🗽

Signature Project

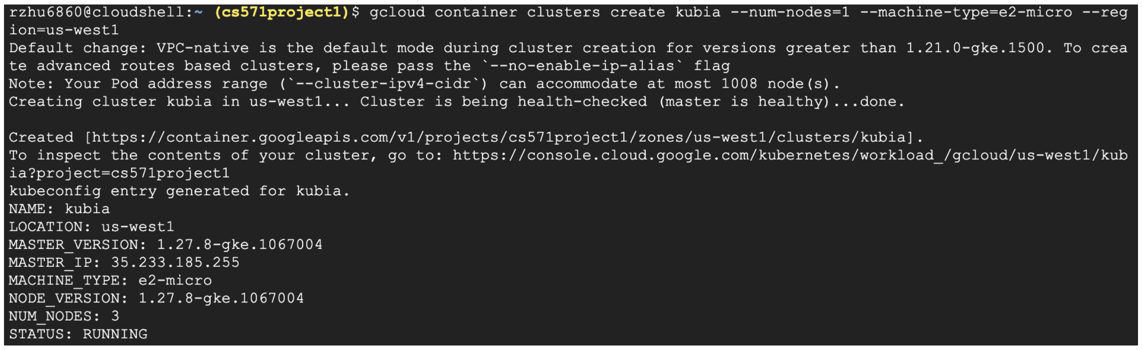
Student ID: 19811

Student Name: Ruichen Zhu

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

1. Create a cluster as usual on GKE

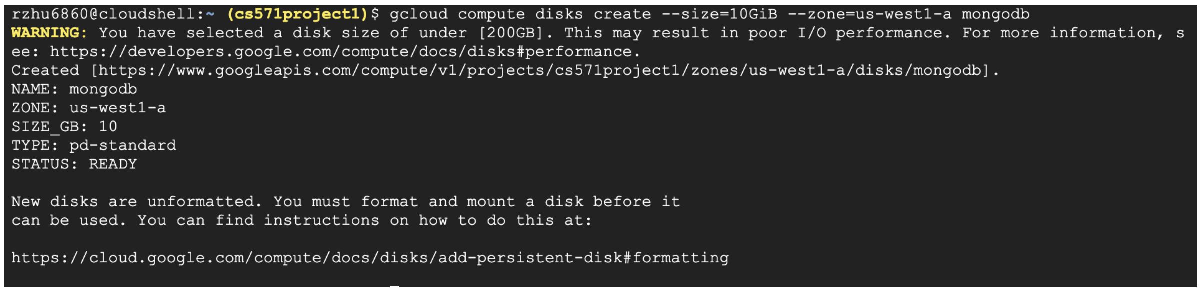
```
gcloud container clusters create kubia --num-nodes=1 --machine-
--region=us-west1
```



```
rzhu6860@cloudshell:~ (cs571project1)$ gcloud container clusters create kubia --num-nodes=1 --machine-type=e2-micro --reg
ion=us-west1
Default change: VPC-native is the default mode during cluster creation for versions greater than 1.21.0-gke.1500. To crea
te advanced routes based clusters, please pass the `--no-enable-ip-alias` flag
Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most 1008 node(s).
Creating cluster kubia in us-west1... Cluster is being health-checked (master is healthy)...done.

Created [https://container.googleapis.com/v1/projects/cs571project1/zones/us-west1/clusters/kubia].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/us-west1/kub
ia?project=cs571project1
kubeconfig entry generated for kubia.
NAME: kubia
LOCATION: us-west1
MASTER_VERSION: 1.27.8-gke.1067004
MASTER_IP: 35.233.185.255
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.27.8-gke.1067004
NUM_NODES: 3
STATUS: RUNNING
```

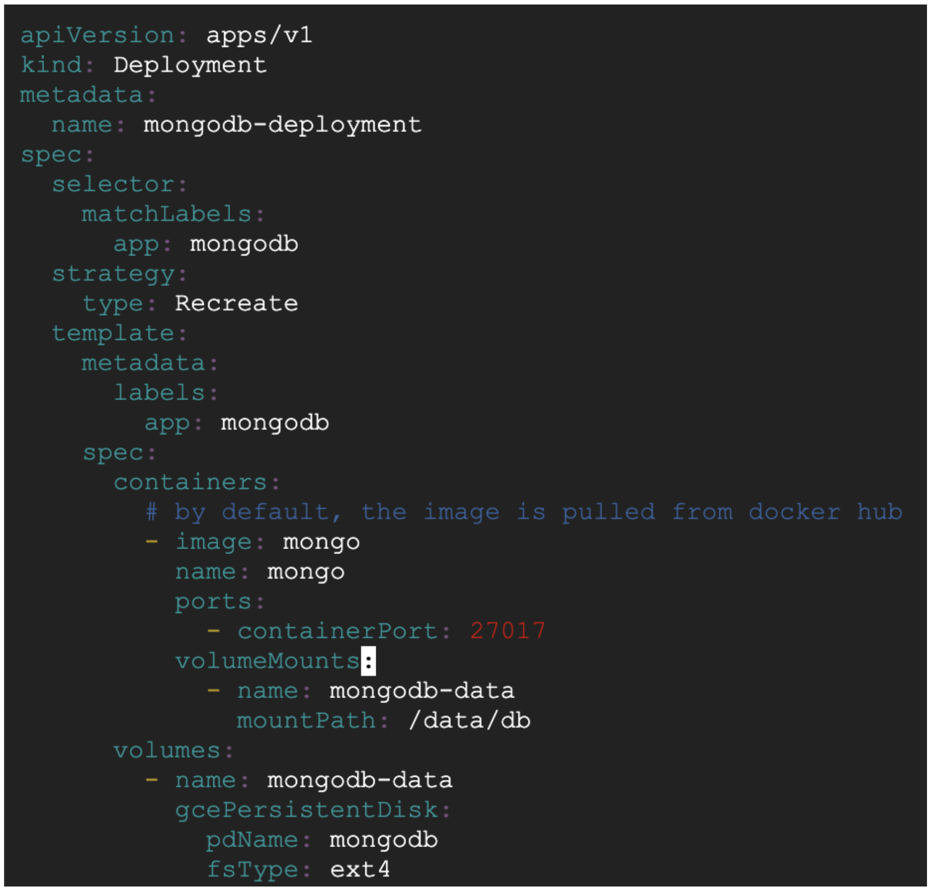
2. Let's create a Persistent Volume first

```
gcloud compute disks create --size=10GiB --zone=us-west1-a mong
```



