Faculty of Engineering and Applied Science

SOFE 4790U Distributed Systems

CRN 44425

Lab # 1

Mamun Hossain - 100553073

Introduction:

The objective of this lab was to familiarize ourselves with the Google Cloud Platform, and the usage of containers and how to access them using Kubernetes.

Discussion:

For this lab, we began to familiarize ourselves with different softwares, such as the Google cloud platform, Docker while. using Kubernetes. Docker is an open platform that allows for development, shipment, and running applications. It will allow you to be able to separate your applications in one infrastructure and transfer that over to another infrastructure without any compatibility errors (eg. Windows to MacOS). Kubernetes, also known as K8s is an open-source system for automating deployment, scaling, and management of containerized applications. There are several built-in commands using “kubectl” that will allow you to deploy applications, rolling out changes, scaling the application to fit your needs, monitoring your applications and so much more, which in turn allows a way to make managing the applications much easier.

In this lab, we ran this software to deploy a MySQL image provided to us. We were able to manage the MySQL image by searching for its IP address, deploying it, and scaling it to the needs required for us in this lab. We were also able to deploy multiple deployments once we were able to grasp an understanding, allowing us to understand how Kubernetes can manage several containers at the same time.

The main thing we want to look at is why use Docker? In our labs, why can’t we just run a VM to make things how we perceive as easier? There are a lot of benefits to a VM, such as:

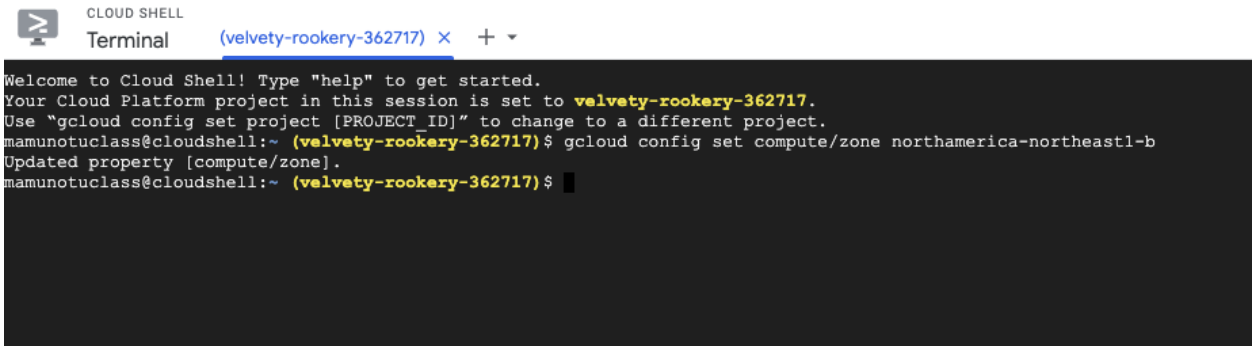
- All OS resources are available to the applications
- It is cheaper than having multiple machines
- Has well known security tools and controls

That being said, a VM isn’t perfect. As we have deployed different containers in the lab, we can see that if we were to run multiple containers on our VM at once, it would put a lot of load on top of the system. The best way to alleviate this issue is to have a server run multiple workloads on an OS, and that’s where Docker comes in. Docker lets you make containers within a program more portable and flexible to use, and Kubernetes orchestrates all that information and helps manage it as well. Some benefits with a container vs a VM would be the following:

