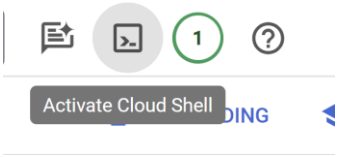
CS571 Signature Project: MongoDB + Python Flask Web Framework + REST API + GKE

Student Name: Xin Wen

Student ID: 19876

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

1. Enable Kubernetes Engine API and Active Google Cloud Shell



create a kubernetes cluster with three nodes:

gcloud container clusters create kubia --num-nodes=3 --machine-type=e2-micro --zone=us-west1-b

```
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ gcloud container clusters create kubia --num-nodes=3 --machine-type=e2-micro --zone=us-west1-b
Default change: VPC-native is the default mode during cluster creation for versions greater than 1.21.0-gke.1500. To create advanced routes based clusters
, please pass the `--no-enable-ip-alias` flag
Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most 1008 node(s).
Creating cluster kubia in us-west1-b... Cluster is being health-checked (master is healthy)...done.
Created [https://container.googleapis.com/v1/projects/signature-fullstack-project/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=signature-fullstack-
project
kubeconfig entry generated for kubia.
NAME: kubia
LOCATION: us-west1-b
MASTER_VERSION: 1.27.8-gke.1067004
MASTER_IP: 34.83.117.142
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.27.8-gke.1067004
NUM_NODES: 3
STATUS: RUNNING
```

2. Let's create a Persistent Volume first,
   gcloud compute disks create --size=10GiB --zone=us-west1-b mongodb

```
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ gcloud compute disks create --size=10GiB --zone=us-west1-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.google.com
/compute/docs/disks#performance.
Created [https://www.googleapis.com/compute/v1/projects/signature-fullstack-project/zones/us-west1-b/disks/mongodb].
NAME: mongodb
ZONE: us-west1-b
SIZE_GB: 10
TYPE: pd-standard
STATUS: READY
```

3. Now create a mongodb deployment with this yaml file:
   $ vim mongodb-deployment.yaml
   $ kubectl apply -f mongodb-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
spec:
  selector:
    matchLabels:
      app: mongodb
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
        - name: mongo
          image: mongo
          ports:
            - containerPort: 27017
          volumeMounts:
            - name: mongodb-data
              mountPath: /data/db
      volumes:
        - name: mongodb-data
          gcePersistentDisk:
            pdName: mongodb
            fsType: ext4
```

```
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
```

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

```
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
mongodb-deployment-594c77dcdf-rm5c2   1/1     Running   0          2m37s
```

5. Create a service for the mongodb, so it can be accessed from the outside
   $ vim mongodb-service.yaml
   $ kubectl apply -f mongodb-service.yaml

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

```
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ vim mongodb-service.yaml
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
```

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

```
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ kubectl get svc
NAME              TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)           AGE
kubernetes        ClusterIP      10.95.144.1     <none>           443/TCP           136m
mongodb-service   LoadBalancer   10.95.159.159   34.105.112.198   27017:31440/TCP   3m55s
```

7. Now try and see if MongoDB is functioning for connections using External-IP
   $ kubectl exec -it mongodb-deployment-replace-with-your-pod-name -- bash

   $ kubectl exec -it mongodb-deployment-594c77dcdf-rm5c2 -- bash

   Now you are inside the mongodb deployment pod

```
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ kubectl exec -it mongodb-deployment-594c77dcdf-rm5c2 -- bash
root@mongodb-deployment-594c77dcdf-rm5c2:/#
```

   Try

   $ mongosh External-IP

   You should see something like this, which means your MongoDB is up and can be accessed using the External-IP

```
root@mongodb-deployment-594c77dcdf-rm5c2:/# mongosh 34.105.112.198
Current Mongosh Log ID: 66047cedacf77cb443db83af
Connecting to:          mongodb://34.105.112.198:27017/?directConnection=true&appName=mongosh+2.2.0
Using MongoDB:          7.0.7
Using Mongosh:          2.2.0

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
   2024-03-27T18:51:18.297+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/
prodnotes-filesystem
   2024-03-27T18:51:19.094+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
   2024-03-27T18:51:19.095+00:00: vm.max_map_count is too low
------

test>
```

8. Type exit to exit mongodb and back to our google console

```
test> exit
root@mongodb-deployment-594c77dcdf-rm5c2:/# exit
exit
wxhtd1220@cloudshell:~ (signature-fullstack-project)$
```

9. We need to insert some records into the MongoDB for later use:
   $ node

