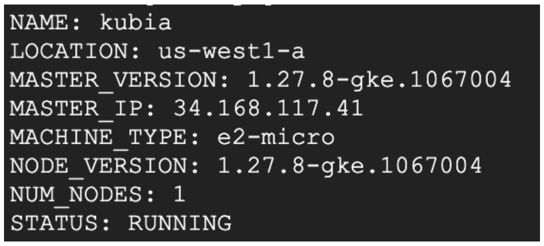
CS571 Signature Project
Name: lianwei Deng
ID: 19874

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

1. Create a cluster as usual on GKE

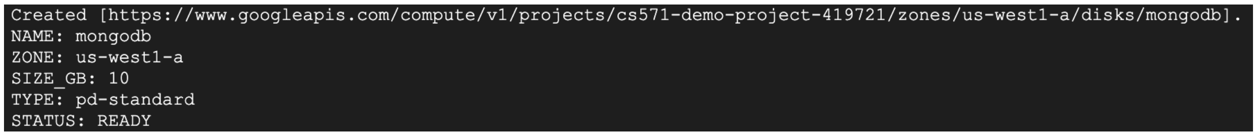
$gcloud container clusters create kubia --num-nodes=1 --machine-type=e2-micro --region=us-west, wait for the creation to finish,



```
NAME: kubia
LOCATION: us-west1-a
MASTER_VERSION: 1.27.8-gke.1067004
MASTER_IP: 34.168.117.41
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.27.8-gke.1067004
NUM_NODES: 1
STATUS: RUNNING
```

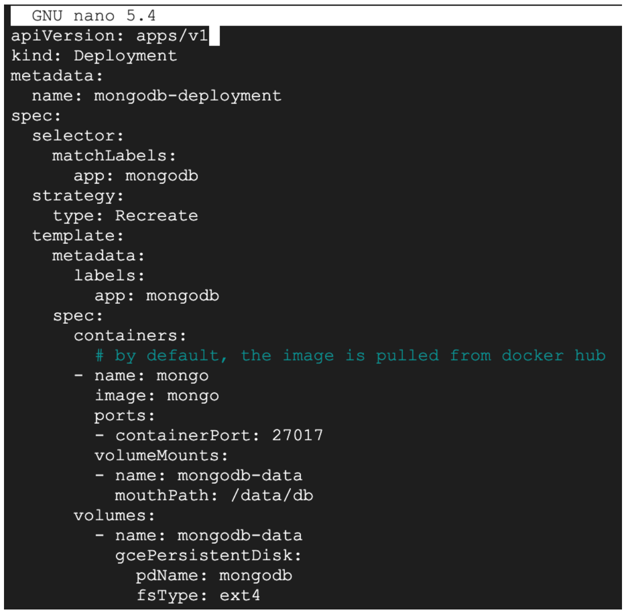
2. Let's create a Persistent Volume first,

$gcloud compute disks create --size=10GiB --zone=us-west1-a mongodb



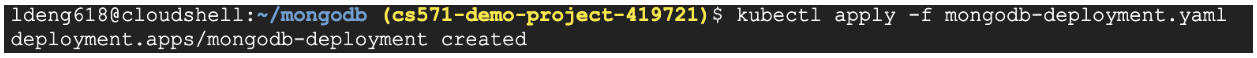
```
Created [https://www.googleapis.com/compute/v1/projects/cs571-demo-project-419721/zones/us-west1-a/disks/mongodb].
NAME: mongodb
ZONE: us-west1-a
SIZE_GB: 10
TYPE: pd-standard
STATUS: READY
```

3. Now create a mongodb deployment with this yaml filec



```
  GNU nano 5.4
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
      - name: mongo
        image: mongo
        ports:
        - containerPort: 27017
        volumeMounts:
        - name: mongodb-data
          mouthPath: /data/db
      volumes:
        - name: mongodb-data
          gcePersistentDisk:
            pdName: mongodb
            fsType: ext4
```

$kubectl apply -f mongodb-deployment.yaml

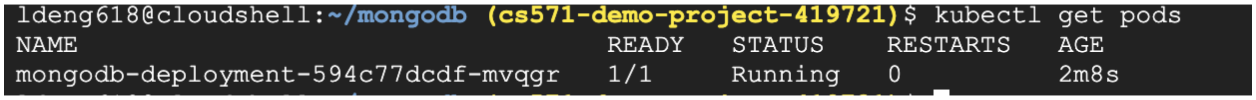


```
ldeng618@cloudshell:~/mongodb (cs571-demo-project-419721)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
```

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

$kubectl get pods

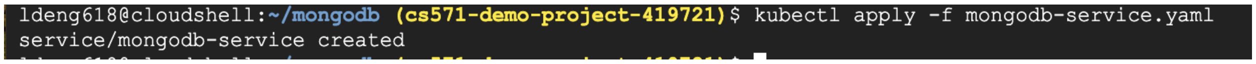
Please wait until you see the STATUS is running, then you can move forward



```
ldeng618@cloudshell:~/mongodb (cs571-demo-project-419721)$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
mongodb-deployment-594c77dcdf-mvqgr   1/1     Running   0          2m8s
```

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

$kubectl apply -f mongodb-service.yaml

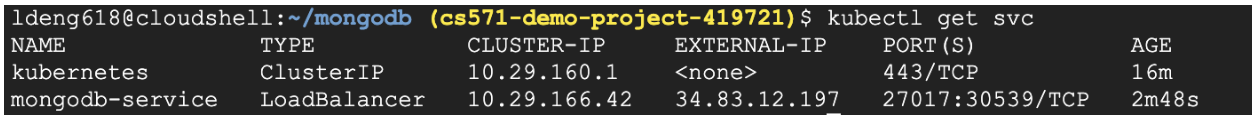


```
ldeng618@cloudshell:~/mongodb (cs571-demo-project-419721)$ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
```

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



```
ldeng618@cloudshell:~/mongodb (cs571-demo-project-419721)$ kubectl get svc
NAME              TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)           AGE
kubernetes        ClusterIP      10.29.160.1    <none>         443/TCP           16m
mongodb-service   LoadBalancer   10.29.166.42   34.83.12.197   27017:30539/TCP   2m48s
```

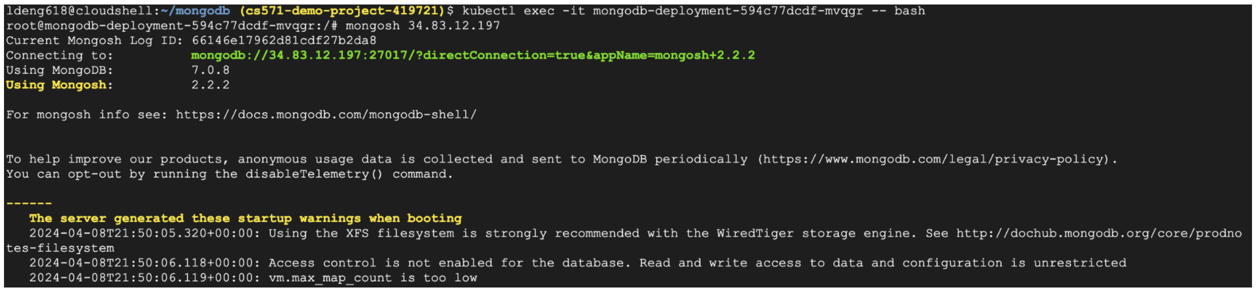
5. Now try and see if mongoDB is functioning for connections using the External-IP

