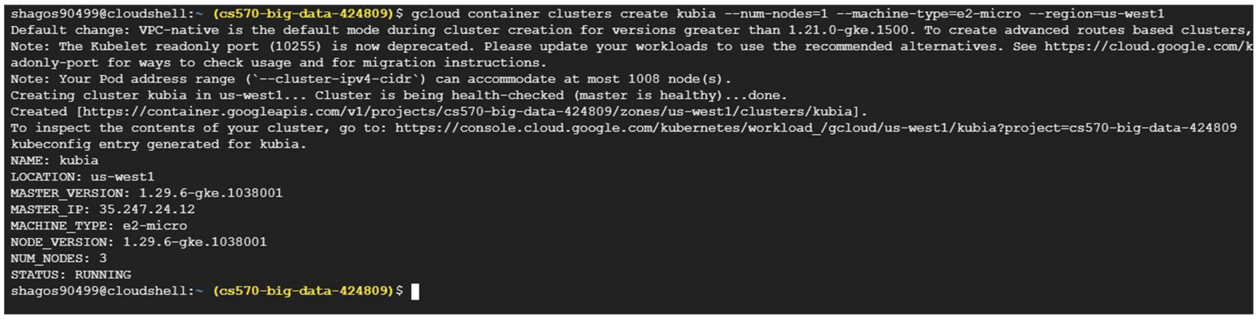
Week 11: Homework 4: Chapter 7: Configmap: Signature Project: MongoDB + Python Flask Web Framework + REST API + GKE

Step 1: Create MongoDB Using Persistent Volume on GKE and Insert Records

1. **Create a GKE Cluster:**

```
gcloud container clusters create kubia --num-nodes=1 --machine-type=e2-
micro --region=us-west1
```

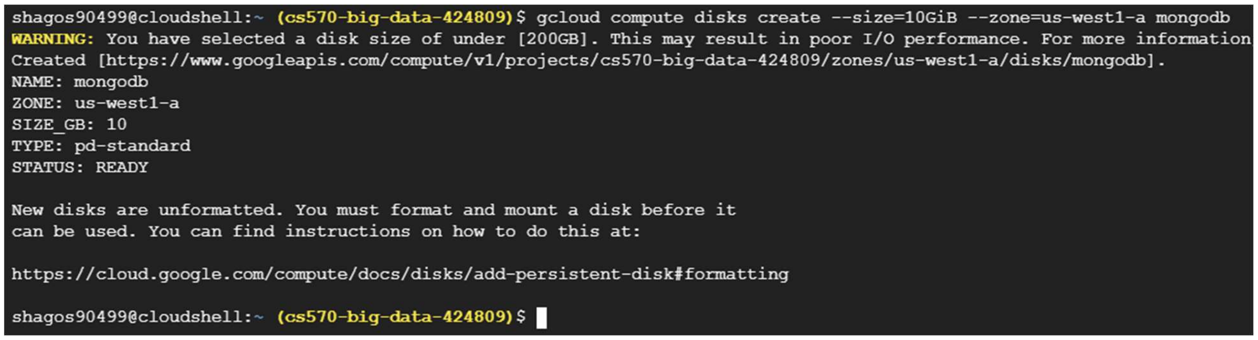
Wait for the cluster creation to finish.



```
shagos90499@cloudshell:~ (cs570-big-data-424809)$ gcloud container clusters create kubia --num-nodes=1 --machine-type=e2-micro --region=us-west1
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,
Note: The Kubelet readonly port (10255) is now deprecated. Please update your workloads to use the recommended alternatives. See https://cloud.google.com/k
adonly-port for ways to check usage and for migration instructions.
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/cs570-big-data-424809/zones/us-west1/clusters/kubia].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/us-west1/kubia?project=cs570-big-data-424809
kubeconfig entry generated for kubia.
NAME: kubia
LOCATION: us-west1
MASTER_VERSION: 1.29.6-gke.1038001
MASTER_IP: 35.247.24.12
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.29.6-gke.1038001
NUM_NODES: 3
STATUS: RUNNING
shagos90499@cloudshell:~ (cs570-big-data-424809)$
```

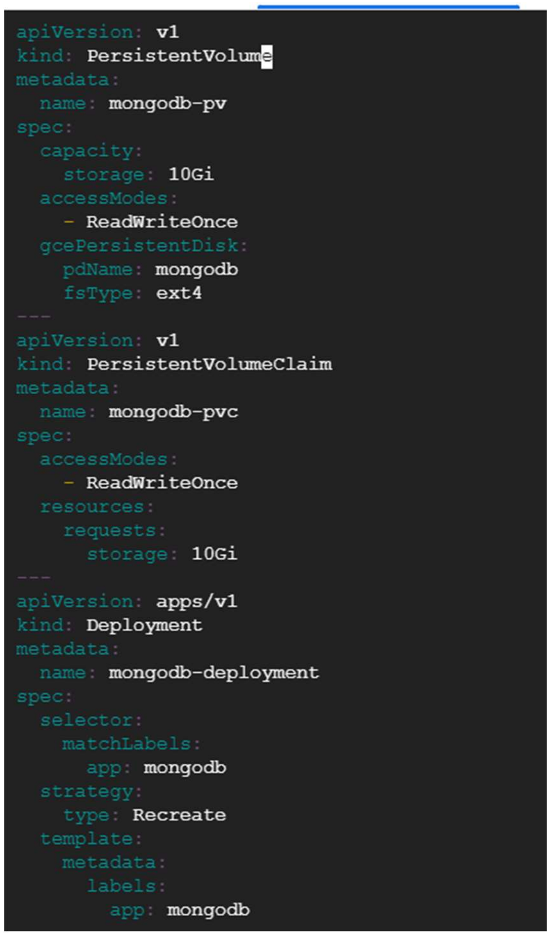
2. **Create Persistent Volume:**

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



```
shagos90499@cloudshell:~ (cs570-big-data-424809)$ 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
Created [https://www.googleapis.com/compute/v1/projects/cs570-big-data-424809/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

shagos90499@cloudshell:~ (cs570-big-data-424809)$
```

3. **Deploy MongoDB:** Apply the `mongodb-deployment.yaml` configuration:

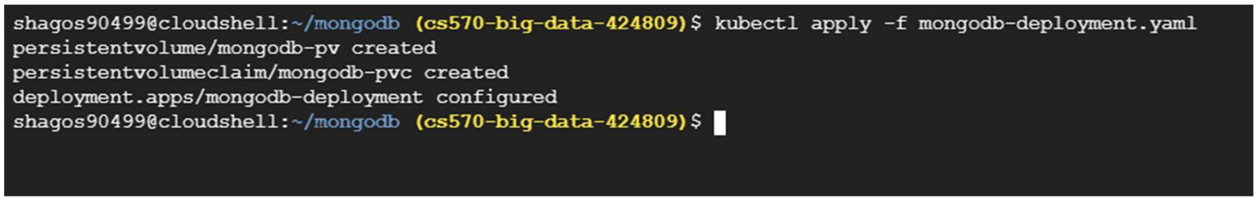


```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mongodb-pv
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  gcePersistentDisk:
    pdName: mongodb
    fsType: ext4
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mongodb-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
spec:
  selector:
    matchLabels:
      app: mongodb
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mongodb
```

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mongodb-pv
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  gcePersistentDisk:
    pdName: mongodb
    fsType: ext4
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mongodb-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
---
apiVersion: apps/v1
```

```
kind: Deployment
metadata:
  name: mongodb-deployment
spec:
  selector:
    matchLabels:
      app: mongodb
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
      - image: mongo
        name: mongo
        ports:
        - containerPort: 27017
        volumeMounts:
        - name: mongodb-data
          mountPath: /data/db
      volumes:
      - name: mongodb-data
        persistentVolumeClaim:
          claimName: mongodb-pvc
```

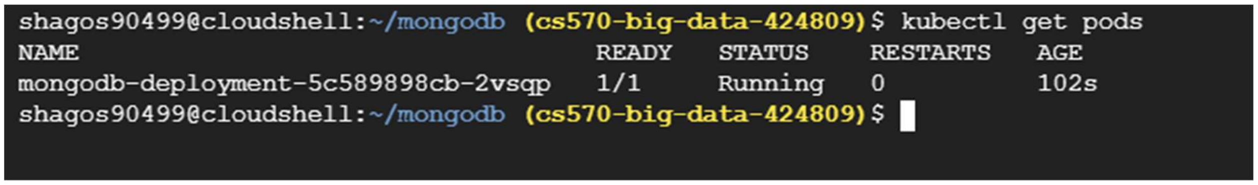
```
kubectl apply -f mongodb-deployment.yaml
```



