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

1. Create a cluster as usual on GKE
    a. gcloud container clusters create kubia --num-nodes=3 --machine-type=e2-micro --region=asia-east1-b
    b.
```
Created [https://container.googleapis.com/v1/projects/psychic-trainer-419714/zones/asia-east1-b/clusters/kubia].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/asia-east1-b/kubia?project=psychic-tra
iner-419714
kubeconfig entry generated for kubia.
NAME: kubia
LOCATION: asia-east1-b
MASTER_VERSION: 1.27.8-gke.1067004
MASTER_IP: 35.221.164.93
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.27.8-gke.1067004
NUM_NODES: 3
STATUS: RUNNING
```
2. Let's create a Persistent Volume first
    a. gcloud compute disks create --size=10GiB --zone=asia-east1-b mongodb
    b.
```
eweng909@cloudshell:~ (psychic-trainer-419714)$ gcloud compute disks create --size=10GiB --zone=asia-east1-b 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.goo
gle.com/compute/docs/disks#performance.
Created [https://www.googleapis.com/compute/v1/projects/psychic-trainer-419714/zones/asia-east1-b/disks/mongodb].
NAME: mongodb
ZONE: asia-east1-b
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
    a. vim mongodb-deployment.yaml
    b.
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
spec:
  selector:
    matchLabels:
      app: mongodb
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
        - image: mongo
          name: mongo
          ports:
            - containerPort: 27017
          volumeMounts:
            - name: mongodb-data
              mountPath: /data/db
      volumes:
        - name: mongodb-data
          gcePersistentDisk:
            pdName: mongodb
            fsType: ext4
```
    c. kubectl apply -f mongodb-deployment.yaml

d.
```
eweng909@cloudshell:~ (psychic-trainer-419714)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
```

4. Check if the deployment pod has been successfully created and started running
    a. kubectl get pods
    b. Check that status is running
    c.
```
eweng909@cloudshell:~ (psychic-trainer-419714)$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
mongodb-deployment-594c77dcdf-glgv6   1/1     Running   0          48s
```

5. Create a service for the mongoDB, so it can be accessed from outside
    a. mongodb-service.yaml
    b.
```
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  type: LoadBalancer
  ports:
    - port: 27017
      targetPort: 27017
  selector:
    app: mongodb
~
~
```
    c. kubectl apply -f mongodb-service.yaml
    d.
```
eweng909@cloudshell:~ (psychic-trainer-419714)$ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
```

6. Wait couple of minutes, and check if the service is up
    a. kubectl get svc
    b. Wait until external-ip is generated
    c.
```
eweng909@cloudshell:~ (psychic-trainer-419714)$ kubectl get svc
NAME              TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)           AGE
kubernetes        ClusterIP      10.14.240.1     <none>           443/TCP           32m
mongodb-service   LoadBalancer   10.14.249.107   35.236.135.126   27017:32002/TCP   11m
```

7. Now try and see if mongoDB is functioning for connections using the External-IP
    a. kubectl exec -it mongodb-deployment-replace-with-your-pod-name -- bash
    b.
```
eweng909@cloudshell:~ (psychic-trainer-419714)$ kubectl exec -it mongodb-deployment-594c77dcdf-glgv6 -- bash
root@mongodb-deployment-594c77dcdf-glgv6:/#
```
    c. Now you are inside the mongodb deployment pod
    d. Try:
        i. Mongosh EXTERNAL-IP

ii.

```
root@mongodb-deployment-594c77dcdf-f8m8g:/#  mongosh 104.199.140.249
Current Mongosh Log ID: 6619393639210a3d657b2da8
Connecting to:          mongodb://104.199.140.249:27017/?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
   2024-04-12T13:33:52.977+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http
://dochub.mongodb.org/core/prodnotes-filesystem
   2024-04-12T13:33:54.756+00:00: Access control is not enabled for the database. Read and write access to data and configurati
on is unrestricted
   2024-04-12T13:33:54.757+00:00: vm.max_map_count is too low
------
```

iii. Enter exit to exit the root

iv.

```
    2024-04-09T15:15:48.117+00:00: vm.max_map_count i
------

test> exit
root@mongodb-deployment-594c77dcdf-8vqhs:/#
```