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

Now you are inside the mongodb deployment pod

Try mongosh External-IP

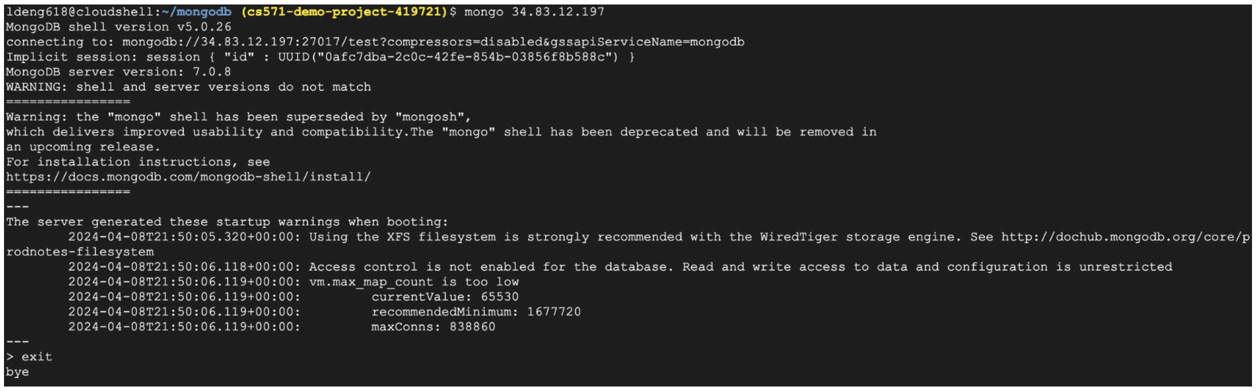
You should see something like this, which means your mongoDB is up and can be accessd using the External-IP



```
ldeng618@cloudshell:~/mongodb (cs571-demo-project-419721)$ kubectl exec -it mongodb-deployment-594c77dcdf-mvqgr -- bash
root@mongodb-deployment-594c77dcdf-mvqgr:/# mongosh 34.83.12.197
Current Mongosh Log ID: 66146e17962d81cdf27b2da8
Connecting to:          mongodb://34.83.12.197:27017/?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
   2024-04-08T21:50:05.320+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodno
tes-filesystem
   2024-04-08T21:50:06.118+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
   2024-04-08T21:50:06.119+00:00: vm.max_map_count is too low
```

Or you can try on gcp shell: mongo external-ip,

Type exit to exit mongodb and back to our google console



```
ldeng618@cloudshell:~/mongodb (cs571-demo-project-419721)$ mongo 34.83.12.197
MongoDB shell version v5.0.26
connecting to: mongodb://34.83.12.197:27017/test?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("0afc7dba-2c0c-42fe-854b-03856f8b588c") }
MongoDB server version: 7.0.8
WARNING: shell and server versions do not match
================
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
================
---
The server generated these startup warnings when booting:
        2024-04-08T21:50:05.320+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/p
rodnotes-filesystem
        2024-04-08T21:50:06.118+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
        2024-04-08T21:50:06.119+00:00: vm.max_map_count is too low
        2024-04-08T21:50:06.119+00:00:         currentValue: 65530
        2024-04-08T21:50:06.119+00:00:         recommendedMinimum: 1677720
        2024-04-08T21:50:06.119+00:00:         maxConns: 838860
---
> exit
bye
```

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

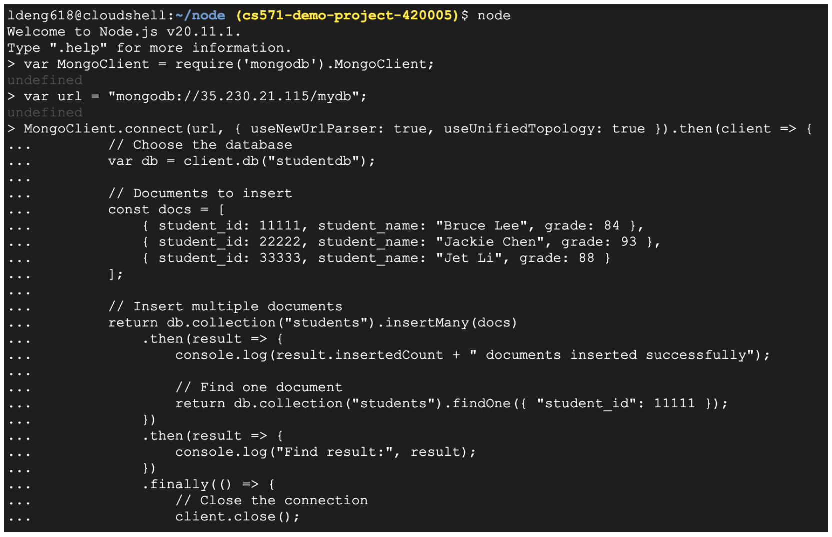
```
node
```

Enter the below content line by line

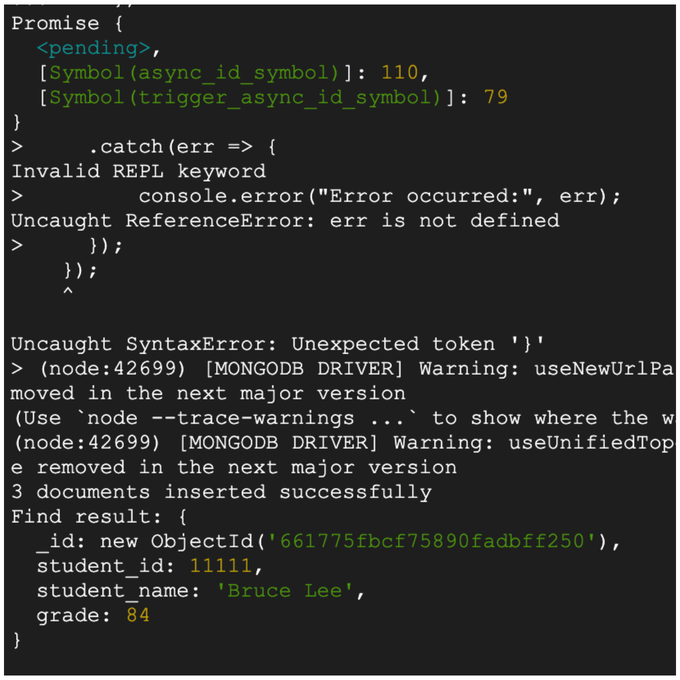
```
// Import MongoDB client
var MongoClient = require('mongodb').MongoClient;

// MongoDB connection URL
var url = "mongodb://EXTERNAL-IP/mydb";

// Connect to the database
MongoClient.connect(url, { useNewUrlParser: true, useUnifiedTopology: true
}).then(client => {
       // Choose the database
       var db = client.db("studentdb");

       // Documents to insert
       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 }
       ];

       // Insert multiple documents
       return db.collection("students").insertMany(docs)
           .then(result => {
               console.log(result.insertedCount + " documents inserted successfully");

               // Find one document
               return db.collection("students").findOne({ "student_id": 11111 });
```

```
            })
            .then(result => {
                console.log("Find result:", result);
            })
            .finally(() => {
                // Close the connection
                client.close();
            });
    })
    .catch(err => {
        console.error("Error occurred:", err);
    });
```

If Everything is correct, you should see this, 3 means three records was inserted, and we tried search for student_id=11111, (ctrl+D exit)



