Signature Project: MongoDB + Python Flask Web Framework + REST API + GKE

1. Create a cluster as usual on GKE
   - gcloud container clusters create kubia --num-nodes=3 --machine-type=e2-micro --zone=us-west1-b

Wait for the creation to finish,

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ gcloud container clusters create kubia --num-nodes=3 --machine-type=e2-micro --zone=us-west1-b
Note: The Kubelet readonly port (10255) is now deprecated. Please update your workloads to use the recommended alternatives. See https://cloud.g
-kubelet-readonly-port for ways to check usage and for migration instructions.
Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most 1008 node(s).
Creating cluster kubia in us-west1-b... Cluster is being health-checked (Kubernetes Control Plane is healthy)...done.
Created [https://container.googleapis.com/v1/projects/able-scope-442020-a1/zones/us-west1-b/clusters/kubia].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/us-west1-b/kubia?project=able-scope
kubeconfig entry generated for kubia.
NAME: kubia
LOCATION: us-west1-b
MASTER_VERSION: 1.30.5-gke.1443001
MASTER_IP: 104.196.241.133
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.30.5-gke.1443001
NUM_NODES: 3
STATUS: RUNNING
hpatel65373@cloudshell:~ (able-scope-442020-a1)$
```

2. Now create a Persistent Volume:
   - gcloud compute disks create mongodb --size=10GiB --zone=us-west1-b

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ gcloud compute disks create mongodb --size=10GiB --zone=us-west1-b
WARNING: You have selected a disk size of under [200GB]. This may result in poor I/O performance. For more informati
Created [https://www.googleapis.com/compute/v1/projects/able-scope-442020-a1/zones/us-west1-b/disks/mongodb].
NAME: mongodb
ZONE: us-west1-b
SIZE_GB: 10
TYPE: pd-standard
STATUS: READY
```

Create a Persistent Volume (PV)

First, create a YAML file for the Persistent Volume that references the disk you created in Google Cloud. The PV will be configured to use the mongodb disk.

Create a file named mongodb-pv.yaml:

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

Apply the PV Configuration

Run the following command to apply the Persistent Volume configuration:
kubectl apply -f mongodb-pv.yaml

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ kubectl apply -f mongodb-pv.yaml
persistentvolume/mongodb-pv created
hpatel65373@cloudshell:~ (able-scope-442020-a1)$
```

Create a Persistent Volume Claim (PVC)

Next, you need to create a Persistent Volume Claim (PVC) that will request storage from the PV. Create a YAML file for the PVC, e.g.,mongodb-pvc.yaml:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mongodb-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: ""
~
~
```

Apply the PVC Configuration

Run the following command to apply the Persistent Volume Claim:
kubectl apply -f mongodb-pvc.yaml

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ kubectl apply -f mongodb-pvc.yaml
persistentvolumeclaim/mongodb-pvc created
hpatel65373@cloudshell:~ (able-scope-442020-a1)$
```

Verify the PV and PVC

To check if your PV and PVC were created successfully and bound to each other, you can run:
kubectl get pv

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ kubectl get pv
NAME         CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   VOLUMEATTRIBUTESCLASS   REASON   AGE
mongodb-pv   5Gi        RWO            Retain           Available                          <unset>                          20m
```

kubectl get pvc

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ kubectl get pvc
NAME          STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
mongodb-pvc   Pending                                      standard-rwo   <unset>                 118s
```

3. Now create a mongodb deployment with this yaml file

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb
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
~
~
```

Now apply mongodb-deployment.yaml file
kubectl apply -f mongodb-deployment.yaml

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb created
hpatel65373@cloudshell:~ (able-scope-442020-a1)$
```

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

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ kubectl get pods
NAME                                 READY   STATUS    RESTARTS   AGE
mongodb-deployment-658f677f67-4cx9l   1/1     Running   0          45s
hpatel65373@cloudshell:~ (able-scope-442020-a1)$
```

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

```yaml
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  type: LoadBalancer
  ports:
    - port: 27017 # Default port for MongoDB
      targetPort: 27017 # Port where the MongoDB container is listening
  selector:
    app: mogodb
~
~
```

Now apply mongodb-service.yaml
kubectl apply -f mongodb-service.yaml

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
hpatel65373@cloudshell:~ (able-scope-442020-a1)$
```

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

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ kubectl get svc
NAME              TYPE           CLUSTER-IP       EXTERNAL-IP    PORT(S)           AGE
kubernetes        ClusterIP      34.118.224.1     <none>         443/TCP           9m1s
mongodb-service   LoadBalancer   34.118.228.251   34.168.11.57   27017:32583/TCP   51s
hpatel65373@cloudshell:~ (able-scope-442020-a1)$
```

