19953 Zhiyu Zhang

CS571: Week 12: Homework: Chapter 7: Configmap: Signature Project: MongoDB + Python Flask Web Framework + REST API + GKE

This document demonstrates a project on GKE with the following steps:

1. Create MongoDB using Persistent Volume and insert records
2. Create and deploy a student server to get records from MongoDB
3. Create and deploy a python Flask bookshelf REST API
4. Create ConfigMap for both applications to store MongoDB URL and name
5. Expose both applications using ingress with traefik
6. Access our applications

Note:

1. In this documentation, if any `code segment` is highlighted in red, please replace it with your own information.

2. This project can be resource demanding. If you want to run everything on GKE, I recommend set the machine type to standard instead of micro when creating the cluster.

1. Create MongoDB using Persistent Volume and insert records

1-1 In GCP terminal, create a Kubernetes cluster with three nodes

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

```
zzhang42305@cloudshell:~ (cs571-a)$ 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/cs571-a/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=cs571-a
kubeconfig entry generated for kubia.
NAME: kubia
LOCATION: us-west1
MASTER_VERSION: 1.27.8-gke.1067004
MASTER_IP: 35.247.18.215
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.27.8-gke.1067004
NUM_NODES: 3
STATUS: RUNNING
```

1-2 Make sure the cluster and all nodes are running

```
$ kubectl get nodes
```

```
zzhang42305@cloudshell:~ (cs571-a)$ kubectl get nodes
NAME                                   STATUS   ROLES    AGE    VERSION
gke-kubia-default-pool-09a6b457-rnbp   Ready    <none>   5m6s   v1.27.8-gke.1067004
gke-kubia-default-pool-3afc3039-t7nk   Ready    <none>   5m6s   v1.27.8-gke.1067004
gke-kubia-default-pool-b57eaa1a-0kv6   Ready    <none>   5m6s   v1.27.8-gke.1067004
```

1-3 Create a PersistentVolume using YAML file

```
$ vim mongodb-pv.yaml
```

with the following code:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mongodb-pv
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: ""
  gcePersistentDisk:
    pdName: mongodb
    fsType: ext4
```

Then apply the YAML file to create the persistent volume:

```
$ kubectl apply -f mongodb-pv.yaml
```

```
zzhang42305@cloudshell:~ (cs571-a)$ vim mongodb-pv.yaml
zzhang42305@cloudshell:~ (cs571-a)$ kubectl apply -f mongodb-pv.yaml
persistentvolume/mongodb-pv created
```

1-4 Create a PersistentVolumeClaim using YAML file

```
$ vim mongodb-pvc.yaml
```

with the following code:

```
apiVersion: v1
kind: PersistentVolumeClaim
```

```
metadata:
  name: mongodb-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: ""
```

Then apply the YAML file to create the persistent volume claim:

```
$ kubectl apply -f mongodb-pvc.yaml
```

```
zzhang42305@cloudshell:~ (cs571-a)$ vim mongodb-pvc.yaml
zzhang42305@cloudshell:~ (cs571-a)$ kubectl apply -f mongodb-pvc.yaml
persistentvolumeclaim/mongodb-pvc created
```

1-5 Create a mongodb deployment using YAML file

```
$ vim mongodb-deployment.yaml
```

with the following code:

```
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
        persistentVolumeClaim:
          claimName: mongodb-pvc
```

Then apply the YAML file to create the mongodb deployment:

```
$ kubectl apply -f mongodb-deployment.yaml
```

```
zzhang42305@cloudshell:~ (cs571-a)$ vim mongodb-deployment.yaml
zzhang42305@cloudshell:~ (cs571-a)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
```

1-6 Check if the deployment pod is running correctly

```
$ kubectl get pods
```

```
zzhang42305@cloudshell:~ (cs571-a)$ kubectl get pods
NAME                                 READY   STATUS    RESTARTS   AGE
mongodb-deployment-59d64c4895-ndpf9   1/1     Running   0          11s
```

1-7 Create a service for the mongodb using YAML file

