Kubernetes Signature Project

Present By: Nisarg Patel

Student Id: 19619

Table Of Contents

Introduction

The project has consist two applications running on different kubernetes pods. **Student record application** running on Node.js + MongoDB + Google kubernetes Engine (GKE) technologies

Bookstore application, using with MongoDB + Python Flask Web Framework + REST API + GKE technologies.

Design

Both applications have same domain name but access in different path with used of Kubernetes Ingress component

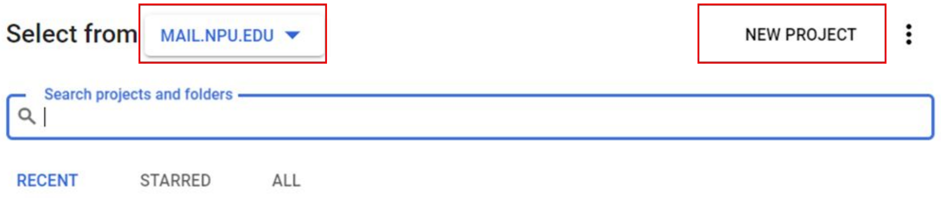
Domain name: **cs571.signatureproject.com**

Student record application: cs571.signatureproject.com/studentserver

Bookstore application: cs571.signatureproject.com/bookshelf

Implementation

Step 1 Create a new project on GCP for project work.



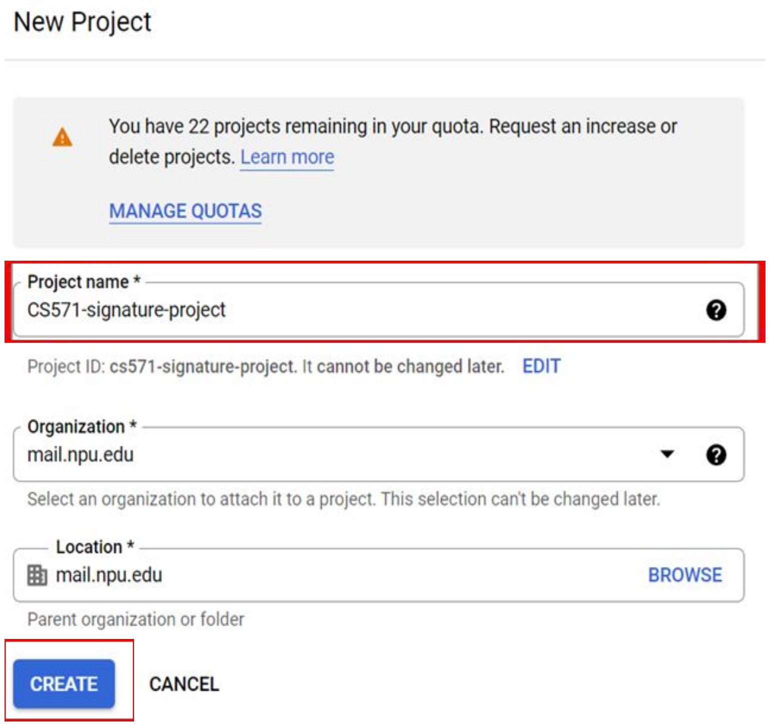
Select from MAIL.NPU.EDU

NEW PROJECT

Search projects and folders

RECENT STARRED ALL

Select the project that you created and open the GCP terminal windows.



New Project

You have 22 projects remaining in your quota. Request an increase or delete projects. Learn more

MANAGE QUOTAS

Project name *: CS571-signature-project

Project ID: cs571-signature-project. It cannot be changed later. EDIT

Organization *: mail.npu.edu

Select an organization to attach it to a project. This selection can't be changed later.

