CS571 Signature Project

MERT KABADAYI

19616

Step1 Create MongoDB using Persistent Volume on GKE, and insert records into it

1) Create a cluster as usual on GKE

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

```
kabadayi19616@cloudshell:~ (cloudhmw)$ gcloud container clusters list
NAME: kubia
LOCATION: us-central1-a
MASTER_VERSION: 1.21.9-gke.1002
MASTER_IP: 35.239.86.111
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.21.9-gke.1002
NUM_NODES: 1
STATUS: RUNNING
```

2) Let's create a Persistent Volume first

```
kabadayi19616@cloudshell:~ (cloudhmw)$ gcloud compute disks create --size=10GiB --zone=us-central1-a mongodb
WARNING: You have selected a disk size of under [200GB]. This may result in poor I/O performance. For more information, see: https
sks#performance.
Created [https://www.googleapis.com/compute/v1/projects/cloudhmw/zones/us-central1-a/disks/mongodb].
NAME: mongodb
ZONE: us-central1-a
SIZE_GB: 10
TYPE: pd-standard
STATUS: READY

New disks are unformatted. You must format and mount a disk before it
```

3) Now create a mongodb deployment with this yaml file

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
spec:
  selector:
    matchLabels:
      app: mongodb
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
       app: mongodb
    spec:
     containers:
     - image: mongo
       name: mongo
       ports:
         - containerPort: 27017
       volumeMounts:
         - name: mongodb-data
           mountPath: /data/db
     volumes:
       - name: mongodb-data
         gcePersistentDisk:
           pdName: mongodb
```

```
kabadayi19616@cloudshell:~ (cloudhmw)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
```

4) Check if the deployment pod has been successfully created and started running

```
kabadayi19616@cloudshell:~ (cloudhmw)$ kubectl get pods
NAME                                 READY   STATUS    RESTARTS   AGE
mongodb-deployment-57dc68b4bd-n9xch  1/1     Running   0          5m11s
```

5) Create a service for the mongoDB, so it can be accessed from outside

```
kabadayi19616@cloudshell:~ (cloudhmw)$ vim mongodb-service.yaml
kabadayi19616@cloudshell:~ (cloudhmw)$ cat mongodb-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  type: LoadBalancer
  ports:
    - port: 27017
      targetPort: 27017
  selector:
    app: mongodb
```

```
kabadayi19616@cloudshell:~ (cloudhmw)$ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
```

6) Wait couple of minutes, and check if the service is up

```
kabadayi19616@cloudshell:~ (cloudhmw)$ kubectl get svc
NAME              TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)           AGE
kubernetes        ClusterIP      10.56.0.1      <none>          443/TCP           45m
mongodb-service   LoadBalancer   10.56.3.243    35.238.10.240   27017:30027/TCP   59s
```

7) Now try and see if mongoDB is functioning for connections using the External-IP

```
kabadayi19616@cloudshell:~ (cloudhmw)$ kubectl exec -it mongodb-deployment-57dc68b4bd-n9xch -- bash
root@mongodb-deployment-57dc68b4bd-n9xch:/#
```

Now you are inside the mongodb deployment pod

Try

mongo External-IP

```
root@mongodb-deployment-57dc68b4bd-n9xch:/# mongo External-IP
MongoDB shell version v5.0.6
connecting to: mongodb://127.0.0.1:27017/External-IP?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("d877ab4a-9d58-4f58-a6b9-5c3c5fdc5357") }
MongoDB server version: 5.0.6
================
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
================
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        https://docs.mongodb.com/
Questions? Try the MongoDB Developer Community Forums
        https://community.mongodb.com
---
The server generated these startup warnings when booting:
        2022-03-31T00:24:23.095+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage en
ilesystem
        2022-03-31T00:24:25.103+00:00: Access control is not enabled for the database. Read and write access to data a
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).
```

see something like this, which means your mongoDB is up and can be accessd using the External-IP

8) Type exit to exit mongodb and back to our google console

```
---
> exit
bye
root@mongodb-deployment-57dc68b4bd-n9xch:/# exit
exit
```

9) We need to insert some records into the mongoDB for later use

