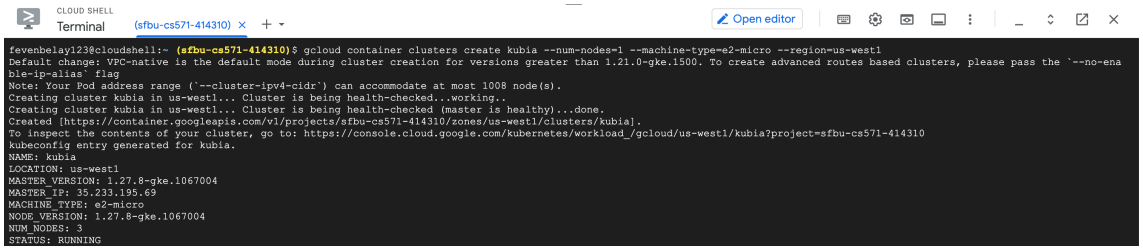
Week 12: Homework: Chapter 7: Configmap: Signature Project: MongoDB + Python Flask Web Framework + REST API + GKE
Name: Feven Belay Araya
ID: 20027

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

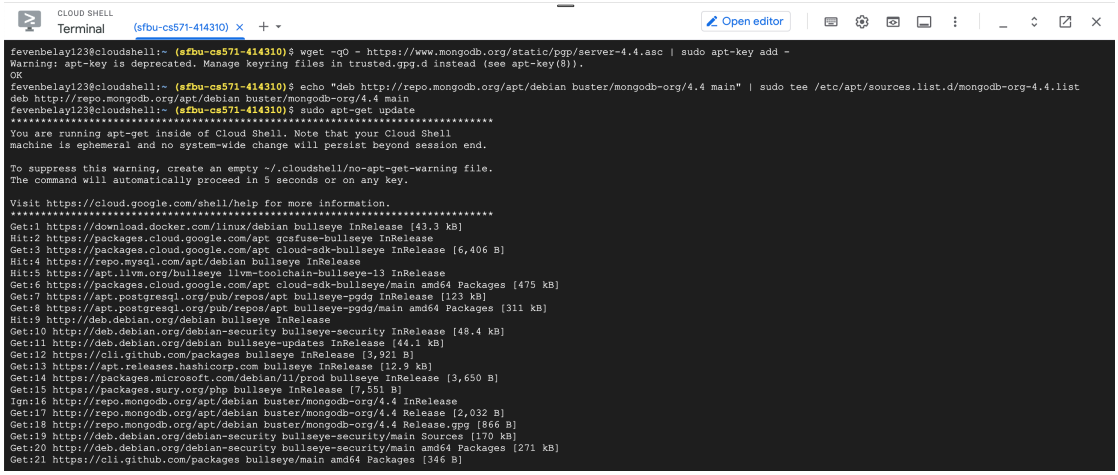
Step1. Create MongoDB using Persistent Volume on GKE, and insert records into it

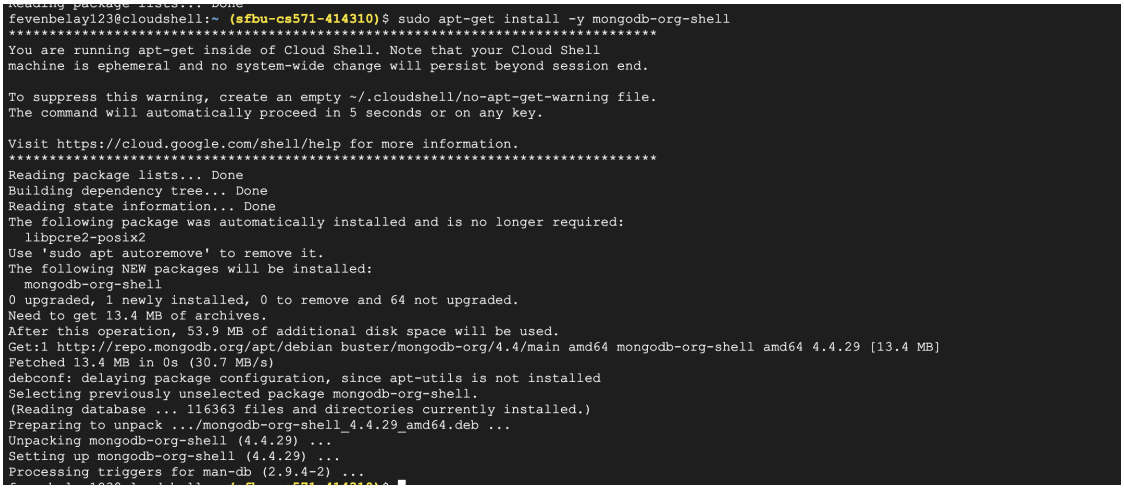
1. Create a cluster as usual on GKE using the command
   gcloud container clusters create kubia --num-nodes=1 --machine-type=e2-micro --region=us-west1