8. We need to insert some records into the mongoDB for later use
    a. • First Make sure node is installed.
        i. Npm install mongodb
9. Insert student records into mongodb
    a. Go to your database inside mongodb
        i. Use mydb
        ii. Insert 3 students records:
        iii. >db.students.insertMany([
            1. { student_id: 11111, student_name: "Bruce Lee", grade: 84 },
            2. { student_id: 22222, student_name: "Jackie Chen", grade: 93 },
            3. { student_id: 33333, student_name: "Jet Li", grade: 88 }
        iv.])

v.

```
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('661941777a14df806a7b2da9'),
    '1': ObjectId('661941777a14df806a7b2daa'),
    '2': ObjectId('661941777a14df806a7b2dab')
  }
}
mydb>
```

b. Exit and go to GCP console

i.

```
mydb> exit
root@mongodb-deployment-594c77dcdf-f8m8g:/#
```

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

1. Create a studentServer

a.

```
const http = require('http');
const { MongoClient } = require('mongodb');
const url = require('url');
const os = require('os'); // Import the os module

// Update this URI to your MongoDB URI
const uri = "mongodb://" + process.env.MONGO_URL + "/" + process.env.MONGO_DATABASE; // Use process.env to ac
ariables
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 = new URL(req.url, 'http://localhost'); // Use new URL() constructor
               const db = client.db();
               // Example: Handle a simple get request
               if (parsedUrl.pathname === '/api/score') {
                       const studentId = parseInt(parsedUrl.searchParams.get('student_id'), 10); // Use sear
ery parameters
                       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) {
"studentServer.js" 50L, 2464B
```

b.

```
                        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)
```

c. Vim studentserver.js

2. Create Dockerfile
    a. vim Dockerfile
    b. Use the code:
        i. # Use a Node.js base image
        ii. FROM node:latest
        iii. # Copy your application code and package.json to the container
        COPY . /app
        iv. WORKDIR /app
        v. # Install dependencies
        vi. RUN npm install mongodb
        vii. # Your application's default command CMD ["node", "studentServer.js"]
        viii.

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

        ix.

```
eweng909@cloudshell:~/mongodb (cs571-project12)$ docker build -t studentserver .
[+] Building 3.6s (9/9) FINISHED                                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                                          0.0s
 => => transferring dockerfile: 295B                                                                                          0.0s
 => [internal] load metadata for docker.io/library/node:latest                                                                0.5s
 => [internal] load .dockerignore                                                                                             0.0s
 => => transferring context: 2B                                                                                               0.0s
 => [internal] load build context                                                                                             0.0s
 => => transferring context: 358B                                                                                             0.0s
 => CACHED [1/4] FROM docker.io/library/node:latest@sha256:162d92c5f1467ad877bf6d8a098d9b04d7303879017a2f3644bfb1de1fc88      0.0s
 => [2/4] COPY . /app                                                                                                         0.0s
 => [3/4] WORKDIR /app                                                                                                        0.0s
 => [4/4] RUN npm install mongodb                                                                                             2.8s
 => exporting to image                                                                                                        0.2s
 => => exporting layers                                                                                                       0.2s
 => => writing image sha256:41e12e64f53faaab376258c57246ae40689083f6f6125f104e7551f4c590d52d                                  0.0s
 => => naming to docker.io/library/studentserver                                                                              0.0s
eweng909@cloudshell:~/mongodb (cs571-project12)$ vim Dockerfile
```