```
shagos90499@cloudshell:~/mongodb (cs570-big-data-424809)$ kubectl apply -f mongodb-deployment.yaml
persistentvolume/mongodb-pv created
persistentvolumeclaim/mongodb-pvc created
deployment.apps/mongodb-deployment configured
shagos90499@cloudshell:~/mongodb (cs570-big-data-424809)$
```

4. **Check Deployment Status:**

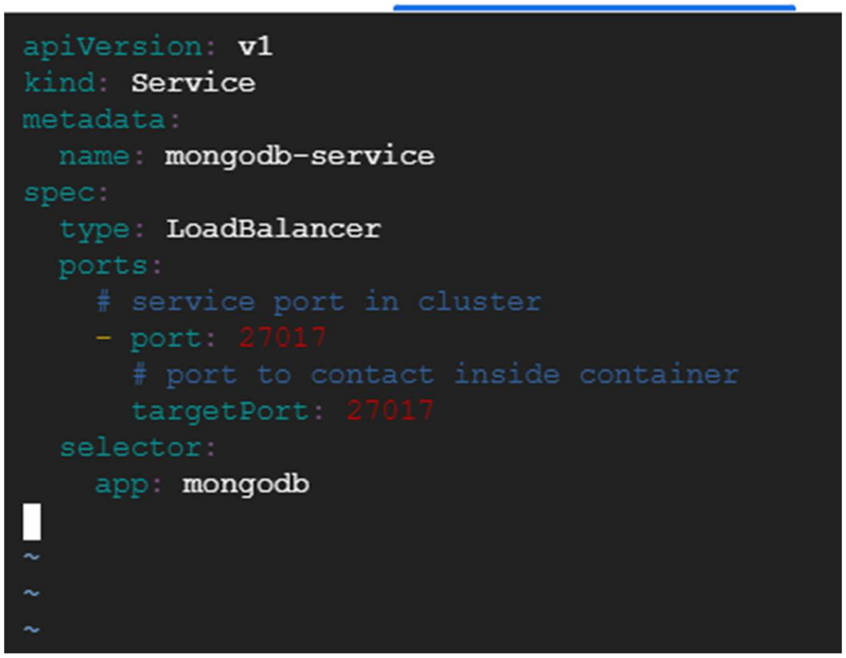
```
kubectl get pods
```



```
shagos90499@cloudshell:~/mongodb (cs570-big-data-424809)$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
mongodb-deployment-5c589898cb-2vsqp   1/1     Running   0          102s
shagos90499@cloudshell:~/mongodb (cs570-big-data-424809)$
```

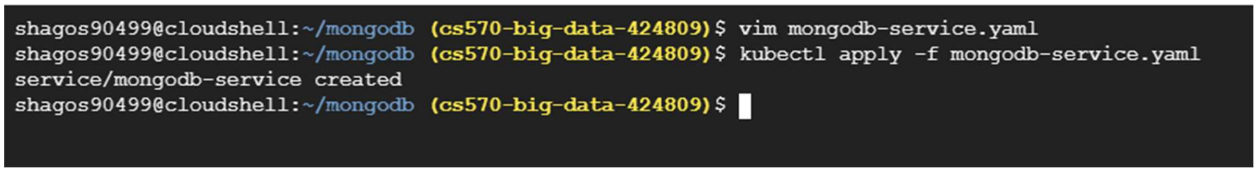
Ensure the pod status is `Running`.

5. **Create MongoDB Service:** Apply the `mongodb-service.yaml` configuration:



```
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
```

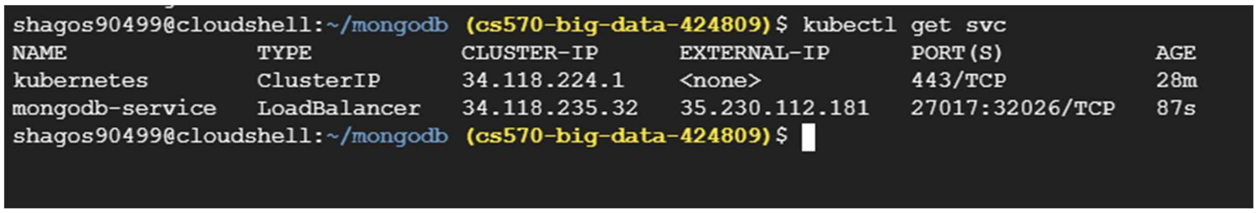


```
shagos90499@cloudshell:~/mongodb (cs570-big-data-424809)$ vim mongodb-service.yaml
shagos90499@cloudshell:~/mongodb (cs570-big-data-424809)$ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
shagos90499@cloudshell:~/mongodb (cs570-big-data-424809)$
```

6. **Verify Service Status:**

```
kubectl get svc
```

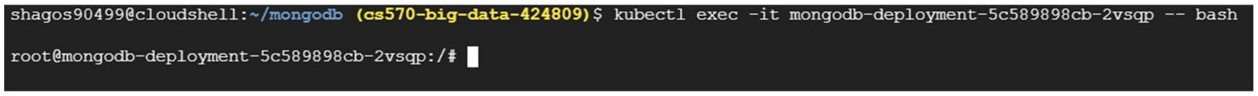
Wait for the `EXTERNAL-IP` to be assigned.



```
shagos90499@cloudshell:~/mongodb (cs570-big-data-424809)$ kubectl get svc
NAME              TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)           AGE
kubernetes        ClusterIP      34.118.224.1    <none>           443/TCP           28m
mongodb-service   LoadBalancer   34.118.235.32   35.230.112.181   27017:32026/TCP   87s
shagos90499@cloudshell:~/mongodb (cs570-big-data-424809)$
```

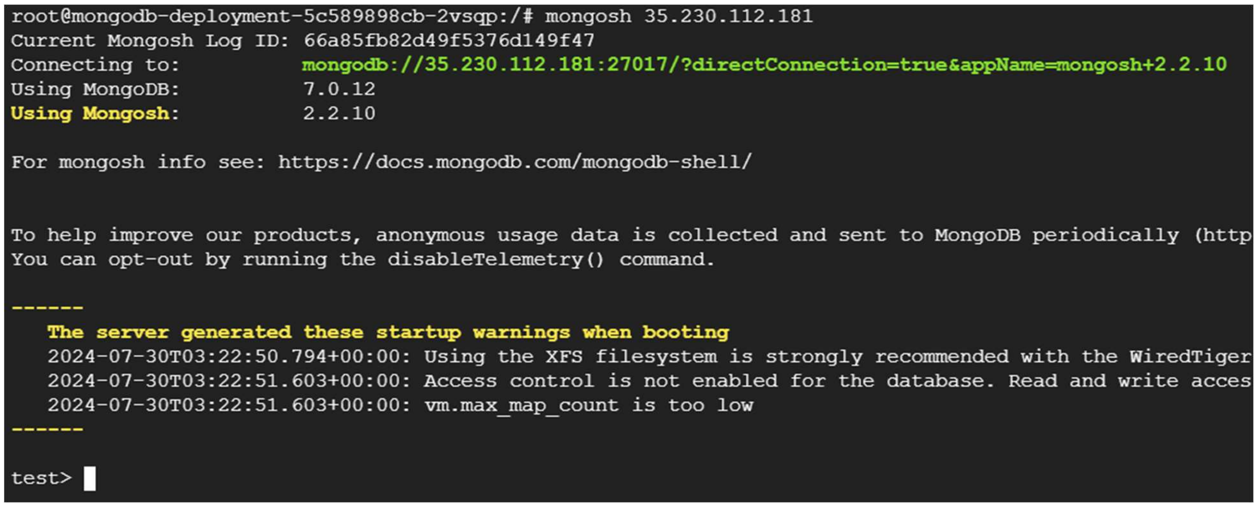
7. **Test MongoDB Connection:** Access the MongoDB pod and connect:

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



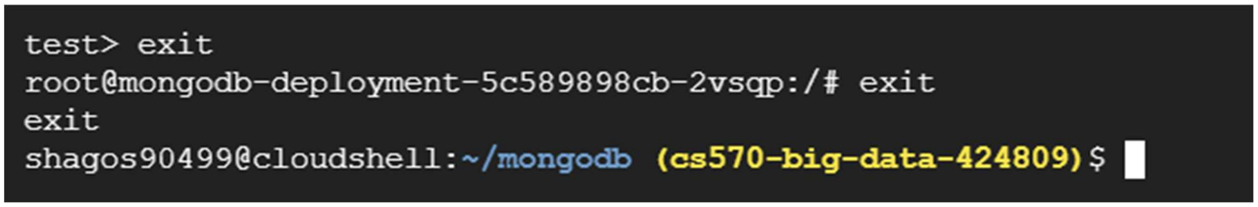
```
shagos90499@cloudshell:~/mongodb (cs570-big-data-424809)$ kubectl exec -it mongodb-deployment-5c589898cb-2vsqp -- bash
root@mongodb-deployment-5c589898cb-2vsqp:/#
```

You should see the MongoDB shell prompt. Test connectivity with:



