Week 12: Homework: Chapter 7: Configmap: Signature Project: MongoDB + Python Flask Web Framework + REST API + GKE

Name: Arsiema Yohannes

ID: 20039

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

1. Create cluster.

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

```
aghebrem423@cloudshell:~ (rugged-filament-414319)$ gcloud container clusters create kubia --num-nodes=1 --machine-type=e2-micro --region=us-west1
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
Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most 1008 node(s).
Creating cluster kubia in us-west1... Cluster is being health-checked (master is healthy)...done.
Created [https://container.googleapis.com/v1/projects/rugged-filament-414319/zones/us-west1/clusters/kubia].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/us-west1/kubia?project=rugged-filament-414319
kubeconfig entry generated for kubia.
NAME: kubia
LOCATION: us-west1
MASTER_VERSION: 1.27.8-gke.1067004
MASTER_IP: 34.105.105.211
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.27.8-gke.1067004
NUM_NODES: 3
STATUS: RUNNING
aghebrem423@cloudshell:~ (rugged-filament-414319)$
```

2. Create disk

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

```
aghebrem423@cloudshell:~ (rugged-filament-414319)$ gcloud compute disks create --size=10GiB --zone=us-west1-c mongodb
WARNING: You have selected a disk size of under [200GB]. This may result in poor I/O performance. For more information, see: https://developers.google.com/compute/docs/disks#performance
.
Created [https://www.googleapis.com/compute/v1/projects/rugged-filament-414319/zones/us-west1-c/disks/mongodb].
NAME: mongodb
ZONE: us-west1-c
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
```

3. Now create a mongodb deployment with this yaml file

   vim mongodb-deployment.yaml

```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:
      - name: mongo
        image: 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

```
aghebrem423@cloudshell:~ (rugged-filament-414319)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
aghebrem423@cloudshell:~ (rugged-filament-414319)$
```

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

kubectl get pods

```
aghebrem423@cloudshell:~ (rugged-filament-414319)$ kubectl get pods
NAME                                 READY   STATUS              RESTARTS   AGE
mongodb-deployment-594c77dcdf-gmx7g   0/1     ContainerCreating   0          9m5s
aghebrem423@cloudshell:~ (rugged-filament-414319)$ kubectl get pods
NAME                                 READY   STATUS    RESTARTS   AGE
mongodb-deployment-594c77dcdf-gmx7g   1/1     Running   0          12m
aghebrem423@cloudshell:~ (rugged-filament-414319)$
```

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

vim mongodb-service.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  type: LoadBalancer
  ports:
  - port: 27017
    targetPort: 27017
  selector:
    app: mongodb
```

kubectl apply -f mongodb-service.yaml

```
aghebrem423@cloudshell:~ (rugged-filament-414319)$ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
aghebrem423@cloudshell:~ (rugged-filament-414319)$
```

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

kubectl get svc

```
aghebrem423@cloudshell:~ (rugged-filament-414319)$ kubectl get svc
NAME              TYPE           CLUSTER-IP     EXTERNAL-IP      PORT(S)           AGE
kubernetes        ClusterIP      10.2.128.1     <none>           443/TCP           65m
mongodb-service   LoadBalancer   10.2.139.212   34.168.223.189   27017:31787/TCP   69s
aghebrem423@cloudshell:~ (rugged-filament-414319)$
```

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

```
aghebrem423@cloudshell:~ (rugged-filament-414319)$ kubectl exec -it mongodb-deployment-594c77dcdf-gmx7g -- bash
root@mongodb-deployment-594c77dcdf-gmx7g:/#
```