3. Tag the docker image and push it to your dockerhub
    a. docker tag studentserver username/studentserver:latest
    b. docker push username/studentserver:latest
    c.

```
eweng909@cloudshell:~/mongodb (cs571-project12)$ docker tag studentserver eweng909/studentserver:latest
eweng909@cloudshell:~/mongodb (cs571-project12)$  docker push eweng909/studentserver:latest
The push refers to repository [docker.io/eweng909/studentserver]
853821d522b8: Pushed
5f70bf18a086: Pushed
f90a1d7a9690: Pushed
3a72264cad04: Mounted from library/node
9f017d2bee1c: Mounted from library/node
47181ad0eb66: Mounted from library/node
3e81cc85b636: Mounted from library/node
893507f6057f: Mounted from library/node
2353f7120e0e: Mounted from library/node
51a9318e6edf: Mounted from library/node
c5bb35826823: Mounted from library/node
latest: digest: sha256:c043a016836dd5ffb52148d87b278a7a9f229dfce01d1cd4ec87266353398154 size: 2629
```

Create and deploy a python Flask bookshelf REST API

1. Create bookshelf.py
    a. Vim bookshelf.py
    b.

```
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!")
```

c.

```
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)))
~
```

2. Create a Dockerfile
    a. vim Dockerfile
    b. FROM python:3.7-slim
    c. COPY . /app
    d. WORKDIR /app
    e. RUN pip install --upgrade pip && pip install -r requirements.txt
    f. ENV PORT 5000
    g. EXPOSE 5000
    h. ENTRYPOINT ["python3"]
    i. CMD ["bookshelf.py"]
3. Create a requirements.txt as well
    a.

```
Flask==1.1.2
Flask-PyMongo==2.3.0
gunicorn==20.0.4
requests==2.25.1
http://storage.googleapis.com/velostrata-release/gce-v2v/gce-v2v.tar.gz
~
~
```

4. Build the bookshelf app docker image
    a. docker build -t bookshelf .

b.

```
eweng909@cloudshell:~/mongodb (cs571-project12)$ docker build -t bookshelf .
[+] Building 15.1s (9/9) FINISHED                                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                                          0.0s
 => => transferring dockerfile: 459B                                                                                          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
 => CACHED [1/4] FROM docker.io/library/python:3.7-slim@sha256:b53f496ca43e5af6994f8e316cf03af31050bf7944e0e4a308ad86c00      0.0s
 => [internal] load build context                                                                                             0.0s
 => => transferring context: 2.40kB                                                                                           0.0s
 => [2/4] COPY . /app                                                                                                         0.0s
 => [3/4] WORKDIR /app                                                                                                        0.0s
 => [4/4] RUN pip install -r requirements.txt                                                                                13.4s
 => exporting to image                                                                                                        1.0s
 => => exporting layers                                                                                                       1.0s
 => => writing image sha256:67c755ec8a1e539387b57846f56e0bd941e69ac4ad69e9ef3f57d366aeb9c607                                  0.0s
 => => naming to docker.io/library/bookshelf                                                                                  0.0s
eweng909@cloudshell:~/mongodb (cs571-project12)$
```

5. Tag the docker image and push it into your dockerhub
    a. docker tag bookshelf username/bookshelf:latest
    b. docker push username/bookshelf:latest

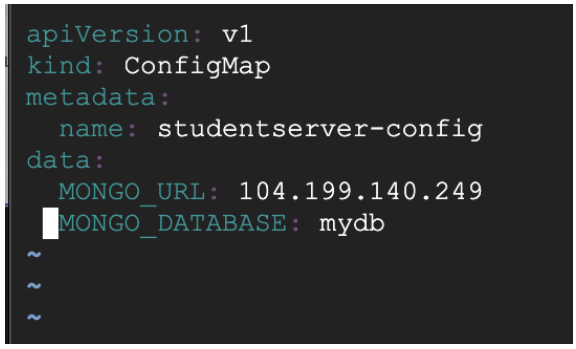
c.

```
eweng909@cloudshell:~/mongodb (cs571-project12)$ docker tag bookshelf eweng909/bookshelf:latest
eweng909@cloudshell:~/mongodb (cs571-project12)$ docker push eweng909/bookshelf:latest
The push refers to repository [docker.io/eweng909/bookshelf]
2a119a99c7e7: Pushed
5f70bf18a086: Mounted from eweng909/studentserver
a7ff4661cc50: 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: sha256:29edf3f143b44173b683865e958b7b74555fbaa1b946be606f51ddaf5ab769e6 size: 1996
```