```
root@mongodb-deployment-5c589898cb-2vsqp:/# mongosh 35.230.112.181
Current Mongosh Log ID: 66a85fb82d49f5376d149f47
Connecting to:          mongodb://35.230.112.181:27017/?directConnection=true&appName=mongosh+2.2.10
Using MongoDB:          7.0.12
Using Mongosh:          2.2.10

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 (http
You can opt-out by running the disableTelemetry() command.

------
   The server generated these startup warnings when booting
   2024-07-30T03:22:50.794+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger
   2024-07-30T03:22:51.603+00:00: Access control is not enabled for the database. Read and write acces
   2024-07-30T03:22:51.603+00:00: vm.max_map_count is too low
------

test>
```

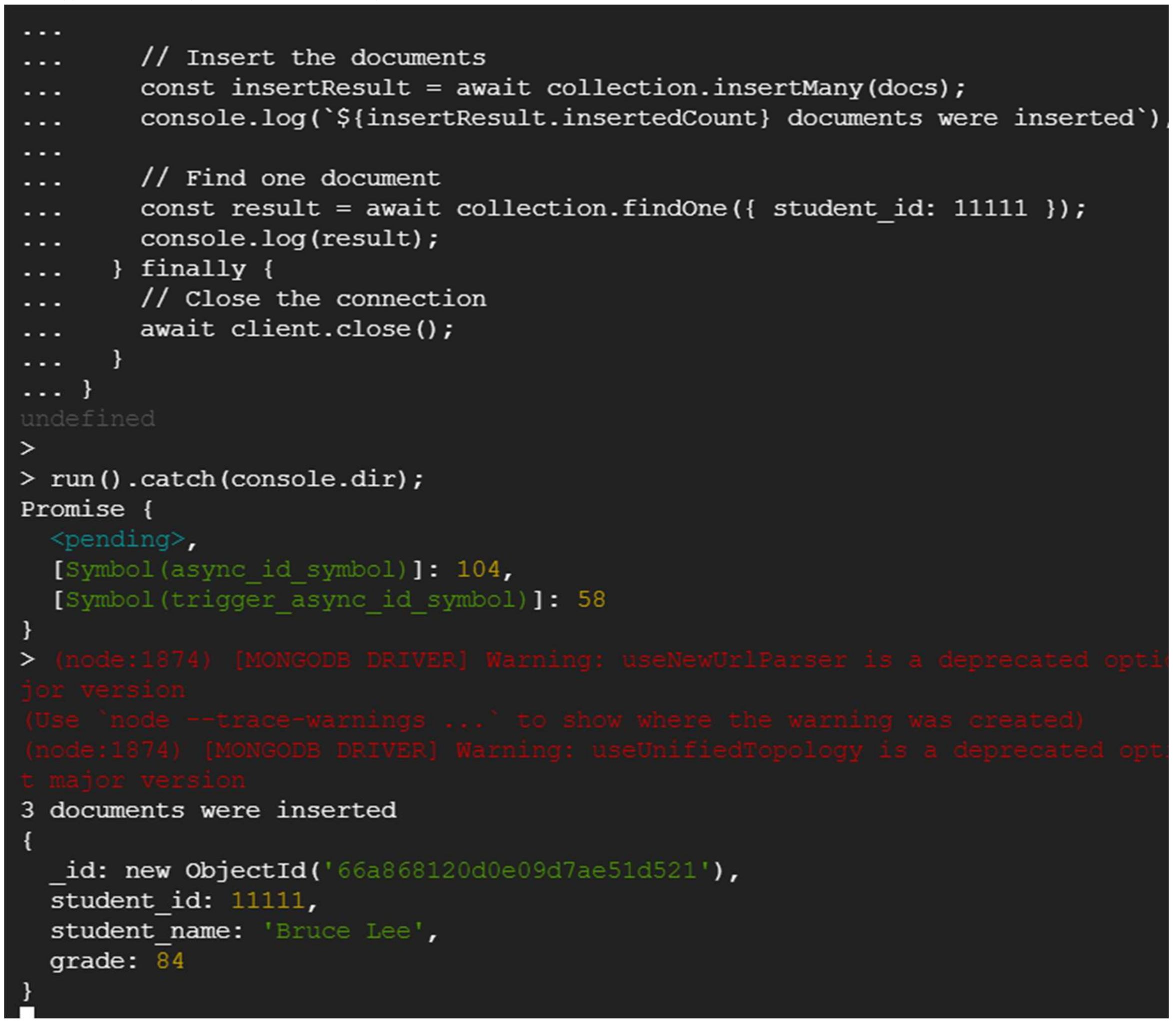
8. Type exit to exit from the shell and go back to the cloud shell



```
test> exit
root@mongodb-deployment-5c589898cb-2vsqp:/# exit
exit
shagos90499@cloudshell:~/mongodb (cs570-big-data-424809)$
```

9. **Insert Records into MongoDB:** we need to insert some data into MongoDB. Execute the following code in a Node.js environment:

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

async function run() {
  const url = "mongodb://35.230.112.181/studentdb"; // Use the correct
IP and port
  const client = new MongoClient(url, { useNewUrlParser: true,
useUnifiedTopology: true });

  try {
    // Connect to the MongoDB cluster
    await client.connect();

    // Specify the database and collection
    const db = client.db("studentdb");
    const collection = db.collection("students");

    // Create documents to be inserted
    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 the documents
    const insertResult = await collection.insertMany(docs);
    console.log(`${insertResult.insertedCount} documents were
inserted`);

    // Find one document
    const result = await collection.findOne({ student_id: 11111 });
    console.log(result);
  } finally {
    // Close the connection
    await client.close();
  }
}
run().catch(console.dir);
```



```
...
...     // Insert the documents
...     const insertResult = await collection.insertMany(docs);
...     console.log(`${insertResult.insertedCount} documents were inserted`)
...
...     // Find one document
...     const result = await collection.findOne({ student_id: 11111 });
...     console.log(result);
...   } finally {
...     // Close the connection
...     await client.close();
...   }
... }
undefined
>
> run().catch(console.dir);
Promise {
  <pending>,
  [Symbol(async_id_symbol)]: 104,
  [Symbol(trigger_async_id_symbol)]: 58
}
> (node:1874) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated opti
jor version
(Use `node --trace-warnings ...` to show where the warning was created)
(node:1874) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated opt
t major version
3 documents were inserted
{
  _id: new ObjectId('66a868120d0e09d7ae51d521'),
  student_id: 11111,
  student_name: 'Bruce Lee',
  grade: 84
}
```