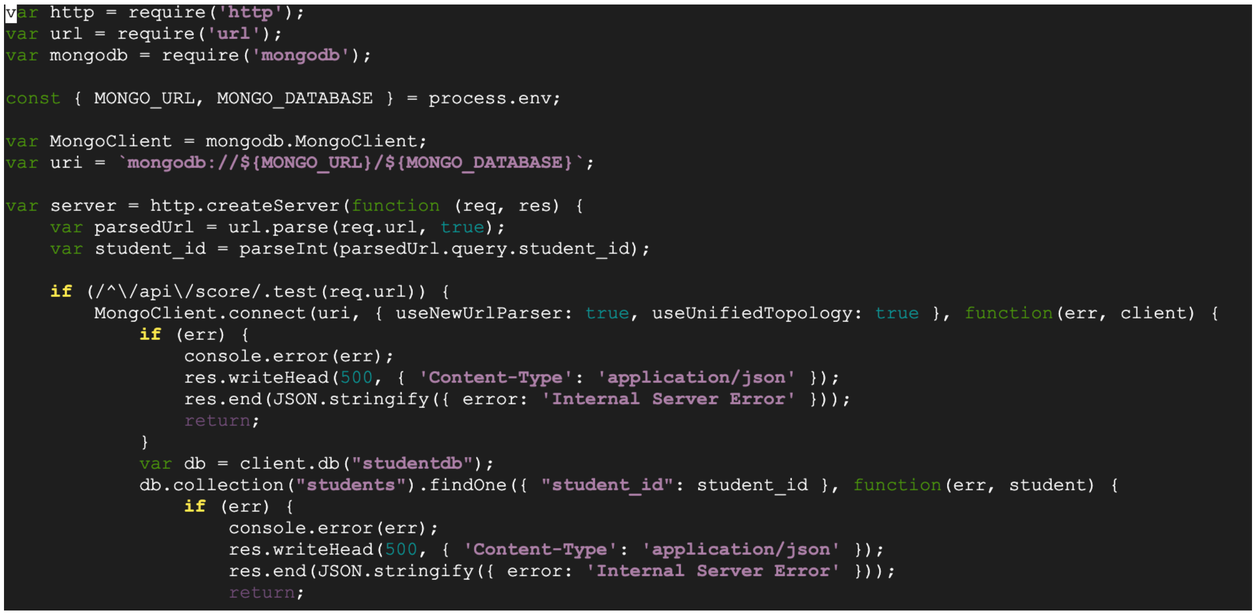
```
ldeng618@cloudshell:~/node (cs571-demo-project-420005)$ node
Welcome to Node.js v20.11.1.
Type ".help" for more information.
> var MongoClient = require('mongodb').MongoClient;
undefined
> var url = "mongodb://35.230.21.115/mydb";
undefined
> MongoClient.connect(url, { useNewUrlParser: true, useUnifiedTopology: true }).then(client => {
...         // Choose the database
...         var db = client.db("studentdb");
...
...         // Documents to insert
...         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 }
...         ];
...
...         // Insert multiple documents
...         return db.collection("students").insertMany(docs)
...             .then(result => {
...                 console.log(result.insertedCount + " documents inserted successfully");
...
...                 // Find one document
...                 return db.collection("students").findOne({ "student_id": 11111 });
...             })
...             .then(result => {
...                 console.log("Find result:", result);
...             })
...             .finally(() => {
...                 // Close the connection
...                 client.close();
```



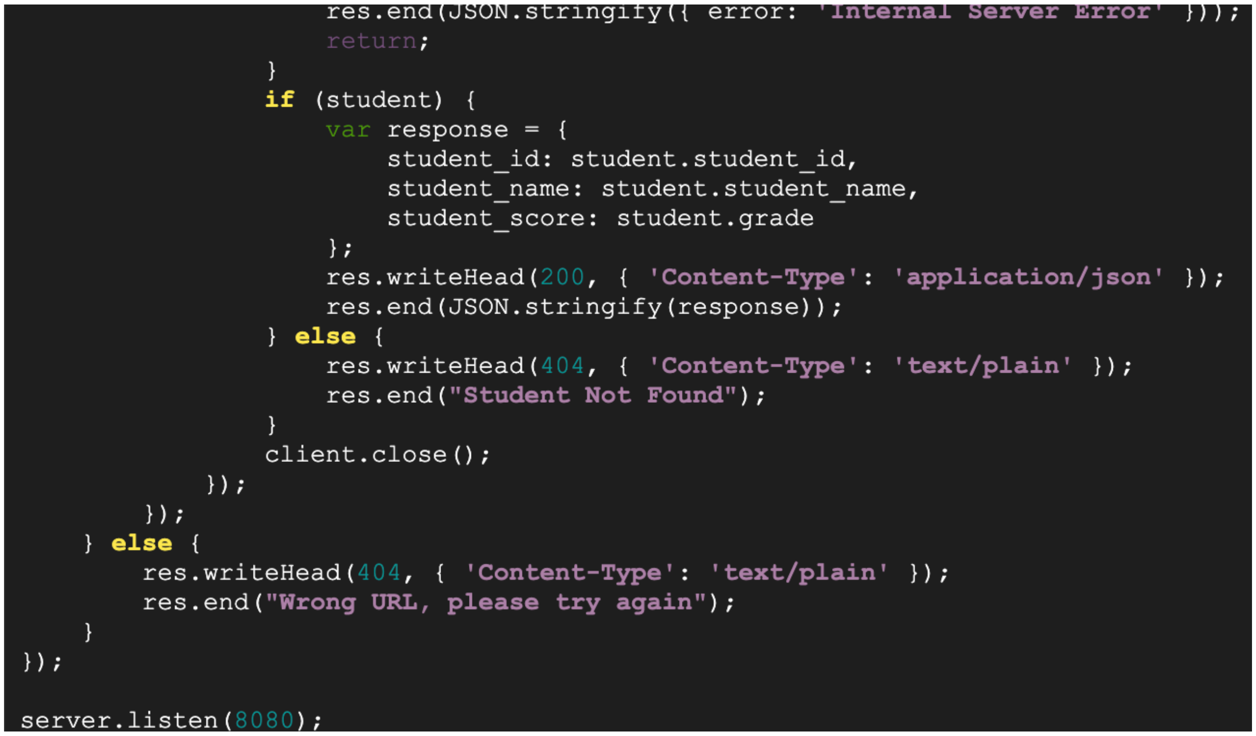
```
Promise {
  <pending>,
  [Symbol(async_id_symbol)]: 110,
  [Symbol(trigger_async_id_symbol)]: 79
}
>     .catch(err => {
Invalid REPL keyword
>         console.error("Error occurred:", err);
Uncaught ReferenceError: err is not defined
>     });
    });
    ^

Uncaught SyntaxError: Unexpected token '}'
> (node:42699) [MONGODB DRIVER] Warning: useNewUrlPa
moved in the next major version
(Use `node --trace-warnings ...` to show where the w
(node:42699) [MONGODB DRIVER] Warning: useUnifiedTop
e removed in the next major version
3 documents inserted successfully
Find result: {
  _id: new ObjectId('661775fbcf75890fadbff250'),
  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.js



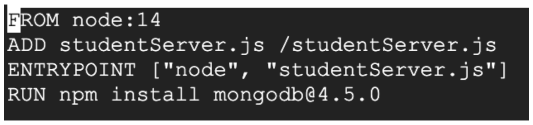
```
var http = require('http');
var url = require('url');
var mongodb = require('mongodb');

const { MONGO_URL, MONGO_DATABASE } = process.env;

var MongoClient = mongodb.MongoClient;
var uri = `mongodb://${MONGO_URL}/${MONGO_DATABASE}`;

var server = http.createServer(function (req, res) {
    var parsedUrl = url.parse(req.url, true);
    var student_id = parseInt(parsedUrl.query.student_id);

    if (/^\/api\/score/.test(req.url)) {
        MongoClient.connect(uri, { useNewUrlParser: true, useUnifiedTopology: true }, function(err, client) {
            if (err) {
                console.error(err);
                res.writeHead(500, { 'Content-Type': 'application/json' });
                res.end(JSON.stringify({ error: 'Internal Server Error' }));
                return;
            }
            var db = client.db("studentdb");
            db.collection("students").findOne({ "student_id": student_id }, function(err, student) {
                if (err) {
                    console.error(err);
                    res.writeHead(500, { 'Content-Type': 'application/json' });
                    res.end(JSON.stringify({ error: 'Internal Server Error' }));
                    return;
```