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 get pods -l app=mongodb

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ kubectl get pods -l app=mongodb
NAME                                  READY   STATUS    RESTARTS   AGE
mongodb-deployment-658f677f67-4cx91   1/1     Running   0          3m56s
hpatel65373@cloudshell:~ (able-scope-442020-a1)$
```

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

- kubectl exec -it mongodb-deployment-658f677f67-4cx91 -- bash

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ kubectl exec -it mongodb-deployment-658f677f67-4cx91 -- bash
root@mongodb-deployment-658f677f67-4cx91:/#
```

Now you are inside the mongodb deployment pod
Try,

mongosh External-IP
mongosh 34.168.11.57

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ kubectl exec -it mongodb-deployment-658f677f67-4cx9l -- bash
root@mongodb-deployment-658f677f67-4cx9l:/# mongosh 34.168.11.57
Current Mongosh Log ID: 6747922d6787881439c1c18b
Connecting to:          mongodb://34.168.11.57:27017/?directConnection=true&appName=mongosh+2.3.3
MongoNetworkError: connect ECONNREFUSED 34.168.11.57:27017
root@mongodb-deployment-658f677f67-4cx9l:/#
```

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.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('673a2fed2a47c59549c1c18c'),
    '1': ObjectId('673a2fed2a47c59549c1c18d'),
    '2': ObjectId('673a2fed2a47c59549c1c18e')
  }
}
mydb>
```

Exit mongodb and go back to GCP console
> exit
exit

```
root@mongodb-deployment-658f677f67-4cx9l:/# exit
exit
command terminated with exit code 1
hpatel65373@cloudshell:~ (able-scope-442020-a1)$
```

2. Create and deploy a student server to get records from MongoDB
Create a studentServer
vim studentServer.js with following code,

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

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

// Connect to MongoDB
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
  }
}

// Handle HTTP Requests
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; // Remove MongoDB's internal `_id` field
        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');
  }
}

// Start the Server
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);
```

Create Dockerfile,
vim Dockerfile with following code,

```
# Use a Node.js base image
FROM node:latest

# Copy your application code and package.json to the container
COPY . /app

# Set the working directory
WORKDIR /app

# Install dependencies
RUN npm install

# Your application's default command
CMD ["node", "studentServer.js"]
```

Build the studentserver docker image,
docker build -t studentserver .

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ docker build -t student-server .
[+] Building 38.5s (11/11) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 338B
 => [internal] load metadata for docker.io/library/node:latest
 => [auth] library/node:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load build context
 => => transferring context: 950.84MB
 => [1/5] FROM docker.io/library/node:latest@sha256:a2fea8b0b74b6e828caa6d83f4b2a0dcb2eb1ff90f30205c32f7bd36ddf976c4
 => CACHED [2/5] WORKDIR /app
 => [3/5] COPY package*.json ./
 => [4/5] RUN npm install
 => [5/5] COPY . .
 => exporting to image
 => => exporting layers
 => => writing image sha256:788e40275062f483770d43c55d58c272b5ef477396310ba65b9290ab58d522d6
 => => naming to docker.io/library/student-server
hpatel65373@cloudshell:~ (able-scope-442020-a1)$
```

Tag the docker image and push it to your dockerhub,

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

With your username,

- $ docker tag studentserver hardik952/studentserver:latest
- $ docker push hardik952/studentserver:latest

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ docker login
Authenticating with existing credentials...
WARNING! Your password will be stored unencrypted in /home/hpatel65373/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credential-stores

Login Succeeded
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ $ docker tag studentserver hardik952/studentserver:latest
-bash: $: command not found
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ docker tag studentserver hardik952/studentserver:latest
Error response from daemon: No such image: studentserver:latest
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ docker push hardik952/studentserver:latest
The push refers to repository [docker.io/hardik952/studentserver]
An image does not exist locally with the tag: hardik952/studentserver
hpatel65373@cloudshell:~ (able-scope-442020-a1)$
```

3. Create and deploy a python Flask bookshelf REST API

Create bookshelf.py
vim bookshelf.py with 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__)

# MongoDB configuration
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):
-- INSERT --
```

Create a Dockerfile,
vim Dockerfile with following code,

```
# Use the Python 3.7 slim image
FROM python:3.7-slim

# Copy the application code to the container
COPY . /app

# Set the working directory
WORKDIR /app

# Upgrade pip and install dependencies from requirements.txt
RUN pip install --upgrade pip && pip install -r requirements.txt

# Set environment variable for the port
ENV PORT 5000

# Expose the application port
EXPOSE 5000

# Set the entry point for the container
ENTRYPOINT ["python3"]

# Define the command to run the application
CMD ["bookshelf.py"]
~
~
```

Build the bookshelf app docker image,

- docker build -t bookshelf .