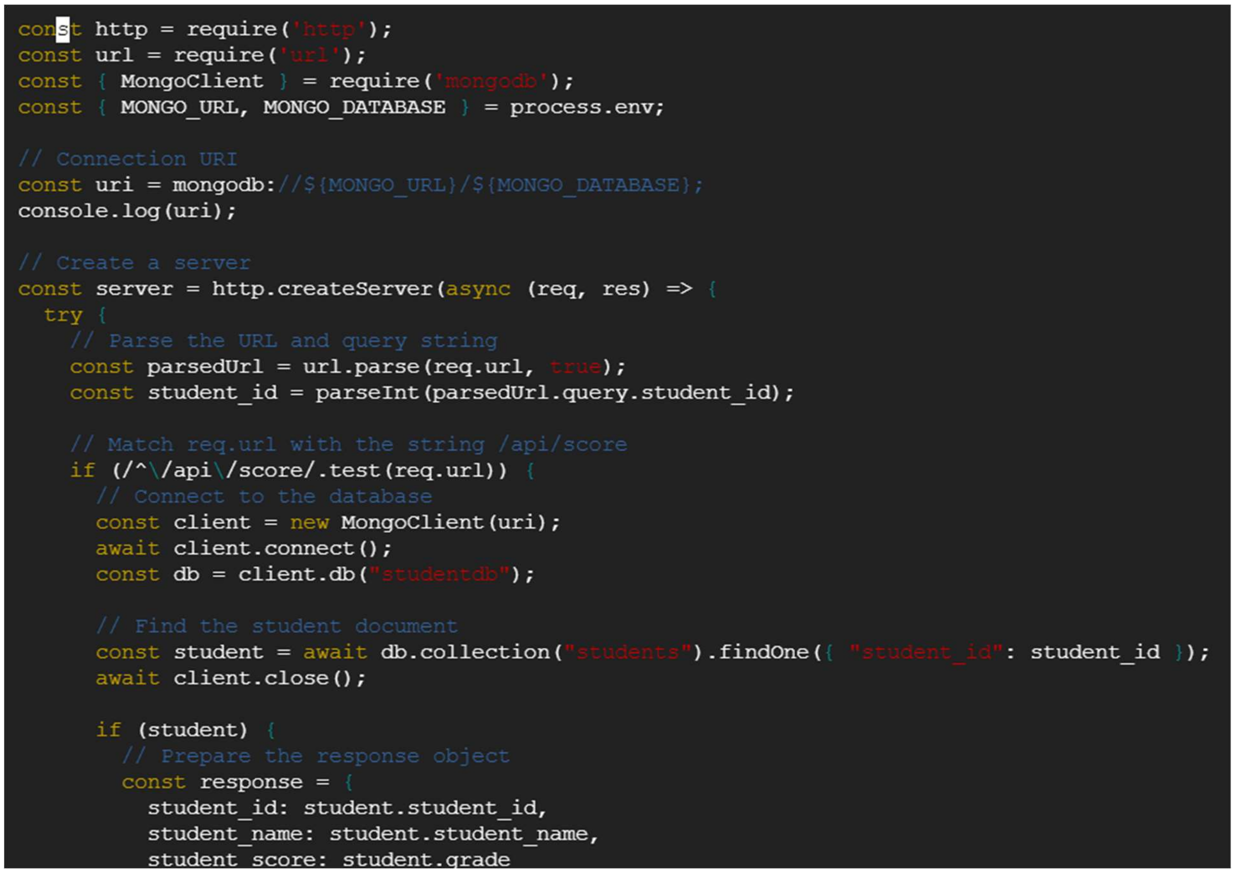
Then we can see that 3 rows were inserted into the database and we could retrieve student data with id 11111.

Step 2: Modify StudentServer to Fetch Records from MongoDB and Deploy to GKE

1. **Create `studentServer.js`:**

```
const http = require('http');
const url = require('url');
const { MongoClient } = require('mongodb');
const { MONGO_URL, MONGO_DATABASE } = process.env;
// Connection URI
const uri = mongodb://${MONGO_URL}/${MONGO_DATABASE};
console.log(uri);

// Create a server
const server = http.createServer(async (req, res) => {
  try {
    // Parse the URL and query string
    const parsedUrl = url.parse(req.url, true);
    const student_id = parseInt(parsedUrl.query.student_id);

    // Match req.url with the string /api/score
    if (/^\/api\/score/.test(req.url)) {
      // Connect to the database
      const client = new MongoClient(uri);
      await client.connect();
      const db = client.db("studentdb");

      // Find the student document
      const student = await db.collection("students").findOne({
"student_id": student_id });
      await client.close();

      if (student) {
        // Prepare the response object
        const response = {
          student_id: student.student_id,
          student_name: student.student_name,
          student_score: student.grade
        };

        // Send the response
        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");
    }
  } catch (err) {
    console.error(err);
    res.writeHead(500);
    res.end("Internal Server Error\n");
  }
});

// Start the server
```

```
server.listen(8080, () => {
  console.log('Server is listening on port 8080');
});
```



```
const http = require('http');
const url = require('url');
const { MongoClient } = require('mongodb');
const { MONGO_URL, MONGO_DATABASE } = process.env;

// Connection URI
const uri = mongodb://${MONGO_URL}/${MONGO_DATABASE};
console.log(uri);

// Create a server
const server = http.createServer(async (req, res) => {
  try {
    // Parse the URL and query string
    const parsedUrl = url.parse(req.url, true);
    const student_id = parseInt(parsedUrl.query.student_id);

    // Match req.url with the string /api/score
    if (/^\/api\/score/.test(req.url)) {
      // Connect to the database
      const client = new MongoClient(uri);
      await client.connect();
      const db = client.db("studentdb");

      // Find the student document
      const student = await db.collection("students").findOne({ "student_id": student_id });
      await client.close();

      if (student) {
        // Prepare the response object
        const response = {
          student_id: student.student_id,
          student_name: student.student_name,
          student_score: student.grade
```

2. **Create Dockerfile: this will create a newer version of node js, install all the dependencies needed from a json file.**

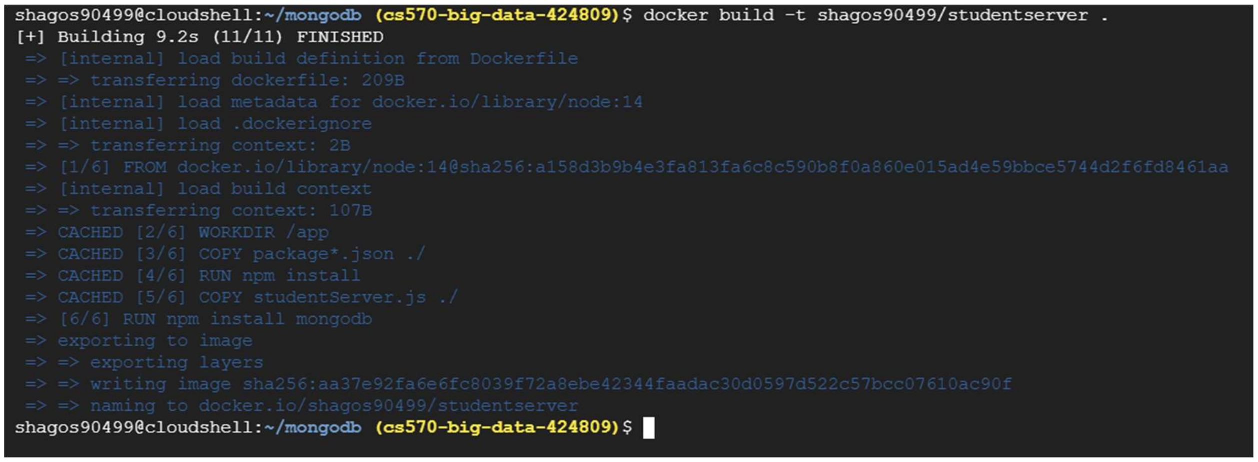
```
FROM node:14
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY studentServer.js ./
EXPOSE 8080
ENTRYPOINT ["node", "studentServer.js"]
```

package.json

```
{
  "name": "studentserver",
  "version": "1.0.0",
  "description": "Student Server",
  "main": "studentServer.js",
  "scripts": {
    "start": "node studentServer.js"
  },
  "dependencies": {
    "mongodb": "^4.0.0",
    "http": "0.0.1-security"
    }
}
```