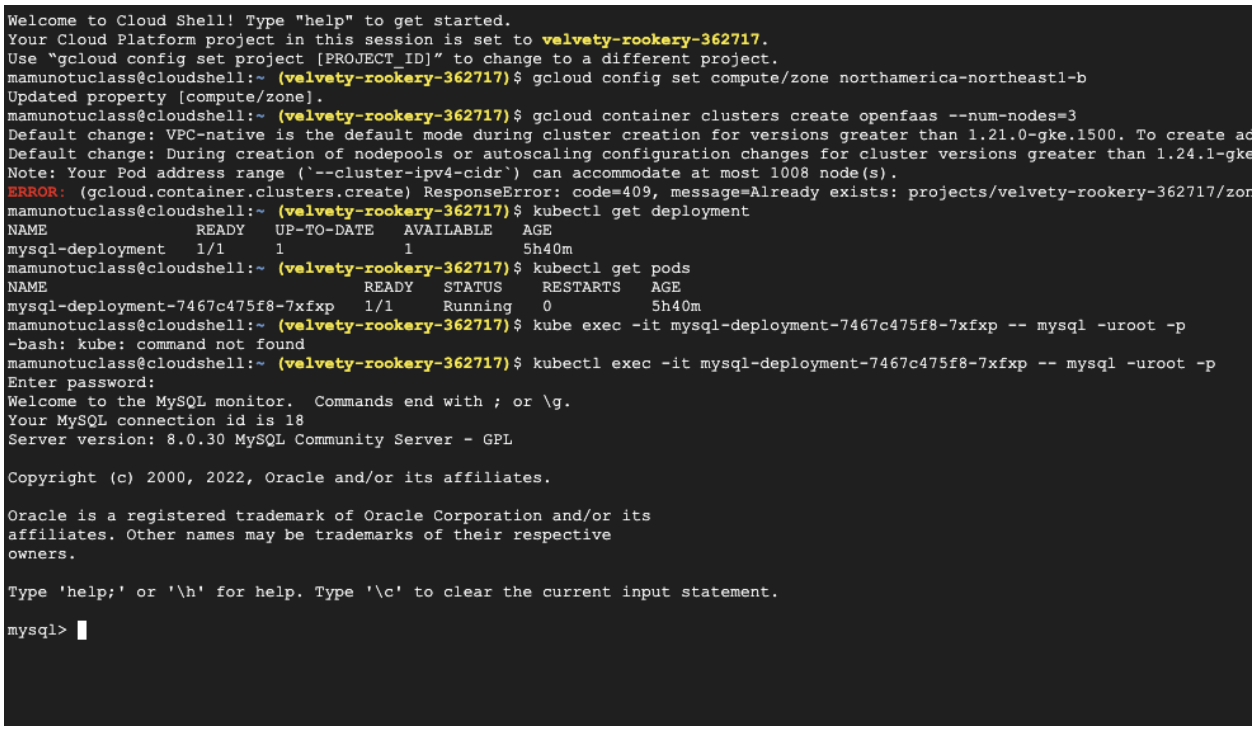
- A container would be very lightweight, where a VM is heavyweight
- A VM starts up in several minutes, whereas a container starts up almost immediately
- VMs require too much memory space, whereas a container requires less
- A VM is very isolated, so it has much stronger security whereas a container is not as secure

With this information, we can see why it would be much easier to deploy multiple containers, such as MySQL and the extra file we had made later on the lab, because if we were to do this in the VM, it would require much more time and it would pose more difficulty as we can just simply push all the commands with the container and Kubernetes to reach the goal we want, without putting too much stress on our resources.

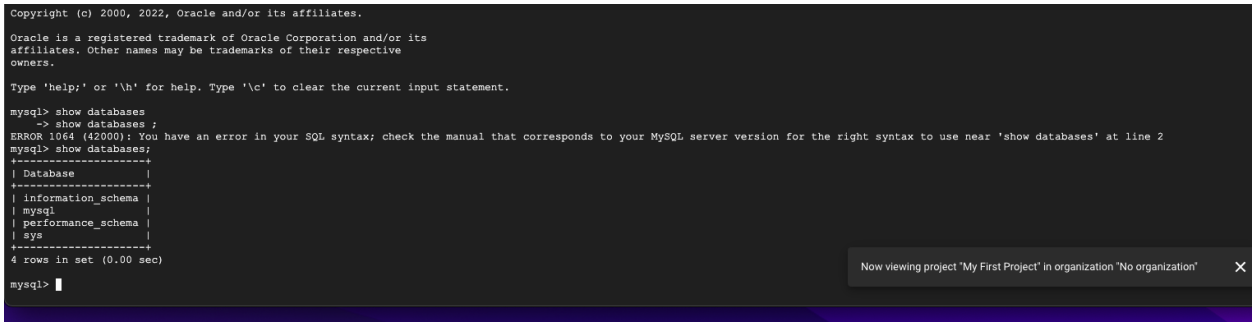
Deployment of MySQL Database:



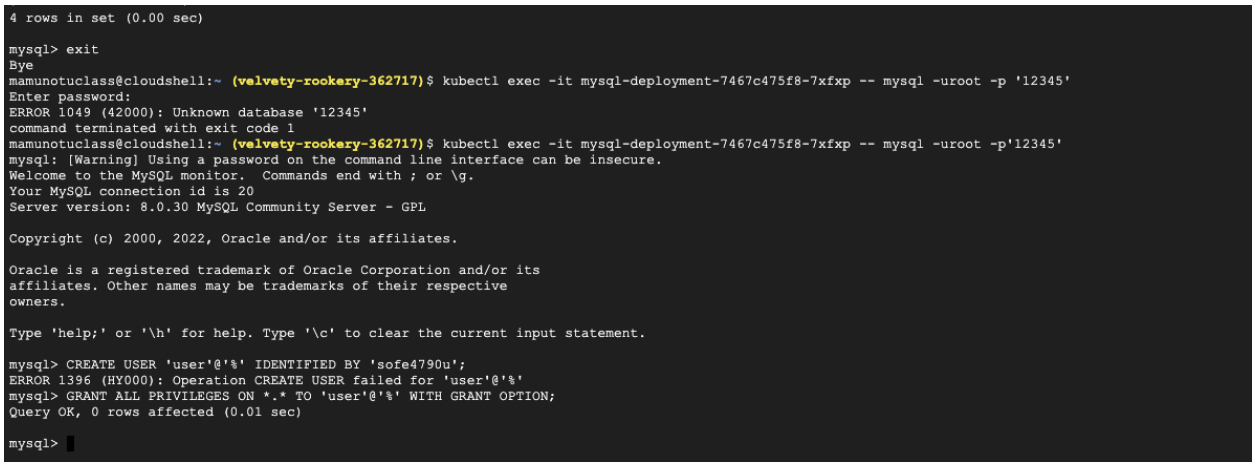
Configured time zone



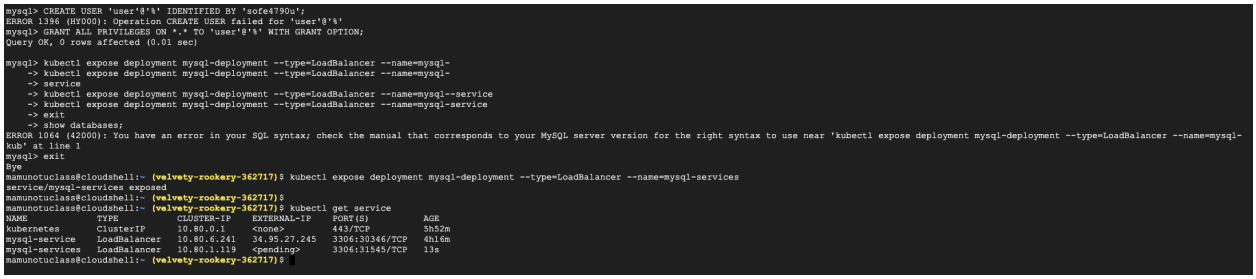
Created container clusters, deployed image, received pod information and accessed MySQL server from that information



Viewed database with Kubernetes commands



Altered password and then exited MySQL server



Retrieved external IP

```
mamunotuclass@cloudshell:~ (velvety-rookery-362717)$ kubectl expose deployment mysql-deployment --type=LoadBalancer --name=mysql-services
service/mysql-services exposed
mamunotuclass@cloudshell:~ (velvety-rookery-362717)$
mamunotuclass@cloudshell:~ (velvety-rookery-362717)$ kubectl get service
NAME             TYPE           CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
kubernetes       ClusterIP      10.80.0.1     <none>         443/TCP          5h52m
mysql-service    LoadBalancer   10.80.6.241   34.95.27.245   3306:30346/TCP   4h16m
mysql-services   LoadBalancer   10.80.1.119   <pending>      3306:31545/TCP   13s
mamunotuclass@cloudshell:~ (velvety-rookery-362717)$ mysql -uuser -psofe4790u -h34.95.27.245
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 21
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>
```

Used external IP to access MySQL server in a different method

```
mamunotuclass@cloudshell:~ (velvety-rookery-362717)$ kubectl apply -f mysql.yaml
Error from server (Invalid): error when creating "mysql.yaml": Service "mongoDB-service" is invalid: metadata.name: Invali
abc-123', regex used for validation is '[a-z]([-a-z0-9]*[a-z0-9])?')
Error from server (Invalid): error when creating "mysql.yaml": Deployment.apps "mongoDB" is invalid: [metadata.name: Inval
 for validation is '[a-z0-9]([-a-z0-9]*[a-z0-9])?(\.[a-z0-9]([-a-z0-9]*[a-z0-9])?)*'), spec.template.spec.containers[0].na
 regex used for validation is '[a-z0-9]([-a-z0-9]*[a-z0-9])?'), spec.template.spec.containers[0].ports[0].name: Invalid va
mamunotuclass@cloudshell:~ (velvety-rookery-362717)$ kubectl apply -f mysql.yaml
service/mysql-service unchanged
deployment.apps/mysql-deployment configured
mamunotuclass@cloudshell:~ (velvety-rookery-362717)$
```