Location *: mail.npu.edu BROWSE

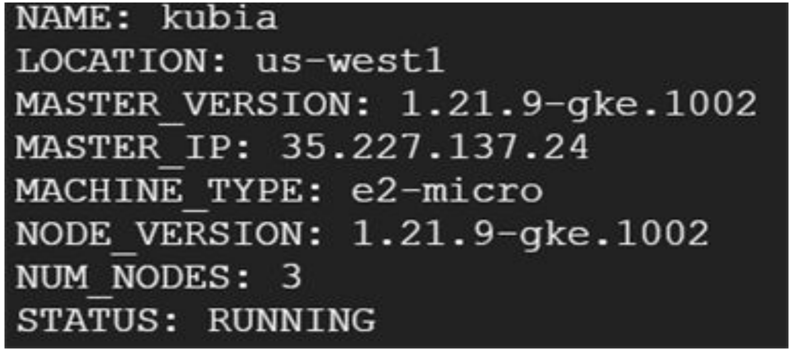
Parent organization or folder

CREATE CANCEL

Implementation

First, launch the kubia cluster

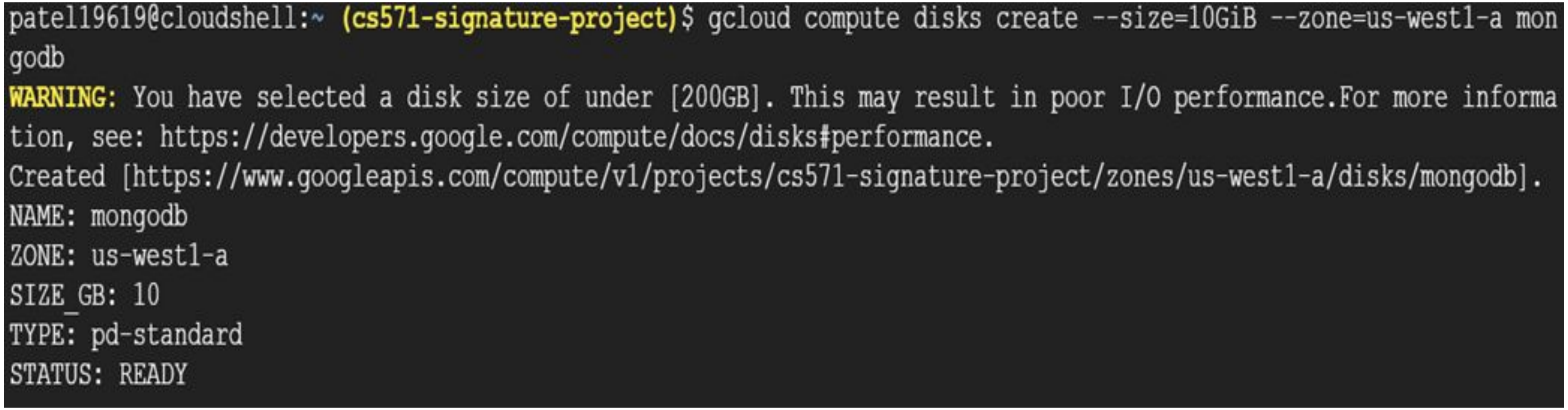
```
gcloud container clusters create kubia --num-nodes=1 --machine-type=e2-micro --region=us-west1
```



```
NAME: kubia
LOCATION: us-west1
MASTER_VERSION: 1.21.9-gke.1002
MASTER_IP: 35.227.137.24
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.21.9-gke.1002
NUM_NODES: 3
STATUS: RUNNING
```

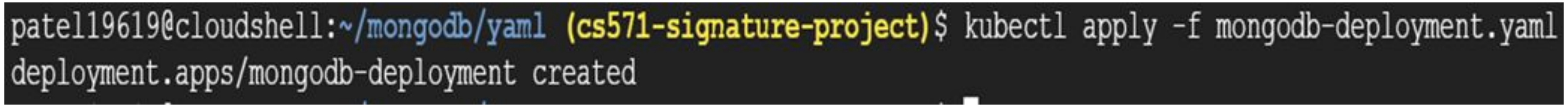
Implementation

Create a GCE Persistent Disk **mongodb** with size of 10GiB.



```
patel19619@cloudshell:~ (cs571-signature-project)$ gcloud compute disks create --size=10GiB --zone=us-west1-a mon
godb
WARNING: You have selected a disk size of under [200GB]. This may result in poor I/O performance.For more informa
tion, see: https://developers.google.com/compute/docs/disks#performance.
Created [https://www.googleapis.com/compute/v1/projects/cs571-signature-project/zones/us-west1-a/disks/mongodb].
NAME: mongodb
ZONE: us-west1-a
SIZE_GB: 10
TYPE: pd-standard
STATUS: READY
```

