Name: Fithi Ghebreamlak

ID: 20068

CS571 Signature Project

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

Wait for the creation to finish,

```
kubeconfig entry generated for kubia.
NAME: kubia
LOCATION: us-west1
MASTER_VERSION: 1.29.6-gke.1038001
MASTER_IP: 35.197.98.163
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.29.6-gke.1038001
NUM_NODES: 3
STATUS: RUNNING
fghebrea408@cloudshell:~ (cs-571-signature-project)$
```

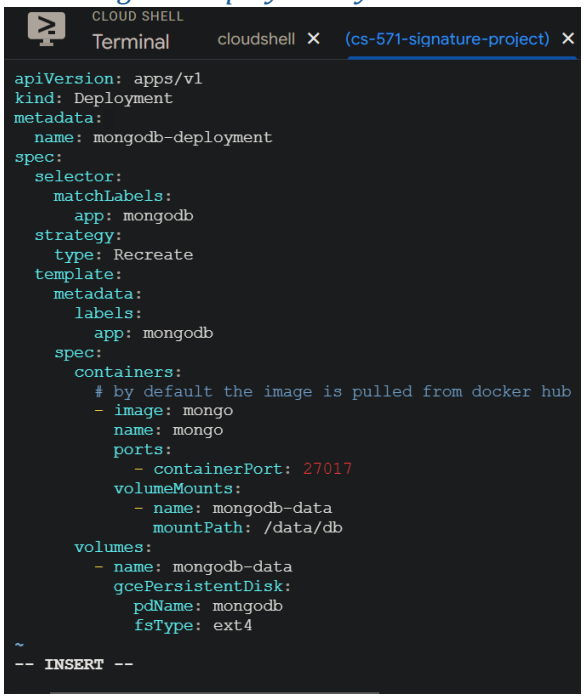
2. Let's create a Persistent Volume first, if you have created a persistent volume for the week10's homework, you can skip this one

gcloud compute disks create --size=10GiB --zone=us-west1-a mongodb

```
Created [https://www.googleapis.com/compute/v1/projects/cs-571-signature-project/zones/us-west1-a/disks/mongodb].
NAME: mongodb
ZONE: us-west1-a
SIZE_GB: 10
TYPE: pd-standard
STATUS: READY

New disks are unformatted. You must format and mount a disk before it
can be used. You can find instructions on how to do this at:

https://cloud.google.com/compute/docs/disks/add-persistent-disk#formatting

fghebrea408@cloudshell:~ (cs-571-signature-project)$
```

3. Now create a mongodb deployment with this yaml file

vim mongodb-deployment.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:
    # by default the image is pulled from docker hub
    - image: mongo
     name: mongo
     ports:
      - containerPort: 27017
     volumeMounts:
      - name: mongodb-data
       mountPath: /data/db
   volumes:
    - name: mongodb-data
     gcePersistentDisk:
      pdName: mongodb
      fsType: ext4
```

kubectl apply -f mongodb-deployment.yaml

```
fghebrea408@cloudshell:~ (cs-571-signature-project)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
fghebrea408@cloudshell:~ (cs-571-signature-project)$
```

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

kubectl get pods

Please wait until you see the STATUS is running, then you can move forward

```
deployment.apps/mongodb-deployment created
fghebrea408@cloudshell:~ (cs-571-signature-project)$ kubectl get pods
NAME                                 READY   STATUS    RESTARTS   AGE
mongodb-deployment-b7579f455-94zj9   1/1     Running   0          70s
fghebrea408@cloudshell:~ (cs-571-signature-project)$
```

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

vim mongodb-service.yaml


```
apiVersion: v1
kind: Service
metadata:
 name: mongodb-service
spec:
 type: LoadBalancer
 ports:
  # service port in cluster
  - port: 27017
   # port to contact inside container
   targetPort: 27017
 selector:
  app: mongodb
```

kubectl apply -f mongodb-service.yaml