Moving onto the advanced method of the deployment, deploying through the mysql.yaml file

```
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> kubectl delete deployment mysql-deployment
    -> show databases;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'kubectl delete deployment mysql-deployment
show databases' at line 1
mysql> exit
Bye
mamunotuclass@cloudshell:~ (velvety-rookery-362717)$ kubectl delete deployment mysql-deployment
deployment.apps "mysql-deployment" deleted
mamunotuclass@cloudshell:~ (velvety-rookery-362717)$ kubectl delete service mysql-service
service "mysql-service" deleted
```

Deleting the previous iterations of the deployments to create ease of access

```
(velvety-rookery-362717) ×                                   Open Editor

mamunotuclass@cloudshell:~ (velvety-rookery-362717)$ kubectl delete deployment mysql-deployment
deployment.apps "mysql-deployment" deleted
mamunotuclass@cloudshell:~ (velvety-rookery-362717)$ kubectl delete service mysql-service
service "mysql-service" deleted
mamunotuclass@cloudshell:~ (velvety-rookery-362717)$ kubectl apply -f mysql.yaml
service/mysql-service created
deployment.apps/mysql-deployment created
mamunotuclass@cloudshell:~ (velvety-rookery-362717)$ kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
mysql-deployment   1/1     1            1           24s
mamunotuclass@cloudshell:~ (velvety-rookery-362717)$ kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
mysql-deployment   1/1     1            1           31s
mamunotuclass@cloudshell:~ (velvety-rookery-362717)$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
mysql-deployment-5496fdc956-zw8qg   1/1     Running   0          40s
mamunotuclass@cloudshell:~ (velvety-rookery-362717)$ kubectl get service
NAME             TYPE           CLUSTER-IP     EXTERNAL-IP      PORT(S)          AGE
kubernetes       ClusterIP      10.80.0.1      <none>           443/TCP          22h
mysql-service    LoadBalancer   10.80.10.133   35.234.253.184   3306:31018/TCP   51s
mysql-services   LoadBalancer   10.80.1.119    34.152.52.239    3306:31545/TCP   16h
mamunotuclass@cloudshell:~ (velvety-rookery-362717)$
```

Retrieved all information again (pod, deployment, service)

mysql.yaml > ...

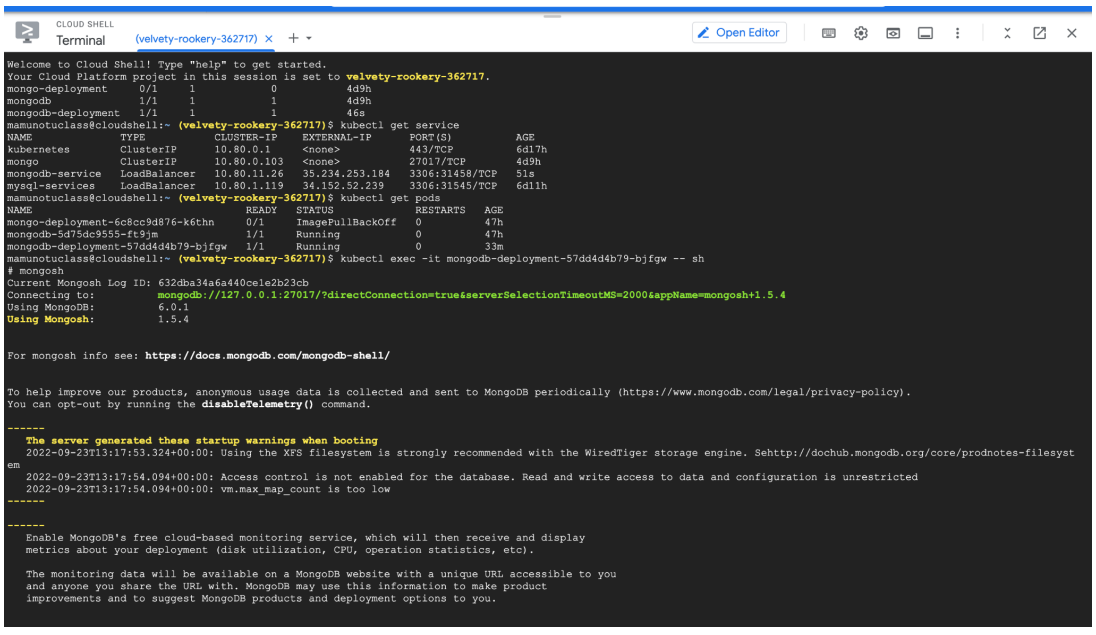
```yaml
apiVersion: v1
kind: Service
metadata:
  name: mysql-service
spec:
  type: LoadBalancer
  ports:
    - port: 3306
  selector:
    app: mysql
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - image: mysql/mysql-server
          name: mysql
          env:
            - name: MYSQL_ROOT_PASSWORD
              value: password
            - name: MYSQL_USER
              value: user
            - name: MYSQL_PASSWORD
              value: sofe4790u
            - name: MYSQL_DATABASE
              value: myDB
          ports:
            - containerPort: 3306
              name: mysql
```