Create ConfigMap for both applications to store MongoDB URL and name

1. Create studentserver-configmap.yaml
    a. vim studentserver-configmap.yaml
        i. the following code:
        ii. apiVersion: v1
        iii. kind: ConfigMap
        iv. Metadata:
            1. name: studentserver-config
        v. Data:
            1. MONGO_URL: your.mongodb.EXTERNAL-IP
        vi. MONGO_DATABASE: mydb

vii.



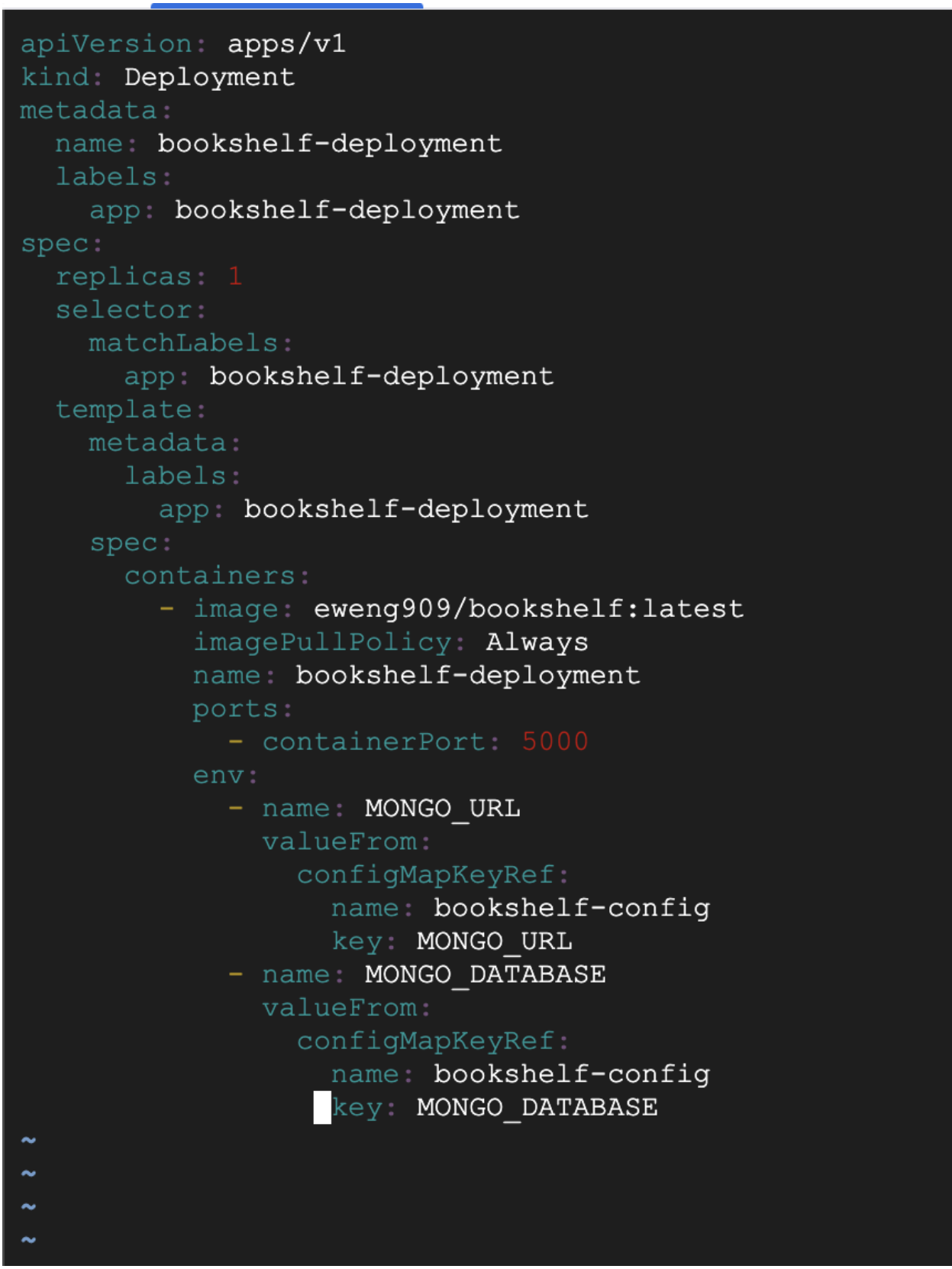
2. Create bookshelf-configmap.yaml
    a. vim bookshelf-configmap.yaml
    b. apiVersion: v1
    c. kind: ConfigMap
    d. metadata:
    e. name: bookshelf-config
    f. data:
    g. # SERVICE_NAME.NAMESPACE.svc.cluster.local:SERVICE_PORT
    h. MONGO_URL: your.mongodb.EXTERNAL-IP
    i. MONGO_DATABASE: mydb