2. Create a Persistent Volume first using the command
   - *gcloud compute disks create --size=10GiB --zone=us-west1-a mongodb*

3. Install mongodb using the following commands
   - *wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | sudo apt-key add -*
   - *echo "deb http://repo.mongodb.org/apt/debian buster/mongodb-org/4.4 main" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.4.list*
   - *sudo apt-get update*
   - *sudo apt-get install -y mongodb-org-shell*





4. Create a mongodb deployment with this yaml file
   - Then *kubectl apply -f mongodb-deployment.yaml*

```
CLOUD SHELL
Terminal (sfbu-cs571-414310) ×
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ nano mongodb-deployment.yaml
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ cat mongodb-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
spec:
  selector:
    matchLabels:
      app: mongodb
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
      - name: mongo
        image: mongo
        ports:
        - containerPort: 27017
        volumeMounts:
        - name: mongodb-data
          mountPath: /data/db
      volumes:
      - name: mongodb-data
        gcePersistentDisk:
          pdName: mongodb
          fsType: ext4
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment configured
```

5. Check if the deployment pod has been successfully created and started running using the command
   - *kubectl get pods*

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ kubectl get pods
NAME                                 READY   STATUS    RESTARTS   AGE
mongodb-deployment-594c77dcdf-j7x8w   1/1     Running   0          58s
```

6. Create a service for the mongoDB, so it can be accessed from outside
   - Then *kubectl apply -f mongodb-service.yaml*

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ nano mongodb-service.yaml
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ cat mongodb-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  type: LoadBalancer
  ports:
  - port: 27017
    targetPort: 27017
  selector:
    app: mongodb
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
```

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

```
service/mongodb-service created
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ kubectl get svc
NAME              TYPE           CLUSTER-IP     EXTERNAL-IP       PORT(S)           AGE
kubernetes        ClusterIP      10.36.0.1      <none>            443/TCP           30m
mongodb-service   LoadBalancer   10.36.15.13    104.196.231.165   27017:32054/TCP   52s
```

8. Now try and see if mongoDB is functioning for connections using the External-IP using the command
   - kubectl exec -it mongodb-deployment-replace-with-your-pod-name -- bash

   Now you are inside the mongodb deployment pod Try
   - *Mongo EXTERNAL-IP*

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ kubectl exec -it mongodb-deployment-594c77dcdf-j7x8w -- bash
root@mongodb-deployment-594c77dcdf-j7x8w:/# mongo 104.196.231.165
bash: mongo: command not found
root@mongodb-deployment-594c77dcdf-j7x8w:/# mongosh 104.196.231.165
Current Mongosh Log ID: 661736efd0ec0aa5447b2da8
Connecting to:          mongodb://104.196.231.165:27017/?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
   2024-04-11T00:58:04.932+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
   2024-04-11T00:58:05.755+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
   2024-04-11T00:58:05.756+00:00: vm.max_map_count is too low
------

test> exit
root@mongodb-deployment-594c77dcdf-j7x8w:/# exit
exit
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$
```

You can also try from outside as follows
   - *Mongo EXTERNAL-IP*

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ mongo 104.196.231.165
MongoDB shell version v4.4.29
connecting to: mongodb://104.196.231.165:27017/test?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("94c4098d-8218-401c-aa77-43569d50d484") }
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:
        2024-04-11T00:58:04.932+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
        2024-04-11T00:58:05.755+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
        2024-04-11T00:58:05.756+00:00: vm.max_map_count is too low
        2024-04-11T00:58:05.756+00:00:         currentValue: 65530
        2024-04-11T00:58:05.756+00:00:         recommendedMinimum: 1677720
        2024-04-11T00:58:05.756+00:00:         maxConns: 838860
---
> exit
bye
```