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ docker build -t bookshelf .
[+] Building 59.9s (10/10) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 547B
 => [internal] load metadata for docker.io/library/python:3.7-slim
 => [auth] library/python:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/4] FROM docker.io/library/python:3.7-slim@sha256:b53f496ca43e5af6994f8e316cf03af31050bf7944e0e4a308ad86c001cf028b
 => => resolve docker.io/library/python:3.7-slim@sha256:b53f496ca43e5af6994f8e316cf03af31050bf7944e0e4a308ad86c001cf028b
 => => sha256:a255ffcb469f2ec40f2958a76beb0c2bbebfe92ce9af67a9b48d84b4cb695ac8 7.54kB / 7.54kB
 => => sha256:a803e7c4b030119420574a882a52b6431e160fceb7620f61b525d49bc2d58886 29.12MB / 29.12MB
 => => sha256:bf3336e84c8e00632cdea35b18fec9a5691711bdc8ac885e3ef54a3d5ff500ba 3.50MB / 3.50MB
 => => sha256:8973eb85275f19b8d72c6047560629116ad902397e5c1885b2508788197de28b 11.38MB / 11.38MB
 => => sha256:b53f496ca43e5af6994f8e316cf03af31050bf7944e0e4a308ad86c001cf028b 1.86kB / 1.86kB
 => => sha256:ffd28e36ef37b3a4a24f6a771a48d7c5499ea42d6309ac911a3f699e122060be 1.37kB / 1.37kB
 => => sha256:f9afc3cc0135aad884dad502f28a5b3d8cd32565116131da818ebf2ea6d46095 244B / 244B
 => => sha256:39312d8b4ab77de264678427265a2668073675bb8666caf723da18c9e4b7e3fc 3.13MB / 3.13MB
 => => extracting sha256:a803e7c4b030119420574a882a52b6431e160fceb7620f61b525d49bc2d58886
 => => extracting sha256:bf3336e84c8e00632cdea35b18fec9a5691711bdc8ac885e3ef54a3d5ff500ba
 => => extracting sha256:8973eb85275f19b8d72c6047560629116ad902397e5c1885b2508788197de28b
 => => extracting sha256:f9afc3cc0135aad884dad502f28a5b3d8cd32565116131da818ebf2ea6d46095
 => => extracting sha256:39312d8b4ab77de264678427265a2668073675bb8666caf723da18c9e4b7e3fc
 => [internal] load build context
 => => transferring context: 1.60MB
 => [2/4] COPY . /app
 => [3/4] WORKDIR /app
 => [4/4] RUN pip install --upgrade pip && pip install -r requirements.txt
 => exporting to image
 => => exporting layers
 => => writing image sha256:e7bfd6e11ab8135c06dd9aa59b2ad7b74f7fbbad5df17738b2c5cddaff793898
 => => naming to docker.io/library/bookshelf
```

Tag the docker image and push it to your dockerhub,

- docker tag bookshelf hardik952/bookshelf:latest
- docker push hardik952/bookshelf:latest

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ docker tag bookshelf hardik952/bookshelf:latest
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ docker push hardik952/bookshelf:latest
The push refers to repository [docker.io/hardik952/bookshelf]
40c10afd57a7: Pushed
5f70bf18a086: Pushed
5efd61e745cd: 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: sha256:715164a59cce372dfce776a9f82554817e029842e78dc139ee92226574ad6acb size: 2002
hpatel65373@cloudshell:~ (able-scope-442020-a1)$
```

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

Create studentserver-configmap.yaml

- vim studentserver-configmap.yaml with following code,

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

Create bookshelf-configmap.yaml

- vim bookshelf-configmap.yaml with 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
~
~
~
```

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

Create studentserver-deployment.yaml

- vim studentserver-deployment.yaml with following code,

```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: username/studentserver:latest
          imagePullPolicy: Always
          name: web
          ports:
            - containerPort: 8080
          env:
```

```yaml
env:
  - name: MONGO_URL
    valueFrom:
      configMapKeyRef:
        name: studentserver-config
        key: MONGO_URL
  - name: MONGO_DATABASE
    valueFrom:
      configMapKeyRef:
        name: studentserver-config
        key: MONGO_DATABASE
```

Create bookshelf-deployment.yaml

- vim bookshelf-deployment.yaml with following code,

```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: username/bookshelf:latest
          imagePullPolicy: Always
          name: bookshelf-deployment
          ports:
            - containerPort: 5000
          env:
```