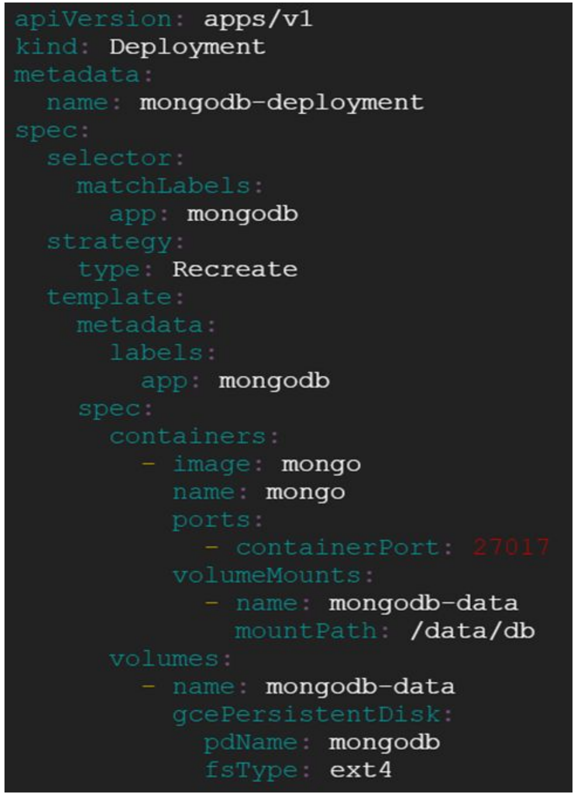
Create a Mongodb-deployment pod on Kubernetes



```
patel19619@cloudshell:~/mongodb/yaml (cs571-signature-project)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
```

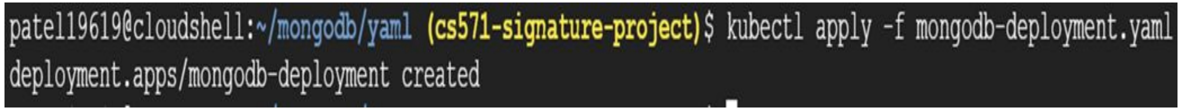
Implementation

Create a Mongodb-deployment.yaml



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

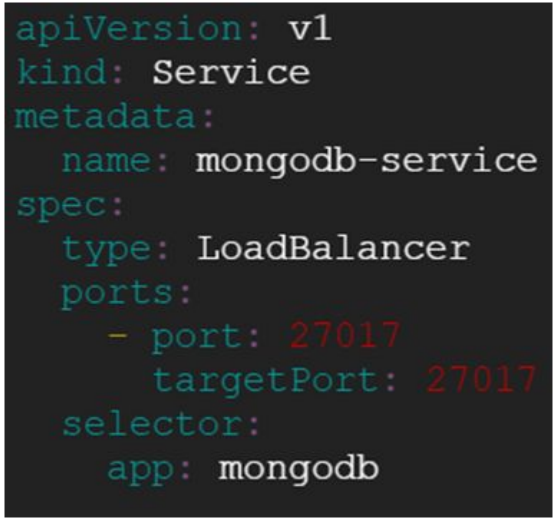
Create a Mongodb-deployment pod on Kubernetes



```
patel19619@cloudshell:~/mongodb/yaml (cs571-signature-project)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
```

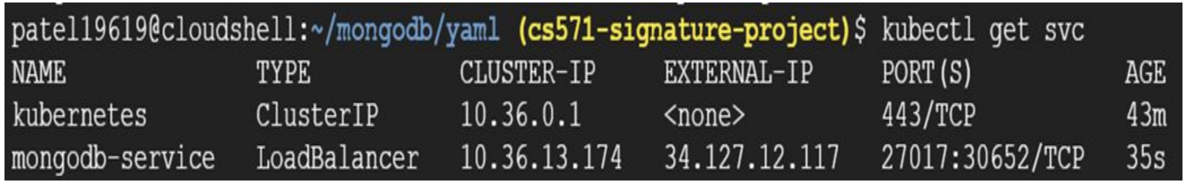
Implementation

Create mongodb service



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

Check the mongodb-service is created and running.