3. **Build Docker Image:**

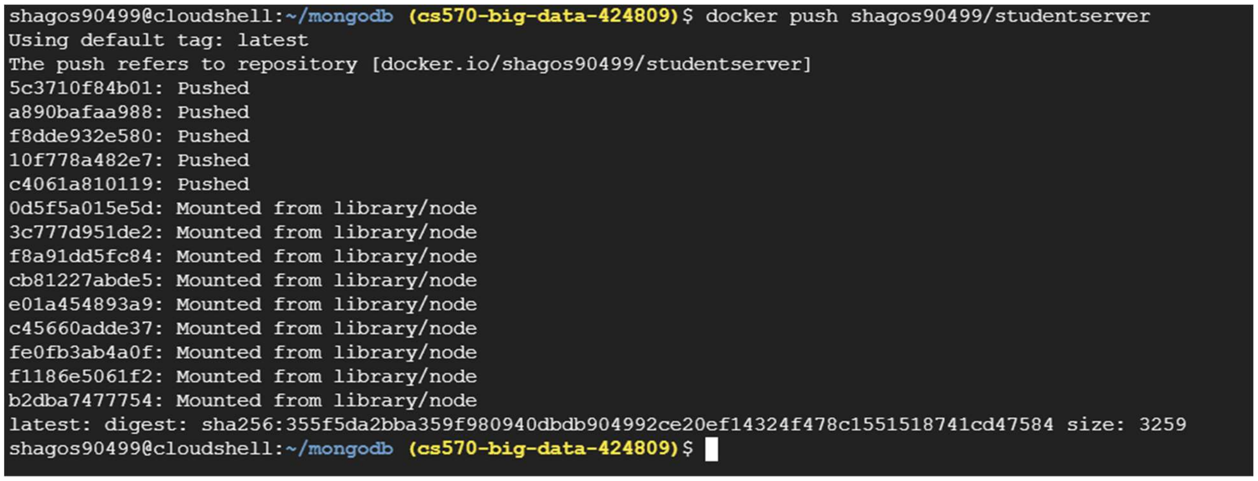
```
docker build -t yourdockerhubID/studentserver .
```



```
shagos90499@cloudshell:~/mongodb (cs570-big-data-424809)$ docker build -t shagos90499/studentserver .
[+] Building 9.2s (11/11) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 209B
 => [internal] load metadata for docker.io/library/node:14
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/6] FROM docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461aa
 => [internal] load build context
 => => transferring context: 107B
 => CACHED [2/6] WORKDIR /app
 => CACHED [3/6] COPY package*.json ./
 => CACHED [4/6] RUN npm install
 => CACHED [5/6] COPY studentServer.js ./
 => [6/6] RUN npm install mongodb
 => exporting to image
 => => exporting layers
 => => writing image sha256:aa37e92fa6e6fc8039f72a8ebe42344faadac30d0597d522c57bcc07610ac90f
 => => naming to docker.io/shagos90499/studentserver
shagos90499@cloudshell:~/mongodb (cs570-big-data-424809)$
```

4. **Push Docker Image to Docker Hub:**

```
docker push yourdockerhubID/studentserver
```



```
shagos90499@cloudshell:~/mongodb (cs570-big-data-424809)$ docker push shagos90499/studentserver
Using default tag: latest
The push refers to repository [docker.io/shagos90499/studentserver]
5c3710f84b01: Pushed
a890bafaa988: Pushed
f8dde932e580: Pushed
10f778a482e7: Pushed
c4061a810119: 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: sha256:355f5da2bba359f980940dbdb904992ce20ef14324f478c1551518741cd47584 size: 3259
shagos90499@cloudshell:~/mongodb (cs570-big-data-424809)$
```

Step 3: Create the Flask Application

1. **Create `bookshelf.py`:**

```
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")+"/"+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_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 saved 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"]
    }})
    message = "Book updated successfully!" if response.matched_count
else "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)})
    message = "Book deleted successfully!" if response.deleted_count
else "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 `requirements.txt`**:

```
Flask==2.2.2
Flask-PyMongo==2.3.0
PyMongo>=3.3
```

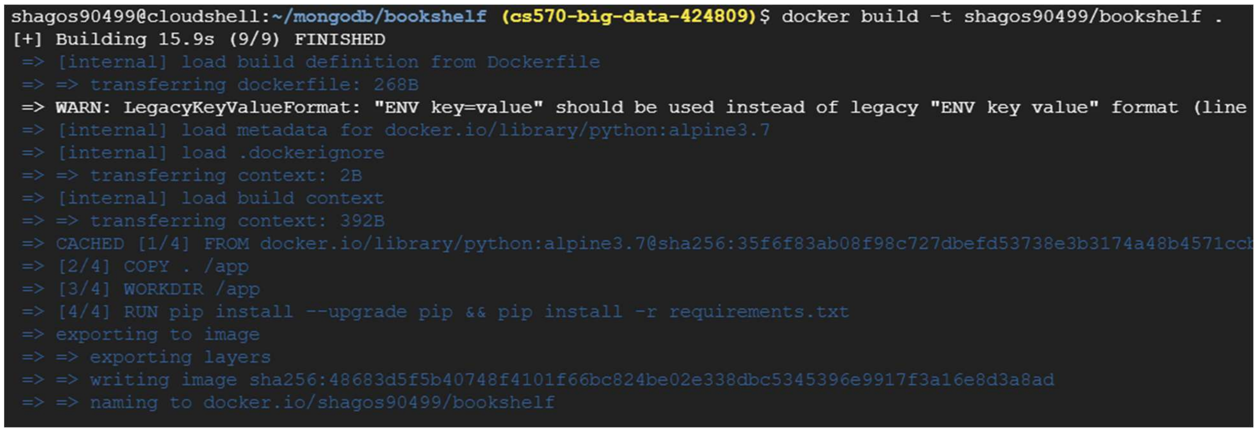
3. **Dockerize the Flask Application**

Create `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"]
```

1. **Build the Docker Image**:

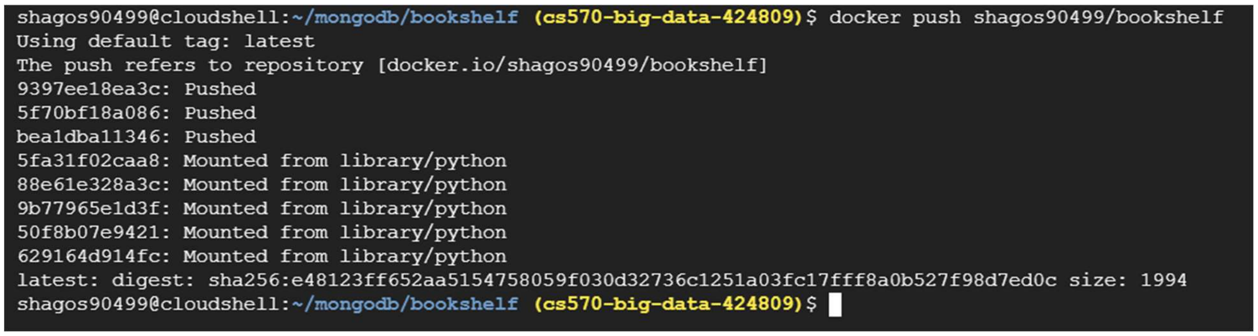
```
docker build -t shagos90499/bookshelf .
```



```
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ docker build -t shagos90499/bookshelf .
[+] Building 15.9s (9/9) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 268B
 => WARN: LegacyKeyValueFormat: "ENV key=value" should be used instead of legacy "ENV key value" format (line
 => [internal] load metadata for docker.io/library/python:alpine3.7
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load build context
 => => transferring context: 392B
 => CACHED [1/4] FROM docker.io/library/python:alpine3.7@sha256:35f6f83ab08f98c727dbefd53738e3b3174a48b4571cc
 => [2/4] COPY . /app
 => [3/4] WORKDIR /app
 => [4/4] RUN pip install --upgrade pip && pip install -r requirements.txt
 => exporting to image
 => => exporting layers
 => => writing image sha256:48683d5f5b40748f4101f66bc824be02e338dbc5345396e9917f3a16e8d3a8ad
 => => naming to docker.io/shagos90499/bookshelf
```