```
$ vim mongodb-service.yaml
```

with the following code:

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

Then apply the YAML file to create the service so we can access it from outside:

```
$ kubectl apply -f mongodb-service.yaml
```

```
zzhang42305@cloudshell:~ (cs571-a)$ vim mongodb-service.yaml
zzhang42305@cloudshell:~ (cs571-a)$ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
```

1-8 Check if the service is running correctly

```
$ kubectl get svc
```

```
zzhang42305@cloudshell:~ (cs571-a)$ kubectl get svc
NAME              TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)           AGE
kubernetes        ClusterIP      10.81.96.1      <none>          443/TCP           29m
mongodb-service   LoadBalancer   10.81.102.215   34.168.247.182  27017:31237/TCP   49s
```

Please wait till you see the EXTERNAL-IP is generated for mongodb-service to proceed.

1-9 Connect to mongodb using the EXTERNAL-IP

```
$ kubectl exec -it your-mongodb-pod-name -- bash
```

```
zzhang42305@cloudshell:~ (cs571-a)$ kubectl exec -it mongodb-deployment-59d64c4895-ndpf9 -- bash
root@mongodb-deployment-59d64c4895-ndpf9:/#
```

Now you are inside the mongodb deployment pod.

```
# mongosh EXTERNAL-IP
```

You should see something like this, which means your mongodb is up and can be accessed with the EXTERNAL-IP:

```
root@mongodb-deployment-59d64c4895-ndpf9:/# mongosh 34.168.247.182
Current Mongosh Log ID: 6613a3c1f51e085e717b2da8
Connecting to:          mongodb://34.168.247.182:27017/?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
   2024-04-08T07:48:58.829+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger
storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
   2024-04-08T07:49:00.447+00:00: Access control is not enabled for the database. Read and write access
 to data and configuration is unrestricted
   2024-04-08T07:49:00.454+00:00: vm.max_map_count is too low
------

test>
```

1-10 Insert student records into mongodb

Go to your database inside mongodb:

```
> use mydb
```

Insert 3 students records:

```
> db.students.insertMany([
  { 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 }
])
```

```
test> use mydb
switched to db mydb
mydb> db.students.find({})

mydb> db.students.insertMany([
...   { 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 }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6614801a3a0bb53ab07b2da9'),
    '1': ObjectId('6614801a3a0bb53ab07b2daa'),
    '2': ObjectId('6614801a3a0bb53ab07b2dab')
  }
}
```

1-11 Exit mongodb and go back to GCP console

```
> exit
# exit
```

```
mydb> exit
root@mongodb-deployment-59d64c4895-ndpf9:/# exit
```

2. Create and deploy a student server to get records from MongoDB

2-1 Create a studentServer

```
$ vim studentServer.js
```

with the following code:

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

// Update this URI to your MongoDB URI
const uri = "mongodb://" + os.getenv("MONGO_URL") + "/" +
os.getenv("MONGO_DATABASE");
const client = new MongoClient(uri);

async function connectToMongoDB() {
  try {
    await client.connect();
    console.log('Connected successfully to MongoDB');
  } catch (error) {
    console.error('Failed to connect to MongoDB:', error);
    process.exit(1); // Exit if cannot connect
  }
}
```

```
async function handleRequest(req, res) {
  try {
    const parsedUrl = url.parse(req.url, true);
    const db = client.db();

    // Example: Handle a simple get request
    if (parsedUrl.pathname === '/api/score') {
      const studentId = parseInt(parsedUrl.query.student_id, 10);
      const student = await db.collection("students").findOne({ student_id:
studentId });

      if (student) {
        delete student._id;
        res.writeHead(200, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify(student) + '\n');
      } else {
        res.writeHead(404, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify({ error: 'Student not found' }) + '\n');
      }
    } else {
      res.writeHead(404, { 'Content-Type': 'application/json' });
      res.end(JSON.stringify({ error: 'Invalid path' }) + '\n');
    }
  } catch (error) {
    console.error('Database operation failed:', error);
    res.writeHead(500, { 'Content-Type': 'application/json' });
    res.end(JSON.stringify({ error: 'Internal server error' }) + '\n');
  }
}

async function startServer() {
  await connectToMongoDB(); // Ensure MongoDB is connected before starting
the server

  const server = http.createServer(handleRequest);
  server.listen(8080, () => console.log('Server is listening on port
8080'));
}

startServer().catch(console.error);
```