Try mongo External-IP

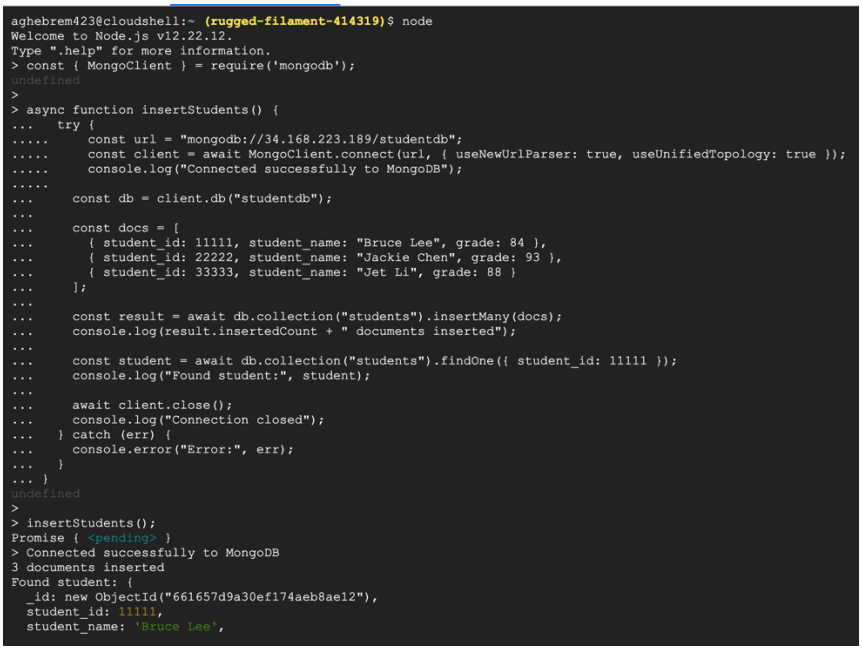
```
aghebrem423@cloudshell:~ (rugged-filament-414319)$ mongo 34.168.223.189
MongoDB shell version v4.4.29
connecting to: mongodb://34.168.223.189:27017/test?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("fb7ffcdb-04ea-493f-8b24-e51771069bb3") }
MongoDB server version: 7.0.8
WARNING: shell and server versions do not match
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:
        2024-04-10T07:52:14.632+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
        2024-04-10T07:52:16.900+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
        2024-04-10T07:52:16.903+00:00: vm.max_map_count is too low
        2024-04-10T07:52:16.903+00:00:          currentValue: 65530
        2024-04-10T07:52:16.903+00:00:          recommendedMinimum: 1677720
        2024-04-10T07:52:16.903+00:00:          maxConns: 838860
---
>
> ^C
bye
```

```
aghebrem423@cloudshell:~ (rugged-filament-414319)$ kubectl exec -it mongodb-deployment-594c77dcdf-c48ch -- bash
root@mongodb-deployment-594c77dcdf-c48ch:/# mongo --version
bash: mongo: command not found
root@mongodb-deployment-594c77dcdf-c48ch:/# mongo 34.168.223.189
bash: mongo: command not found
root@mongodb-deployment-594c77dcdf-c48ch:/# mongosh 34.168.223.189
Current Mongosh Log ID: 661655323453cd3d7a7b2da8
Connecting to:          mongodb://34.168.223.189:27017/?directConnection=true&appName=mongosh+2.2.2
Using MongoDB:          7.0.8
Using Mongosh:          2.2.2

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
   2024-04-10T07:52:14.632+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
   2024-04-10T07:52:16.900+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
   2024-04-10T07:52:16.903+00:00: vm.max_map_count is too low
------

test>
```

```
aghebrem423@cloudshell:~ (rugged-filament-414319)$ sudo npm install -g mongosh
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'mongosh@2.2.3',
npm WARN EBADENGINE   required: { node: '>=16.15.0' },
npm WARN EBADENGINE   current: { node: 'v12.22.12', npm: '7.5.2' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: '@mongosh/cli-repl@2.2.3',
npm WARN EBADENGINE   required: { node: '>=16.15.0' },
npm WARN EBADENGINE   current: { node: 'v12.22.12', npm: '7.5.2' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: '@mongosh/arg-parser@2.2.3',
npm WARN EBADENGINE   required: { node: '>=14.15.1' },
```

```
root@mongodb-deployment-594c77dcdf-c48ch:/# exit
exit
aghebrem423@cloudshell:~ (rugged-filament-414319)$
```