```
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ node
Welcome to Node.js v20.11.1.
Type ".help" for more information.
>
```

Enter the following line by line:

```
const { MongoClient } = require('mongodb');
const url = 'mongodb://34.105.112.198/mydb'; // Replace with your actual external IP

async function run() {
  try {
    const client = await MongoClient.connect(url);
    console.log("Connected correctly to server");
    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} documents were inserted`);

    const findResult = await db.collection("students").findOne({ "student_id": 11111
    console.log(findResult);

    client.close();
  } catch (err) {
    console.error(err);
  }
}

run();
```

If everything is correct, you should see this:

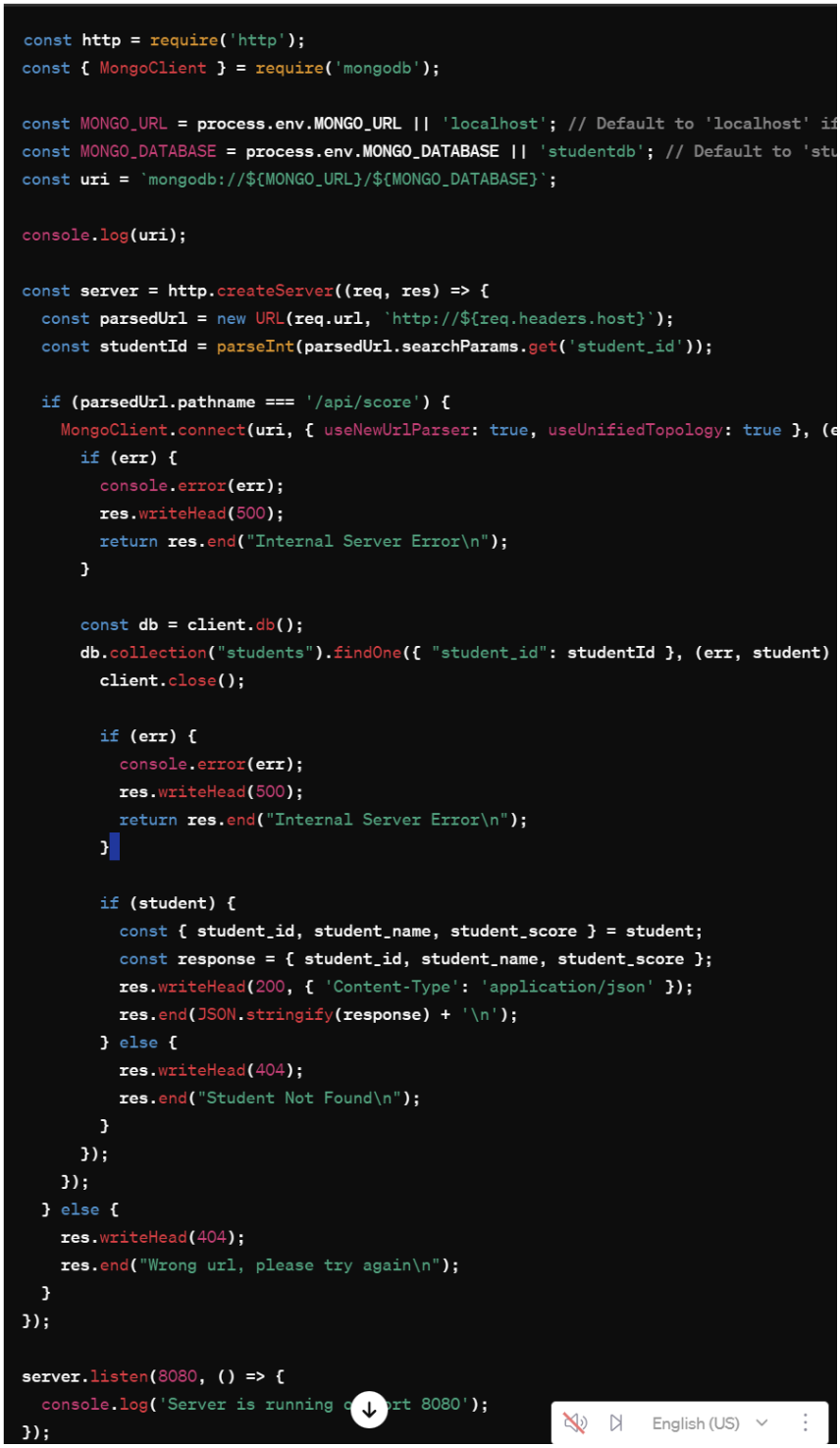
3 means three records was inserted, and we tried search for student_id=11111