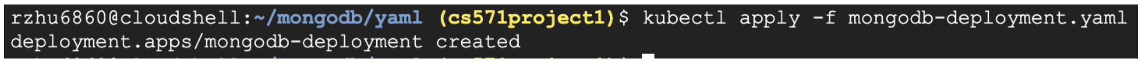
```
rzhu6860@cloudshell:~ (cs571project1)$ gcloud compute disks create --size=10GiB --zone=us-west1-a mongodb
WARNING: You have selected a disk size of under [200GB]. This may result in poor I/O performance. For more information, s
ee: https://developers.google.com/compute/docs/disks#performance.
Created [https://www.googleapis.com/compute/v1/projects/cs571project1/zones/us-west1-a/disks/mongodb].
NAME: mongodb
ZONE: us-west1-a
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



```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:
        # by default, the image is pulled from docker hub
        - image: mongo
          name: mongo
          ports:
            - containerPort: 27017
          volumeMounts:
            - name: mongodb-data
              mountPath: /data/db
      volumes:
        - name: mongodb-data
          gcePersistentDisk:
            pdName: mongodb
            fsType: ext4
```

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

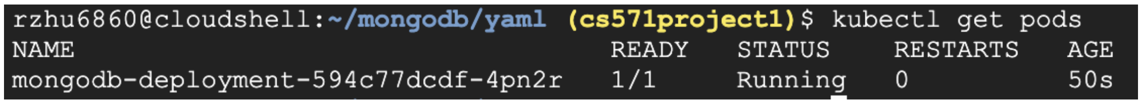


```
rzhu6860@cloudshell:~/mongodb/yaml (cs571project1)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
```

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

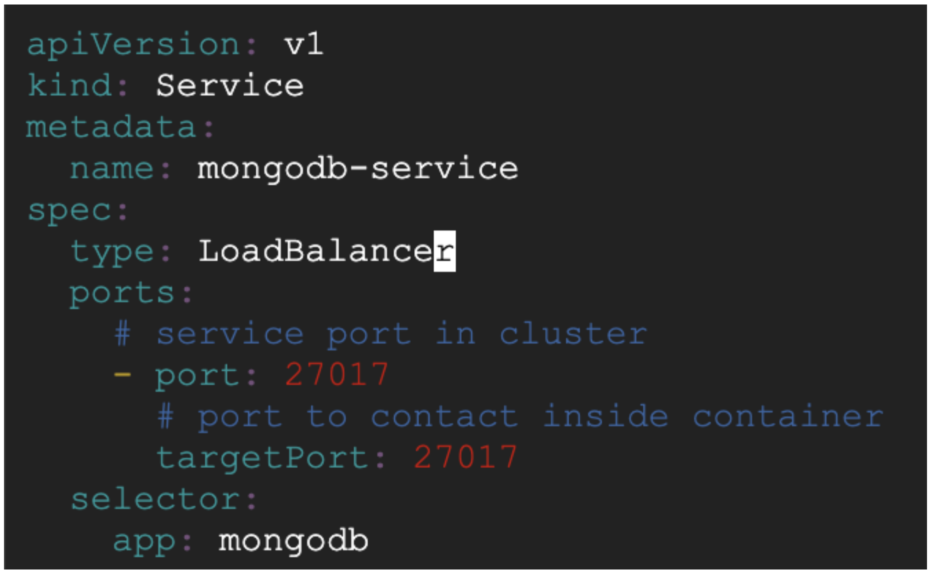
```
kubectl get pods
```

Please wait until you see the STATUS is running, then you can move forward.



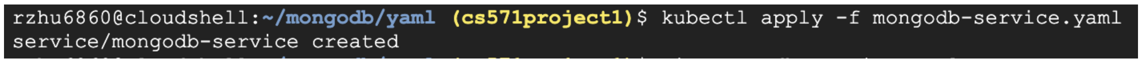
```
rzhu6860@cloudshell:~/mongodb/yaml (cs571project1)$ kubectl get pods
NAME                                 READY   STATUS    RESTARTS   AGE
mongodb-deployment-594c77dcdf-4pn2r  1/1     Running   0          50s
```

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



