Kubernetes Signature Project

Present By: Nisarg Patel
Student Id: 19619

Table Of Contents

Introduction

The project has consist two applications running on different kubernetes pods. The first **student record application** running on Node.js + MongoDB + Google kubernetes Engine (GKE) technologies and the second one, **bookstore application**, using with MongoDB + Python Flask Web Framework + REST API + GKE technologies.

Design

Both applications have same domain name but access in different path with used of Kubernetes Ingress component

Domain name: **cs571.signatureproject.com**

- Student record application: cs571.signatureproject.com/studentserver
- Bookstore application: cs571.signatureproject.com/bookshelf

Implementation

First, launch the kubia cluster

gcloud container clusters create kubia --num-nodes=1 --machine-type=e2-micro --region=us-west1

```
NAME: kubia
LOCATION: us-west1
MASTER_VERSION: 1.21.9-gke.1002
MASTER_IP: 35.227.137.24
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.21.9-gke.1002
NUM_NODES: 3
STATUS: RUNNING
```

Implementation

Create a GCE Persistent Disk **mongodb** with size of 10GiB.

```
patel19619@cloudshell:~ (cs571-signature-project)$ gcloud compute disks create --size=10GiB --zone=us-west1-a mon
godb
WARNING: You have selected a disk size of under [200GB]. This may result in poor I/O performance.For more informa
tion, see: https://developers.google.com/compute/docs/disks#performance.
Created [https://www.googleapis.com/compute/v1/projects/cs571-signature-project/zones/us-west1-a/disks/mongodb].
NAME: mongodb
ZONE: us-west1-a
SIZE_GB: 10
TYPE: pd-standard
STATUS: READY
```

Create a Mongodb-deployment pod on Kubernetes

```
patel19619@cloudshell:~/mongodb/yaml (cs571-signature-project)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
```

Implementation

Create a External service for mongodb pod to access database with EXTERNAL-IP address

```
patel19619@cloudshell:~/mongodb/yaml (cs571-signature-project)$ kubectl get svc
NAME              TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)           AGE
kubernetes        ClusterIP      10.36.0.1      <none>          443/TCP           43m
mongodb-service   LoadBalancer   10.36.13.174   34.127.12.117   27017:30652/TCP   35s
```

Successfully launched the mongodb database service on gcp.

Now follow the same procedure to complete the configuration of student server application and bookshelf python application.

For student server application first create the **studentServer.js** file with working code of fetch the student information.

Create the Dockerfile to build studentserver docker image

Implementation

vim Dockerfile

```
FROM node:14
ADD studentServer.js /studentServer.js
RUN npm install mongodb
ENTRYPOINT ["node", "studentServer.js"]
```

Build the student server docker image

docker build -t yourdockerhubID/studentserver .

```
Successfully built a43e3412ef6c
Successfully tagged cs571/studentserver:latest
```

Push the builded studentserver docker image to dock hub repository

```
latest: digest: sha256:a2e1699f3edbe5493e80ef2411c1c974e48cd21ed80bcdc76dae70b2f59cf1df size: 2424
```

Implementation

Create ConfigMap to store mongodb configuration to avoid re-building docker image

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  # SERVICE_N                                    l:SERVICE_PORT
  MONGO_URL:  Your_mongodb_service_externalIP
  MONGO_DATABASE: mydb
```

Now deploy the studentserver-deployment with configmap reference of mongodb database URL address and name.

Create the studentserver LoadBalancer service to communicate with studentserver container application from cluster.

Implementation

To deploy the bookshelf python application follow the same procedure same as student server application deployment.

Once the both application successfully deployed and mongoDB database is running properly, you would get same cluster configuration

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ kubectl get all
NAME                                          READY   STATUS    RESTARTS   AGE
pod/bookshelf-deployment-59b57bf758-lwwtl     1/1     Running   0          9m46s
pod/mongodb-deployment-57dc68b4bd-tmv4g       1/1     Running   0          8h
pod/studentserver-deployment-84cbd9dcd4-snjs6 1/1     Running   1          8h

NAME                           TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)          AGE
service/bookshelf-service      LoadBalancer   10.36.15.181   34.127.84.154   5000:32540/TCP   8m30s
service/kubernetes             ClusterIP      10.36.0.1      <none>          443/TCP          8h
service/mongodb-service        LoadBalancer   10.36.3.133    34.145.21.88    27017:31337/TCP  8h
service/studentserver-service  LoadBalancer   10.36.1.37     35.247.18.41    8080:31228/TCP   8h

NAME                                     READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/bookshelf-deployment     1/1     1            1           9m47s
deployment.apps/mongodb-deployment       1/1     1            1           8h
deployment.apps/studentserver-deployment 1/1     1            1           8h

NAME                                                DESIRED   CURRENT   READY   AGE
replicaset.apps/bookshelf-deployment-59b57bf758     1         1         1       9m47s
replicaset.apps/mongodb-deployment-57dc68b4bd       1         1         1       8h
replicaset.apps/studentserver-deployment-84cbd9dcd4 1         1         1       8h
```