```
fghebrea408@cloudshell:~ (cs-571-signature-project)$ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
fghebrea408@cloudshell:~ (cs-571-signature-project)$
```

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

kubectl get svc

```
fghebrea408@cloudshell:~ (cs-571-signature-project)$ kubectl get svc
NAME              TYPE           CLUSTER-IP       EXTERNAL-IP     PORT(S)           AGE
kubernetes        ClusterIP      34.118.224.1     <none>          443/TCP           45m
mongodb-service   LoadBalancer   34.118.225.179   35.233.176.19   27017:31858/TCP   5m16s
fghebrea408@cloudshell:~ (cs-571-signature-project)$
```

Please wait until you see the external-ip is generated for mongodb-service, then you can move forward

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

kubectl exec -it mongodb-deployment-replace-with-your-pod-name – bash

```
fghebrea408@cloudshell:~ (cs-571-signature-project)$ kubectl exec -it mongodb-deployment-b7579f455-94zj9 -- bash
root@mongodb-deployment-b7579f455-94zj9:/#
```

Now you are inside the mongodb deployment pod. Try

mongo External-IP

```
root@fd090b1d71ff:/# mongosh 35.233.176.19
Current Mongosh Log ID: 66ac70e0b91f95d459838725
Connecting to:          mongodb://35.233.176.19:27017/?directConnection=true&appName=mongosh+2.2.15
Using MongoDB:          7.0.12
Using Mongosh:          2.2.15

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
   2024-08-02T05:18:47.919+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnot
es-filesystem
   2024-08-02T05:18:50.526+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
   2024-08-02T05:18:50.529+00:00: vm.max_map_count is too low
------

test>

test> exit
```

You should 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

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

node

```
fghebrea408@cloudshell:~ (cs-571-signature-project)$ node
Welcome to Node.js v20.15.1.
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,

```
..             { 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);
......     });
.. });
ndefined
3

 _id: 605bdad4e16e6507b7674872,
 student_id: 11111,
 student_name: 'Bruce Lee',
 grade: 84
```

3 means three records was inserted, and we tried search for student_id=11111

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

Docker build -t yourdockerhubID/studentserver .

```
fghebrea408@cloudshell:~/mongo-node (cs-571-signature-project)$ docker build -t justfifi/studentserver .
[+] Building 0.4s (9/9) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 153B
 => [internal] load metadata for docker.io/library/node:18
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/4] FROM docker.io/library/node:18@sha256:11b742eda0142d9ea809fad8c506cbcadb2802c7d4b32e044e6b976691df36b1
 => [internal] load build context
 => => transferring context: 38B
 => CACHED [2/4] WORKDIR /usr/src/app
 => CACHED [3/4] COPY studentServer.js .
 => CACHED [4/4] RUN npm install mongodb
 => exporting to image
 => => exporting layers
 => => writing image sha256:c628d0534d158a7586731b2cca550c1f52467012af107240be2566560cd8f93c
 => => naming to docker.io/justfifi/studentserver
fghebrea408@cloudshell:~/mongo-node (cs-571-signature-project)$
```