```
                    res.end(JSON.stringify({ error: 'Internal Server Error' }));
                    return;
```

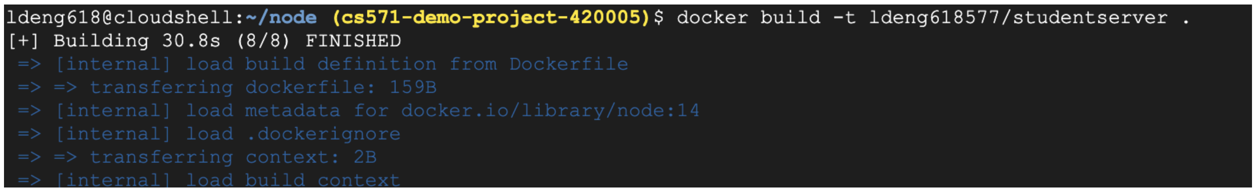
```
                }
                if (student) {
                    var response = {
                        student_id: student.student_id,
                        student_name: student.student_name,
                        student_score: student.grade
                    };
                    res.writeHead(200, { 'Content-Type': 'application/json' });
                    res.end(JSON.stringify(response));
                } else {
                    res.writeHead(404, { 'Content-Type': 'text/plain' });
                    res.end("Student Not Found");
                }
                client.close();
            });
        });
    } else {
        res.writeHead(404, { 'Content-Type': 'text/plain' });
        res.end("Wrong URL, please try again");
    }
});

server.listen(8080);
```

Dockerfile:



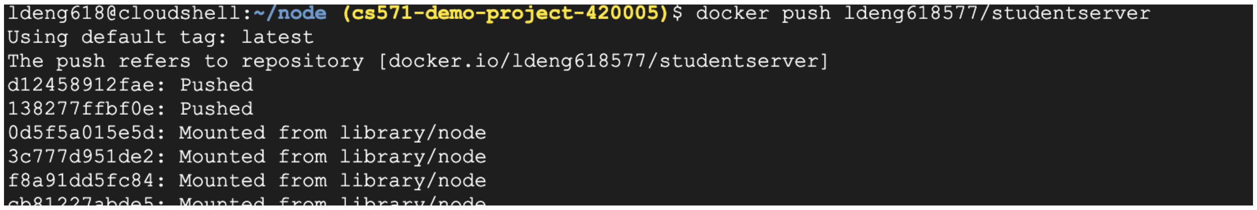
```
FROM node:14
ADD studentServer.js /studentServer.js
ENTRYPOINT ["node", "studentServer.js"]
RUN npm install mongodb@4.5.0
```

Build the studentserver docker image
$docker build -t yourdockerhubID/studentserver .
Make sure there is no error



```
ldeng618@cloudshell:~/node (cs571-demo-project-420005)$ docker build -t ldeng618577/studentserver .
[+] Building 30.8s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 159B
 => [internal] load metadata for docker.io/library/node:14
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load build context
```

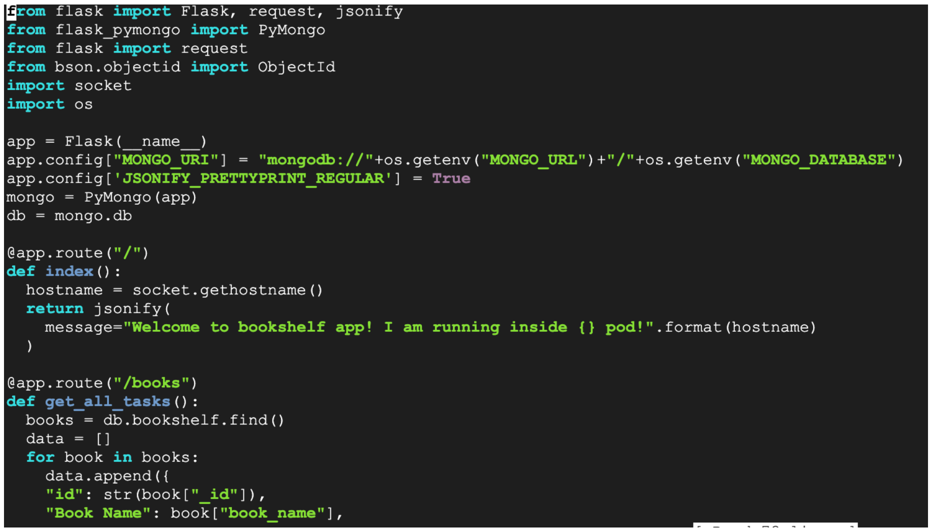
2. Push the docker image
   docker push yourdockerhubID/studentserver



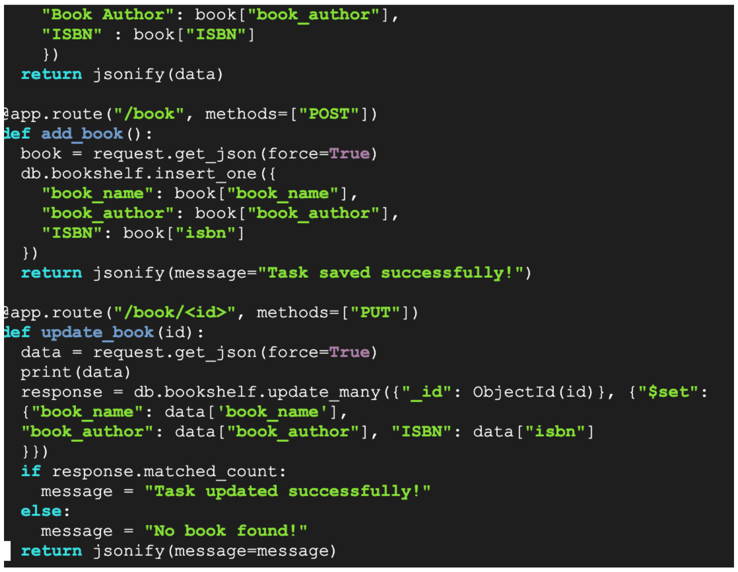
```
ldeng618@cloudshell:~/node (cs571-demo-project-420005)$ docker push ldeng618577/studentserver
Using default tag: latest
The push refers to repository [docker.io/ldeng618577/studentserver]
d12458912fae: Pushed
138277ffbf0e: Pushed
0d5f5a015e5d: Mounted from library/node
3c777d951de2: Mounted from library/node
f8a91dd5fc84: Mounted from library/node
cb81227abde5: Mounted from library/node
```

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

1. Create bookshelf.py

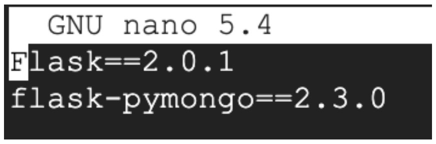


```python
from flask import Flask, request, jsonify
from flask_pymongo import PyMongo
from flask import request
from bson.objectid import ObjectId
import socket
import os

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"],
```



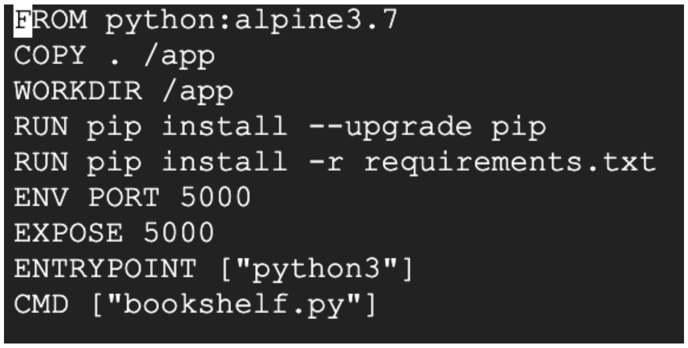
```
    "Book Author": book["book_author"],
    "ISBN" : book["ISBN"]
    })
  return jsonify(data)

@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)
  print(data)
  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)
```

Requirements.txt:



```
  GNU nano 5.4
Flask==2.0.1
flask-pymongo==2.3.0
```

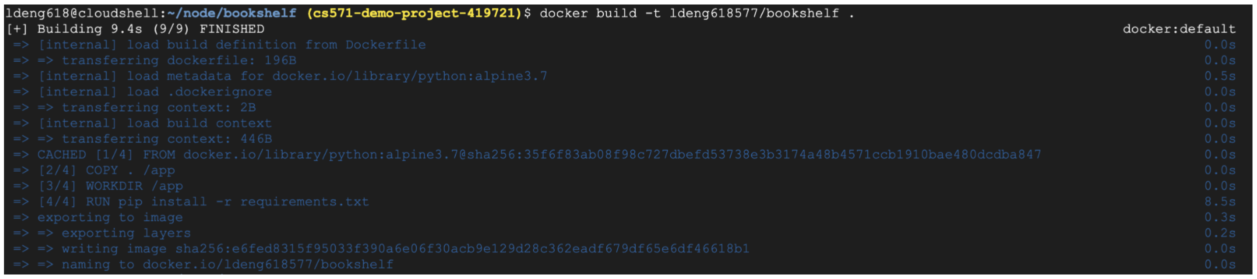
2. Create a Dockerfile