```
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  type: LoadBalancer
  ports:
    # service port in cluster
    - port: 27017
      # port to contact inside container
      targetPort: 27017
  selector:
    app: mongodb
```

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

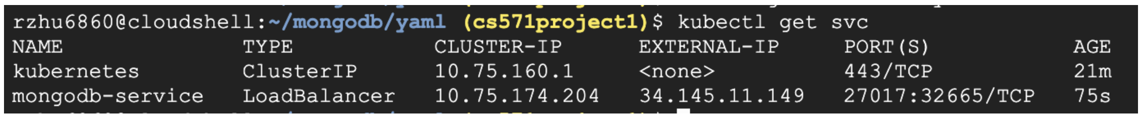


```
rzhu6860@cloudshell:~/mongodb/yaml (cs571project1)$ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
```

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

```
kubectl get svc
```

Please wait until you see the external-ip is generated for mongodb-service, then you can move forward.

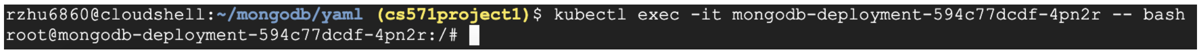


```
rzhu6860@cloudshell:~/mongodb/yaml (cs571project1)$ kubectl get svc
NAME             TYPE          CLUSTER-IP     EXTERNAL-IP    PORT(S)          AGE
kubernetes       ClusterIP     10.75.160.1    <none>         443/TCP          21m
mongodb-service  LoadBalancer  10.75.174.204  34.145.11.149  27017:32665/TCP  75s
```

7. Now try and see if mongoDB is functioning for connections using the External-IP

   ```
   kubectl exec -it mongodb-deployment-replace-with-your-pod-name
   ```

   Now you are inside the mongodb deployment pod:

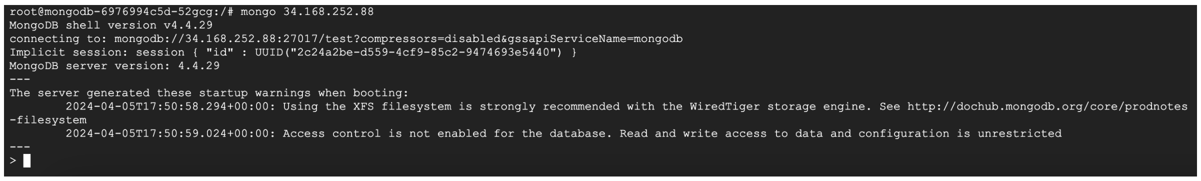
   

   ```
   rzhu6860@cloudshell:~/mongodb/yaml (cs571project1)$ kubectl exec -it mongodb-deployment-594c77dcdf-4pn2r -- bash
   root@mongodb-deployment-594c77dcdf-4pn2r:/#
   ```

   Try:

   ```
   mongo External-IP
   ```

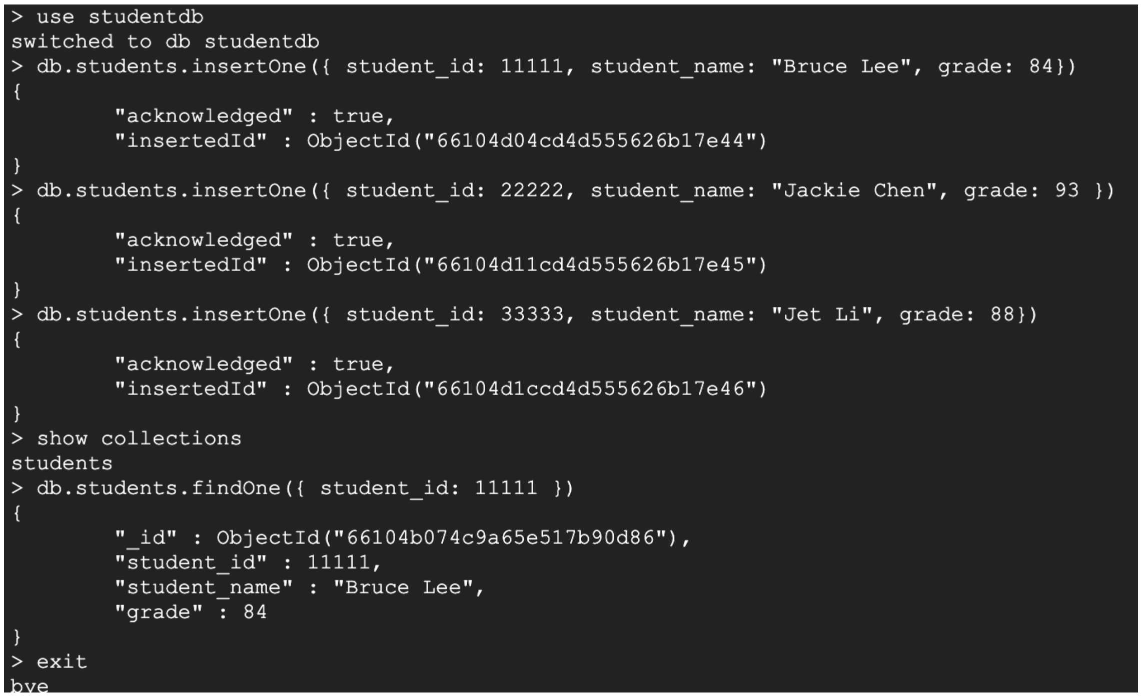
   You should see something like this, which means your mongoDB is up and can be accessed using the External-IP

   ```
   root@mongodb-6976994c5d-52gcg:/# mongo 34.168.252.88
   MongoDB shell version v4.4.29
   connecting to: mongodb://34.168.252.88:27017/test?compressors=disabled&gssapiServiceName=mongodb
   Implicit session: session { "id" : UUID("2c24a2be-d559-4cf9-85c2-9474693e5440") }
   MongoDB server version: 4.4.29
   ---
   The server generated these startup warnings when booting:
           2024-04-05T17:50:58.294+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes
   -filesystem
           2024-04-05T17:50:59.024+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
   ---
   >
   ```

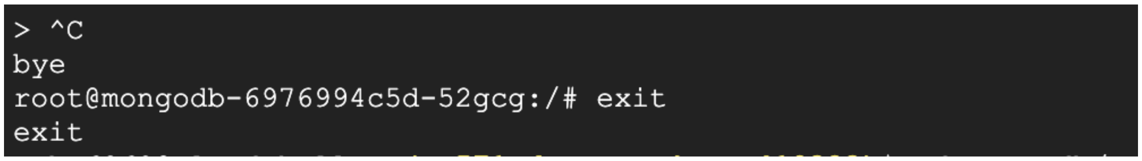
8. We need to insert some records into the mongoDB for later use

   Enter the following line by line