Make sure there is no error

4. Push the docker image

docker push yourDockerID/studentserver

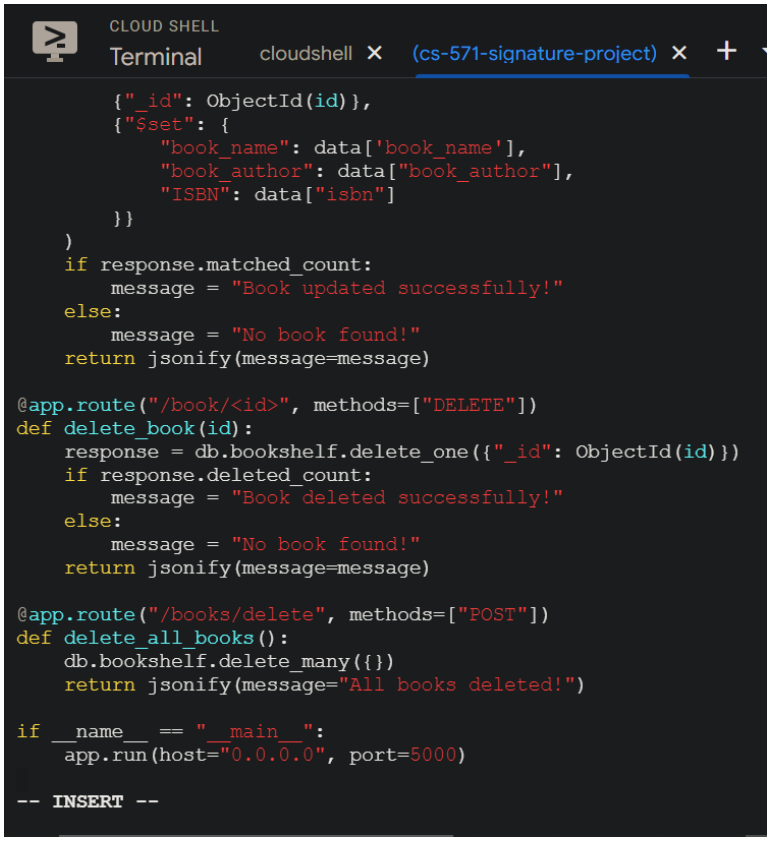
```
fghebrea408@cloudshell:~/mongo-node (cs-571-signature-project)$ docker push justfifi/studentserver
Using default tag: latest
The push refers to repository [docker.io/justfifi/studentserver]
bcdeb40d9677: Pushed
6cbcfc180b70: Pushed
ea75cdcb3ae9: Pushed
9e1b92f36696: Mounted from library/node
b0b3b4d0e4e3: Mounted from library/node
f5bad50165a4: Mounted from library/node
2e06e6e0c554: Mounted from library/node
ffe60aac26fc: Mounted from library/node
0905150af928: Mounted from library/node
7cfafa82cfd2: Mounted from library/node
f6faf32734e0: Mounted from library/node
latest: digest: sha256:0b3e33f08fb2b0ced9a736720ea17c7ecd6fc53a0f869458073f874387006ca1 size: 2629
fghebrea408@cloudshell:~/mongo-node (cs-571-signature-project)$
```

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

1. Create bookshelf.py

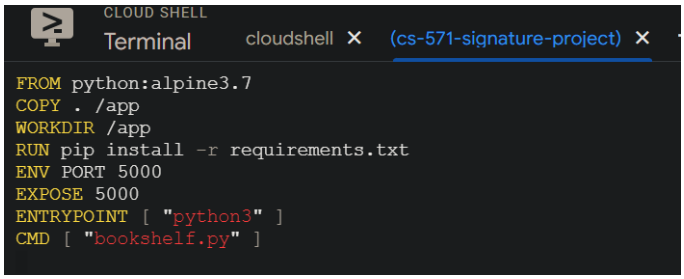
```
from flask import Flask, request, jsonify
from flask_pymongo import PyMongo
from flask import request
from bson.objectid import ObjectId
import socket
import os
app = Flask(__name__)
app.config["MONGO_URI"] =
"mongodb://"+os.getenv("MONGO_URL")+"/"+os.getenv("MONGO_DATABASE")
app.config['JSONIFY_PRETTYPRINT_REGULAR']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



Create requirements.txt if you don't have one.

Add the following to the file

```
Flask==2.0.3
Flask-PyMongo==2.3.0
pymongo==4.4.0
```

3. Build the bookshelf app into a docker image

docker build -t yourDockerID/bookshelf .