```
kabadayi19616@cloudshell:~ (cloudhmw)$ node
Welcome to Node.js v12.14.1.
Type ".help" for more information.
>
```

Enter the following line by line

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://EXTERNAL-IP/mydb"
// Connect to the db
MongoClient.connect(url,{ useNewUrlParser: true, useUnifiedTopology: true },
function(err, client){
if (err)
throw err;
// create a document to be inserted
var db = client.db("studentdb");
const docs = [
{ student_id: 11111, student_name: "Bruce Lee", grade: 84},
{ student_id: 22222, student_name: "Jackie Chen", grade: 93 },
{ student_id: 33333, student_name: "Jet Li", grade: 88}
]
db.collection("students").insertMany(docs, function(err, res){
if(err) throw err;
console.log(res.insertedCount);
client.close();
});
db.collection("students").findOne({"student_id": 11111},
function(err, result){
console.log(result);
});
});
```

If Everything is correct, you should see this, 3 means three records was inserted, and we tried search for student_id=11111

```
... function(err, client){
..... if (err)
..... throw err;
..... // create a document to be inserted
..... var db = client.db("studentdb");
..... const docs = [
..... { student_id: 11111, student_name: "Bruce Lee", grade: 84},
..... { student_id: 22222, student_name: "Jackie Chen", grade: 93 },
..... { student_id: 33333, student_name: "Jet Li", grade: 88}
..... ]
..... db.collection("students").insertMany(docs, function(err, res){
....... if(err) throw err;
....... console.log(res.insertedCount);
....... //client.close();
....... });
..... db.collection("students").findOne({"student_id": 11111},
....... function(err, result){
......... if(err) throw err;
......... console.log(result);
......... });
..... });
undefined
> 3
{
  _id: new ObjectId("62450f9c87e62387e532cd05"),
  student_id: 11111,
  student_name: 'Bruce Lee',
  grade: 84
}
>
```

Step2 Modify our studentServer to get records from MongoDB and deploy to GKE

1) Create a studentServer

var http = require('http');

var url = require('url');

var mongodb = require('mongodb');

const {

MONGO_URL,

MONGO_DATABASE

} = process.env;

// - Expect the request to contain a query

// string with a key 'student_id' and a student ID as

```
// the value. For example
// /api/score?student_id=1111
// - The JSON response should contain only 'student_id', 'student_name'
// and 'student_score' properties. For example:
//
// {
// "student_id": 1111,
// "student_name": Bruce Lee,
// "student_score": 84
// }
//
var MongoClient = mongodb.MongoClient;
var uri = `mongodb://${MONGO_URL}/${MONGO_DATABASE}`;
// Connect to the db
console.log(uri);
var server = http.createServer(function (req, res) {
var result;
// req.url = /api/score?student_id=11111
var parsedUrl = url.parse(req.url, true);
var student_id = parseInt(parsedUrl.query.student_id);
// match req.url with the string /api/score
if (/^\/api\/score/.test(req.url)) {
// e.g., of student_id 1111
MongoClient.connect(uri,{ useNewUrlParser: true, useUnifiedTopology:
true }, function(err, client){
if (err)
throw err;
var db = client.db("studentdb");
```

```
db.collection("students").findOne({"student_id":student_id},
(err, student) => {
if(err)
throw new Error(err.message, null);
if (student) {
res.writeHead(200, { 'Content-Type': 'application/json'
})
res.end(JSON.stringify(student)+ '\n')
}else {
res.writeHead(404);
res.end("Student Not Found \n");
}
});
});
} else {
res.writeHead(404);
res.end("Wrong url, please try again\n");
}
});
server.listen(8080);
```

2) Create Dockerfile

```
FROM node:7
ADD studentServer.js /studentServer.js
ENTRYPOINT ["node", "studentServer.js"]
RUN npm install mongodb
```

3) Build the studentserver docker image

```
    `-- smart-buffer@4.2.0

npm WARN enoent ENOENT: no such file or directory, open '/package.json'
npm WARN !invalid#1 No description
npm WARN !invalid#1 No repository field.
npm WARN !invalid#1 No README data
npm WARN !invalid#1 No license field.
npm info ok
Removing intermediate container c2aa4f6a81ab
 ---> 969872ac180b
Successfully built 969872ac180b
Successfully tagged herre0/studentserver:latest
```