```yaml
 env:
   - name: MONGO_URL
     valueFrom:
       configMapKeyRef:
         name: bookshelf-config
         key: MONGO_URL
   - name: MONGO_DATABASE
     valueFrom:
       configMapKeyRef:
         name: bookshelf-config
         key: MONGO_DATABASE
```

Create studentserver-service.yaml

- vim studentserver-service.yaml with following code,

```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
~
~
```

Create bookshelf-service.yaml

- vim bookshelf-service.yaml with following code,

```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
~
~
```

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

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ kubectl apply -f studentserver-deployment.yaml
deployment.apps/web created
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ kubectl apply -f studentserver-configmap.yaml
configmap/studentserver-config created
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ kubectl apply -f studentserver-service.yaml
service/web created
hpatel65373@cloudshell:~ (able-scope-442020-a1)$
```

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

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
hpatel65373@cloudshell:~ (able-scope-442020-a1)$
```

5-7 Check if all the pods are running correctly

- kubectl get pods

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ kubectl get pods
NAME                                  READY   STATUS             RESTARTS   AGE
bookshelf-deployment-5765fb8668-gpmqx   0/1     ImagePullBackOff   0          2m5s
mongodb-deployment-658f677f67-4cx91     1/1     Running            0          94m
web-5c485bc9b-8t7vg                   0/1     ErrImagePull       0          3m33s
hpatel65373@cloudshell:~ (able-scope-442020-a1)$
```

5-8 Check if all the services are running correctly

- kubectl get svc

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ kubectl get svc
NAME                TYPE           CLUSTER-IP       EXTERNAL-IP     PORT(S)           AGE
bookshelf-service   LoadBalancer   34.118.234.59    35.197.13.4     5000:32709/TCP    2m22s
kubernetes          ClusterIP      34.118.224.1     <none>          443/TCP           100m
mongodb-service     LoadBalancer   34.118.228.251   34.168.11.57    27017:32583/TCP   92m
web                 LoadBalancer   34.118.227.159   34.168.110.17   8080:30672/TCP    3m43s
hpatel65373@cloudshell:~ (able-scope-442020-a1)$
```

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

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ helm repo add traefik https://helm.traefik.io/traefik
"traefik" has been added to your repositories
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "traefik" chart repository
Update Complete. ⎈Happy Helming!⎈
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ helm install traefik traefik/traefik
NAME: traefik
LAST DEPLOYED: Wed Nov 27 22:59:07 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
traefik with docker.io/traefik:v3.2.0 has been deployed successfully on default namespace !
hpatel65373@cloudshell:~ (able-scope-442020-a1)$
```

5-10 Create an ingress service using YAML file

- vim my-ingress.yaml with following code,

```
apiVersion: traefik.io/v1alpha1
kind: IngressRoute
metadata:
  name: my-ingressroute
  namespace: default # Set the namespace to 'default' where Traefik is running
spec:
  entryPoints:
    - web # Ensure 'web' is defined as an entry point in your Traefik config
  routes:
    - match: Host(`35.230.26.107`) && PathPrefix(`/studentserver`)
      kind: Rule
      services:
        - name: web
          port: 8080
    - match: Host(`34.82.173.215`) && 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

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ kubectl apply -f my-ingress.yaml
ingressroute.traefik.io/my-ingressroute created
hpatel65373@cloudshell:~ (able-scope-442020-a1)$
```

5-11 Check if ingressroute service is running

- kubectl get svc

```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ kubectl get svc
NAME                TYPE           CLUSTER-IP       EXTERNAL-IP     PORT(S)                      AGE
bookshelf-service   LoadBalancer   34.118.234.59    35.197.13.4     5000:32709/TCP               8m20s
kubernetes          ClusterIP      34.118.224.1     <none>          443/TCP                      106m
mongodb-service     LoadBalancer   34.118.228.251   34.168.11.57    27017:32583/TCP              98m
traefik             LoadBalancer   34.118.238.196   34.169.107.36   80:30785/TCP,443:30318/TCP   3m26s
web                 LoadBalancer   34.118.227.159   34.168.110.17   8080:30672/TCP               9m41s
hpatel65373@cloudshell:~ (able-scope-442020-a1)$
```

5-12 Add Address to /etc/hosts

- sudo vi /etc/hosts

EXTERNAL-IP your.domain.com

34.118.238196 cs571project.20107.com

```
# /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-163545214915-default
34.118.238196 cs571project.20107.com
```

6. Access our applications

6-1 Access student server

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

- curl your.domain.com/studentserver/api/score?student_id=11111
- curl http://cs571project.20107.com/studentserver/api/score?student_id=11111

6-2 Access the bookshelf application

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

List all books:

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

Update a book:

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

Delete a book:

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

Done!! 🙂