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




```
const { MongoClient } = require('mongodb');

async function insertStudents() {
  try {
    const url = "mongodb://35.230.76.15/studentdb";
    const client = await MongoClient.connect(url, { useNewUrlParser: true, useUnifiedTopology: true });
    console.log("Connected successfully to MongoDB");
    const 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 }
    ];
    const result = await db.collection("students").insertMany(docs);
    console.log(result.insertedCount + " documents inserted");
    const student = await db.collection("students").findOne({ student_id: 11111 });
```

```
    console.log("Found student:", student);
    await client.close();
    console.log("Connection closed");
  } catch (err) {
    console.error("Error:", err);
  }
}
insertStudents();
```

B. Modify our studentServer to get records from MongoDB and deploy to GKE

1. Create studentServer JavaScript File:

vim studentServer.js

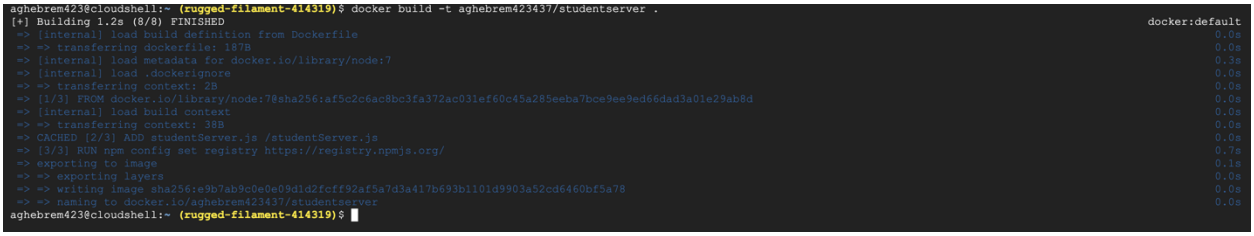
```
var http = require('http');
var url = require('url');
var mongodb = require('mongodb');

const { MONGO_URL, MONGO_DATABASE } = process.env;

var MongoClient = mongodb.MongoClient;
var uri = `mongodb://${MONGO_URL}/${MONGO_DATABASE}`;

var server = http.createServer(function (req, res) {
    var parsedUrl = url.parse(req.url, true);
    var student_id = parseInt(parsedUrl.query.student_id);

    if (/^\/api\/score/.test(req.url)) {
        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 }, function(err, student) {
                if (err)
                    throw new Error(err.message);

                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");
                }
                client.close();
            });
        });
    } else {
        res.writeHead(404);
        res.end("Wrong URL, please try again\n");
    }
});

server.listen(8080);
```