4) Push the docker image

```
kabadayi19616@cloudshell:~ (cloudhmw)$ docker push herre0/studentserver
Using default tag: latest
The push refers to repository [docker.io/herre0/studentserver]
64449004d399: Pushed
7377bbee496b: Pushed
ab90d83fa34a: Mounted from library/node
8ee318e54723: Mounted from library/node
e6695624484e: Mounted from library/node
da59b99bbd3b: Mounted from library/node
5616a6292c16: Mounted from library/node
f3ed6cb59ab0: Mounted from library/node
654f45ecb7e3: Mounted from library/node
2c40c66f7667: Mounted from library/node
latest: digest: sha256:ca689511addd545c0f27f2c614277d689c03e16a31b84591fb4711abc7aa2abb size: 2424
```

Step3 Create a python Flask bookshelf REST API and deploy on GKE

1) Create bookshelf.py

```
from flask import Flask, request, jsonify
from flask_pymongo import PyMongo
from flask import request
from bson.objectid import ObjectId
import socket
import os
app = Flask(__name__)
app.config["MONGO_URI"] =
"mongodb://"+os.getenv("MONGO_URL")+"/"+os.getenv("MONGO_DATABASE")
app.config['JSONIFY_PRETTYPRINT_REGULAR'] = True
```

```
mongo = PyMongo(app)
db = mongo.db
@app.route("/")
def index():
hostname = socket.gethostname()
return jsonify(
message="Welcome to bookshelf app! I am running inside {}
pod!".format(hostname)
)
@app.route("/books")
def get_all_tasks():
books = db.bookshelf.find()
data = []
for book in books:
data.append({
"id": str(book["_id"]),
"Book Name": book["book_name"],
"Book Author": book["book_author"],
"ISBN" : book["ISBN"]
})
return jsonify(
data
)
@app.route("/book", methods=["POST"])
def add_book():
book = request.get_json(force=True)
db.bookshelf.insert_one({
"book_name": book["book_name"],
```

```
"book_author": book["book_author"],
"ISBN": book["isbn"]
})
return jsonify(
message="Task saved successfully!"
)
@app.route("/book/<id>", methods=["PUT"])
def update_book(id):
data = request.get_json(force=True)
print(data)
response = db.bookshelf.update_many({"_id": ObjectId(id)}, {"$set":
{"book_name": data['book_name'],
"book_author": data["book_author"], "ISBN": data["isbn"]
}})
if response.matched_count:
message = "Task updated successfully!"
else:
message = "No book found!"
return jsonify(
message=message
)
@app.route("/book/<id>", methods=["DELETE"])
def delete_task(id):
response = db.bookshelf.delete_one({"_id": ObjectId(id)})
if response.deleted_count:
message = "Task deleted successfully!"
else:
message = "No book found!"
```

```
return jsonify(
message=message
)
@app.route("/tasks/delete", methods=["POST"])
def delete_all_tasks():
db.bookshelf.remove()
return jsonify(
message="All Books deleted!"
)
if __name__ == "__main__":
app.run(host="0.0.0.0", port=5000)
```

2)Create a Dockerfile

```
FROM python:alpine3.7
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
ENV PORT 5000
EXPOSE 5000
ENTRYPOINT [ "python3" ]
CMD [ "bookshelf.py" ]
```

3)Build the bookshelf app into a docker image

```
kabadayi19616@cloudshell:~ (cloudhmw)$ docker build -t herre0/bookshelf .
Sending build context to Docker daemon  39.37MB
Step 1/8 : FROM python:alpine3.7
alpine3.7: Pulling from library/python
48ecbb6b270e: Pull complete
692f29ee68fa: Pull complete
6439819450d1: Pull complete
3c7be240f7bf: Pull complete
ca4b349df8ed: Pull complete
Digest: sha256:35f6f83ab08f98c727dbefd53738e3b3174a48b4571ccb1910bae480dcdba847
Status: Downloaded newer image for python:alpine3.7
 ---> 00be2573e9f7
Step 2/8 : COPY . /app
 ---> 417447eafde9
Step 3/8 : WORKDIR /app
 ---> Running in c0213b0f5388
Removing intermediate container c0213b0f5388
 ---> baeb4fbe01bb
Step 4/8 : RUN pip install -r requirements.txt
 ---> Running in 52cb3b024d5b
Could not open requirements file: [Errno 2] No such file or directory: 'requirements.txt'
You are using pip version 19.0.1, however version 22.0.4 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
The command '/bin/sh -c pip install -r requirements.txt' returned a non-zero code: 1
```