Code required to run the server with all the necessary information (ie. Port, Load balancer, image, type, name, apps)

```
mysql> exit
Bye
mamunotuclass@cloudshell:~ (velvety-rookery-362717)$ kubectl get service
NAME              TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
kubernetes        ClusterIP      10.80.0.1       <none>           443/TCP          23h
mysql-service     LoadBalancer   10.80.10.133    35.234.253.184   3306:31018/TCP   19m
mysql-services    LoadBalancer   10.80.1.119     34.152.52.239    3306:31545/TCP   17h
mamunotuclass@cloudshell:~ (velvety-rookery-362717)$ mysql -uuser -psofe4790u -h35.234.253.184
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 24
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 (0.04 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 (0.03 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>
```

Entered commands from the lab to display database information

MongoDB deployment:



Following a similar approach to the advanced deployment method for MySQL, we deploy the MongoDB.yaml in similar fashion retrieving the pods, deploying the image, and retrieving the service for the IP. Then accessing the server for mongoDB, we will then be able to access mydB to alter the database for mongoDB as well.

```
test> use myDb
switched to db myDb
myDb> yse test
Uncaught:
SyntaxError: Missing semicolon. (1:3)

> 1 | yse test
    |    ^
  2 |

myDb> use test
switched to db test
test> show dbs
admin   40.00 KiB
config  60.00 KiB
local   40.00 KiB
test>

test> db.test.insert({name: "jack", age: 24})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("632dbb1182404578747715e0") }
}
test> db.test.inset{(name: "bob", age: 31})
Uncaught:
SyntaxError: Missing semicolon. (1:13)

> 1 | db.test.inset{(name: "bob", age: 31})
    |              ^
  2 |

test> db.test.insert({name: "bob", age: 31})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("632dbb4282404578747715e1") }
}
test> db.test.insert({name:
...
... db.test.insert
... clear
Uncaught:
SyntaxError: Unexpected token, expected "," (4:0)
```

Seeing the kibibytes for each test, and then manually adding in the information for the database.test to ensure our code works.

```
SyntaxError: Unexpected token, expected "," (4:0)

  2 |
  3 | db.test.insert
> 4 | clear
    | ^
  5 |

test> db.test.insert({name: "george", age: 28})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("632dbb7f82404578747715e2") }
}
test> show collections
test
test> db.test.find()
[
  { _id: ObjectId("632dbb1182404578747715e0"), name: 'jack', age: 24 },
  { _id: ObjectId("632dbb4282404578747715e1"), name: 'bob', age: 31 },
  {
    _id: ObjectId("632dbb7f82404578747715e2"),
    name: 'george',
    age: 28
  }
```

Testing our code to make sure we can find the ages that are greater than 23 and ensure that the database is working the way we want.

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
          ports:
            - containerPort: 3306
              name: mongodb
```

Code required to access mongoDB server.

Conclusion:

In conclusion, we were able to successfully utilize Kubernetes and enhance our knowledge on Docker to set up different containers and use different applications with .yaml files and increase our knowledge on containerizations.