2-2 Create Dockerfile

```
$ vim Dockerfile
```

with the following code:

```
# Use a Node.js base image
FROM node:latest
# Copy your application code and package.json to the container
COPY . /app
WORKDIR /app
# Install dependencies
RUN npm install
# Your application's default command
CMD ["node", "studentServer.js"]
```

2-3 Build the studentserver docker image

```
$ docker build -t studentserver .
```

```
zzhang42305@cloudshell:~ (cs571-a)$ docker build -t studentserver .
[+] Building 0.2s (7/7) FINISHED                                                docker:default
 => [internal] load build definition from Dockerfile                                     0.0s
 => => transferring dockerfile: 128B                                                     0.0s
 => [internal] load metadata for docker.io/library/node:7                                0.1s
 => [internal] load .dockerignore                                                        0.0s
 => => transferring context: 2B                                                          0.0s
 => [internal] load build context                                                        0.0s
 => => transferring context: 38B                                                         0.0s
 => [1/2] FROM docker.io/library/node:7@sha256:af5c2c6ac8bc3fa372ac031ef60c45a285eeba7bce9ee9ed6  0.0s
 => CACHED [2/2] ADD studentServer.js /studentServer.js                                  0.0s
 => exporting to image                                                                   0.0s
 => => exporting layers                                                                  0.0s
 => => writing image sha256:986048d48812f7a935b24589d3b3a43a8c59f7a1f552ec2a41986cb4613fbfaf  0.0s
 => => naming to docker.io/library/studentserver                                         0.0s
```

2-4 Tag the docker image and push it to your dockerhub

```
$ docker tag studentserver username/studentserver:latest
$ docker push username/studentserver:latest
```

```
zzhang42305@cloudshell:~ (cs571-a)$ docker tag studentserver zoeyzhiyu/studentserver:latest
zzhang42305@cloudshell:~ (cs571-a)$ docker push zoeyzhiyu/studentserver:latest
The push refers to repository [docker.io/zoeyzhiyu/studentserver]
6088c34e7df5: 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: sha256:014b96d216bcb515d9d2c53ef7418a93ad127dff59fc67395dd07e94ef37cc2c size: 2213
```

3. Create and deploy a python Flask bookshelf REST API

3-1 Create bookshelf.py

```
$ vim bookshelf.py
```

with the following code:

```
from flask import Flask, jsonify, request
from flask_pymongo import PyMongo
from bson.objectid import ObjectId
import socket
import os

app = Flask(__name__)

# Corrected the MONGO_URI line to be on a single line
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 the bookshelf app! I am running inside {}
pod!".format(hostname)
    )
```

```
@app.route("/books")
def get_all_books():
    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="Book added successfully!")

@app.route("/book/<id>", methods=["PUT"])
def update_book(id):
    data = request.get_json(force=True)
    response = db.bookshelf.update_one({"_id": ObjectId(id)}, {"$set":
data})
    if response.matched_count:
        message = "Book updated successfully!"
    else:
        message = "No book found!"
    return jsonify(message=message)

@app.route("/book/<id>", methods=["DELETE"])
def delete_book(id):
    response = db.bookshelf.delete_one({"_id": ObjectId(id)})
    if response.deleted_count:
        message = "Book deleted successfully!"
    else:
        message = "No book found!"
    return jsonify(message=message)

@app.route("/books/delete", methods=["POST"])
def delete_all_books():
    db.bookshelf.delete_many({})
    return jsonify(message="All books deleted!")

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=int(os.getenv('PORT', 5000)))
```