2. Create Dockerfile

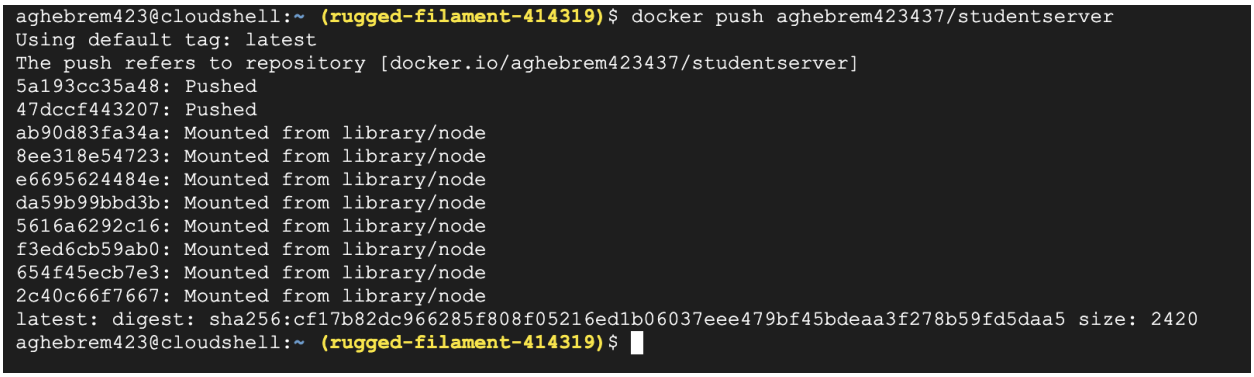
FROM node:7

ADD studentServer.js /studentServer.js
ENTRYPOINT ["node", "studentServer.js"]
RUN npm config set registry https://registry.npmjs.org/

3. Build the studentserver docker image
   docker build -t yourdockerhubID/studentserver



4. Push the docker image
   docker push aghebrem423437/studentserver



C. Create a python Flask bookshelf REST API and deploy on GKE
   1. Create bookshelf.py:
      vim bookshelf.py

```
from flask import Flask, request, jsonify

from flask_pymongo import PyMongo

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)
    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 Dockerfile

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

requirements.txt

```
Flask==2.0.1
flask-pymongo==2.3.0
```

3. Build the bookshelf app into a docker image
   docker build -t yourdockerhubID/studentserver .

4. Push the docker image to your dockerhub
   docker push yourdockerhubID/bookshelf

```
aghebrem423@cloudshell:~ (rugged-filament-414319)$ docker push aghebrem423437/bookshelf
Using default tag: latest
The push refers to repository [docker.io/aghebrem423437/bookshelf]
e525af63780e: Pushed
b91c7b20d680: Pushed
5f70bf18a086: Pushed
de7df7e17f67: 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:c6e0a24c8178847c942812cf18966c0e9ef316c4d115f3b892bf78171c28873b size: 2208
aghebrem423@cloudshell:~ (rugged-filament-414319)$
```

D. 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:
    MONGO_URL: Change-this-to-your-mongoDB-EXTERNAL-IP
    MONGO_DATABASE: mydb

E. 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: aghebrem423437/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: aghebrem423437/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 studentserver-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: web
spec:
  type: LoadBalancer
  ports:
  - port: 8080
    targetPort: 8080
  selector:
    app: web
```

4. Create bookshelf-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: bookshelf-service
spec:
  type: LoadBalancer
```

ports:

- port: 5000

  targetPort: 5000

selector:

  app: bookshelf-deployment

5. Start minikube

minikube start

```
aghebrem423@cloudshell:~ (rugged-filament-414319)$ minikube start
* minikube v1.32.0 on Debian 11.9 (amd64)
  - MINIKUBE_FORCE_SYSTEMD=true
  - MINIKUBE_HOME=/google/minikube
  - MINIKUBE_WANTUPDATENOTIFICATION=false
* Using the docker driver based on existing profile
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Updating the running docker "minikube" container ...

X Docker is nearly out of disk space, which may cause deployments to fail! (96% of capacity). You can pass '--force' to skip this check.
* Suggestion:

    Try one or more of the following to free up space on the device:

    1. Run "docker system prune" to remove unused Docker data (optionally with "-a")
    2. Increase the storage allocated to Docker for Desktop by clicking on:
    Docker icon > Preferences > Resources > Disk Image Size
    3. Run "minikube ssh -- docker system prune" if using the Docker container runtime
* Related issue: https://github.com/kubernetes/minikube/issues/9024

* Preparing Kubernetes v1.28.3 on Docker 24.0.7 ...
  - kubelet.cgroups-per-qos=false
  - kubelet.enforce-node-allocatable=""
* 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
```

6. Start Ingress
   minikube addons enable ingress

```
aghebrem423@cloudshell:~ (rugged-filament-414319)$ 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:v1.9.4
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20231011-8b53cabe0
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20231011-8b53cabe0
* Verifying ingress addon...
* The 'ingress' addon is enabled
aghebrem423@cloudshell:~ (rugged-filament-414319)$
```

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