```
> use studentdb
switched to db studentdb
> db.students.insertOne({ student_id: 11111, student_name: "Bruce Lee", grade: 84})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("66104d04cd4d555626b17e44")
}
> db.students.insertOne({ student_id: 22222, student_name: "Jackie Chen", grade: 93 })
{
        "acknowledged" : true,
        "insertedId" : ObjectId("66104d11cd4d555626b17e45")
}
> db.students.insertOne({ student_id: 33333, student_name: "Jet Li", grade: 88})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("66104d1ccd4d555626b17e46")
}
> show collections
students
> db.students.findOne({ student_id: 11111 })
{
        "_id" : ObjectId("66104b074c9a65e517b90d86"),
        "student_id" : 11111,
        "student_name" : "Bruce Lee",
        "grade" : 84
}
> exit
bye
```

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



```
> ^C
bye
root@mongodb-6976994c5d-52gcg:/# exit
exit
```

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;

var MongoClient = mongodb.MongoClient;
var uri = `mongodb://${MONGO_URL}/${MONGO_DATABASE}`;
console.log(uri);
```

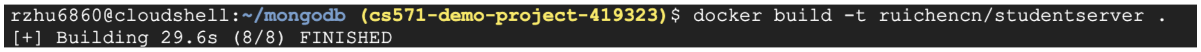
```
var server = http.createServer(function (req, res) {
  var result;
  // req.url = /api/score?student_id=11111
  var parsedUrl = url.parse(req.url, true);
  var student_id = parseInt(parsedUrl.query.student_id);
  // match req.url with the string /api/score
  if (/^\/api\/score/.test(req.url)) {
    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
              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);
```

2. Create Dockerfile

```
FROM node:14
ADD studentServer.js /studentServer.js
ENTRYPOINT ["node", "studentServer.js"]
RUN npm install mongodb@4.5.0
```

3. Build the studentserver docker image

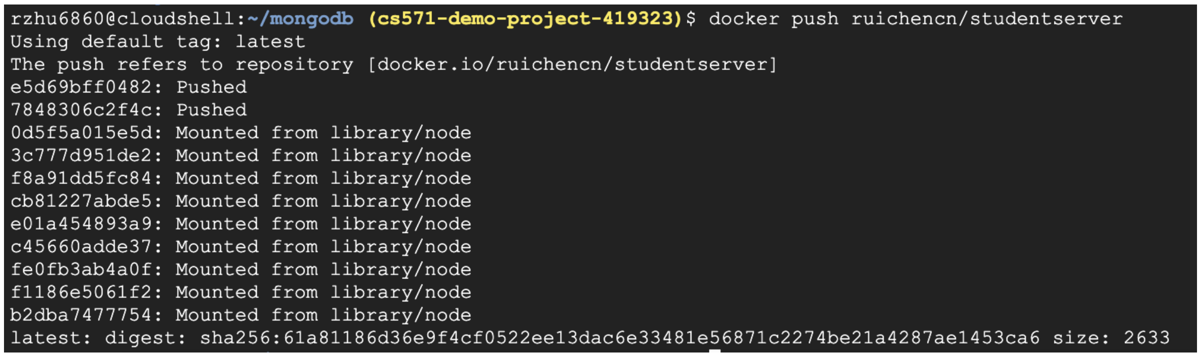
```
docker build -t yourdockerhubID/studentserver .
```



```
rzhu6860@cloudshell:~/mongodb (cs571-demo-project-419323)$ docker build -t ruichencn/studentserver .
[+] Building 29.6s (8/8) FINISHED
```