3-2 Create a Dockerfile

```
$ vim Dockerfile
```

with the following code:

```
FROM python:3.7-slim
COPY . /app
WORKDIR /app
```

```
RUN pip install --upgrade pip && pip install -r requirements.txt
ENV PORT 5000
EXPOSE 5000
ENTRYPOINT ["python3"]
CMD ["bookshelf.py"]
```

3-3 Build the bookshelf app docker image

```
$ docker build -t bookshelf .
```

```
zzhang42305@cloudshell:~ (cs571-a)$ docker build -t bookshelf .
[+] Building 42.6s (9/9) FINISHED                                                                                docker:default
 => [internal] load build definition from Dockerfile                                                                        0.0s
 => => transferring dockerfile: 238B                                                                                        0.0s
 => [internal] load metadata for docker.io/library/python:3.7-slim                                                          0.6s
 => [internal] load .dockerignore                                                                                           0.0s
 => => transferring context: 2B                                                                                             0.0s
 => [internal] load build context                                                                                           0.1s
 => => transferring context: 241.97kB                                                                                       0.1s
 => [1/4] FROM docker.io/library/python:3.7-slim@sha256:b53f496ca43e5af6994f8e316cf03af31050bf79                            3.4s
 => => resolve docker.io/library/python:3.7-slim@sha256:b53f496ca43e5af6994f8e316cf03af31050bf79                            0.0s
 => => sha256:a803e7c4b030119420574a882a52b6431e160fceb7620f61b525d49bc2d58886 29.12MB / 29.12MB                           0.5s
 => => sha256:bf3336e84c8e00632cdea35b18fec9a5691711bdc8ac885e3ef54a3d5ff500ba 3.50MB / 3.50MB                             0.5s
 => => sha256:8973eb85275f19b8d72c6047560629116ad902397e5c1885b2508788197de28b 11.38MB / 11.38MB                           0.3s
 => => sha256:b53f496ca43e5af6994f8e316cf03af31050bf7944e0e4a308ad86c001cf028b 1.86kB / 1.86kB                              0.0s
 => => sha256:ffd28e36ef37b3a4a24f6a771a48d7c5499ea42d6309ac911a3f699e122060be 1.37kB / 1.37kB                              0.0s
 => => sha256:a255ffcb469f2ec40f2958a76beb0c2bbebfe92ce9af67a9b48d84b4cb695ac8 7.54kB / 7.54kB                              0.0s
 => => sha256:f9afc3cc0135aad884dad502f28a5b3d8cd32565116131da818ebf2ea6d46095 244B / 244B                                  0.5s
 => => extracting sha256:a803e7c4b030119420574a882a52b6431e160fceb7620f61b525d49bc2d58886                                   1.6s
 => => sha256:39312d8b4ab77de264678427265a2668073675bb8666caf723da18c9e4b7e3fc 3.13MB / 3.13MB                              0.7s
 => => extracting sha256:bf3336e84c8e00632cdea35b18fec9a5691711bdc8ac885e3ef54a3d5ff500ba                                   0.2s
 => => extracting sha256:8973eb85275f19b8d72c6047560629116ad902397e5c1885b2508788197de28b                                   0.5s
 => => extracting sha256:f9afc3cc0135aad884dad502f28a5b3d8cd32565116131da818ebf2ea6d46095                                   0.0s
 => => extracting sha256:39312d8b4ab77de264678427265a2668073675bb8666caf723da18c9e4b7e3fc                                   0.3s
 => [2/4] COPY . /app                                                                                                       1.2s
 => [3/4] WORKDIR /app                                                                                                      0.0s
 => [4/4] RUN pip install --upgrade pip setuptools wheel && pip install -r requirements.txt                               33.5s
 => exporting to image                                                                                                      3.8s
 => => exporting layers                                                                                                     3.8s
 => => writing image sha256:7a3c8e846d024f94ee74b1a377e86185f4db7699b7164aaed330296d3d1de003                                0.0s
 => => naming to docker.io/library/bookshelf                                                                                0.0s
```