j.

Expose both applications using ingress with traefik

1. Create studentserver-deployment.yaml
    a. vim studentserver-deployment.yaml
    b. Replace the username in image: username/studentserver:latest

c.



```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: eweng909/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
~
~
~
~
```

2. Create bookshelf-deployment.yaml

a. vim bookshelf-deployment.yaml
b. Replace the username in image: username/bookshelf:latest
c.

```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: eweng909/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
~
```

3. Create studentserver-service.yaml

a. vim studentserver-service.yaml

b.
```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
   a. vim bookshelf-service.yaml
   b.
```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. Create all the studentserver related pods and start service using the above YAML files
   a. kubectl apply -f studentserver-deployment.yaml
   b. kubectl apply -f studentserver-configmap.yaml
   c. kubectl apply -f studentserver-service.yaml
   d.
```
eweng909@cloudshell:~/mongodb (cs571-project12)$ kubectl apply -f studentserver-deployment.yaml
deployment.apps/web created
```
   e.
```
eweng909@cloudshell:~/mongodb (cs571-project12)$ kubectl apply -f studentserver-configmap.yaml
configmap/studentserver-config created
```
   f.
```
eweng909@cloudshell:~/mongodb (cs571-project12)$ kubectl apply -f studentserver-service.yaml
service/web created
```

6. Create all the bookshelf related pods and start service using the above YAML files
   a. kubectl apply -f bookshelf-deployment.yaml

b. kubectl apply -f bookshelf-configmap.yaml
c. kubectl apply -f bookshelf-service.yaml
d.

```
eweng909@cloudshell:~/mongodb (cs571-project12)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created
eweng909@cloudshell:~/mongodb (cs571-project12)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
eweng909@cloudshell:~/mongodb (cs571-project12)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
eweng909@cloudshell:~/mongodb (cs571-project12)$
```

7. Check if all the pods are running correctly

a.

```
eweng909@cloudshell:~/mongodb (cs571-project12)$ kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
bookshelf-deployment-548554fd7c-59tfm   1/1     Running   0          2s
mongodb-deployment-594c77dcdf-f8m8g     1/1     Running   0          3h9m
web-7d5d677cbf-sdtc9                    1/1     Running   0          52m
eweng909@cloudshell:~/mongodb (cs571-project12)$
```

8. Check if all the services are running correctly
   a. kubectl get svc
   b.

```
eweng909@cloudshell:~/mongodb (cs571-project12)$ kubectl get svc
NAME                TYPE           CLUSTER-IP     EXTERNAL-IP       PORT(S)          AGE
bookshelf-service   LoadBalancer   10.18.12.47    35.185.147.138    5000:31036/TCP   31m
kubernetes          ClusterIP      10.18.0.1      <none>            443/TCP          3h46m
mongodb-service     LoadBalancer   10.18.1.26     104.199.140.249   27017:30210/TCP  3h9m
web                 LoadBalancer   10.18.13.225   35.221.129.115    8080:32179/TCP   50m
eweng909@cloudshell:~/mongodb (cs571-project12)$
```

