Faculty of Engineering and Applied Science

SOFE 4790U Distributed Systems

CRN 44425

Lab #1

Alexander Campbell

100703650

Introduction:

The goal of this lab was to introduce us to utilizing the Google Cloud Platform and the containerization of systems such as Docker.

Discussion:

Throughout this lab and the course work we've begun to learn about Docker. Docker allows us to containerize software in a lightweight environment called the Docker Engine. You can develop, ship, and run software using this engine. Due to these containers developers can easily share their software environments with other developers without the hassle of downloading all the supporting software required to run an application. Docker utilizes a client-server architecture in which your user-controlled client talks to a daemon through a REST API. This daemon runs the major functions of Docker such as building, running, and distribution of containers. Docker can be run using images, which are essentially templates or instructions given to the engine on how to format and create a specific docker container, such as in this lab we utilize a pre-built mySQL image. To deploy Docker containers you can use your local machine or software such as Kubernetes. Kubernetes is a software that can be used to manage large networks of containers. It does so by automating many aspects of a distributed system such as deployment, management, and scaling. For example, Kubernetes can be used to automatically deploy more containers given that your user traffic picks up.

We were also able to explore the major differences between using Docker versus a virtual machine. The main difference between the two is that a virtual machine emulates the hardware of a computer while Docker is only the isolated operating system. This makes it so Docker is lighter weight in comparison as it doesn't have to take up the same size due to being isolated only at the OS-level, rather than having the bulky hardware emulation of a virtual machine. Even though Docker is generally more lightweight than a virtual machine, that isn't always what is required. Virtual machines offer more security and isolation than Docker as it is a separate system on its own, rather than an application being run on an existing OS. Since Docker shares a host kernel with the application that runs it is inherently less secure than a virtual machine, which has its own kernel.

In most use cases it does make sense to use Docker as it's more lightweight, more portable, and generally speaking most applications do not require hardware level emulation.

Procedure for MySQL Database:

```
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ gcloud config set compute/zone northamerica-northeast1-b
Updated property [compute/zone].
```

```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ gcloud container clusters create openfaas --num-nodes=3
Default change: VPC-native is the default mode during cluster creation for versions greater than 1.21.0-gke.1500. To create advanced routes based clusters, please pass the `--no-enable-ip-alias` flag
Default change: During creation of nodepools or autoscaling configuration changes for cluster versions greater than 1.24.1-gke.800 a default location policy is applied. For Spot and PVM it defaults to ANY, and for all other VM kinds a B
ALANCED policy is used. To change the default values use the `--location-policy` flag.
Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most 1008 node(s).
Creating cluster openfaas in northamerica-northeast1-b... Cluster is being health-checked (master is healthy)...done.
Created [https://container.googleapis.com/v1/projects/dev-solstice-362019/zones/northamerica-northeast1-b/clusters/openfaas].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/northamerica-northeast1-b/openfaas?project=dev-solstice-362019
kubeconfig entry generated for openfaas.
NAME: openfaas
LOCATION: northamerica-northeast1-b
MASTER_VERSION: 1.22.11-gke.400
MASTER_IP: 34.152.53.12
MACHINE_TYPE: e2-medium
NODE_VERSION: 1.22.11-gke.400
NUM_NODES: 3
STATUS: RUNNING
```

Created cluster of three nodes

```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl create deployment mysql-deployment --image mysql/mysql-server --port=3306
deployment.apps/mysql-deployment created
```

```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
mysql-deployment   0/1     1            0           5s
```

```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get pods
NAME                                READY   STATUS              RESTARTS   AGE
mysql-deployment-7467c475f8-zlzlj   0/1     ContainerCreating   0          12s
```

Successfully deployed mySQL image to cluster

```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl logs mysql-deployment-7467c475f8-zlzlj 2>&1 |grep GENERATED
[Entrypoint] GENERATED ROOT PASSWORD: a1Krt2+@2e*8kLNKM6O92:l;r_RL7.M,
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl exec -it mysql-deployment-7467c475f8-zlzlj  -- mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.30

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> ALTER USER 'root'@'localhost' IDENTIFIED BY '123' ;
Query OK, 0 rows affected (0.01 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.01 sec)

mysql> exit
Bye
```

Accessed the mySQL database with generated password and changed the password to the root user

```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl exec -it  <pod-name>  -- mysql -uroot -p123
-bash: pod-name: No such file or directory
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl exec -it mysql-deployment-7467c475f8-zlzlj  -- mysql -uroot -p123
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 10
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> CREATE USER 'user'@'%' IDENTIFIED BY 'sofe4790u';
Query OK, 0 rows affected (0.02 sec)

mysql> GRANT ALL PRIVILEGES ON *.* TO 'user'@'%' WITH GRANT OPTION;
Query OK, 0 rows affected (0.01 sec)

mysql> exit
Bye
```