9. We need to insert some records into the mongoDB for later using
   - *node*

and then enter the below contents.
   - 3 means three records was inserted, and we tried search for student_id=11111

```
fba8584@cloudshell:~/node_modules/mongodb (cs571-demo-project-419705)$ node
Welcome to Node.js v20.11.1.
Type ".help" for more information.
> const MongoClient = require('mongodb').MongoClient;
undefined
> const url = "mongodb://104.198.110.9/mydb";
undefined
>
> async function run() {
...     try {
...         const client = await MongoClient.connect(url, { useNewUrlParser: true, useUnifiedTopology: true });
...         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 insertResult = await db.collection("students").insertMany(docs);
...         console.log(insertResult.insertedCount);
...
...         const findResult = await db.collection("students").findOne({ "student_id": 11111 });
...         console.log(findResult);
...
...         client.close();
...     } catch (err) {
...         console.error(err);
...     }
... }
undefined
>
> run();
Promise {
  <pending>,
  [Symbol(async_id_symbol)]: 64,
  [Symbol(trigger_async_id_symbol)]: 6
}
> (node:5544) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
(Use `node --trace-warnings ...` to show where the warning was created)
(node:5544) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
3
{
  _id: new ObjectId('66160f8d2ae6593acdb89728'),
  student_id: 11111,
  student_name: 'Bruce Lee',
  grade: 84
}
```

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

1. Create a studentServer

```
var http = require('http');
var url = require('url');
var mongodb = require('mongodb');
const {
  MONGO_URL,
  MONGO_DATABASE
} = process.env;
// - Expect the request to contain a query
// string with a key 'student_id' and a student ID as
// the value. For example
// /api/score?student_id=11111
// - The JSON response should contain only 'student_id', 'student_name'
// and 'student_score' properties. For example:
//
// {
//   "student_id": 1111,
//   "student_name": Bruce Lee,
//   "student_score": 84
// }
//
var MongoClient = mongodb.MongoClient;
var uri = `mongodb://${MONGO_URL}/${MONGO_DATABASE}`;
// Connect to the db
console.log(uri);

var server = http.createServer(function (req, res) {
```

```
  var result;
  // req.url = /api/score?student_id=11111
  var parsedUrl = url.parse(req.url, true);
  var student_id = parseInt(parsedUrl.query.student_id);

  // match req.url with the string /api/score
  if (/^\/api\/score/.test(req.url)) {
    // e.g., of student_id 11111
    MongoClient.connect(uri, { useNewUrlParser: true, useUnifiedTopology: true }, function (err,
client) {
      if (err)
        throw err;
      var db = client.db("studentdb");
      db.collection("students").findOne({ "student_id": student_id }, (err, student) => {
        if (err)
          throw new Error(err.message, null);
        if (student) {
          res.writeHead(200, { 'Content-Type': 'application/json' });
          res.end(JSON.stringify(student) + '\n');
        } else {
          res.writeHead(404);
          res.end("Student Not Found \n");
        }
      });
    });
  } else {
    res.writeHead(404);
    res.end("Wrong url, please try again\n");
  }
});
server.listen(8080);
```

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ cat studentServer.js
var http = require('http');
var url = require('url');
var mongodb = require('mongodb');
const {
  MONGO_URL,
  MONGO_DATABASE
} = process.env;
// - Expect the request to contain a query
// string with a key 'student_id' and a student ID as
// the value. For example
// /api/score?student_id=11111
// - The JSON response should contain only 'student_id', 'student_name'
// and 'student_score' properties. For example:
//
// {
//   "student_id": 1111,
//   "student_name": Bruce Lee,
//   "student_score": 84
// }
//
var MongoClient = mongodb.MongoClient;
var uri = `mongodb://${MONGO_URL}/${MONGO_DATABASE}`;
// Connect to the db
console.log(uri);

var server = http.createServer(function (req, res) {
  var result;
  // req.url = /api/score?student_id=11111
  var parsedUrl = url.parse(req.url, true);
  var student_id = parseInt(parsedUrl.query.student_id);