2. **Push the Docker Image to Docker Hub**:

```
docker push zhou19539/bookshelf
```



```
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ docker push shagos90499/bookshelf
Using default tag: latest
The push refers to repository [docker.io/shagos90499/bookshelf]
9397ee18ea3c: Pushed
5f70bf18a086: Pushed
bea1dba11346: 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:e48123ff652aa5154758059f030d32736c1251a03fc17fff8a0b527f98d7ed0c size: 1994
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$
```

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

1. **Create a ConfigMap for `studentServer`**

   Create a file named `studentserver-configmap.yaml` with the following content:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  MONGO_URL: Change-this-to-your-mongoDB-EXTERNAL-IP
  MONGO_DATABASE: mydb
```

2. **Create a ConfigMap for `bookshelf`**

   Create a file named `bookshelf-configmap.yaml` with the following content:

```
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
```

Notice:

- The reason for creating these two ConfigMaps is to avoid re-building the Docker image again if the MongoDB pod restarts with a different External-IP.

Step 5: Expose Two Applications Using Ingress with Nginx

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: zhou19539/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: zhou19539/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 `studentserver-service.yaml`**

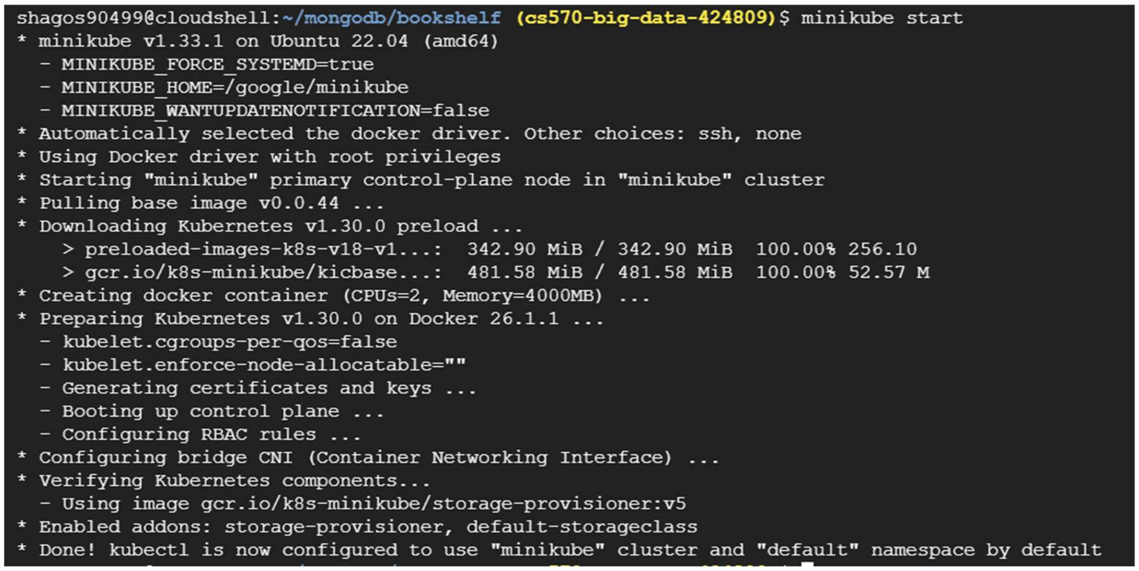
```
apiVersion: v1
kind: Service
metadata:
  name: web
spec:
  type: LoadBalancer
  ports:
  - port: 8080
    targetPort: 8080
  selector:
    app: web
```

4. **Create `bookshelf-service.yaml`**

```
apiVersion: v1
kind: Service
metadata:
  name: bookshelf-service
spec:
  type: LoadBalancer
  ports:
  - port: 5000
    targetPort: 5000
  selector:
    app: bookshelf-deployment
```

5. **Start Minikube**

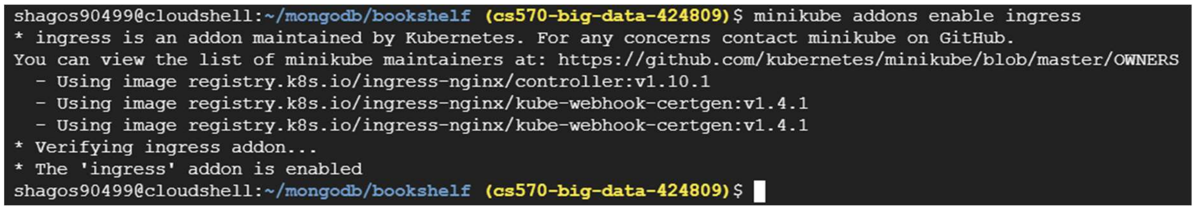
```
minikube start
```



```
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ minikube start
* minikube v1.33.1 on Ubuntu 22.04 (amd64)
  - MINIKUBE_FORCE_SYSTEMD=true
  - MINIKUBE_HOME=/google/minikube
  - MINIKUBE_WANTUPDATENOTIFICATION=false
* Automatically selected the docker driver. Other choices: ssh, none
* Using Docker driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.44 ...
* Downloading Kubernetes v1.30.0 preload ...
    > preloaded-images-k8s-v18-v1...:  342.90 MiB / 342.90 MiB  100.00% 256.10
    > gcr.io/k8s-minikube/kicbase...:  481.58 MiB / 481.58 MiB  100.00% 52.57 M
* Creating docker container (CPUs=2, Memory=4000MB) ...
* Preparing Kubernetes v1.30.0 on Docker 26.1.1 ...
  - 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) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

6. **Start Ingress**

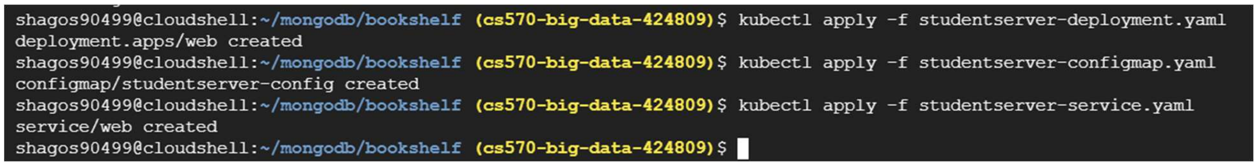
```
minikube addons enable ingress
```



```
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ 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:v1.10.1
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.1
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.1
* Verifying ingress addon...
* The 'ingress' addon is enabled
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$
```

7. **Create `studentserver` Related Pods and Start Service**

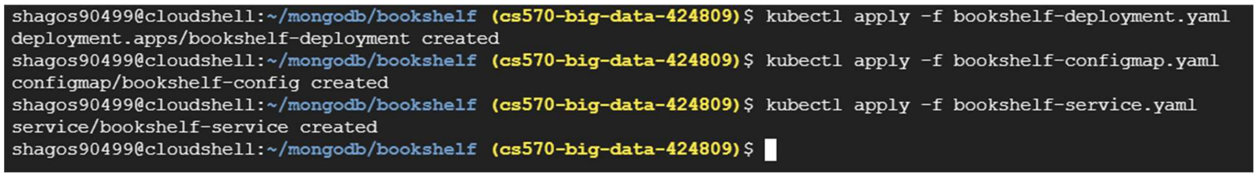
```
kubectl apply -f studentserver-deployment.yaml
kubectl apply -f studentserver-configmap.yaml
kubectl apply -f studentserver-service.yaml
```



```
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ kubectl apply -f studentserver-deployment.yaml
deployment.apps/web created
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ kubectl apply -f studentserver-configmap.yaml
configmap/studentserver-config created
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ kubectl apply -f studentserver-service.yaml
service/web created
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$
```