```
...
...     const insertResult = await db.collection("students").insertMany(docs);
...     console.log(`${insertResult.insertedCount} documents were inserted`);
...
...     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)]: 1545,
  [Symbol(trigger_async_id_symbol)]: 6
}
> Connected correctly to server
3 documents were inserted
{
  _id: new ObjectId('66049326a4f4f9c7ad73e464'),
  student_id: 11111,
  student_name: 'Bruce Lee',
  grade: 84
}
```

.

Step2: Modify our student Server to get records from MongoDB and deploy to GKE

1. Create a studentServer.js
   $ vim studentServer.js



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

const MONGO_URL = process.env.MONGO_URL || 'localhost'; // Default to 'localhost' if
const MONGO_DATABASE = process.env.MONGO_DATABASE || 'studentdb'; // Default to 'stu
const uri = `mongodb://${MONGO_URL}/${MONGO_DATABASE}`;

console.log(uri);

const server = http.createServer((req, res) => {
  const parsedUrl = new URL(req.url, `http://${req.headers.host}`);
  const studentId = parseInt(parsedUrl.searchParams.get('student_id'));

  if (parsedUrl.pathname === '/api/score') {
    MongoClient.connect(uri, { useNewUrlParser: true, useUnifiedTopology: true }, (e
      if (err) {
        console.error(err);
        res.writeHead(500);
        return res.end("Internal Server Error\n");
      }

      const db = client.db();
      db.collection("students").findOne({ "student_id": studentId }, (err, student)
        client.close();

        if (err) {
          console.error(err);
          res.writeHead(500);
          return res.end("Internal Server Error\n");
        }

        if (student) {
          const { student_id, student_name, student_score } = student;
          const response = { student_id, student_name, student_score };
          res.writeHead(200, { 'Content-Type': 'application/json' });
          res.end(JSON.stringify(response) + '\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, () => {
  console.log('Server is running o rt 8080');
});
```

2. Create a Dockerfile

   $ vim Dockerfile

```
FROM node:16
ADD studentServer.js /studentServer.js
WORKDIR /usr/src/app
ENTRYPOINT ["node", "studentServer.js"]
RUN npm install mongodb
```

3. Build the studentserver docker image:

   Docker login first:

```
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ docker login
Authenticating with existing credentials...
WARNING! Your password will be stored unencrypted in /home/wxhtd1220/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

   $ docker build -t wxhtd1220/studentserver

```
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ docker build -t wxhtd1220/studentserver .
[+] Building 29.9s (9/9) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 177B
 => [internal] load metadata for docker.io/library/node:16
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/4] FROM docker.io/library/node:16@sha256:f77a1aef2da8d83e45ec990f45df50f1a286c5fe8bbfb8c6e4246c6389705c0b
 => => resolve docker.io/library/node:16@sha256:f77a1aef2da8d83e45ec990f45df50f1a286c5fe8bbfb8c6e4246c6389705c0b
 => => sha256:c94b82f9827cab6e421b350965a9ef11b25b13ffbd1030536203d541f55dcbe2 2.00kB / 2.00kB
 => => sha256:311da66c465ea1576925360eba391bcd32dece9be95960a0bc9ffcb25fe712017 50.50MB / 50.50MB
 => => sha256:7e9bf1145880c05b2df612b083b96593f3b8dbf51647aa6138a50d09d42df2454 17.58MB / 17.58MB
 => => extracting sha256:ca266fd6192108b67fb57b74753a8c4ca5d8bd458baae3d4df7ce9f42dedcc1d
 => => extracting sha256:ee7d78be1eb92caf6ae84fc3af736b23eca018d5dedc967ae5bdee6d7082403b
 => [internal] load build context
 => => transferring context: 1.73kB
 => [2/4] ADD studentServer.js /studentServer.js
 => [3/4] WORKDIR /usr/src/app
 => [4/4] RUN npm install mongodb
 => exporting to image
 => => exporting layers
 => => writing image sha256:a6fffe980d868943e8ade0117a0cb0d39bfb1be3dc4a78f0196c5e8877d2bb06
```

   Make sure the image create successfully:

   $ docker images

```
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ docker images
REPOSITORY                TAG       IMAGE ID       CREATED          SIZE
wxhtd1220/studentserver   latest    a6fffe980d86   2 minutes ago    921MB
```

4. Push the docker image to dockerhub:

   $ docker push wxhtd1220/studentserver