Implementation

Launch the minkube to enable egress service: $ **minikune start**

$ **minikube addons enable ingress**

Create the ingress service for both applications with same domain name but different path

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ kubectl get ingress
NAME              CLASS   HOSTS                        ADDRESS        PORTS   AGE
project-server    nginx   cs571.signatureproject.com   192.168.49.2   80      15s
```

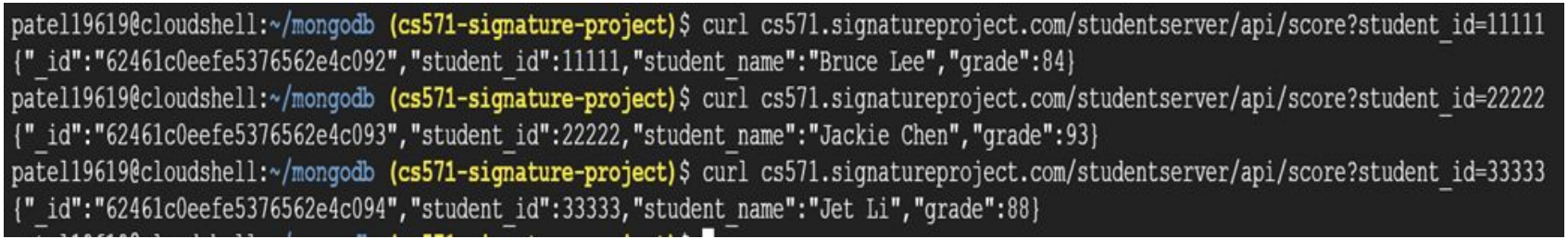
Add the ingress service ip Address to /etc/hosts

```
# Kubernetes-managed hosts file.
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
fe00::0 ip6-mcastprefix
fe00::1 ip6-allnodes
fe00::2 ip6-allrouters
172.17.0.4      cs-810844977107-default
192.168.49.2    cs571.signatureproject.com
```

Test

To test the student server Node.js application

curl cs571.signatureproject.com/studentserver/api/score?student_id=11111/22222/33333



Test

For Bookshelf python + REST API application do following operations

- Add Books

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ curl -X POST -d "{\"book_name\": \"cloud com
puting\",\"book_author\": \"unkown\", \"isbn\": \"123456\" }" http://cs571.signatureproject.com/bookshelf/book
{
  "message": "Book saved successfully!"
}
```

- Get Book list

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ curl cs571.signatureproject.com/bookshelf/bo
oks
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "624626504b936020a8225e5f"
  }
]
```

Test

- Update Book

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ curl -X PUT -d "{\"book_name\": \"123\",\"bo
ok_author\": \"test\", \"isbn\": \"123updated\" }" http://cs571.signatureproject.com/bookshelf/book/624626504b936
020a8225e5f
{
  "message": "Book updated successfully!"
}
```

- Get Updated Book

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ curl cs571.signatureproject.com/bookshelf/bo
oks
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123updated",
    "id": "624626504b936020a8225e5f"
  }
]
```

Test

- Delete Book

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ curl -X DELETE cs571.signatureproject.com/bo
okshelf/book/624626504b936020a8225e5f
{
  "message": "Book deleted successfully!"
}
```

- Book list

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ curl cs571.signatureproject.com/bookshelf/bo
oks
[]
```

Enhancement Ideas

- Try getting a real domain name, and host both applications under it for public test
- Adding TLS to MongoDB to secure the database to prevent malicious access

Conclusion

- Kubernetes useful for scale resources and applications in real time
- Orchestrate containers on multiple hosts
- Control and automate deployments and updates
- Multiple applications can be hosted on single domain name (i.e. cs.signatureproject.com)
- Save money by optimizing infrastructural resources with more efficient use of hardware