4. Push the docker image

```
docker push yourdockerhubID/studentserver
```



```
rzhu6860@cloudshell:~/mongodb (cs571-demo-project-419323)$ docker push ruichencn/studentserver
Using default tag: latest
The push refers to repository [docker.io/ruichencn/studentserver]
e5d69bff0482: Pushed
7848306c2f4c: Pushed
0d5f5a015e5d: Mounted from library/node
3c777d951de2: Mounted from library/node
f8a91dd5fc84: Mounted from library/node
cb81227abde5: Mounted from library/node
e01a454893a9: Mounted from library/node
c45660adde37: Mounted from library/node
fe0fb3ab4a0f: Mounted from library/node
f1186e5061f2: Mounted from library/node
b2dba7477754: Mounted from library/node
latest: digest: sha256:61a81186d36e9f4cf0522ee13dac6e33481e56871c2274be21a4287ae1453ca6 size: 2633
```

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")+"
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 runn

@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["

    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"],"bo
    return jsonify(message="Task saved successfully!")

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

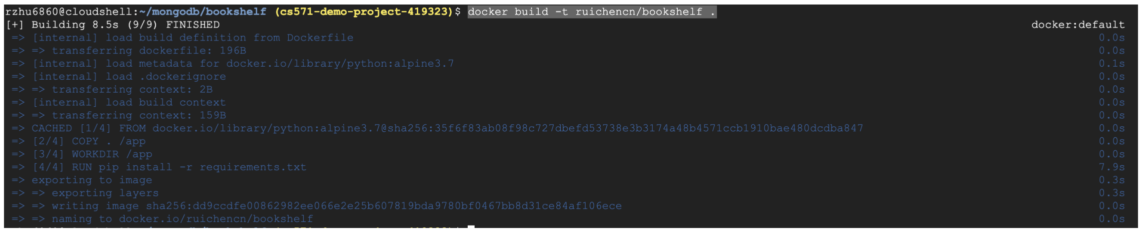
```
@app.route("/book/<id>", methods=["DELETE"])
def delete_task(id):
    response = db.bookshelf.delete_one({"_id": ObjectId(id)})
    if response.deleted_count:
        message = "Task deleted successfully!"
    else:
        message = "No book found!"
    return jsonify(message=message)

@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

```
FROM python:alpine3.7
COPY . /app
WORKDIR /app
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

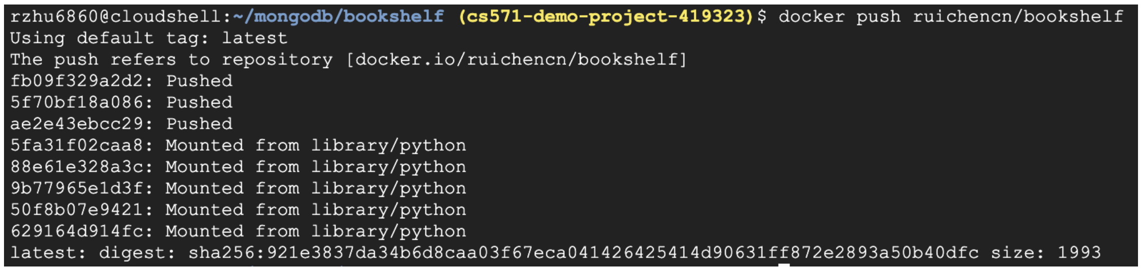
```
docker build -t ruichencn/bookshelf .
```



```
rzhu6860@cloudshell:~/mongodb/bookshelf (cs571-demo-project-419323)$ docker build -t ruichencn/bookshelf .
[+] Building 8.5s (9/9) FINISHED                                                                                   docker:default
 => [internal] load build definition from Dockerfile                                                                         0.0s
 => => transferring dockerfile: 196B                                                                                         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                                                                                            0.0s
 => => transferring context: 159B                                                                                            0.0s
 => CACHED [1/4] FROM docker.io/library/python:alpine3.7@sha256:35f6f83ab08f98c727dbefd53738e3b3174a48b4571ccb1910bae480dcdba847  0.0s
 => [2/4] COPY . /app                                                                                                        0.0s
 => [3/4] WORKDIR /app                                                                                                       0.0s
 => [4/4] RUN pip install -r requirements.txt                                                                                7.9s
 => exporting to image                                                                                                       0.3s
 => => exporting layers                                                                                                      0.3s
 => => writing image sha256:dd9ccdfe00862982ee066e2e25b607819bda9780bf0467bb8d31ce84af106ece                                 0.0s
 => => naming to docker.io/ruichencn/bookshelf                                                                               0.0s
```

4. Push the docker image to your dockerhub

```
docker push yourdockerhubID/bookshelf
```



```
rzhu6860@cloudshell:~/mongodb/bookshelf (cs571-demo-project-419323)$ docker push ruichencn/bookshelf
Using default tag: latest
The push refers to repository [docker.io/ruichencn/bookshelf]
fb09f329a2d2: Pushed
5f70bf18a086: Pushed
ae2e43ebcc29: 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: sha256:921e3837da34b6d8caa03f67eca041426425414d90631ff872e2893a50b40dfc size: 1993
```