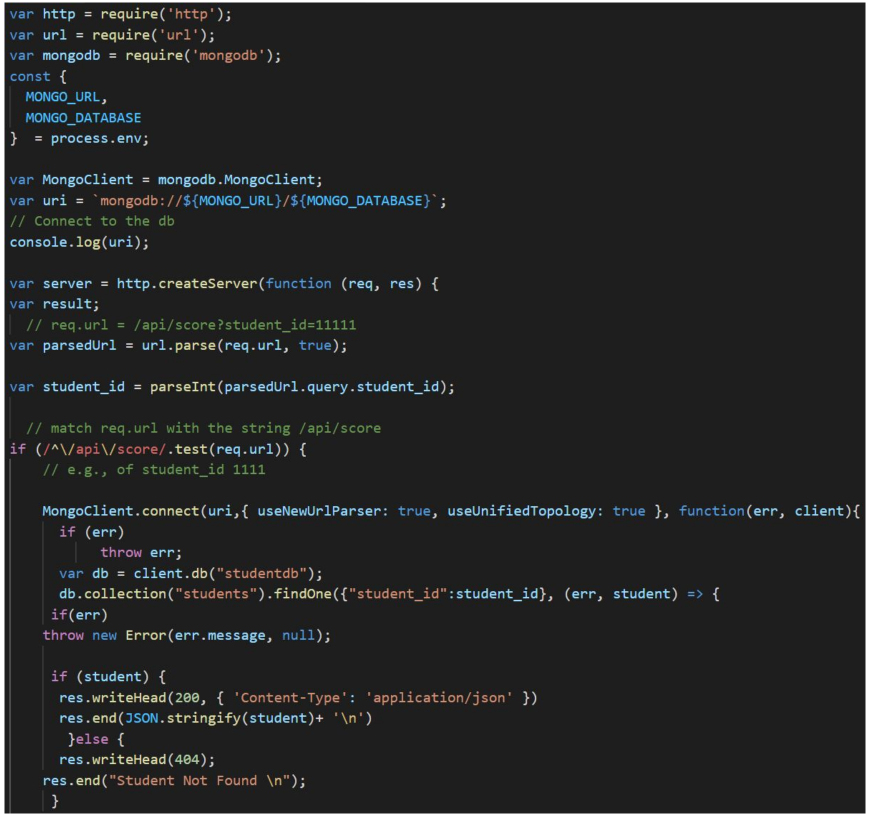
```
patel19619@cloudshell:~/mongodb/yaml (cs571-signature-project)$ kubectl get svc
NAME              TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)             AGE
kubernetes        ClusterIP      10.36.0.1      <none>          443/TCP             43m
mongodb-service   LoadBalancer   10.36.13.174   34.127.12.117   27017:30652/TCP     35s
```

Successfully launched the mongodb database service on gcp.

Implementation

Follow the same procedure to do configuration of student server application and bookshelf python application.

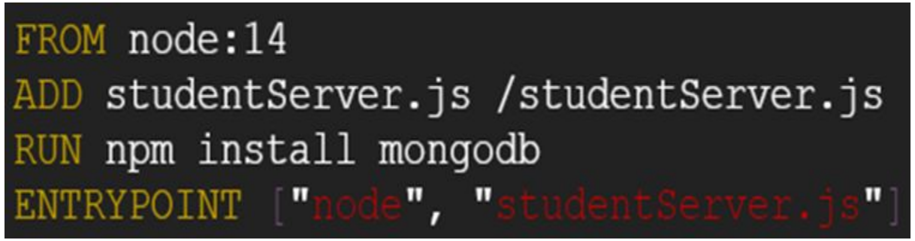
Create the **studentServer.js** file with working code of fetch the student information.



```
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}`;
// Connect to the db
console.log(uri);

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

var student_id = parseInt(parsedUrl.query.student_id);

  // match req.url with the string /api/score
if (/^\/api\/score/.test(req.url)) {
    // e.g., of student_id 1111

    MongoClient.connect(uri,{ useNewUrlParser: true, useUnifiedTopology: true }, function(err, client){
      if (err)
           throw err;
      var db = client.db("studentdb");
      db.collection("students").findOne({"student_id":student_id}, (err, student) => {
     if(err)
    throw new Error(err.message, null);

     if (student) {
      res.writeHead(200, { 'Content-Type': 'application/json' })
      res.end(JSON.stringify(student)+ '\n')
       }else {
      res.writeHead(404);
    res.end("Student Not Found \n");
     }
```

Implementation

vim Dockerfile

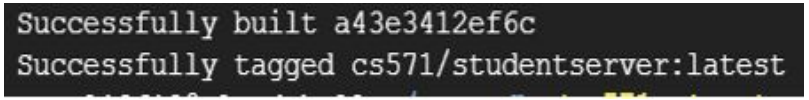


```
FROM node:14
ADD studentServer.js /studentServer.js
RUN npm install mongodb
ENTRYPOINT ["node", "studentServer.js"]
```

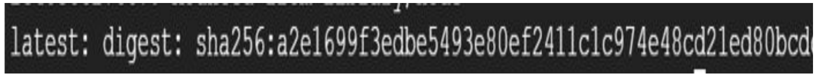
Build the student server docker image

docker build -t yourdockerhubID/studentserver .

docker push yourdockerhubID/studentserver



```
Successfully built a43e3412ef6c
Successfully tagged cs571/studentserver:latest
```