8. **Create `bookshelf` Related Pods and Start Service**

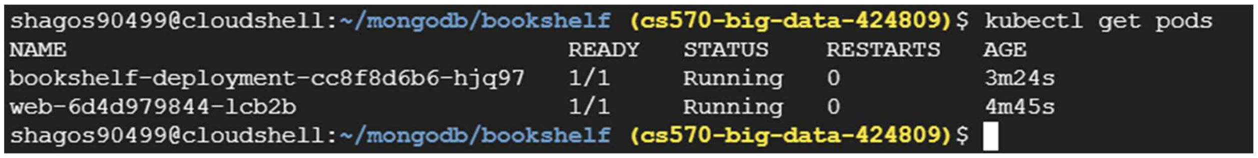
```
kubectl apply -f bookshelf-deployment.yaml
kubectl apply -f bookshelf-configmap.yaml
kubectl apply -f bookshelf-service.yaml
```



```
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$
```

9. **Check if All Pods are Running Correctly**

```
kubectl get pods
```



```
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ kubectl get pods
NAME                                   READY   STATUS    RESTARTS   AGE
bookshelf-deployment-cc8f8d6b6-hjq97   1/1     Running   0          3m24s
web-6d4d979844-lcb2b                   1/1     Running   0          4m45s
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$
```

10. **Create Ingress Service YAML File**

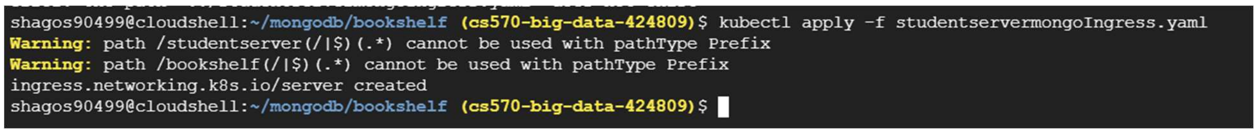
Create a file named `studentservermongoIngress.yaml` with the following content:

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

```
          pathType: Prefix
          backend:
            service:
              name: bookshelf-service
              port:
                number: 5000
```

11. **Create the Ingress Service**

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

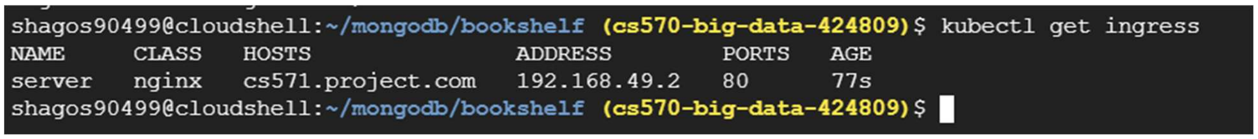


```
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ kubectl apply -f studentservermongoIngress.yaml
Warning: path /studentserver(/|$)(.*) cannot be used with pathType Prefix
Warning: path /bookshelf(/|$)(.*) cannot be used with pathType Prefix
ingress.networking.k8s.io/server created
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$
```

12. **Check if Ingress is Running**

```
kubectl get ingress
```

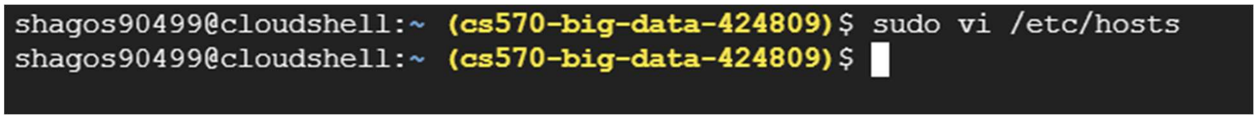
Wait until you see the `Address`, then move forward.



```
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ kubectl get ingress
NAME     CLASS   HOSTS                ADDRESS        PORTS   AGE
server   nginx   cs571.project.com    192.168.49.2   80      77s
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$
```

13. **Add Address to `/etc/hosts`**

```
sudo vi /etc/hosts
```



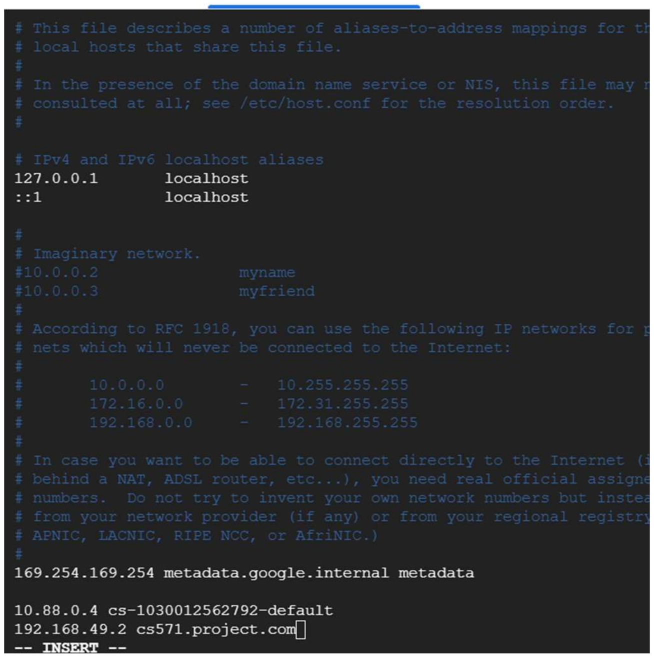
```
shagos90499@cloudshell:~ (cs570-big-data-424809)$ sudo vi /etc/hosts
shagos90499@cloudshell:~ (cs570-big-data-424809)$
```

Add the address you got from the previous step to the end of the file:

```
Your-address cs571.project.com
```

Your `/etc/hosts` file should look something like this after adding the line, but your address should be different from mine:

```
192.168.49.2 cs571.project.com
```

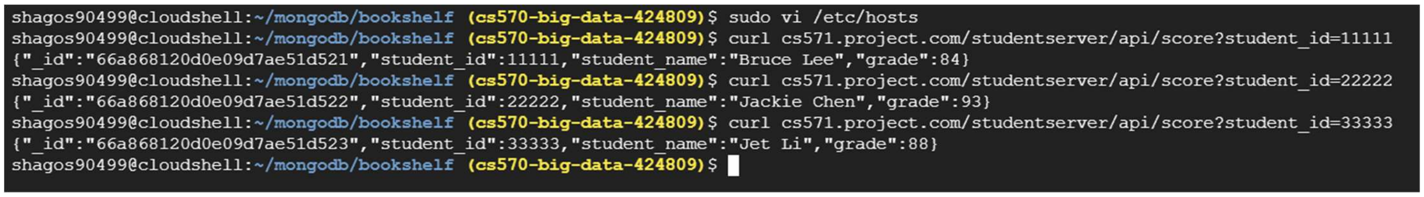


```
# This file describes a number of aliases-to-address mappings for th
# local hosts that share this file.
#
# In the presence of the domain name service or NIS, this file may n
# consulted at all; see /etc/host.conf for the resolution order.
#

# IPv4 and IPv6 localhost aliases
127.0.0.1       localhost
::1             localhost

#
# Imaginary network.
#10.0.0.2               myname
#10.0.0.3               myfriend
#
# According to RFC 1918, you can use the following IP networks for p
# nets which will never be connected to the Internet:
#
#       10.0.0.0        -   10.255.255.255
#       172.16.0.0      -   172.31.255.255
#       192.168.0.0     -   192.168.255.255
#
# In case you want to be able to connect directly to the Internet (i
# behind a NAT, ADSL router, etc...), you need real official assigne
# numbers.  Do not try to invent your own network numbers but instea
# from your network provider (if any) or from your regional registry
# APNIC, LACNIC, RIPE NCC, or AfriNIC.)
#
169.254.169.254 metadata.google.internal metadata

10.88.0.4 cs-1030012562792-default
192.168.49.2 cs571.project.com
-- INSERT --
```

14. **Test Your Applications**

- Retrieve all students from server: If everything goes smoothly, you should be able to access your applications