```
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ docker push wxhtd1220/studentserver
Using default tag: latest
The push refers to repository [docker.io/wxhtd1220/studentserver]
7c17987554d8: Pushed
484f21ad76ee: Pushed
d8d9905d8cdd: Pushed
be322b479aee: Layer already exists
d41bcd3a037b: Layer already exists
fe0d845e767b: Layer already exists
f25ec1d93a58: Layer already exists
794ce8b1b516: Layer already exists
3220beed9b06: Layer already exists
684f82921421: Layer already exists
9af5f53e8f62: Layer already exists
latest: digest: sha256:c5e67cc593b32e1469717ceca9004c9ed5c107dd08437d4deeb36b96885fbdf0 size: 2629
```

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

1. Created bookshelf.py

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

app = Flask(__name__)
app.config["MONGO_URI"] = "mongodb://" + os.getenv("MONGO_URL", "localhost") + "/" +
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!".fo

@app.route("/books")
def get_all_books():
    books = db.bookshelf.find()
    data = []
    for book in books:
        data.append({
            "id": str(book["_id"]),
            "Book Name": book["book_name"],
```

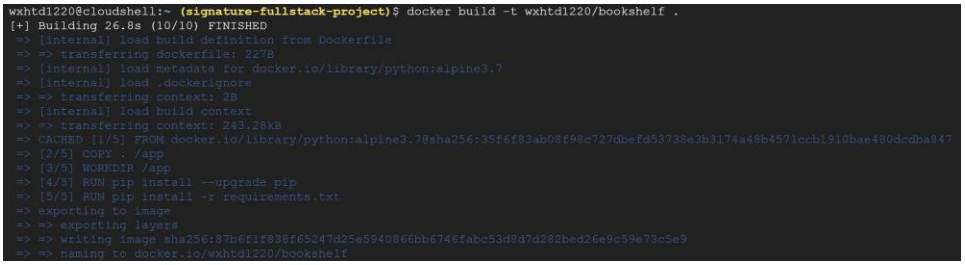
```
            "Book Author": book["book_author"],
            "ISBN": book["ISBN"]
        })
    return jsonify(data)

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

@app.route("/book/<id>", methods=["PUT"])
def update_book(id):
    data = request.get_json(force=True)
    response = db.bookshelf.update_one({"_id": ObjectId(id)}, {"$set": {
        "book_name": data['book_name'],
        "book_author": data["book_author"], "ISBN": data["isbn"]
    }})
    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=5000)
```

2. Create a requirements.txt file and a Dockerfile:

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

```
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. Build the bookshelf app into a docker image
   $ docker build -t wxhtd1220/bookshelf
   Make sure this step build successfully




```
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ docker images
REPOSITORY                TAG       IMAGE ID       CREATED             SIZE
wxhtd1220/bookshelf       latest    87b6f1f838f6   7 minutes ago       406MB
wxhtd1220/studentserver   latest    a6fffe980d86   About an hour ago   921MB
```

4. Push the docker image to your dockerhub
   $ docker push wxhtd1220/bookshelf

```
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ docker push wxhtd1220/bookshelf
Using default tag: latest
The push refers to repository [docker.io/wxhtd1220/bookshelf]
fea9dab8764b: Pushed
36213c0fe716: Pushed
5f70bf18a086: Pushed
157ca30a5e8a: 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: sha256:7b89a7a1c8073d53d0d1689e27b6b64cc83dc60b5e751fc3b125c3d28c72e07e size: 2209
```

Step4 Create ConfigMap for both applications to store MongoDB URL and MongoDB name

1. Create a file named studentserver-configmap.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  MONGO_URL: 34.105.112.198
  MONGO_DATABASE: mydb
~
```

2. Create a file named bookshelf-configmap.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: bookshelf-config
data:
  MONGO_URL: 34.105.112.198
  MONGO_DATABASE: mydb
~
```

In MONGO_URL section, you need to change it to your own External_IP address

Notice: the reason of creating those two ConfigMap is to avoid re-building docker image again if the mongoDB pod restarts with a different External-IP

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

1. Create studentserver-deployment.yaml

```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: wxhtd1220/studentserver
          imagePullPolicy: Always
          name: web
          ports:
            - containerPort: 8080
          env:
            - name: MONGO_URL
              valueFrom:
                configMapKeyRef:
                  name: studentserver-config
                  key: MONGO_URL
            - name: MONGO_DATABASE
              valueFrom:
                configMapKeyRef:
                  name: studentserver-config
                  key: MONGO_DATABASE
```

2. Create bookshelf-deployment.yaml