```
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"]
```

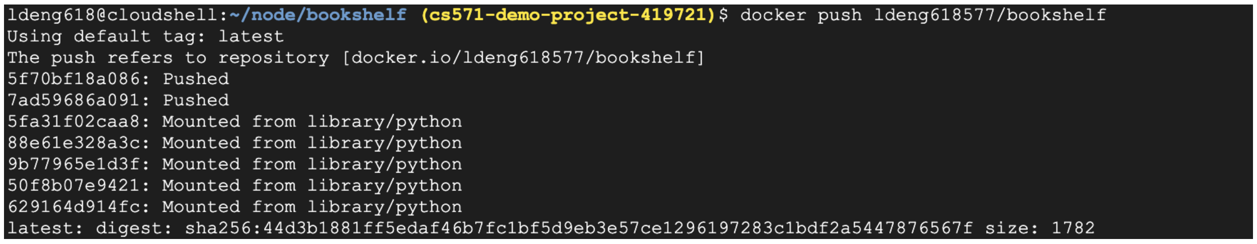
3. Build the bookshelf app into a docker image

docker build -t dockerid/repository



```
ldeng618@cloudshell:~/node/bookshelf (cs571-demo-project-419721)$ docker build -t ldeng618577/bookshelf .
[+] Building 9.4s (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                                          0.5s
 => [internal] load .dockerignore                                                                            0.0s
 => => transferring context: 2B                                                                              0.0s
 => [internal] load build context                                                                            0.0s
 => => transferring context: 446B                                                                            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                                                                8.5s
 => exporting to image                                                                                       0.3s
 => => exporting layers                                                                                      0.2s
 => => writing image sha256:e6fed8315f95033f390a6e06f30acb9e129d28c362eadf679df65e6df46618b1                 0.0s
 => => naming to docker.io/ldeng618577/bookshelf                                                             0.0s
```

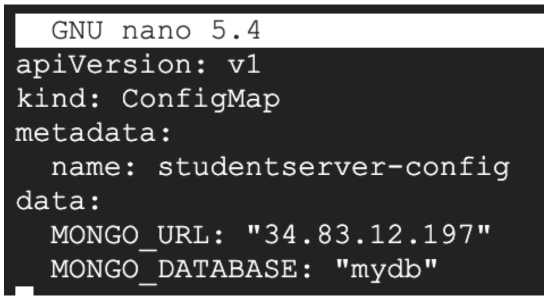
4. Push the docker image to your dockerhub
   $docker push yourdockerhubID/bookshelf



```
ldeng618@cloudshell:~/node/bookshelf (cs571-demo-project-419721)$ docker push ldeng618577/bookshelf
Using default tag: latest
The push refers to repository [docker.io/ldeng618577/bookshelf]
5f70bf18a086: Pushed
7ad59686a091: 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: sha256:44d3b1881ff5edaf46b7fc1bf5d9eb3e57ce1296197283c1bdf2a5447876567f size: 1782
```

<u>Step4 Create ConfigMap for both applications to store MongoDB URL and MongoDB name</u>

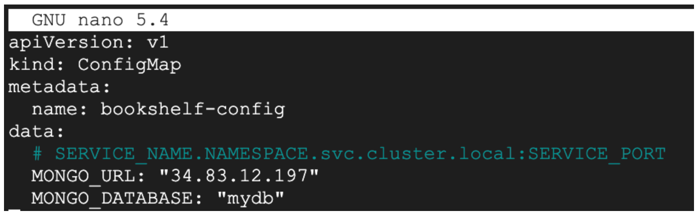
1. Create a file named studentserver-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
name: studentserver-config
data:
MONGO_URL: Change-this-to-your-mongoDB-EXTERNAL-IP
MONGO_DATABASE: mydb



```
  GNU nano 5.4
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  MONGO_URL: "34.83.12.197"
  MONGO_DATABASE: "mydb"
```

2. Create a file named bookshelf-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
name: bookshelf-config
data:
# SERVICE_NAME.NAMESPACE.svc.cluster.local:SERVICE_PORT

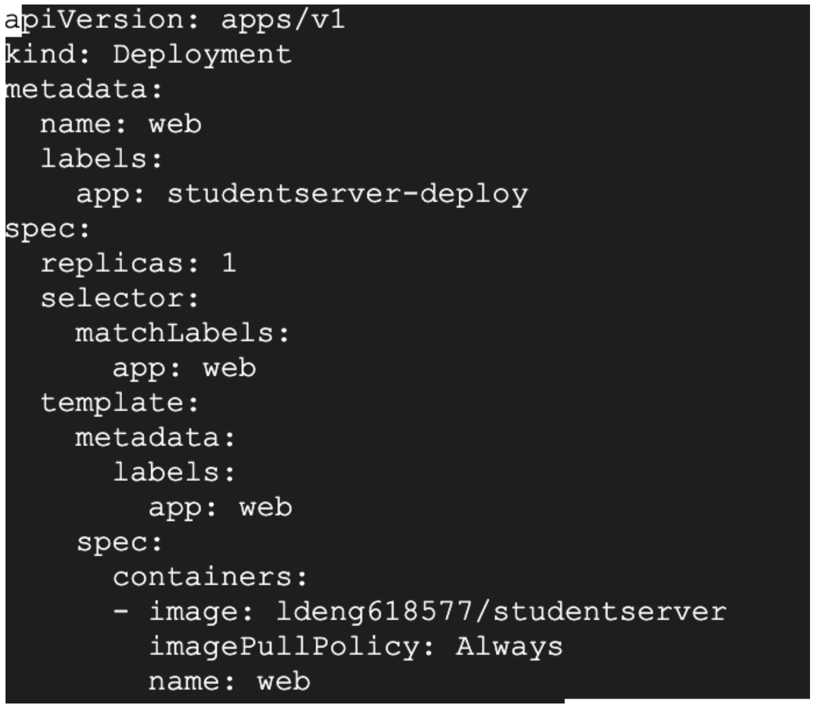
MONGO_URL: Change-this-to-your-mongoDB-EXTERNAL-IP
MONGO_DATABASE: mydb



```
  GNU nano 5.4
apiVersion: v1
kind: ConfigMap
metadata:
  name: bookshelf-config
data:
  # SERVICE_NAME.NAMESPACE.svc.cluster.local:SERVICE_PORT
  MONGO_URL: "34.83.12.197"
  MONGO_DATABASE: "mydb"
```

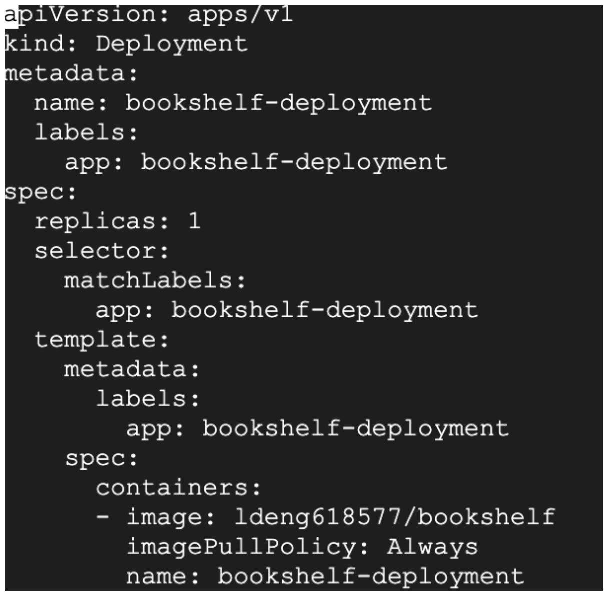
Step5 Expose 2 application using ingress with Nginx, so we can put them on the same Domain but different PATH

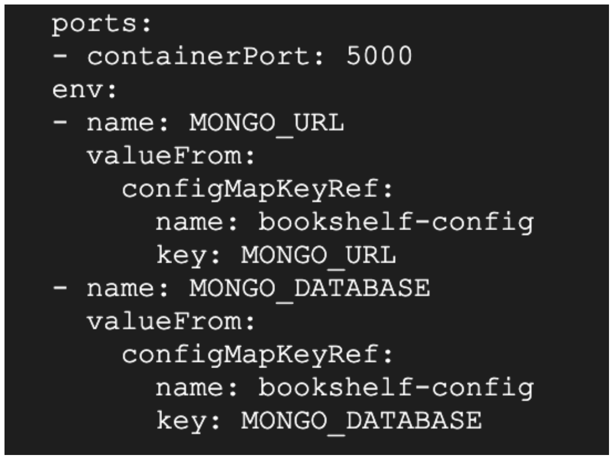
1. Create studentserver-deployment.yaml



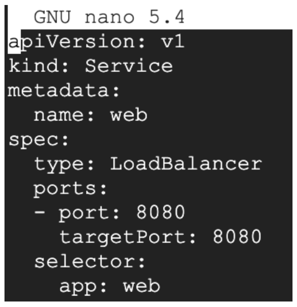
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
  labels:
    app: studentserver-deploy
spec:
  replicas: 1
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
      - image: ldeng618577/studentserver
        imagePullPolicy: Always
        name: web
```