3-4 Tag the docker image and push it to your dockerhub

```
$ docker tag bookshelf username/bookshelf:latest
$ docker push username/bookshelf:latest
```

```
zzhang42305@cloudshell:~ (cs571-a)$ docker tag bookshelf zoeyzhiyu/bookshelf:latest
zzhang42305@cloudshell:~ (cs571-a)$ docker push zoeyzhiyu/bookshelf:latest
The push refers to repository [docker.io/zoeyzhiyu/bookshelf]
c2e4e7912edb: Pushed
5f70bf18a086: Pushed
8eeaab143a79: Pushed
b8594deafbe5: Mounted from library/python
8a55150afecc: Mounted from library/python
ad34ffec41dd: Mounted from library/python
f19cb1e4112d: Mounted from library/python
d310e774110a: Mounted from library/python
latest: digest: sha256:9a0f96640460bb08ca3aac68a7763685abee362b87a9a2274be551ed42ec1c0d size: 2000
```

4. Create ConfigMap for both applications to store MongoDB URL and name

4-1 Create studentserver-configmap.yaml

```
$ vim studentserver-configmap.yaml
```

with the following code:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  MONGO_URL: your.mongodb.EXTERNAL-IP
  MONGO_DATABASE: mydb
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  MONGO_URL: 34.168.247.182
  MONGO_DATABASE: mydb
```

4-2 Create bookshelf-configmap.yaml

```
$ vim bookshelf-configmap.yaml
```

with the following code:

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

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: bookshelf-config
data:
  # SERVICE_NAME.NAMESPACE.svc.cluster.local:SERVICE_PORT
  MONGO_URL: 34.168.247.182
  MONGO_DATABASE: mydb
```

The reason of creating two ConfigMap is to avoid re-building docker image again if the mongodb pod restarts with a different EXTERNAL-IP.

5. Expose both applications using ingress with traefik

5-1 Create studentserver-deployment.yaml

```
$ vim studentserver-deployment.yaml
```

with the following code:

```
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: username/studentserver:latest
          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
```

5-2 Create bookshelf-deployment.yaml

```
$ vim bookshelf-deployment.yaml
```

with the following code:

```
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: username/bookshelf:latest
          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
```

5-3 Create studentserver-service.yaml

```
$ vim studentserver-service.yaml
```

with the following code:

```
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
```

5-4 Create bookshelf-service.yaml

```
$ vim bookshelf-service.yaml
```

with the following code:

```
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
```

5-5 Create all the studentserver related pods and start service using the above YAML files

```
$ kubectl apply -f studentserver-deployment.yaml
$ kubectl apply -f studentserver-configmap.yaml
$ kubectl apply -f studentserver-service.yaml
```

```
zzhang42305@cloudshell:~ (cs571-a)$ kubectl apply -f studentserver-deployment.yaml
deployment.apps/web created
zzhang42305@cloudshell:~ (cs571-a)$ kubectl apply -f studentserver-configmap.yaml
configmap/studentserver-config created
zzhang42305@cloudshell:~ (cs571-a)$ kubectl apply -f studentserver-service.yaml
service/web created
```

5-6 Create all the bookshelf related pods and start service using the above YAML files

```
$ kubectl apply -f bookshelf-deployment.yaml
$ kubectl apply -f bookshelf-configmap.yaml
$ kubectl apply -f bookshelf-service.yaml
```

```
zzhang42305@cloudshell:~ (cs571-a)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created
zzhang42305@cloudshell:~ (cs571-a)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
zzhang42305@cloudshell:~ (cs571-a)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
```