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

1. Create a file named studentserver-configmap.yaml

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  MONGO_URL: 34.168.252.88
  MONGO_DATABASE: studentdb
```

2. Create a file named bookshelf-configmap.yaml

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: bookshelf-config
```

```yaml
data:
  # SERVICE_NAME.NAMESPACE.svc.cluster.local:SERVICE_PORT
  MONGO_URL: 34.168.252.88
  MONGO_DATABASE: studentdb
```

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: ruichencn/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

```
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: ruichencn/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 sutdentserver-service.yaml

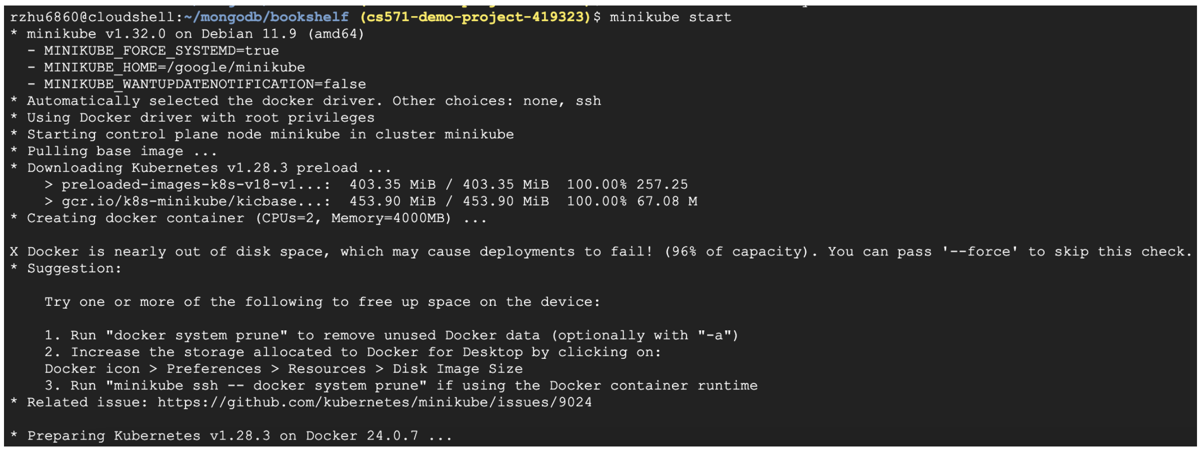
```
apiVersion: v1
kind: Service
metadata:
  name: web
spec:
  type: LoadBalancer
  ports:
    # service port in cluster
    - port: 8080
      # port to contact inside container
      targetPort: 8080
  selector:
    app: web
```

4. Create 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
```

5. Start minikube

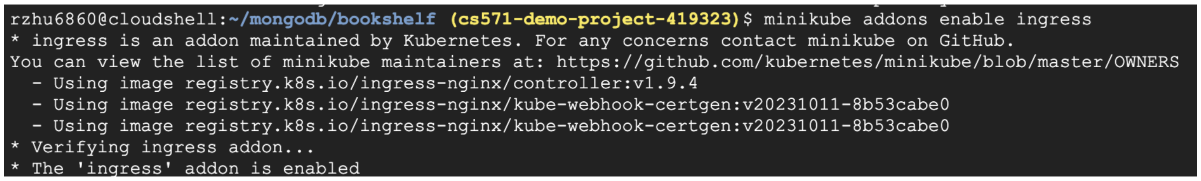
```
minikube start
```



```
rzhu6860@cloudshell:~/mongodb/bookshelf (cs571-demo-project-419323)$ 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% 257.25
    > gcr.io/k8s-minikube/kicbase...:  453.90 MiB / 453.90 MiB  100.00% 67.08 M
* Creating docker container (CPUs=2, Memory=4000MB) ...

X Docker is nearly out of disk space, which may cause deployments to fail! (96% 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 ...
```

6. Start Ingress

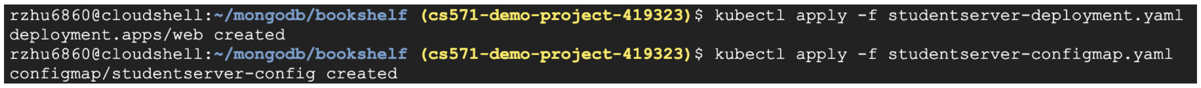
```
minikube addons enable ingress
```



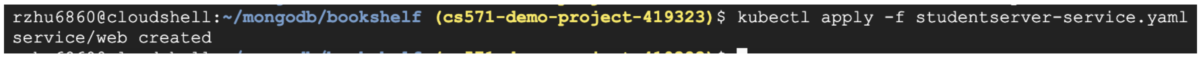
```
rzhu6860@cloudshell:~/mongodb/bookshelf (cs571-demo-project-419323)$ minikube addons enable ingress
* ingress is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
  - 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 file

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



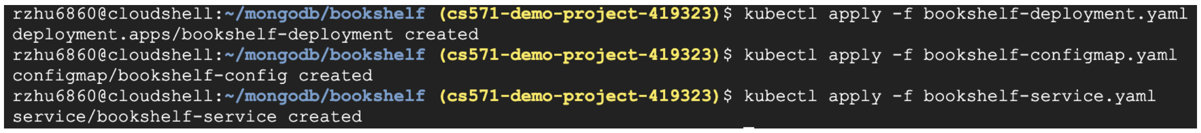
```
rzhu6860@cloudshell:~/mongodb/bookshelf (cs571-demo-project-419323)$ kubectl apply -f studentserver-deployment.yaml
deployment.apps/web created
rzhu6860@cloudshell:~/mongodb/bookshelf (cs571-demo-project-419323)$ kubectl apply -f studentserver-configmap.yaml
configmap/studentserver-config created
```