2. Create bookshelf-deployment.yaml

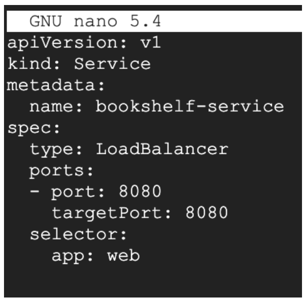

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: bookshelf-deployment
  labels:
    app: bookshelf-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: bookshelf-deployment
  template:
    metadata:
      labels:
        app: bookshelf-deployment
    spec:
      containers:
      - image: ldeng618577/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

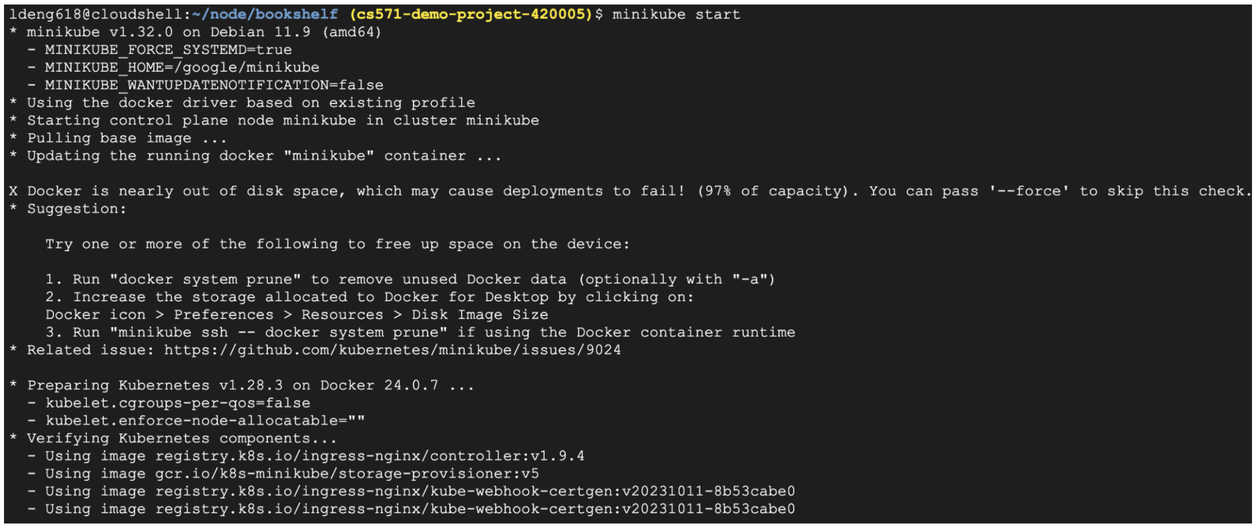

```
  GNU nano 5.4
apiVersion: v1
kind: Service
metadata:
  name: web
spec:
  type: LoadBalancer
  ports:
  - port: 8080
    targetPort: 8080
  selector:
    app: web
```

4. Create bookshelf-service.yaml



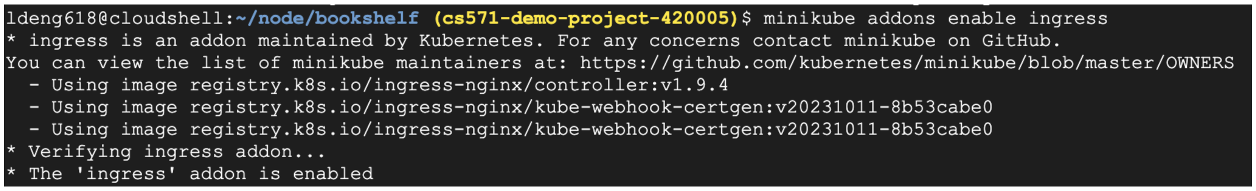
```
  GNU nano 5.4
apiVersion: v1
kind: Service
metadata:
  name: bookshelf-service
spec:
  type: LoadBalancer
  ports:
  - port: 8080
    targetPort: 8080
  selector:
    app: web
```

5. Start minikube minikube start



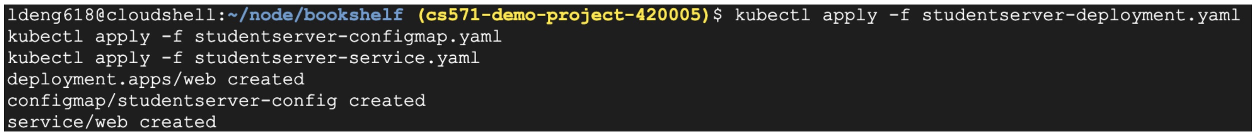
```
ldeng618@cloudshell:~/node/bookshelf (cs571-demo-project-420005)$ minikube start
* minikube v1.32.0 on Debian 11.9 (amd64)
  - MINIKUBE_FORCE_SYSTEMD=true
  - MINIKUBE_HOME=/google/minikube
  - MINIKUBE_WANTUPDATENOTIFICATION=false
* Using the docker driver based on existing profile
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Updating the running docker "minikube" container ...

X Docker is nearly out of disk space, which may cause deployments to fail! (97% 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=""
* Verifying Kubernetes components...
  - Using image registry.k8s.io/ingress-nginx/controller:v1.9.4
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20231011-8b53cabe0
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20231011-8b53cabe0
```

6. Start Ingress minikube addons enable ingress



```
ldeng618@cloudshell:~/node/bookshelf (cs571-demo-project-420005)$ 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
```

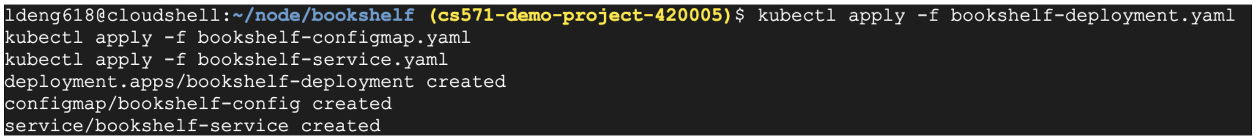
7. Create studentserver related pods and start service using the above yaml file
   kubectl apply -f studentserver-deployment.yaml
   kubectl apply -f studentserver-configmap.yaml
   kubectl apply -f studentserver-service.yaml



```
ldeng618@cloudshell:~/node/bookshelf (cs571-demo-project-420005)$ kubectl apply -f studentserver-deployment.yaml
kubectl apply -f studentserver-configmap.yaml
kubectl apply -f studentserver-service.yaml
deployment.apps/web created
configmap/studentserver-config created
service/web created
```