  // match req.url with the string /api/score
  if (/^\/api\/score/.test(req.url)) {
    // e.g., of student_id 11111
    MongoClient.connect(uri, { useNewUrlParser: true, useUnifiedTopology: true }, function (err, client) {
      if (err)
        throw err;
```

2. Create Dockerfile as follows

FROM node:7
ADD studentServer.js /studentServer.js
ENTRYPOINT ["node", "studentServer.js"]
RUN npm config set registry https://registry.npmjs.org/

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ cat 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 .

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ docker login -u fevenbelay123225
Password:
WARNING! Your password will be stored unencrypted in /home/fevenbelay123/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ docker build -t fevenbelay123225/studentserver .
[+] Building 0.4s (8/8) FINISHED                                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                                          0.0s
 => => transferring dockerfile: 186B                                                                                          0.0s
 => [internal] load metadata for docker.io/library/node:7                                                                     0.3s
 => [internal] load .dockerignore                                                                                             0.0s
 => => transferring context: 2B                                                                                               0.0s
 => [internal] load build context                                                                                             0.0s
 => => transferring context: 38B                                                                                              0.0s
 => [1/3] FROM docker.io/library/node:7@sha256:af5c2c6ac8bc3fa372ac031ef60c45a285eeba7bce9ee9ed66dad3a01e29ab8d                0.0s
 => CACHED [2/3] ADD studentServer.js /studentServer.js                                                                       0.0s
 => CACHED [3/3] RUN npm config set registry https://registry.npmjs.org/                                                      0.0s
 => exporting to image                                                                                                        0.0s
 => => exporting layers                                                                                                       0.0s
 => => writing image sha256:7226f304ff6b5255acbd3d0f65e57725d14db2ec0552ef5515ff7ff4b17128c3                                  0.0s
 => => naming to docker.io/fevenbelay123225/studentserver                                                                     0.0s
```