```
rzhu6860@cloudshell:~/mongodb/bookshelf (cs571-demo-project-419323)$ kubectl apply -f studentserver-service.yaml
service/web created
```

8. Create bookshelf related pods and start service using the above yaml file

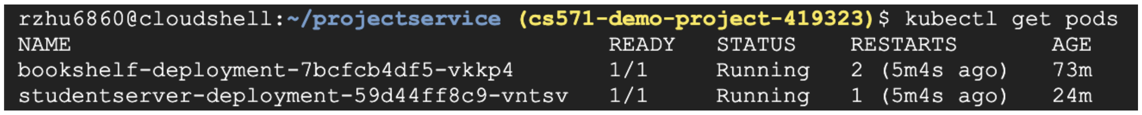
```
kubectl apply -f bookshelf-deployment.yaml
kubectl apply -f bookshelf-configmap.yaml
kubectl apply -f bookshelf-service.yaml
```



```
rzhu6860@cloudshell:~/mongodb/bookshelf (cs571-demo-project-419323)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created
rzhu6860@cloudshell:~/mongodb/bookshelf (cs571-demo-project-419323)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
rzhu6860@cloudshell:~/mongodb/bookshelf (cs571-demo-project-419323)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
```

9. Check if all the pods are running correctly

```
kubectl get pods
```



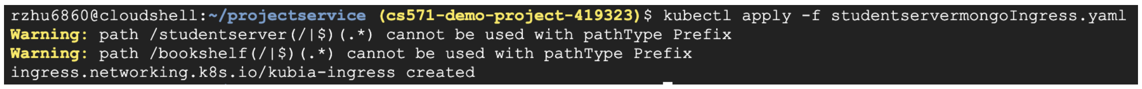
```
rzhu6860@cloudshell:~/projectservice (cs571-demo-project-419323)$ kubectl get pods
NAME                                     READY   STATUS    RESTARTS       AGE
bookshelf-deployment-7bcfcb4df5-vkkp4    1/1     Running   2 (5m4s ago)   73m
studentserver-deployment-59d44ff8c9-vntsv 1/1     Running   1 (5m4s ago)   24m
```

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

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: kubia-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
  rules:
  - host: cs571.project.com
    http:
      paths:
      - path: /studentserver(/|$)(.*)
        pathType: Prefix
        backend:
          service:
            name: studentserver-service
            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

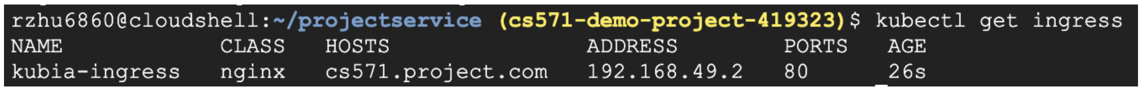
```
kubectl apply -f studentservermongoIngress.yaml
```



```
rzhu6860@cloudshell:~/projectservice (cs571-demo-project-419323)$ 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/kubia-ingress created
```

12. Check ingress status

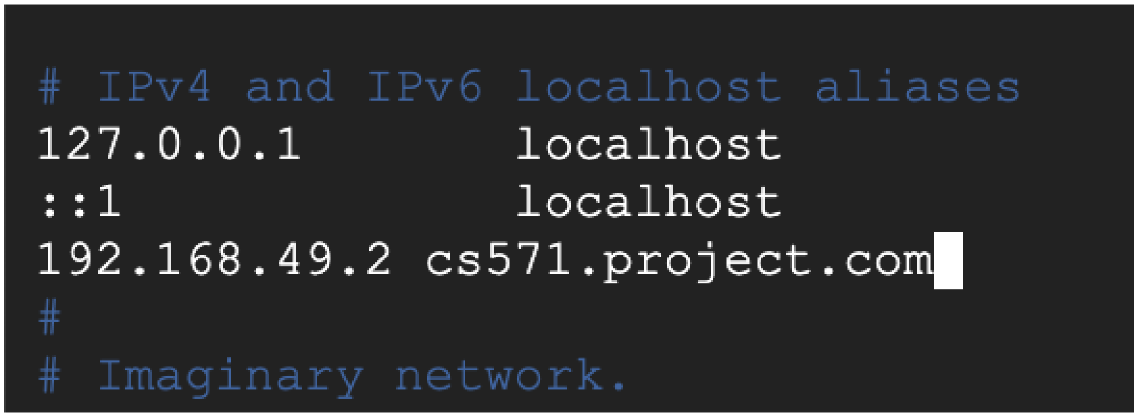
```
kubectl get ingress
```



```
rzhu6860@cloudshell:~/projectservice (cs571-demo-project-419323)$ kubectl get ingress
NAME            CLASS   HOSTS               ADDRESS        PORTS   AGE
kubia-ingress   nginx   cs571.project.com   192.168.49.2   80      26s
```

13. Add Addreee to /etc/hosts