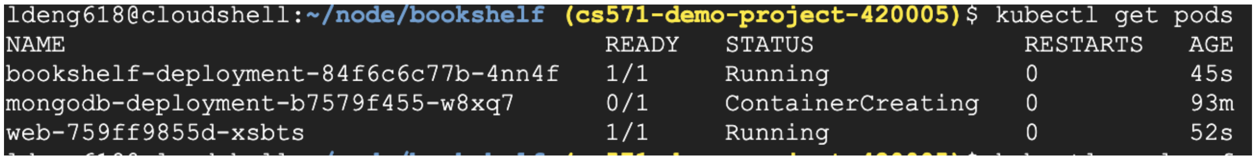
8. Create bookshelf related pods and start service using the above yaml file

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



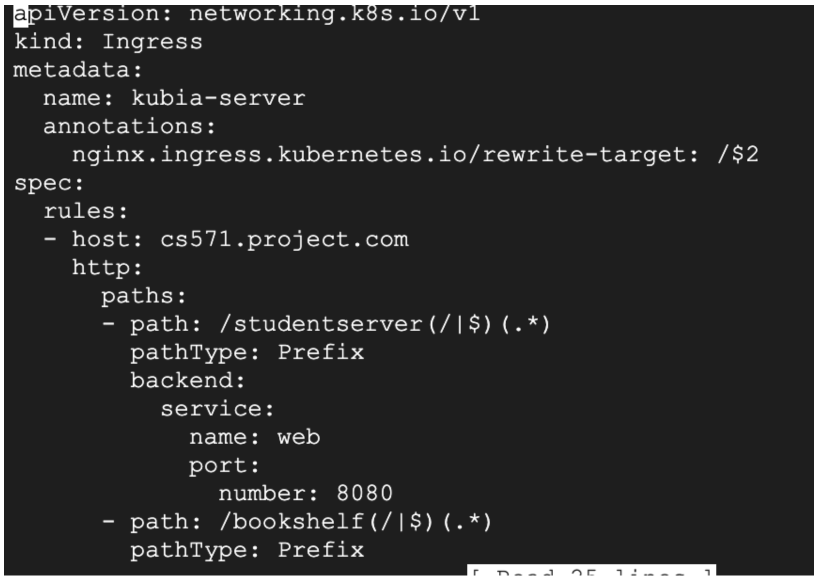
```
ldeng618@cloudshell:~/node/bookshelf (cs571-demo-project-420005)$ kubectl apply -f bookshelf-deployment.yaml
kubectl apply -f bookshelf-configmap.yaml
kubectl apply -f bookshelf-service.yaml
deployment.apps/bookshelf-deployment created
configmap/bookshelf-config created
service/bookshelf-service created
```

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



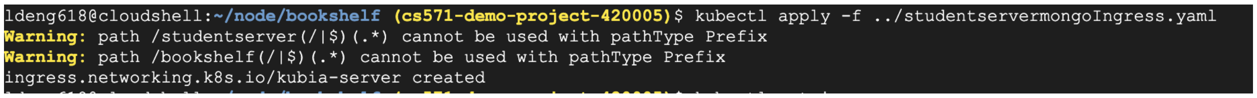
```
ldeng618@cloudshell:~/node/bookshelf (cs571-demo-project-420005)$ kubectl get pods
NAME                                   READY   STATUS              RESTARTS   AGE
bookshelf-deployment-84f6c6c77b-4nn4f  1/1     Running             0          45s
mongodb-deployment-b7579f455-w8xq7     0/1     ContainerCreating   0          93m
web-759ff9855d-xsbts                   1/1     Running             0          52s
```

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



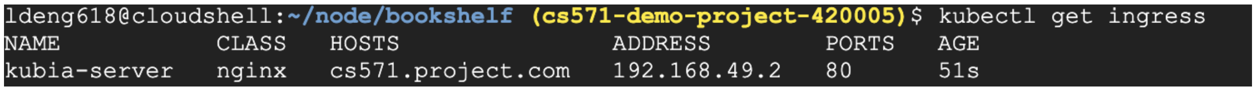
```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: kubia-server
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
  rules:
  - host: cs571.project.com
    http:
      paths:
      - path: /studentserver(/|$)(.*)
        pathType: Prefix
        backend:
          service:
            name: web
            port:
              number: 8080
      - path: /bookshelf(/|$)(.*)
        pathType: Prefix
```

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



```
ldeng618@cloudshell:~/node/bookshelf (cs571-demo-project-420005)$ 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-server created
```

12. Check if ingress is running
kubectl get ingress
Please wait until you see the Address, then move forward

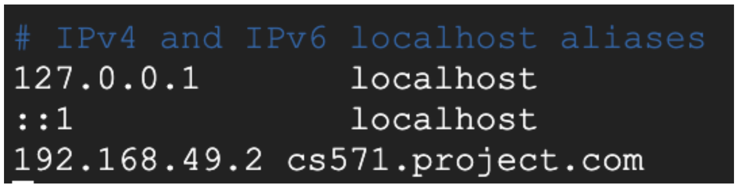


```
ldeng618@cloudshell:~/node/bookshelf (cs571-demo-project-420005)$ kubectl get ingress
NAME           CLASS   HOSTS               ADDRESS        PORTS   AGE
kubia-server   nginx   cs571.project.com   192.168.49.2   80      51s
```

13. Add Address to /etc/hosts

    vi /etc/hosts

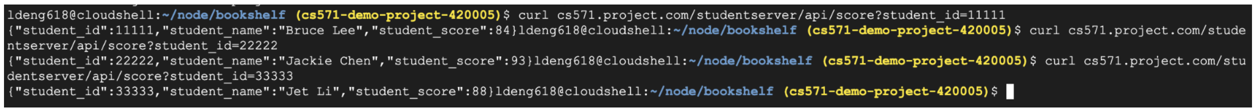
    Add the address you got from above step to the end of the file Your-address cs571.project.com



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

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

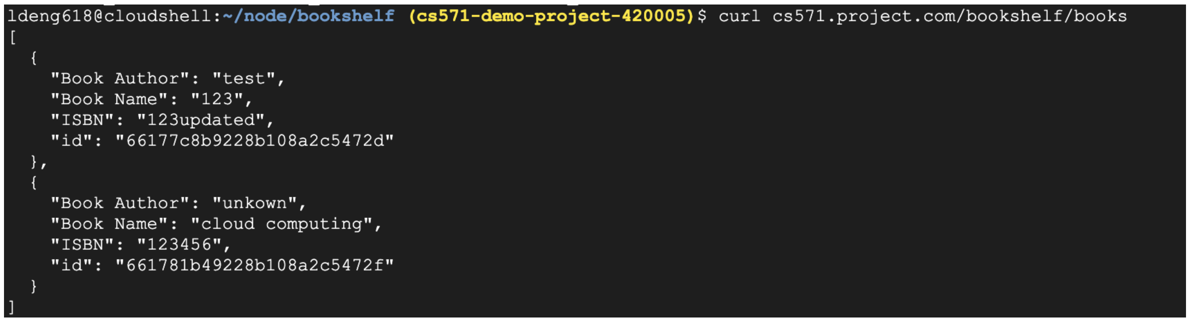
    curl cs571.project.com/studentserver/api/score?student_id=11111



```
ldeng618@cloudshell:~/node/bookshelf (cs571-demo-project-420005)$ curl cs571.project.com/studentserver/api/score?student_id=11111
{"student_id":11111,"student_name":"Bruce Lee","student_score":84}ldeng618@cloudshell:~/node/bookshelf (cs571-demo-project-420005)$ curl cs571.project.com/stude
ntserver/api/score?student_id=22222
{"student_id":22222,"student_name":"Jackie Chen","student_score":93}ldeng618@cloudshell:~/node/bookshelf (cs571-demo-project-420005)$ curl cs571.project.com/stu
dentserver/api/score?student_id=33333
{"student_id":33333,"student_name":"Jet Li","student_score":88}ldeng618@cloudshell:~/node/bookshelf (cs571-demo-project-420005)$
```