```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: wxhtd1220/bookshelf
          imagePullPolicy: Always
          name: bookshelf-deployment
          ports:
            - containerPort: 5000
          env:
            - name: MONGO_URL
              valueFrom:
                configMapKeyRef:
                  name: bookshelf-config
                  key: MONGO_URL
            - name: MONGO_DATABASE
              valueFrom:
                configMapKeyRef:
                  name: bookshelf-config
                  key: MONGO_DATABASE
```

3. Create a studentserver-service.yaml

```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 a bookshelf-service.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: bookshelf-service
spec:
  type: LoadBalancer
  ports:
      # service port in cluster
    - port: 5000
      # port to conncect inside container
      targetPort: 5000
  selector:
    app: bookshelf-deployment
~
```

5. Start minikube

   $ minikube start

```
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ 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% 183.20
    > gcr.io/k8s-minikube/kicbase...:  453.90 MiB / 453.90 MiB  100.00% 100.18
* Creating docker container (CPUs=2, Memory=4000MB) ...

X Docker is nearly out of disk space, which may cause deployments to fail! (94% 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
   $ minikube addons enable ingress

```
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ minikube addons enable ingress
* ingress is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
  - Using image registry.k8s.io/ingress-nginx/controller:v1.9.4
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20231011-8b53cabe0
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20231011-8b53cabe0
* Verifying ingress addon...
* The 'ingress' addon is enabled
```

7. Create 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

```
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ kubectl apply -f studentserver-deployment.yaml
deployment.apps/web created
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ kubectl apply -f studentserver-configmap.yaml
configmap/studentserver-config created
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ kubectl apply -f studentserver-service.yaml
service/web created
```

8. Create 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

```
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
```

9. Check if all the pods are running correctly
   $ kubectl get pods
10. Create an ingress service yaml file called studentservermongoIngress.yaml

```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
            backyend:
              service:
                name: web
                port:
                  number: 8080
          - path: /bookshelf(/|$)(.*)
            pathType: Prefix
            backend:
              service:
                name: bookshelf-service
                port:
                  number: 5000
```

11. Create the ingress service using the above yaml file
    $ kubectl apply -f ../studentservermongoIngress.yaml

```
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ 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
```

12. Check if ingress is running:
    $ kubectl get ingress

```
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ kubectl get ingress
NAME     CLASS   HOSTS               ADDRESS        PORTS   AGE
server   nginx   cs571.project.com   192.168.49.2   80      59s
```

13. Add Address to /etc/host
    $ sudo vi etc/host
    Add the address you got in the last step in the file

```
#       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-289653573644-default
192.168.49.2 cs571.project.com
~
```

14. If everthing goes smoothly, you should be able to access your application:
    $ curl cs571.project.com/studentserver/api/score?student_id=11111

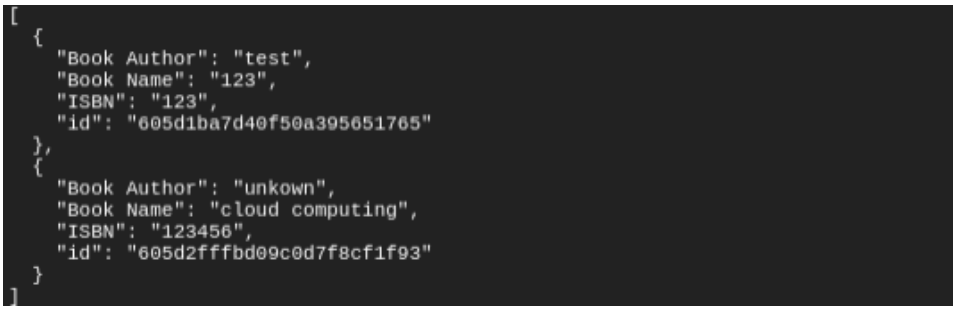
```
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ curl cs571.project.com/studentserver/api/score?student_id=11111
{"_id":"605a6b49c3a15527de9d0f9b","student_id":11111,"student_name":"Bruce Lee","grade":84}
```

    $ curl cs571.project.com/bookshelf/bookscurl

```
wxhtd1220@cloudshell:~ (signature-fullstack-project)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123",
    "id": "605d1ba7d40f50a395651765"
  }
]
```

15. Add a book:
    $ curl -X POST -d "{\"book_name\": \"cloud computing\",\"book_author\": \"unkown\", \"isbn\": \"123456\" }" http://cs571.project.com/bookshelf/book

$ curl cs571.project.com/bookshelf/books



Update a book

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

Delete a book

$ curl -X DELETE cs571.project.com/bookshelf/book/6052fffbd09c0d7f8cf1f93