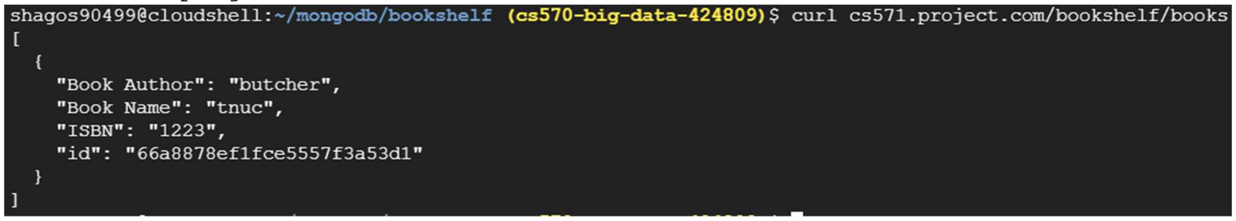
  `curl cs571.project.com/studentserver/api/score?student_id=11111`



```
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ sudo vi /etc/hosts
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ curl cs571.project.com/studentserver/api/score?student_id=11111
{"_id":"66a868120d0e09d7ae51d521","student_id":11111,"student_name":"Bruce Lee","grade":84}
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ curl cs571.project.com/studentserver/api/score?student_id=22222
{"_id":"66a868120d0e09d7ae51d522","student_id":22222,"student_name":"Jackie Chen","grade":93}
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ curl cs571.project.com/studentserver/api/score?student_id=33333
{"_id":"66a868120d0e09d7ae51d523","student_id":33333,"student_name":"Jet Li","grade":88}
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$
```

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

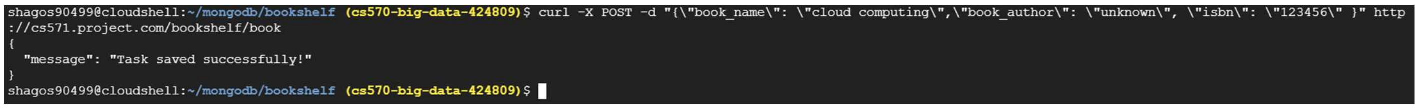
  `curl cs571.project.com/bookshelf/books`



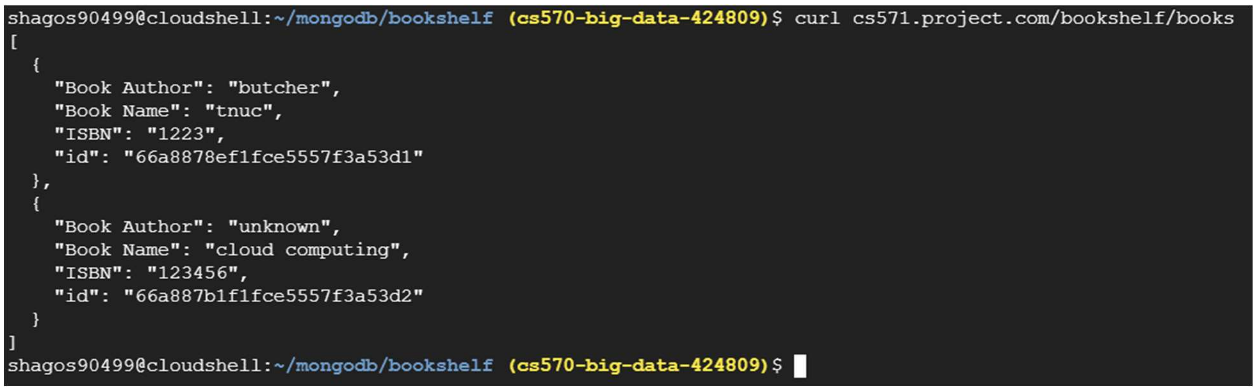
```
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "butcher",
    "Book Name": "tnuc",
    "ISBN": "1223",
    "id": "66a8878ef1fce5557f3a53d1"
  }
]
```

- Add a book:

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



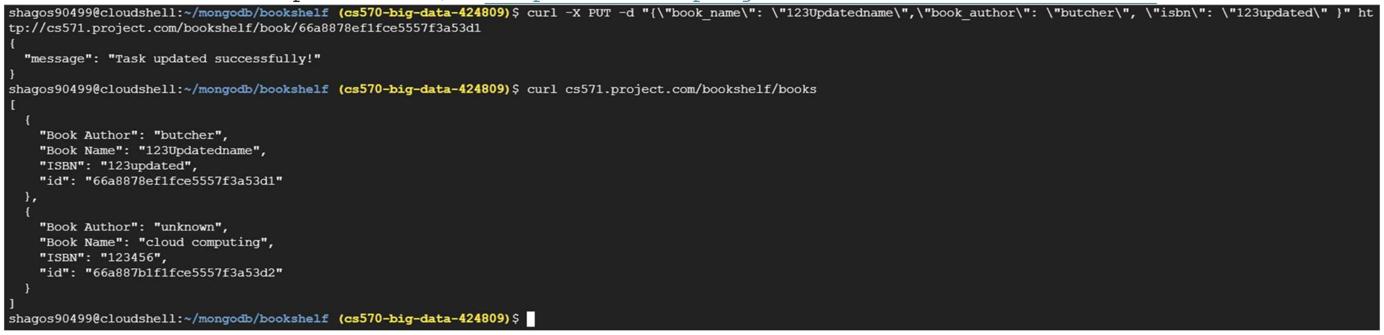
```
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ curl -X POST -d "{\"book_name\": \"cloud computing\",\"book_author\": \"unknown\", \"isbn\": \"123456\" }" http
://cs571.project.com/bookshelf/book
{
  "message": "Task saved successfully!"
}
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$
```



```
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "butcher",
    "Book Name": "tnuc",
    "ISBN": "1223",
    "id": "66a8878ef1fce5557f3a53d1"
  },
  {
    "Book Author": "unknown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "66a887b1f1fce5557f3a53d2"
  }
]
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$
```

- Update a book:

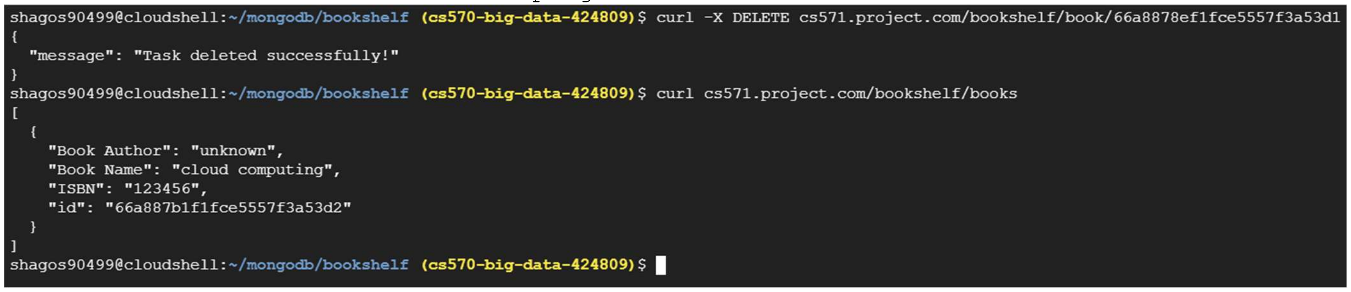
```
curl -X PUT -d "{\"book_name\":
\"123Updatedname\",\"book_author\": \"butcher\", \"isbn\":
\"123updated\" }" http://cs571.project.com/bookshelf/book/id
```



```
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ curl -X PUT -d "{\"book_name\": \"123Updatedname\",\"book_author\": \"butcher\", \"isbn\": \"123updated\" }" ht
tp://cs571.project.com/bookshelf/book/66a8878ef1fce5557f3a53d1
{
  "message": "Task updated successfully!"
}
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "butcher",
    "Book Name": "123Updatedname",
    "ISBN": "123updated",
    "id": "66a8878ef1fce5557f3a53d1"
  },
  {
    "Book Author": "unknown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "66a887b1f1fce5557f3a53d2"
  }
]
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$
```

- Delete a book:

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



```
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ curl -X DELETE cs571.project.com/bookshelf/book/66a8878ef1fce5557f3a53d1
{
  "message": "Task deleted successfully!"
}
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "unknown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "66a887b1f1fce5557f3a53d2"
  }
]
shagos90499@cloudshell:~/mongodb/bookshelf (cs570-big-data-424809)$
```