1           10s
preetppatel87@cloudshell:~ (nth-agent-362522)$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
mysql-deployment-5496fdc956-c5g9b   1/1     Running   0          14s
preetppatel87@cloudshell:~ (nth-agent-362522)$ kubectl get services
NAME            TYPE           CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
kubernetes      ClusterIP      10.80.0.1     <none>         443/TCP          16h
mysql-service   LoadBalancer   10.80.6.30    <pending>      3306:30976/TCP   29s
preetppatel87@cloudshell:~ (nth-agent-362522)$ kubectl get services
NAME            TYPE           CLUSTER-IP    EXTERNAL-IP     PORT(S)          AGE
kubernetes      ClusterIP      10.80.0.1     <none>          443/TCP          16h
mysql-service   LoadBalancer   10.80.6.30    34.152.44.158   3306:30976/TCP   49s
```

```
preetppatel87@cloudshell:~ (nth-agent-362522)$ mysql -uuser -psofe4790u -h34.152.44.158
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.30 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use myDB;
Database changed
mysql> create table person( id int, age int, name varchar(50));
Query OK, 0 rows affected (0.07 sec)

mysql> insert into person values(1,30,'tom');
Query OK, 1 row affected (0.05 sec)

mysql> insert into person values(2,23,'adam');
Query OK, 1 row affected (0.03 sec)

mysql> insert into person values(3,79,'Joe');
Query OK, 1 row affected (0.04 sec)
```

```
mysql> select * from person where age>=30;
+------+------+------+
| id   | age  | name |
+------+------+------+
|    1 |   30 | tom  |
|    3 |   79 | Joe  |
+------+------+------+
2 rows in set (0.03 sec)
```

```
mysql> select * from person;
+------+------+------+
| id   | age  | name |
+------+------+------+
|    1 |   30 | tom  |
|    2 |   23 | adam |
|    3 |   79 | Joe  |
+------+------+------+
3 rows in set (0.03 sec)
```

MongoDB Deployment:

- **Execution Images:**

```
preetppatel87@cloudshell:~ (nth-agent-362522)$ kubectl apply -f mongodb.yaml
service/mongodb created
deployment.apps/mongodb created
preetppatel87@cloudshell:~ (nth-agent-362522)$ kubectl get deployments
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
mongodb   1/1     1            1           7s
preetppatel87@cloudshell:~ (nth-agent-362522)$ kubectl get pods
NAME                       READY   STATUS    RESTARTS   AGE
mongodb-68b46ffbb6-jbsx6   1/1     Running   0          12s
preetppatel87@cloudshell:~ (nth-agent-362522)$ kubectl get service
NAME         TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)          AGE
kubernetes   ClusterIP      10.80.0.1      <none>         443/TCP          16h
mongodb      LoadBalancer   10.80.15.128   <pending>      3308:31812/TCP   43s
preetppatel87@cloudshell:~ (nth-agent-362522)$ kubectl get service
NAME         TYPE           CLUSTER-IP     EXTERNAL-IP      PORT(S)          AGE
kubernetes   ClusterIP      10.80.0.1      <none>           443/TCP          16h
mongodb      LoadBalancer   10.80.15.128   35.203.15.176    3308:31812/TCP   46s
preetppatel87@cloudshell:~ (nth-agent-362522)$
```

```
preetppatel87@cloudshell:~ (nth-agent-362522)$ kubectl exec -it mongodb-68b46ffbb6-jbsx6 -- mongosh -u root -p password
Current Mongosh Log ID: 632341ed03577d9bbf64b30e
Connecting to:          mongodb://<credentials>@127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+1.5.4
Using MongoDB:          6.0.1
Using Mongosh:          1.5.4

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2022-09-15T15:15:48.847+00:00: vm.max_map_count is too low
------

------
   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
   improvements and to suggest MongoDB products and deployment options to you.

   To enable free monitoring, run the following command: db.enableFreeMonitoring()
   To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
------
```

```
test> db.createCollection("car");
{ ok: 1 }
test> db.car.insertMany([ { make: "Toyota", name: "Prius", age: 3 }, { make: "Honda", name: "Odyssey", age: 2 }, { make: "Ford", name: "Escape", age: 1 }])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("63234336b97b552afcd748ca"),
    '1': ObjectId("63234336b97b552afcd748cb"),
    '2': ObjectId("63234336b97b552afcd748cc")
  }
}
test> db.car.find()
[
  {
    _id: ObjectId("63234336b97b552afcd748ca"),
    make: 'Toyota',
    name: 'Prius',
    age: 3
  },
  {
    _id: ObjectId("63234336b97b552afcd748cb"),
    make: 'Honda',
    name: 'Odyssey',
    age: 2
  },
  {
    _id: ObjectId("63234336b97b552afcd748cc"),
    make: 'Ford',
    name: 'Escape',
    age: 1
  }
]
```