5-7 Check if all the pods are running correctly

```
$ kubectl get pods
```

```
zzhang42305@cloudshell:~ (cs571-a)$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
bookshelf-deployment-86ccdf6697-2svf2   1/1     Running   0          32s
mongodb-deployment-59d64c4895-ndpf9     1/1     Running   0          144m
web-6994b97dd6-2xkdk                    1/1     Running   0          10m
```

5-8 Check if all the services are running correctly

```
$ kubectl get svc
```

```
zzhang42305@cloudshell:~ (cs571-a)$ kubectl get svc
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)           AGE
bookshelf-service   LoadBalancer   10.81.104.229   34.83.50.53      5000:32533/TCP    15h
kubernetes          ClusterIP      10.81.96.1      <none>           443/TCP           19h
mongodb-service     LoadBalancer   10.81.99.44     34.168.247.182   27017:30172/TCP   18h
web                 LoadBalancer   10.81.101.135   34.145.94.122    8080:30037/TCP    41s
```

Please wait till you see the EXTERNAL-IP is generated for web and bookshelf-service to proceed.

5-9 Install traefik so we can use it to create ingressroute later

```
$ helm repo add traefik https://helm.traefik.io/traefik
$ helm repo update
$ helm install traefik traefik/traefik
```

```
zzhang42305@cloudshell:~ (cs571-a)$ helm repo add traefik https://helm.traefik.io/traefik
"traefik" has been added to your repositories
zzhang42305@cloudshell:~ (cs571-a)$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "ingress-nginx" chart repository
...Successfully got an update from the "traefik" chart repository
Update Complete. ⎈Happy Helming!⎈
zzhang42305@cloudshell:~ (cs571-a)$ helm install traefik traefik/traefik
NAME: traefik
LAST DEPLOYED: Tue Apr  9 02:55:55 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Traefik Proxy v2.11.0 has been deployed successfully on default namespace !
```

5-10 Create an ingress service using YAML file

```
$ vim my-ingress.yaml
```

with the following code:

```
apiVersion: traefik.containo.us/v1alpha1
kind: IngressRoute
metadata:
  name: my-ingressroute
spec:
  entryPoints:
    - web
  routes:
    - match: Host(`your.domain.com`) && PathPrefix(`/studentserver`)
      kind: Rule
      services:
        - name: web
          port: 8080
    - match: Host(`your.domain.com`) && PathPrefix(`/bookshelf`)
      kind: Rule
      services:
        - name: bookshelf-service
          port: 5000
```

Then apply the YAML file to create the ingressroute:

```
$ kubectl apply -f my-ingress.yaml
```

```
zzhang42305@cloudshell:~ (cs571-a)$ vim my-ingress.yaml
zzhang42305@cloudshell:~ (cs571-a)$ kubectl apply -f my-ingress.yaml
ingressroute.traefik.containo.us/my-ingressroute created
```

5-11 Check if ingressroute service is running

```
$ kubectl get svc
```

```
zzhang42305@cloudshell:~ (cs571-a)$ kubectl get svc
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)                      AGE
bookshelf-service   LoadBalancer   10.81.104.229   34.83.50.53      5000:32533/TCP               16h
kubernetes          ClusterIP      10.81.96.1      <none>           443/TCP                      20h
mongodb-service     LoadBalancer   10.81.99.44     34.168.247.182   27017:30172/TCP              19h
traefik             LoadBalancer   10.81.109.255   35.230.89.163    80:30057/TCP,443:30138/TCP   7m33s
web                 LoadBalancer   10.81.101.135   34.145.94.122    8080:30037/TCP               52m
```

Please wait till you see the EXTERNAL-IP is generated for traefik to proceed.