4) Push the docker image to your dockerhub

```
kabadayi19616@cloudshell:~ (cloudhmw)$ docker push herre0/bookshelf
Using default tag: latest
The push refers to repository [docker.io/herre0/bookshelf]
50edcfb9375c: Pushed
9a758fd0f1dd: Pushed
5fa31f02caa8: Mounted from library/python
88e61e328a3c: Mounted from library/python
9b77965e1d3f: Mounted from library/python
50f8b07e9421: Mounted from library/python
629164d914fc: Mounted from library/python
latest: digest: sha256:400745deb40d48b2095e54a3dc8a79c780c1e11aa55da6b1fd62c4a831a210d5 size: 1790
```

Step4 Create ConfigMap for both applications to store MongoDB URL and MongoDB name

1) Create a file named studentserver-configmap.yaml

apiVersion: v1

kind: ConfigMap

metadata:

name: studentserver-config

data:

MONGO_URL: Change-this-to-your-mongoDB-EXTERNAL-IP

MONGO_DATABASE: mydb

2) Create a file named bookshelf-configmap.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
name: bookshelf-config
data:
# SERVICE_NAME.NAMESPACE.svc.cluster.local:SERVICE_PORT
MONGO_URL: Change-this-to-your-mongoDB-EXTERNAL-IP
MONGO_DATABASE: mydb
```

Step5 Expose 2 application using ingress with Nginx, so we can put them on the same Domain but different PATH

1) Create studentserver-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
name: web
labels:
app: studentserver-deploy
spec:
replicas: 1
selector:
matchLabels:
app: web
template:
metadata:
labels:
app: web
```

spec:

containers:

- image: zhou19539/studentserver

imagePullPolicy: Always

name: web

ports:

- containerPort: 8080

env:

- name: MONGO_URL

valueFrom:

configMapKeyRef:

name: studentserver-config

key: MONGO_URL

- name: MONGO_DATABASE

valueFrom:

configMapKeyRef:

name: studentserver-config

key: MONGO_DATABASE

2) Create bookshelf-deployment.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
name: bookshelf-deployment
labels:
app: bookshelf-deployment
spec:
replicas: 1
selector:
matchLabels:
app: bookshelf-deployment
template:
metadata:

labels:
app: bookshelf-deployment
spec:
containers:
- image: zhou19539/bookshelf
imagePullPolicy: Always
name: bookshelf-deployment
ports:
- containerPort: 5000
env:
- name: MONGO_URL
valueFrom:
configMapKeyRef:
name: bookshelf-config
key: MONGO_URL
- name: MONGO_DATABASE
valueFrom:
configMapKeyRef:
name: bookshelf-config
key: MONGO_DATABASE

3) Create sutdentserver-service.yaml

apiVersion: v1

kind: Service

metadata:

name: web

spec:

type: LoadBalancer

ports:

service port in cluster

- port: 8080

port to contact inside container

targetPort: 8080

selector:

app: web

4) Create bookshelf-service.yaml

apiVersion: v1

kind: Service

metadata:

name: bookshelf-service

spec:

type: LoadBalancer

ports:

service port in cluster

- port: 5000

port to contact inside container

targetPort: 5000

selector:

app: bookshelf-deployment

5) Start minikube

```
kabadayi19616@cloudshell:~ (cloudhmw)$ minikube start
* minikube v1.25.2 on Debian 11.2 (amd64)
  - MINIKUBE_FORCE_SYSTEMD=true
  - MINIKUBE_HOME=/google/minikube
  - MINIKUBE_WANTUPDATENOTIFICATION=false
* Automatically selected the docker driver. Other choices: ssh, none
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Downloading Kubernetes v1.23.3 preload ...
    > preloaded-images-k8s-v17-v1...: 505.68 MiB / 505.68 MiB  100.00% 213.49 M
* Creating docker container (CPUs=2, Memory=4000MB) ...
* Preparing Kubernetes v1.23.3 on Docker 20.10.12 ...
  - kubelet.cgroups-per-qos=false
  - kubelet.enforce-node-allocatable=""
  - kubelet.housekeeping-interval=5m
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

6) Start Ingress