```
aghebrem423@cloudshell:~ (rugged-filament-414319)$ kubectl apply -f studentserver-deployment.yaml
deployment.apps/web created
aghebrem423@cloudshell:~ (rugged-filament-414319)$ kubectl apply -f studentserver-configmap.yaml
configmap/studentserver-config created
aghebrem423@cloudshell:~ (rugged-filament-414319)$ kubectl apply -f studentserver-service.yaml
service/web created
aghebrem423@cloudshell:~ (rugged-filament-414319)$
```

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

```
aghebrem423@cloudshell:~ (rugged-filament-414319)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created
aghebrem423@cloudshell:~ (rugged-filament-414319)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
aghebrem423@cloudshell:~ (rugged-filament-414319)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
aghebrem423@cloudshell:~ (rugged-filament-414319)$
```

9. Check if all the pods are running correctly
   kubectl get pods

```
aghebrem423@cloudshell:~ (rugged-filament-414319)$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS        AGE
bookshelf-deployment-6666768654-ss8g9   1/1     Running   0               20m
web-854644fb44-6kssc                  1/1     Running   6 (3m45s ago)   6m45s
aghebrem423@cloudshell:~ (rugged-filament-414319)$
```

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

```
aghebrem423@cloudshell:~ (rugged-filament-414319)$ 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
aghebrem423@cloudshell:~ (rugged-filament-414319)$
```

12. Check if ingress is running
    kubectl get ingress

```
aghebrem423@cloudshell:~ (rugged-filament-414319)$ kubectl get ingress
NAME     CLASS   HOSTS               ADDRESS        PORTS   AGE
server   nginx   cs571.project.com   192.168.49.2   80      47s
aghebrem423@cloudshell:~ (rugged-filament-414319)$
```

13. Add Addreee to /etc/hosts
    vim /etc/hosts
    Add the address you got from above step to the end of the file
    Your-address cs571.project.com

```
  GNU nano 5.4
# /etc/hosts: Local Host Database
#
# This file describes a number of aliases-to-address mappings for the for
# local hosts that share this file.
#
# In the presence of the domain name service or NIS, this file may not be
# consulted at all; see /etc/host.conf for the resolution order.
#

# IPv4 and IPv6 localhost aliases
127.0.0.1       localhost
::1             localhost

#
# Imaginary network.
#10.0.0.2               myname
#10.0.0.3               myfriend
#
# According to RFC 1918, you can use the following IP networks for private
# nets which will never be connected to the Internet:
#
#       10.0.0.0        -   10.255.255.255
#       172.16.0.0      -   172.31.255.255
#       192.168.0.0     -   192.168.255.255
#
# In case you want to be able to connect directly to the Internet (i.e. not
# behind a NAT, ADSL router, etc...), you need real official assigned
# numbers.  Do not try to invent your own network numbers but instead get one
# from your network provider (if any) or from your regional registry (ARIN,
# APNIC, LACNIC, RIPE NCC, or AfriNIC.)
#
169.254.169.254 metadata.google.internal metadata

10.88.0.4 cs-21299623887-default
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

```
aghebrem423@cloudshell:~ (rugged-filament-414319)$ curl cs571.project.com/studentserver/api/score?student_id=11111
{"_id":"661729855450d30db3f0910c","student_id":11111,"student_name":"Bruce Lee","grade":84}
aghebrem423@cloudshell:~ (rugged-filament-414319)$
```

15. On another path, you should be able to use the REST API with bookshelf application
    curl cs571.project.com/bookshelf/books

```
aghebrem423@cloudshell:~ (rugged-filament-414319)$ curl http://cs571.project.com/bookshelf
{
  "message": "Welcome to bookshelf app! I am running inside bookshelf-deployment-6666768654-ss8g9 pod!"
}
```

```
aghebrem423@cloudshell:~ (rugged-filament-414319)$ 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

Update a book

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

Delete a book

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