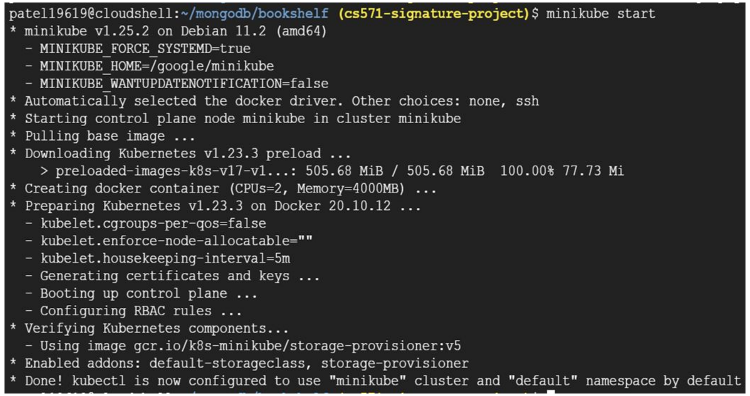


```
latest: digest: sha256:a2e1699f3edbe5493e80ef2411c1c974e48cd21ed80bcd
```

Implementation

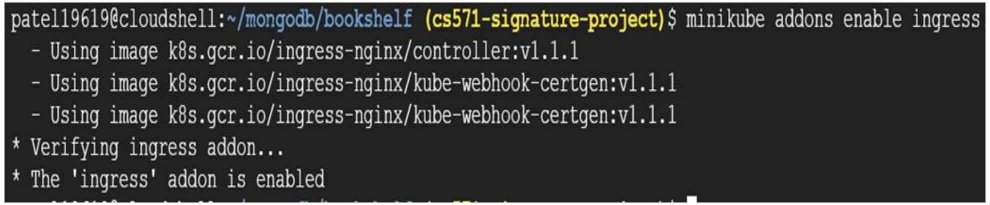
Start the single instance cluster to configure the application to host

minikube start



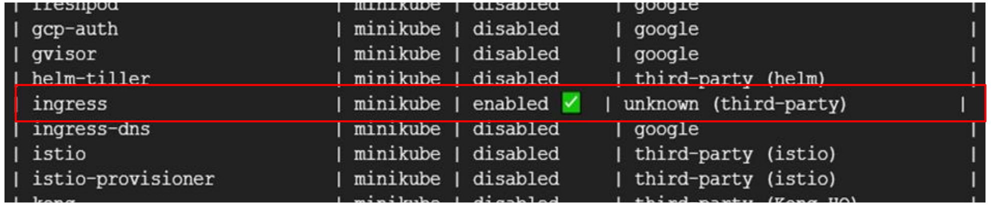
```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ minikube start
* minikube v1.25.2 on Debian 11.2 (amd64)
  - MINIKUBE_FORCE_SYSTEMD=true
  - MINIKUBE_HOME=/google/minikube
  - MINIKUBE_WANTUPDATENOTIFICATION=false
* Automatically selected the docker driver. Other choices: none, ssh
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Downloading Kubernetes v1.23.3 preload ...
    > preloaded-images-k8s-v17-v1...: 505.68 MiB / 505.68 MiB  100.00% 77.73 Mi
* Creating docker container (CPUs=2, Memory=4000MB) ...
* Preparing Kubernetes v1.23.3 on Docker 20.10.12 ...
  - kubelet.cgroups-per-qos=false
  - kubelet.enforce-node-allocatable=""
  - kubelet.housekeeping-interval=5m
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

minikube addons enable ingress



```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ minikube addons enable ingress
  - Using image k8s.gcr.io/ingress-nginx/controller:v1.1.1
  - Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
  - Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
* Verifying ingress addon...
* The 'ingress' addon is enabled
```

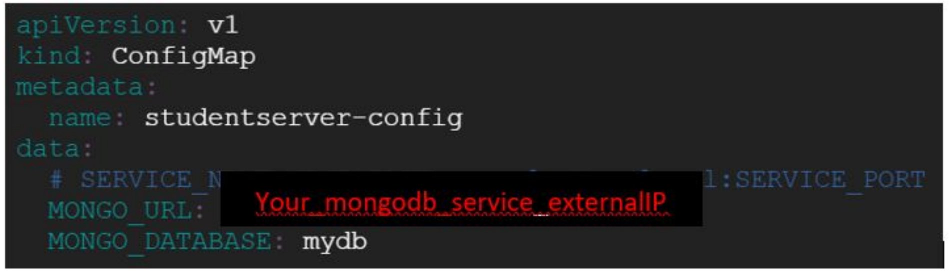
minikube addons list