```
fghebrea408@cloudshell:~/mongo-node (cs-571-signature-project)$ docker build -t justfifi/bookshelf .
[+] Building 9.1s (9/9) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 212B
 => [internal] load metadata for docker.io/library/python:alpine3.7
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load build context
 => => transferring context: 283B
 => CACHED [1/4] FROM docker.io/library/python:alpine3.7@sha256:35f6f83ab08f98c727dbefd53738e3b3174a48b4571ccb1910bae4
 => [2/4] COPY . /app
 => [3/4] WORKDIR /app
 => [4/4] RUN pip install --no-cache-dir -r requirements.txt
 => exporting to image
 => => exporting layers
 => => writing image sha256:faab28b2a638059b7a250c634a4432a1fca9eb115fe0799b5fdf6a227bda0540
 => => naming to docker.io/justfifi/bookshelf

 1 warning found (use --debug to expand):
 - LegacyKeyValueFormat: "ENV key=value" should be used instead of legacy "ENV key value" format (line 5)
fghebrea408@cloudshell:~/mongo-node (cs-571-signature-project)$
```

Make sure this step build successfully.

4. Push the docker image to your dockerhub

docker push justfifi/bookshelf

```
fghebrea408@cloudshell:~/mongo-node (cs-571-signature-project)$ docker push justfifi/bookshelf
Using default tag: latest
The push refers to repository [docker.io/justfifi/bookshelf]
c1556cfa7841: Pushed
5f70bf18a086: Pushed
37564a94f055: 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: sha256:0eaea3aee9742b3c24b2ff77f56b03d6ea16c227795c97662f64eae7e9b788fa size: 1997
fghebrea408@cloudshell:~/mongo-node (cs-571-signature-project)$
```

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
```

Notice: the reason of creating those two ConfigMap is to avoid re-building docker image again, if the mongoDB pod restarts with a different External-IP

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

minikube start

```
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf (cs-571-signature-project)$ minikube start
* minikube v1.33.1 on Ubuntu 22.04 (amd64)
  - MINIKUBE_FORCE_SYSTEMD=true
  - MINIKUBE_HOME=/google/minikube
  - MINIKUBE_WANTUPDATENOTIFICATION=false
* Automatically selected the docker driver. Other choices: none, ssh
* Using Docker driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.44 ...
* Downloading Kubernetes v1.30.0 preload ...
    > preloaded-images-k8s-v18-v1...:  342.90 MiB / 342.90 MiB  100.00% 204.97
    > gcr.io/k8s-minikube/kicbase...:  481.58 MiB / 481.58 MiB  100.00% 93.97 M
* Creating docker container (CPUs=2, Memory=4000MB) ...
* Preparing Kubernetes v1.30.0 on Docker 26.1.1 ...
  - kubelet.cgroups-per-qos=false
  - kubelet.enforce-node-allocatable=""
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf (cs-571-signature-project)$
```

6. Start Ingress

minikube addons enable ingress

```
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf (cs-571-signature-project)$ minikube addons enable ingress
* ingress is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
  - Using image registry.k8s.io/ingress-nginx/controller:v1.10.1
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.1
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.1
* Verifying ingress addon...
* The 'ingress' addon is enabled
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf (cs-571-signature-project)$
```

7. Create studentserver related pods and start service using the above yaml file

kubectl apply -f studentserver-deployment.yaml

kubectl apply -f studentserver-configmap.yaml

kubectl apply -f studentserver-service.yaml

```
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf (cs-571-signature-project)$ kubectl apply -f studentserver-deployment.yaml
deployment.apps/web created
```

```
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf (cs-571-signature-project)$ kubectl apply -f studentserver-configmap.yaml
configmap/studentserver-config created
```

```
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf (cs-571-signature-project)$ kubectl apply -f studentserver-service.yaml
service/web created
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf (cs-571-signature-project)$
```

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

kubectl apply -f bookshelf-deployment.yaml

kubectl apply -f bookshelf-configmap.yaml

kubectl apply -f bookshelf-service.yaml

```
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf (cs-571-signature-project)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created

fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf (cs-571-signature-project)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created

fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf (cs-571-signature-project)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf (cs-571-signature-project)$
```

9. Check if all the pods are running correctly

kubectl get pods

```
NAME                                    READY   STATUS    RESTARTS   AGE
bookshelf-deployment-646c59bd88-grrtn   1/1     Running   0          64m
web-59b45585db-cnslb                    1/1     Running   0          61m
```