15. On another path, you should be able to use the REST API with bookshelf application I.e list all books

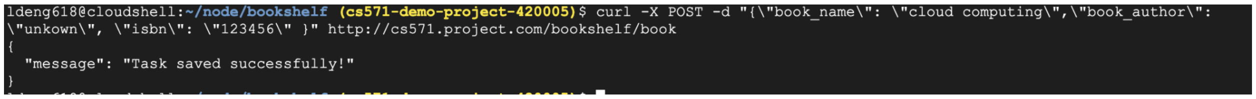
    curl cs571.project.com/bookshelf/books



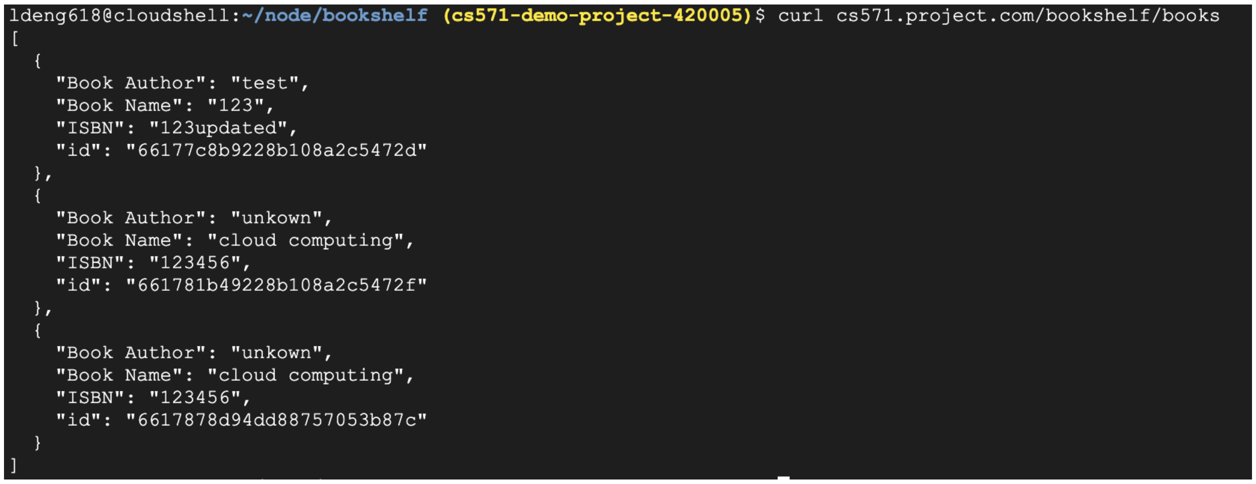
```
ldeng618@cloudshell:~/node/bookshelf (cs571-demo-project-420005)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123updated",
    "id": "66177c8b9228b108a2c5472d"
  },
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "661781b49228b108a2c5472f"
  }
]
```

Add a book

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



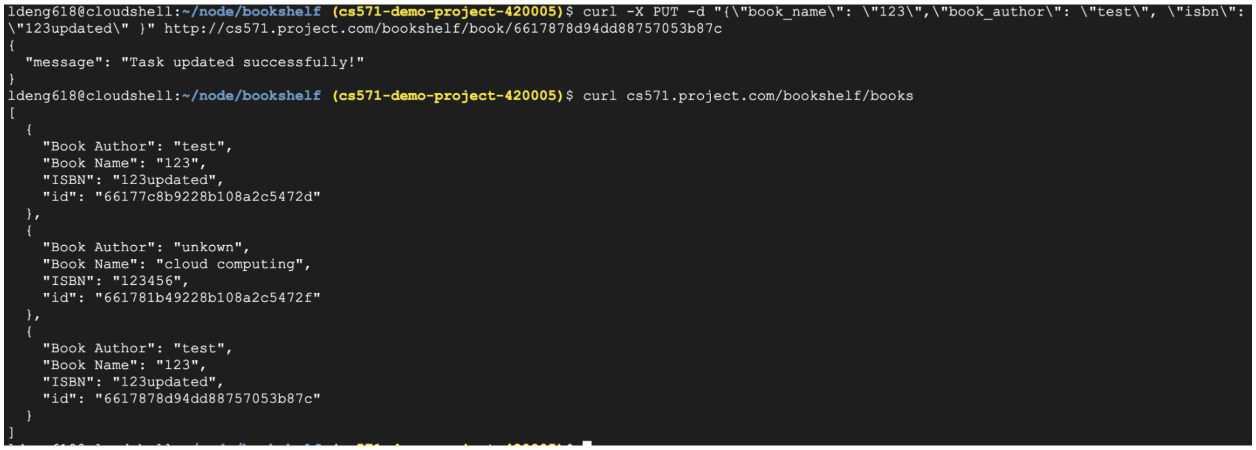
```
ldeng618@cloudshell:~/node/bookshelf (cs571-demo-project-420005)$ curl -X POST -d "{\"book_name\": \"cloud computing\",\"book_author\":
\"unkown\", \"isbn\": \"123456\" }" http://cs571.project.com/bookshelf/book
{
  "message": "Task saved successfully!"
}
```



```
ldeng618@cloudshell:~/node/bookshelf (cs571-demo-project-420005)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123updated",
    "id": "66177c8b9228b108a2c5472d"
  },
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "661781b49228b108a2c5472f"
  },
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "6617878d94dd88757053b87c"
  }
]
```

Update a book

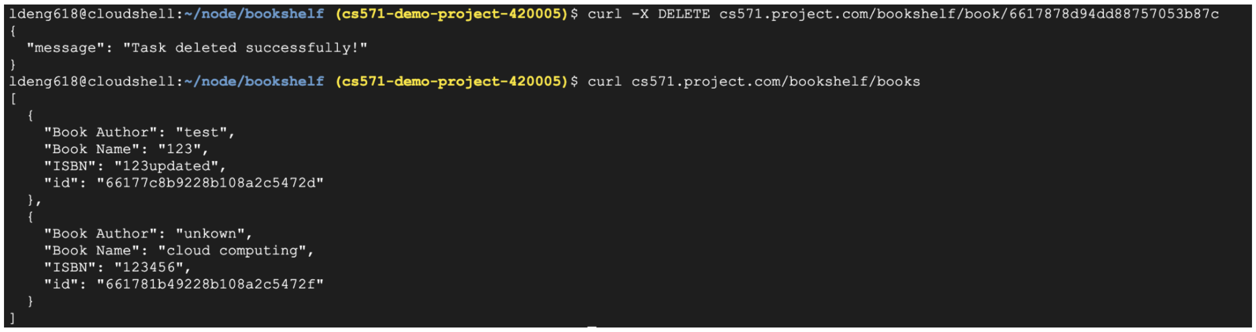
curl -X PUT -d "{\"book_name\": \"123\",\"book_author\": \"test\", \"isbn\": \"123updated\" }" http://cs571.project.com/bookshelf/book/id



```
ldeng618@cloudshell:~/node/bookshelf (cs571-demo-project-420005)$ curl -X PUT -d "{\"book_name\": \"123\",\"book_author\": \"test\", \"isbn\":
\"123updated\" }" http://cs571.project.com/bookshelf/book/6617878d94dd88757053b87c
{
  "message": "Task updated successfully!"
}
ldeng618@cloudshell:~/node/bookshelf (cs571-demo-project-420005)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123updated",
    "id": "66177c8b9228b108a2c5472d"
  },
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "661781b49228b108a2c5472f"
  },
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123updated",
    "id": "6617878d94dd88757053b87c"
  }
]
```

Delete a book

curl -X DELETE cs571.project.com/bookshelf/book/id



```
ldeng618@cloudshell:~/node/bookshelf (cs571-demo-project-420005)$ curl -X DELETE cs571.project.com/bookshelf/book/6617878d94dd88757053b87c
{
  "message": "Task deleted successfully!"
}
ldeng618@cloudshell:~/node/bookshelf (cs571-demo-project-420005)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123updated",
    "id": "66177c8b9228b108a2c5472d"
  },
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "661781b49228b108a2c5472f"
  }
]
```