```
| freshpod                    | minikube | disabled     | google                 |
| gcp-auth                    | minikube | disabled     | google                 |
| gvisor                      | minikube | disabled     | google                 |
| helm-tiller                 | minikube | disabled     | third-party (helm)     |
| ingress                     | minikube | enabled ✅   | unknown (third-party)  |
| ingress-dns                 | minikube | disabled     | google                 |
| istio                       | minikube | disabled     | third-party (istio)    |
| istio-provisioner           | minikube | disabled     | third-party (istio)    |
```

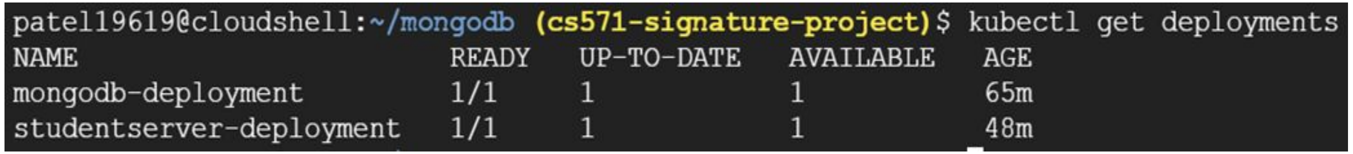
Implementation

Create ConfigMap to store mongodb configuration to avoid re-building docker image



```
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  # SERVICE_N[redacted]1:SERVICE_PORT
  MONGO_URL:  Your_mongodb_service_externalIP
  MONGO_DATABASE: mydb
```

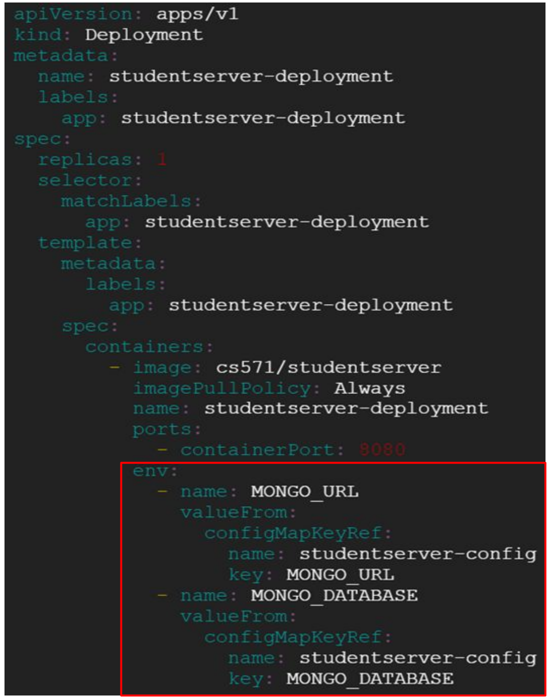
Now deploy the studentserver-deployment with configmap reference of mongodb database URL address and name.



```
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ kubectl get deployments
NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
mongodb-deployment        1/1     1            1           65m
studentserver-deployment  1/1     1            1           48m
```

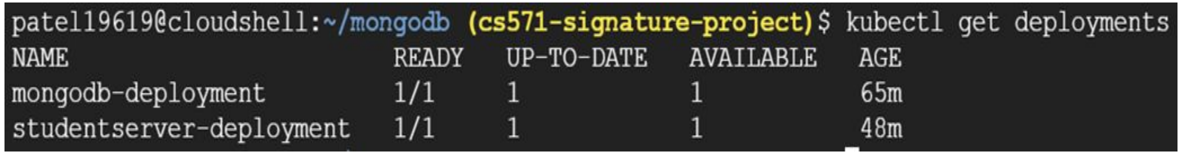
Implementation

Create studentserver-deployment.yaml



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: studentserver-deployment
  labels:
    app: studentserver-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: studentserver-deployment
  template:
    metadata:
      labels:
        app: studentserver-deployment
    spec:
      containers:
        - image: cs571/studentserver
          imagePullPolicy: Always
          name: studentserver-deployment
          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
```