10. 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
```

11. Create the ingress service using the above yaml file

kubectl apply -f studentservermongoIngress.yaml

12. Check if ingress is running

kubectl get ingress

```
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf (cs-571-signature-project)$ kubectl apply -f studentservermongoIngress.yaml
Warning: path /studentserver(/|$)(.*) cannot be used with pathType Prefix
Warning: path /bookshelf(/|$)(.*) cannot be used with pathType Prefix
ingress.networking.k8s.io/server created
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf (cs-571-signature-project)$ kubectl get ingress
NAME     CLASS   HOSTS              ADDRESS   PORTS   AGE
server   nginx   cs571.project.com            80      19s
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf (cs-571-signature-project)$
```

Please wait until you see the Address, then move forward

```
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf (cs-571-signature-project)$ kubectl get ingress
NAME     CLASS   HOSTS              ADDRESS         PORTS   AGE
server   nginx   cs571.project.com   192.168.49.2   80      83s
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf (cs-571-signature-project)$
```

13. Add Address to /etc/hosts

vi /etc/hosts

Add the address you got from above step to the end of the file

Your-address cs571.project.com

Your /etc/hosts file should look something like this after adding the line, but your address should be different from mine

```
# 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
```

14. If everything goes smoothly, you should be able to access your applications

curl cs571.project.com/studentserver/api/score?student_id=11111

```
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf $ curl cs571.project.com/studentserver/api/score?student_id=11111
{"_id":"605a6b49c3a15527de9d0f9b","student_id":11111,"student_name":"Bruce Lee","grade":84}
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf $ curl cs571.project.com/studentserver/api/score?student_id=22222
{"_id":"605a6b49c3a15527de9d0f9c","student_id":22222,"student_name":"Jackie Chen","grade":93}
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf $ curl cs571.project.com/studentserver/api/score?student_id=33333
{"_id":"605a6b49c3a15527de9d0f9d","student_id":33333,"student_name":"Jet Li","grade":88}
```

On another path, you should be able to use the REST API with bookshelf application

I.e list all books

curl cs571.project.com/bookshelf/books

```
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf $ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123",
    "id": "605d1ba7d40f50a395651765"
  }
]
```

Add a book

curl -X POST -d "{\"book_name\": \"cloud computing\",\"book_author\": \"unkown\", \"isbn\": \"123456\" }" http://cs571.project.com/bookshelf/book

```
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf $ curl -X POST -d "{\"book_name\": \"cloud computing\",\"book_author\": \"unkown\", \"isbn\": \"123456\" }" http://cs571.project.com/bookshelf/book
{
  "message": "Task saved successfully!"
}
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf $ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123",
    "id": "605d1ba7d40f50a395651765"
  },
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "605d2fffbd09c0d7f8cf1f93"
  }
]
```

Update a book

curl -X PUT -d "{\"book_name\": \"123\",\"book_author\": \"test\", \"isbn\": \"123updated\" }" http://cs571.project.com/bookshelf/book/id

```
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf $ curl -X PUT -d "{\"book_name\": \"123\",\"book_author\": \"test\", \"isbn\": \"123updated\" }" http://cs571.project.com/bookshelf/book/605d1ba7d40f50a395651765
{
  "message": "Task updated successfully!"
}
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf $ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123updated",
    "id": "605d1ba7d40f50a395651765"
  },
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "605d2fffbd09c0d7f8cf1f93"
  }
]
```

Delete a book

curl -X DELETE cs571.project.com/bookshelf/book/id

```
fghebrea408@cloudshell:~/mongo-node/mongodb/bookshelf $ curl -X DELETE cs571.project.com/bookshelf/book/605d1ba7d40f50a395651765
{
  "message": "Task deleted successfully!"
}
```

```
fqhebrea408@cloudshell:~/mongo-node/mongodb/bookshelf$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "605d2fffbd09c0d7f8cf1f93"
  }
]
```