4. Push the docker image

- docker push yourdockerhubID/studentserver

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ docker push fevenbelay123225/studentserver
Using default tag: latest
The push refers to repository [docker.io/fevenbelay123225/studentserver]
8abfe0124acd: Pushed
b2a1c4433ea1: Pushed
ab90d83fa34a: Pushed
8ee318e54723: Pushed
e6695624484e: Pushed
da59b99bbd3b: Pushed
5616a6292c16: Pushed
f3ed6cb59ab0: Pushed
654f45ecb7e3: Pushed
2c40c66f7667: Pushed
latest: digest: sha256:f3a6c7600bf8ba24b970a03f9dd673d54b2f26955712528861352fa104d87b0a size: 2420
```

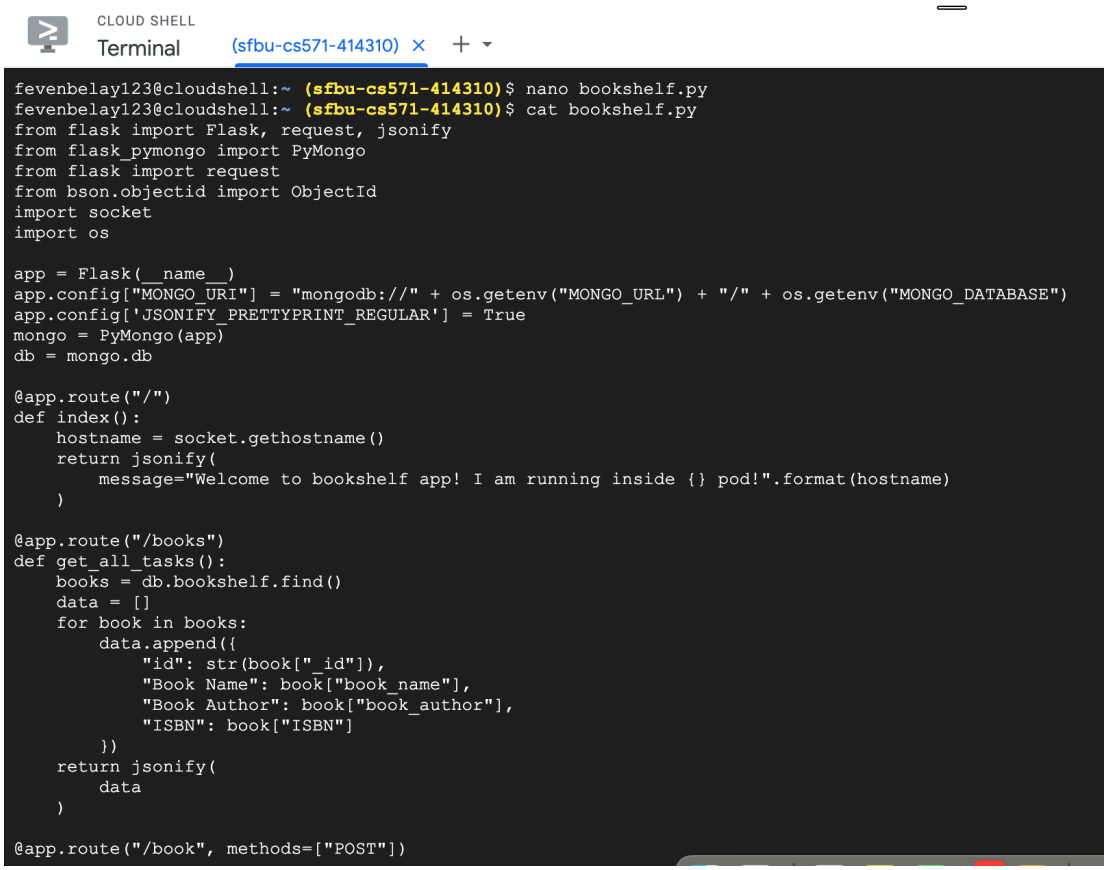
Step3 Create a python Flask bookshelf REST API and deploy on GKE

1. Create bookshelf.py

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

app = Flask(__name__)
app.config["MONGO_URI"] = "mongodb://" + os.getenv("MONGO_URL") + "/" +
os.getenv("MONGO_DATABASE")
app.config['JSONIFY_PRETTYPRINT_REGULAR'] = True
mongo = PyMongo(app)
db = mongo.db

@app.route("/")
def index():
    hostname = socket.gethostname()
    return jsonify(
        message="Welcome to bookshelf app! I am running inside {} pod!".format(hostname)
    )

@app.route("/books")
def get_all_tasks():
    books = db.bookshelf.find()
    data = []
    for book in books:
        data.append({
            "id": str(book["_id"]),
            "Book Name": book["book_name"],
            "Book Author": book["book_author"],
            "ISBN": book["ISBN"]
        })
    return jsonify(
```

```
        data
    )

@app.route("/book", methods=["POST"])
def add_book():
    book = request.get_json(force=True)
    db.bookshelf.insert_one({
        "book_name": book["book_name"],
        "book_author": book["book_author"],
        "ISBN": book["isbn"]
    })
    return jsonify(
        message="Task saved successfully!"
    )

@app.route("/book/<id>", methods=["PUT"])
def update_book(id):
    data = request.get_json(force=True)
    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)
```



```
@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 a Dockerfile

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ nano Dockerfile
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ cat 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" ]
```

3. Create requirements.txt

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ nano requirements.txt
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ cat requirements.txt
Flask==2.0.1
flask-pymongo==2.3.0

fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$
```

4. Build the bookshelf app into a docker image
   - docker build -t zhou19539/bookshelf .

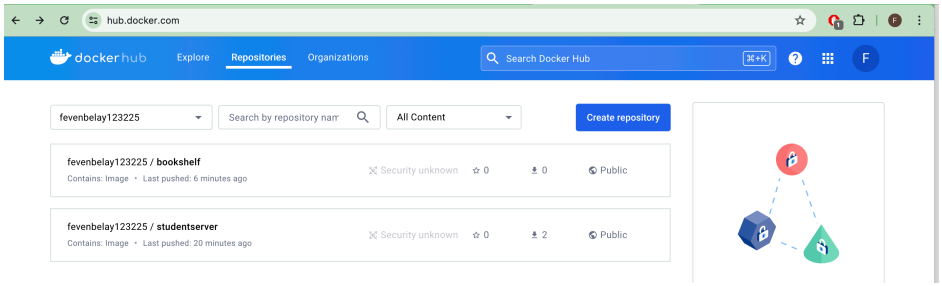
Make sure this step build successfully

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ docker build -t fevenbelay123225/bookshelf .
[+] Building 17.4s (10/10) FINISHED                                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                                          0.0s
 => => transferring dockerfile: 222B                                                                                          0.0s
 => [internal] load metadata for docker.io/library/python:alpine3.7                                                           0.1s
 => [internal] load .dockerignore                                                                                             0.0s
 => => transferring context: 2B                                                                                               0.0s
 => [internal] load build context                                                                                             0.1s
 => => transferring context: 85.32kB                                                                                          0.1s
 => CACHED [1/5] FROM docker.io/library/python:alpine3.7@sha256:35f6f83ab08f98c727dbefd53738e3b3174a48b4571ccb1910bae480dcdba847  0.0s
 => [2/5] COPY . /app                                                                                                         0.9s
 => [3/5] WORKDIR /app                                                                                                        0.0s
 => [4/5] RUN pip install --upgrade pip                                                                                       4.5s
 => [5/5] RUN pip install -r requirements.txt                                                                                10.5s
 => exporting to image                                                                                                        1.1s
 => => exporting layers                                                                                                       1.1s
 => => writing image sha256:b12f7abfa89245968bc2ffcc5656d3339ce7a463c040a676355076db5d8d1421                                  0.0s
 => => naming to docker.io/fevenbelay123225/bookshelf                                                                         0.0s
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$
```