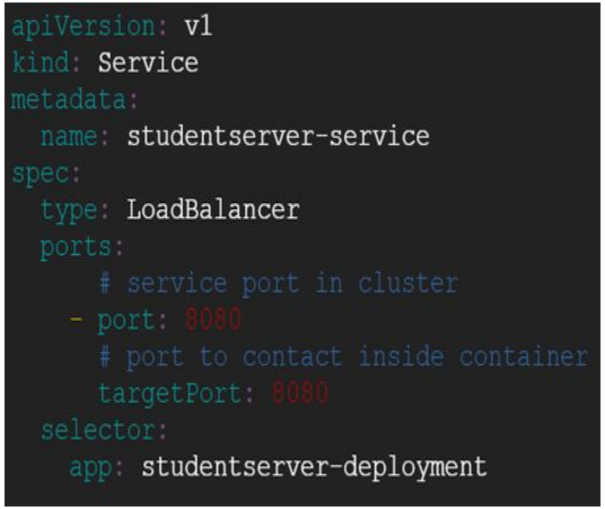
Now deploy the studentserver-deployment with configmap reference of mongodb database URL address and name.



```
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ kubectl get deployments
NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
mongodb-deployment         1/1     1            1           65m
studentserver-deployment   1/1     1            1           48m
```

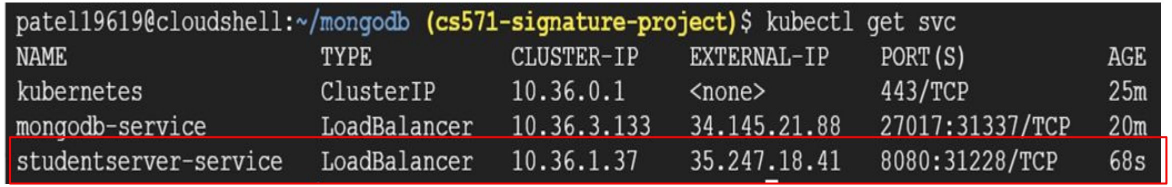
Implementation

Create the student-server LoadBalancer service to communicate with cluster.



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

kubectl get svc

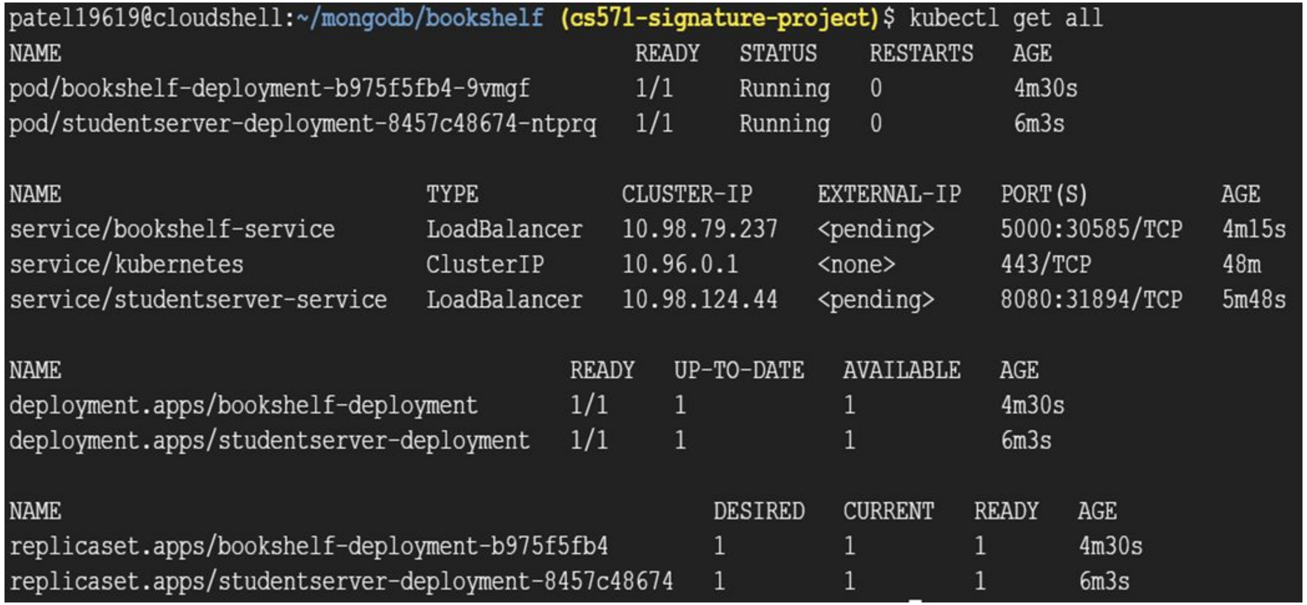


```
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ kubectl get svc
NAME                   TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)             AGE
kubernetes             ClusterIP      10.36.0.1      <none>         443/TCP             25m
mongodb-service        LoadBalancer   10.36.3.133    34.145.21.88   27017:31337/TCP     20m
studentserver-service  LoadBalancer   10.36.1.37     35.247.18.41   8080:31228/TCP      68s
```

Implementation

To deploy the bookshelf python application follow the same procedure of student server..