```
sudo vim /etc/hosts

# ip-address cs571.project.com
192.168.49.2 cs571.project.com
```

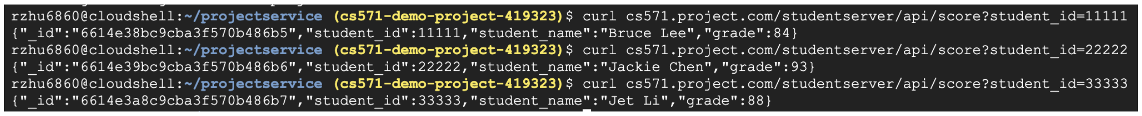


```
# IPv4 and IPv6 localhost aliases
127.0.0.1       localhost
::1             localhost
192.168.49.2 cs571.project.com
#
# Imaginary network.
```

14. Try to access api

    Get student information:

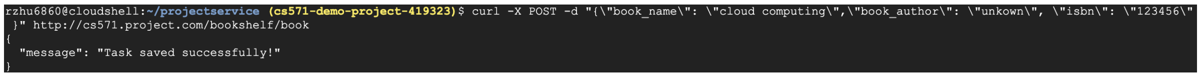
    ```
    curl cs5711.project.com/studentserver/api/score?student_id=11111
    ```

    ```
    rzhu6860@cloudshell:~/projectservice (cs571-demo-project-419323)$ curl cs571.project.com/studentserver/api/score?student_id=11111
    {"_id":"6614e38bc9cba3f570b486b5","student_id":11111,"student_name":"Bruce Lee","grade":84}
    rzhu6860@cloudshell:~/projectservice (cs571-demo-project-419323)$ curl cs571.project.com/studentserver/api/score?student_id=22222
    {"_id":"6614e39bc9cba3f570b486b6","student_id":22222,"student_name":"Jackie Chen","grade":93}
    rzhu6860@cloudshell:~/projectservice (cs571-demo-project-419323)$ curl cs571.project.com/studentserver/api/score?student_id=33333
    {"_id":"6614e3a8c9cba3f570b486b7","student_id":33333,"student_name":"Jet Li","grade":88}
    ```

    For bookshelf add book first:

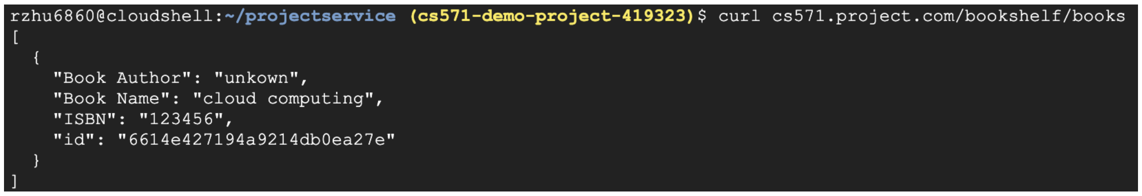
    ```
    curl -X POST -d "{\"book_name\": \"cloud computing\",\"book_aut
    ```

    ```
    rzhu6860@cloudshell:~/projectservice (cs571-demo-project-419323)$ curl -X POST -d "{\"book_name\": \"cloud computing\",\"book_author\": \"unkown\", \"isbn\": \"123456\"
     }" http://cs571.project.com/bookshelf/book
    {
      "message": "Task saved successfully!"
    }
    ```

    And then get the book

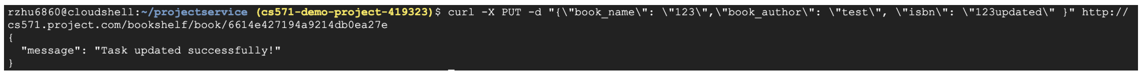
    ```
    curl cs571.project.com/bookshelf/books
    ```

    ```
    rzhu6860@cloudshell:~/projectservice (cs571-demo-project-419323)$ curl cs571.project.com/bookshelf/books
    [
      {
        "Book Author": "unkown",
        "Book Name": "cloud computing",
        "ISBN": "123456",
        "id": "6614e427194a9214db0ea27e"
      }
    ]
    ```

    Update book

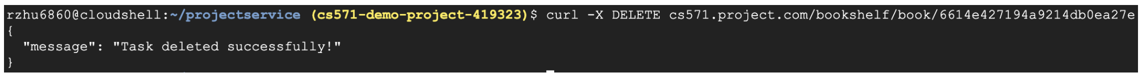
```
curl -X PUT -d "{\"book_name\": \"123\",\"book_author\": \"test
```



```
rzhu6860@cloudshell:~/projectservice (cs571-demo-project-419323)$ curl -X PUT -d "{\"book_name\": \"123\",\"book_author\": \"test\", \"isbn\": \"123updated\" }" http://
cs571.project.com/bookshelf/book/6614e427194a9214db0ea27e
{
  "message": "Task updated successfully!"
}
```

Delete book

```
curl -X DELETE cs571.project.com/bookshelf/book/6614e427194a921
```



```
rzhu6860@cloudshell:~/projectservice (cs571-demo-project-419323)$ curl -X DELETE cs571.project.com/bookshelf/book/6614e427194a9214db0ea27e
{
  "message": "Task deleted successfully!"
}
```

GitHub

https://github.com/RuichenCN/Cloud_Computing_Infrastructure/tree/main/Kubernetes

REFERENCE

END