   c. Wait for external ip to show then continue
9. Install traefik so we can use it to create ingressroute later
   a. helm repo add traefik https://helm.traefik.io/traefik
   b. helm repo update
   c. helm install traefik traefik/traefik
   d.

```
eweng909@cloudshell:~/mongodb (cs571-project12)$ helm repo add traefik https://helm.traefik.io/traefik
"traefik" has been added to your repositories
eweng909@cloudshell:~/mongodb (cs571-project12)$ 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!⎈
eweng909@cloudshell:~/mongodb (cs571-project12)$ helm install traefik traefik/traefik
NAME: traefik
LAST DEPLOYED: Fri Apr 12 16:45:22 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Traefik Proxy v2.11.2 has been deployed successfully on default namespace !
eweng909@cloudshell:~/mongodb (cs571-project12)$
```

10. Create an ingress service using YAML file
    a. vim my-ingress.yaml

b.

```
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
~
```

11. Apply the YAML file to create the ingressroute
    a. kubectl apply -f my-ingress.yaml
    b.

```
eweng909@cloudshell:~/mongodb (cs571-project12)$  kubectl apply -f my-ingress.yaml
ingressroute.traefik.containo.us/my-ingressroute created
eweng909@cloudshell:~/mongodb (cs571-project12)$
```

12. Check if ingressroute service is running
    a. kubectl get svc
    b.

```
eweng909@cloudshell:~/mongodb (cs571-project12)$ kubectl get svc
\NAME                TYPE           CLUSTER-IP     EXTERNAL-IP       PORT(S)                      AGE
bookshelf-service    LoadBalancer   10.18.12.47    35.185.147.138    5000:31036/TCP               36m
kubernetes           ClusterIP      10.18.0.1      <none>            443/TCP                      3h51m
mongodb-service      LoadBalancer   10.18.1.26     104.199.140.249   27017:30210/TCP              3h14m
traefik              LoadBalancer   10.18.9.243    35.194.179.81     80:31198/TCP,443:30589/TCP   3m18s
web                  LoadBalancer   10.18.13.225   35.221.129.115    8080:32179/TCP               54m
eweng909@cloudshell:~/mongodb (cs571-project12)$
```

13. Add Address to /etc/hosts
    a. sudo vi /etc/hosts
    b. Add the EXTERNAL-IP and youe domain name to the end of the file:
    EXTERNAL-IP your.domain.com
    c.

```
#
169.254.169.254 metadata.google.internal metadata

10.88.0.4 cs-1095463139174-default

104.199.140.249 cs571project.20016.com
~
~
~
```

Access our applications

1. Access student server
    a. Access existing record:
    b. curl your.domain.com/studentserver/api/score?student_id=11111

c. curl your.domain.com/studentserver/api/score?student_id=22222

2. Access the bookshelf application

a.

```
eweng909@cloudshell:~/mongodb (cs571-project12)$ curl http://cs571project.20016.com/studentserver/api/score?student_id=22222
404 page not found
eweng909@cloudshell:~/mongodb (cs571-project12)$ curl http://cs571project.20016.com/studentserver/api/score?student_id=33333
404 page not found
eweng909@cloudshell:~/mongodb (cs571-project12)$ curl http://cs571project.20016.com/studentserver/api/score?student_id=44444
404 page not found
eweng909@cloudshell:~/mongodb (cs571-project12)$ curl -X POST -d "{\"book_name\": \"star wars\",\"book_author\":
\"unkown\", \"isbn\": \"654321\" }" http://cs571project.20016.com/bookshelf/book
404 page not found
eweng909@cloudshell:~/mongodb (cs571-project12)$ curl http://cs571project.20016.com/bookshelf/books
404 page not found
```

Github link:

Comments:
Repeatedly keeps saying 404 not found and bookshelf deployment keeps crashing in the end.
I've worked on this till Friday for many days. and will try again later on.