```
kabadayi19616@cloudshell:~ (cloudhmw)$ minikube addons enable ingress
  - Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
  - Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
  - Using image k8s.gcr.io/ingress-nginx/controller:v1.1.1
* Verifying ingress addon...
* The 'ingress' addon is enabled
```

7) Create studentserver related pods and start service using the yaml files

kubectl apply -f studentserver-deployment.yaml

kubectl apply -f studentserver-configmap.yaml

kubectl apply -f studentserver-service.yaml

```
kabadayi19616@cloudshell:~ (cloudhmw)$ kubectl apply -f studentserver-deployment.yaml
deployment.apps/web created
kabadayi19616@cloudshell:~ (cloudhmw)$ kubectl apply -f studentserver-configmap.yaml
configmap/studentserver-config created
kabadayi19616@cloudshell:~ (cloudhmw)$ kubectl apply -f studentserver-service.yaml
error: error parsing studentserver-service.yaml: error converting YAML to JSON: yaml: line 8: did not find expected '-' indicator
kabadayi19616@cloudshell:~ (cloudhmw)$ vim studentserver-service.yaml
kabadayi19616@cloudshell:~ (cloudhmw)$ kubectl apply -f studentserver-service.yaml
service/web created
```

8) Create bookshelf related pods and start service using the above yaml file

```
kabadayi19616@cloudshell:~ (cloudhmw)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created
kabadayi19616@cloudshell:~ (cloudhmw)$ kubectl apply -f bookshelf-configmap.yaml
error: error validating "bookshelf-configmap.yaml": error validating data: [Validation
v1.ConfigMap, ValidationError(ConfigMap): unknown field "name" in io.k8s.api.core.v1.C
kabadayi19616@cloudshell:~ (cloudhmw)$ vim bookshelf-configmap.yaml
kabadayi19616@cloudshell:~ (cloudhmw)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
kabadayi19616@cloudshell:~ (cloudhmw)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
```

9) Create an ingress service yaml file called studentservermongoIngress.yaml

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: server

annotations:

nginx.ingress.kubernetes.io/rewrite-target: /$2

spec:

rules:

- host: cs571.project.com

http:

paths:

- path: /studentserver(/|$)(.*)

pathType: Prefix

backend:

service:

name: web

port:

number: 8080

- path: /bookshelf(/|$)(.*)

pathType: Prefix

backend:

service:

name: bookshelf-service

port:

number: 5000

10) Create the ingress service using the above yaml file

```
kabadayi19616@cloudshell:~ (cloudhmw)$ kubectl apply -f studentservermongoIngress.yaml
ingress.networking.k8s.io/server created
```

11) Check if ingress is running

```
kabadayi19616@cloudshell:~ (cloudhmw)$ kubectl get ingress
NAME     CLASS   HOSTS              ADDRESS        PORTS   AGE
server   nginx   cs571.project.com  192.168.49.2   80      59s
```

12) Add 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-738046022024-default-boost-dlp9q
192.168.49.2 cs571.project.com
```

13)Check end points if they work

```
kabadayi19616@cloudshell:~ (cloudhmw)$ curl cs571.project.com/studentserver/api/score?student_id=11111
{"_id":"62450f9c87e62387e532cd05","student_id":11111,"student_name":"Bruce Lee","grade":84}
```

On another path, you should be able to use the REST API with bookshelf application I.e list all books

```
kabadayi19616@cloudshell:~ (cloudhmw)$ curl cs571.project.com/bookshelf/books
[]
```

Add a book

```
kabadayi19616@cloudshell:~ (cloudhmw)$ curl -X POST -d "{\"book_name\": \"cloud computing\",\"book_author\":\"unkown\", \"isbn\": \"123456\" }" http://cs571.project.com/bookshelf/book
{
  "message": "Task saved successfully!"
}
```

```
kabadayi19616@cloudshell:~ (cloudhmw)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "6245473f4db61263a771c71c"
  }
]
```

Delete a book

```
kabadayi19616@cloudshell:~ (cloudhmw)$ curl -X DELETE cs571.project.com/bookshelf/book/6245473f4db61263a771c71c
{
  "message": "Task deleted successfully!"
}
```

The same domain and 2 different applications that’re built in different languages.