Accessed mySQL database with set password and added in ‘sofe4790u’ user to database

```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl expose deployment mysql-deployment --type=LoadBalancer --name=mysql-service
service/mysql-service exposed
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get service
NAME            TYPE           CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
kubernetes      ClusterIP      10.80.0.1     <none>        443/TCP          5m44s
mysql-service   LoadBalancer   10.80.8.109   <pending>     3306:31911/TCP   10s
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get service
NAME            TYPE           CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
kubernetes      ClusterIP      10.80.0.1     <none>        443/TCP          6m11s
mysql-service   LoadBalancer   10.80.8.109   <pending>     3306:31911/TCP   37s
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get service
NAME            TYPE           CLUSTER-IP    EXTERNAL-IP     PORT(S)          AGE
kubernetes      ClusterIP      10.80.0.1     <none>          443/TCP          8m58s
mysql-service   LoadBalancer   10.80.8.109   34.152.39.101   3306:31911/TCP   3m24s
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ mysql -uuser -psofe4790u -h34.152.39.101
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 11
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> exit
Bye
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl delete deployment mysql-deployment
deployment.apps "mysql-deployment" deleted
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl delete service mysql-service
service "mysql-service" deleted
```

Added a load balancer service to the deployment, connected to deployed mySQL server through the external IP, and deleted the deployment and service

mysql.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: mysql-service
spec:
  type: LoadBalancer
  ports:
    - port: 3306
  selector:
    app: mysql
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-deployment
```

```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl apply -f mysql.yaml
service/mysql-service created
deployment.apps/mysql-deployment created
```

```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get services
NAME            TYPE           CLUSTER-IP    EXTERNAL-IP     PORT(S)          AGE
kubernetes      ClusterIP      10.80.0.1     <none>          443/TCP          3h36m
mysql-service   LoadBalancer   10.80.11.25   34.152.39.101   3306:30605/TCP   89s
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
mysql-deployment   1/1     1            1           97s
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
mysql-deployment-5496fdc956-8xcpc   1/1     Running   0          99s
```

Deployed mySQL using the provided YAML file

```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ mysql -uuser -psofe4790u -h34.152.39.101
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 (0.05 sec)

mysql> insert into person values(1,30,'tom');
Query OK, 1 row affected (0.03 sec)

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

mysql> nsert into person values(3,79,'Joe');
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to
mysql> insert into person values(3,79,'Joe');
Query OK, 1 row affected (0.03 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> exit
Bye
```

Entered the generated SQL server through its external IP and ran the specified SQL statements in manual

Procedure for MongoDB Database:

mongodb.yaml | mysql.yaml

mongodb.yaml > ...

```yaml
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  type: LoadBalancer
  ports:
    - port: 3306
  selector:
    app: mongo
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
        - image: mongo
          name: mongodb
          env:
            - name: MONGODB_ROOT_PASSWORD
              value: password
            - name: MONGODBL_USER
              value: user
            - name: MONGODB_PASSWORD
              value: sofe4790u
            - name: MONGODBL_DATABASE
              value: myDB
```

ode minikube

Created YAML file with needed information for a MongoDB database

```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl apply -f mongodb.yaml
service/mongodb-service created
deployment.apps/mongodb-deployment created
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
mongodb-deployment-57dd4d4b79-6phcz   1/1     Running   0          19s
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get deployments
NAME                 READY   UP-TO-DATE   AVAILABLE   AGE
mongodb-deployment   1/1     1            1           24s
```

```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get service
NAME              TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)          AGE
kubernetes        ClusterIP      10.80.0.1      <none>          443/TCP          8m7s
mongodb-service   LoadBalancer   10.80.1.176    35.203.43.155   3306:32376/TCP   55s
```

Applied file to create deployment and service

```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl exec -it mongodb-deployment-57dd4d4b79-6phcz -- sh
# mongo
sh: 1: mongo: not found
# mongo
sh: 2: mongo: not found
# mongosh
Current Mongosh Log ID: 632a5f893b3516eef5a66b75
Connecting to:          mongodb://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/


To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

------
   The server generated these startup warnings when booting
   2022-09-21T00:32:30.552+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
   2022-09-21T00:32:31.271+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
   2022-09-21T00:32:31.271+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> use myDb
switched to db myDb
myDb> use test
switched to db test
test> show dbs
admin   40.00 KiB
config  12.00 KiB
local   40.00 KiB
```

Entered deployed database using the pod name

```
test> db.test.insert({name: "tom", age: 30})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("632a64c3c74c3ad694629b3d") }
}
test> db.test.insert({name: "adam", age: 23})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("632a64ccc74c3ad694629b3e") }
}
test> show collections
admin
test
test> db.test.find()
[
  { _id: ObjectId("632a64c3c74c3ad694629b3d"), name: 'tom', age: 30 },
  { _id: ObjectId("632a64ccc74c3ad694629b3e"), name: 'adam', age: 23 }
]
test> db.test.insert({name: "joe", age: 79})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("632a6556c74c3ad694629b3f") }
}
test> db.test.find()
[
  { _id: ObjectId("632a64c3c74c3ad694629b3d"), name: 'tom', age: 30 },
  { _id: ObjectId("632a64ccc74c3ad694629b3e"), name: 'adam', age: 23 },
  { _id: ObjectId("632a6556c74c3ad694629b3f"), name: 'joe', age: 79 }
]
```

Inserted data into database and viewed the inserted data

```
test> db.test.find({"age":{$gt:23}}).pretty()
[
  { _id: ObjectId("632a64c3c74c3ad694629b3d"), name: 'tom', age: 30 },
  { _id: ObjectId("632a6556c74c3ad694629b3f"), name: 'joe', age: 79 }
]
test>
```

Queried the data to find only the entries with an age above 23 like in the previous procedure

Conclusion:

In conclusion this lab was a great experience utilizing Kubernetes to set up various databases and increased my knowledge in containedbrization.