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

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ docker push fevenbelay123225/bookshelf
Using default tag: latest
The push refers to repository [docker.io/fevenbelay123225/bookshelf]
c1be6ab8c18d: Pushed
4c3c2183f9ea: Pushed
5f70bf18a086: Pushed
ac83c14bba1e: 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:e83e4dfb2b1c9d9e093ab395eb04f699608df9c45b7b68e5196167d2427e2a89 size: 2208
```

6. Check the repositories in the dockerhub.



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

1. Create studentserver-configmapl file as follows
2. Create bookshelf-configmap file as follows

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ cat studentserver-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  MONGO_URL: mongodb://104.196.231.165
  MONGO_DATABASE: mydb
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ cat bookshelf-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: bookshelf-config
data:
  # SERVICE_NAME.NAMESPACE.svc.cluster.local:SERVICE_PORT
  MONGO_URL: mongodb://104.196.231.165
  MONGO_DATABASE: mydb
```

Step5 Expose 2 application using ingress with Nginx, so we can put them on the same Domain but different PATH

1. Create bookshelf-deployment.yaml file as follows

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ nano bookshelf-deployment.yaml
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ cat bookshelf-deployment.yaml
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: fevenbelay123225/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
```

2. Create studentserver-service.yaml file as follows

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ nano sutdentserver-service.yaml
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ cat sutdentserver-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: web
spec:
  type: LoadBalancer
  ports:
    # service port in cluster
    - port: 8080
      # port to contact inside container
      targetPort: 8080
  selector:
    app: web
```

3. Create bookshelf-service.yaml file as follows

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ nano bookshelf-service.yaml
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ cat bookshelf-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: bookshelf-service
spec:
  type: LoadBalancer
  ports:
    # service port in cluster
    - port: 5000
      # port to contact inside container
      targetPort: 5000
  selector:
    app: bookshelf-deployment
```

4. Create studentserver-configmap.yaml file as follows

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ cat studentserver-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  MONGO_URL: mongodb://104.196.231.165
  MONGO_DATABASE: mydb
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ cat  bookshelf-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: bookshelf-config
data:
  # SERVICE_NAME.NAMESPACE.svc.cluster.local:SERVICE_PORT
  MONGO_URL: mongodb://104.196.231.165
  MONGO_DATABASE: mydb
```

5. Start minikube

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ minikube start
* minikube v1.32.0 on Debian 11.9 (amd64)
  - MINIKUBE_FORCE_SYSTEMD=true
  - MINIKUBE_HOME=/google/minikube
  - MINIKUBE_WANTUPDATENOTIFICATION=false
* Automatically selected the docker driver. Other choices: none, ssh
* Using Docker driver with root privileges
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Downloading Kubernetes v1.28.3 preload ...
    > preloaded-images-k8s-v18-v1...:  403.35 MiB / 403.35 MiB  100.00% 64.24 M
    > gcr.io/k8s-minikube/kicbase...:  453.90 MiB / 453.90 MiB  100.00% 31.72 M
* Creating docker container (CPUs=2, Memory=4000MB) ...