5-12 Add Address to /etc/hosts

```
$ sudo vi /etc/hosts
```

Add the EXTERNAL-IP and you domain name to the end of the file:

```
EXTERNAL-IP your.domain.com
```

```
# 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-302443420602-default
35.230.89.163 cs571project.19953.com
~
```

6. Access our applications

6-1 Access student server

Access existing record:

```
$ curl your.domain.com/studentserver/api/score?student_id=11111
$ curl your.domain.com/studentserver/api/score?student_id=22222
```

Error handling: student not in records:

```
$ curl your.domain.com/studentserver/api/score?student_id=44444
```

Error handling: incorrect path:

```
$ curl your.domain.com/studentserver/apis/score?student_id=44444
```

```
zzhang42305@cloudshell:~ (cs571-a)$ curl http://cs571project.19953.com/studentserver/api/score?student_id=11111
{"student_id":11111,"student_name":"Bruce Lee","grade":84}
zzhang42305@cloudshell:~ (cs571-a)$ curl http://cs571project.19953.com/studentserver/api/score?student_id=22222
{"student_id":22222,"student_name":"Jackie Chen","grade":93}
zzhang42305@cloudshell:~ (cs571-a)$ curl http://cs571project.19953.com/studentserver/api/score?student_id=44444
{"error":"Student not found"}
zzhang42305@cloudshell:~ (cs571-a)$ curl http://cs571project.19953.com/studentserver/apis/score?student_id=44444
{"error":"Invalid path"}
```

6-2 Access the bookshelf application

Add books:

```
$ curl -X POST -d "{\"book_name\": \"star wars\",\"book_author\":
\"unkown\", \"isbn\": \"654321\" }" http://your.domain.com/bookshelf/book

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

```
zzhang42305@cloudshell:~ (cs571-a)$ curl -X POST -d "{\"book_name\": \"star wars\",\"book_author\": \"unkown\", \"isbn\"
: \"654321\" }" http://cs571project.19953.com/bookshelf/book
{
  "message": "Book added successfully!"
}
zzhang42305@cloudshell:~ (cs571-a)$ curl -X POST -d "{\"book_name\": \"cloud computing\",\"book_author\": \"unkown\", \"
isbn\": \"123456\" }" http://cs571project.19953.com/bookshelf/book
{
  "message": "Book added successfully!"
}
```

List all books:

```
$ curl your.domain.com/bookshelf/books
```

```
zzhang42305@cloudshell:~ (cs571-a)$ curl cs571project.19953.com/bookshelf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "star wars",
    "ISBN": "654321",
    "id": "6614b5d12bbb0d18bcabd75d"
  },
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "6614b6dd2bbb0d18bcabd75f"
  }
]
```

Update a book:

```
$ curl -X PUT -d "{\"book_name\": \" cloud computing system\",\"book_author\": \"testing\", \"isbn\": \"123updated\" }" http://your.domain.com/bookshelf/book/id
```

```
zzhang42305@cloudshell:~ (cs571-a)$ curl -X PUT -d "{\"book_name\": \" cloud computing system\",\"book_author\": \"testing\", \"isbn\": \"123updated\" }" http://cs571project.19953.com/bookshelf/book/6614b6dd2bbb0d18bcabd75f
{
  "message": "Book updated successfully!"
}
zzhang42305@cloudshell:~ (cs571-a)$ curl cs571project.19953.com/bookshelf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "star wars",
    "ISBN": "654321",
    "id": "6614b5d12bbb0d18bcabd75d"
  },
  {
    "Book Author": "testing",
    "Book Name": " cloud computing system",
    "ISBN": "123456",
    "id": "6614b6dd2bbb0d18bcabd75f"
  }
]
```

Delete a book:

```
$ curl -X DELETE your.domain.com/bookshelf/book/id
```

```
zzhang42305@cloudshell:~ (cs571-a)$ curl -X DELETE cs571project.19953.com/bookshelf/book/6614b6dd2bbb0d18bcabd75f
{
  "message": "Book deleted successfully!"
}
zzhang42305@cloudshell:~ (cs571-a)$ curl cs571project.19953.com/bookshelf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "star wars",
    "ISBN": "654321",
    "id": "6614b5d12bbb0d18bcabd75d"
  }
]
```