X Docker is nearly out of disk space, which may cause deployments to fail! (95% 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=""
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Verifying Kubernetes components...
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

6. Start Ingress

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ 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/kube-webhook-certgen:v20231011-8b53cabe0
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20231011-8b53cabe0
  - Using image registry.k8s.io/ingress-nginx/controller:v1.9.4
* Verifying ingress addon...
* The 'ingress' addon is enabled
```

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

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ kubectl apply -f sutdentserver-service.yaml
service/web created
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ kubectl apply -f studentserver-deployment.yaml
deployment.apps/web created
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ kubectl apply -f studentserver-configmap.yaml
configmap/studentserver-config created
```

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

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
```

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

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS      AGE
bookshelf-deployment-8468fd7d44-z69tk   1/1     Running   7 (65m ago)   71m
web-685b4c68f5-jjqkt                  1/1     Running   0             32s
```

10. Create an ingress service yaml file called studentservermongoIngress.yaml

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ nano studentservermongoIngress.yaml
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ cat 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(/|$)(.*)
        pathType: Prefix
        backend:
          service:
            name: bookshelf-service
            port:
              number: 5000
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ 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
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$
```

11. Check if ingress is running kubectl get ingress

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ kubectl get ingress
NAME      CLASS    HOSTS                ADDRESS         PORTS   AGE
server    nginx    cs571.project.com    192.168.49.2    80      84s
fevenbelay123@cloudshell:~ (sfbu-cs571-414310)$ sudo vi /etc/hosts
```

12. Add Address to /etc/hosts vi /etc/hosts

```
# /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-84622221263-default

192.168.49.2 cs571.project.com
~
"/etc/hosts" [readonly] 36L, 1177B
```

13. If everything goes smoothly, you should be able to access your applications curl *cs571.project.com/studentserver/api/score?student_id=11111*

```
fevenbelay123@cloudshell : ~ (sfbu-cs571-414308)$ curl cs571.project.com/studentserver/api/score?student_id=11111
{"student_id": "11111", "student_name": "Bruce Lee", "grade" : "84" }
fevenbelay123@cloudshell :~ (sfbu-cs571-414308)$ curl cs571.project.com/studentserver/api/score?student_id=22222
{"student_id": "22222", "student_name": "Jackie Chan", "grade" : "93"}
fevenbelay123@cloudshell : ~ (sfbu-cs571-414308)$ curl cs571.project.com/studentserver/api/score?student_id=33333
{"student_id": "33333", "student_name": "Jet Li", "grade": "88"}
```

14. On another path, you should be able to use the REST API with bookshelf application I.e list all books curl cs571.project.com/bookshelf/books

```
fevenbelay123@cloudshell :~ (sfbu-cs571-414308)$ curl cs571.project.com/bookshelf/books
[
   {
      "Book Author": "test",
      "Book Name": "123",
      "ISBN": "123",
      "id" : "605dbda70f4d0fa5306551765"
   }
]
```

Add a book

- Use this command

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414308$ curl -X POST -d "{\"book_name\":\"Cloud
Computing\",\"book_author\":\"unknown\",\"isbn\":\"123456\"}" http://cs571.project.com/bookshelf/book
{
  "message": "Task saved successfully!"
}
```

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414308$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123",
    "id": "605db1a7d40f50a395651765"
  },
  {
    "Book Author": "unknown",
    "Book Name": "Cloud Computing",
    "ISBN": "123456",
    "id": "605d2ffbd09c0d7f8cf1f93"
  }
]
```

Update a book

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414308)$ curl -X PUT -d
"{\"book_name\":\"123\",\"book_author\":\"test\",\"isbn\":\"123updated\"}" http://cs571.project.com/bookshelf/book/id
{
  "message": "Task updated successfully!"
}
```

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414308)$ curl cs571.project.com/bookshelf/books
[
 {
   "Book Author": "test",
   "Book Name": "123",
   "ISBN": "123updated",
   "id": "605d1ba7d40f50a395651765"
 },
 {
   "Book Author": "unknown",
   "Book Name": "cloud computing",
   "ISBN": "123456",
   "id": "605d2ffbd09c0d7f8cf1f93"
 }
]
```

Delete a Book

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414308)$ curl -X DELETE
cs571.project.com/bookshelf/book/605d1ba7d40f50a395651765
{
  "message": "Task deleted successfully!"
}
```

```
fevenbelay123@cloudshell:~ (sfbu-cs571-414308)$ curl cs571.project.com/bookshelf/books
[
 {
   "Book Author": "unknown",
   "Book Name": "cloud computing",
   "ISBN": "123456",
   "id": "605d2ffbd09c0d7f8cf1f93"
 }
]
```