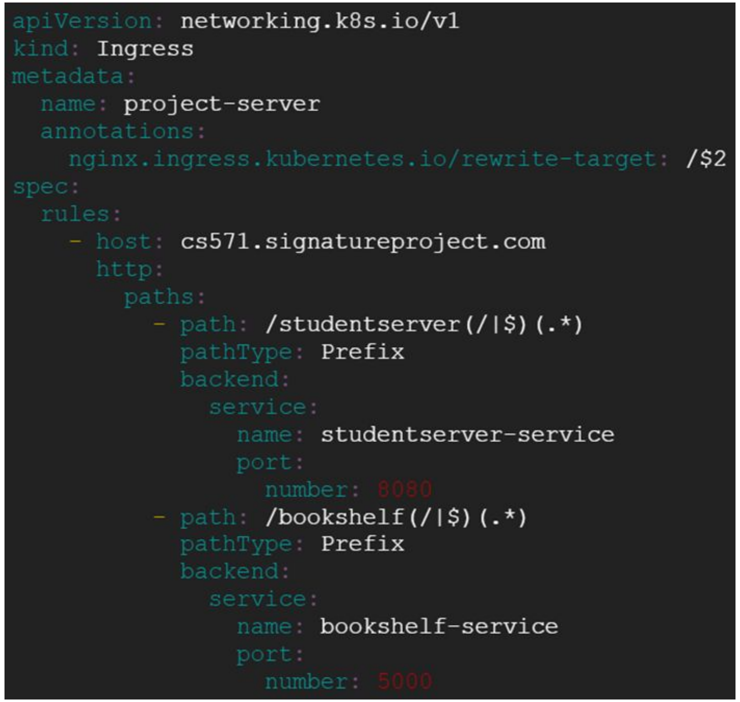
Deployed Successfully and mongoDB database is running properly, you would get same cluster configuration



```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ kubectl get all
NAME                                           READY   STATUS    RESTARTS   AGE
pod/bookshelf-deployment-b975f5fb4-9vmgf       1/1     Running   0          4m30s
pod/studentserver-deployment-8457c48674-ntprq  1/1     Running   0          6m3s

NAME                           TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
service/bookshelf-service      LoadBalancer   10.98.79.237   <pending>     5000:30585/TCP   4m15s
service/kubernetes             ClusterIP      10.96.0.1      <none>        443/TCP          48m
service/studentserver-service  LoadBalancer   10.98.124.44   <pending>     8080:31894/TCP   5m48s

NAME                                      READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/bookshelf-deployment      1/1     1            1           4m30s
deployment.apps/studentserver-deployment  1/1     1            1           6m3s

NAME                                                 DESIRED   CURRENT   READY   AGE
replicaset.apps/bookshelf-deployment-b975f5fb4       1         1         1       4m30s
replicaset.apps/studentserver-deployment-8457c48674  1         1         1       6m3s
```

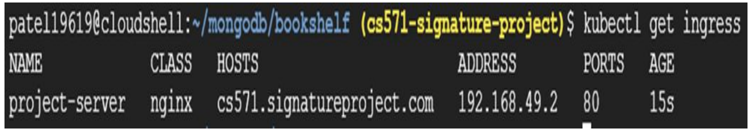
Implementation

Create the ingress service for both applications with same domain name but different path



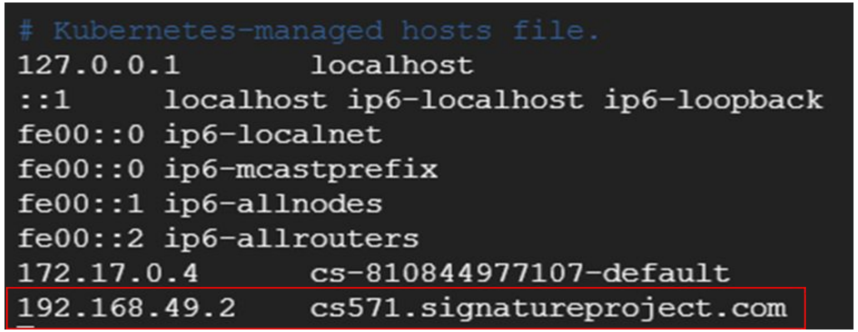
```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: project-server
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
  rules:
    - host: cs571.signatureproject.com
      http:
        paths:
          - path: /studentserver(/|$)(.*)
            pathType: Prefix
            backend:
              service:
                name: studentserver-service
                port:
                  number: 8080
          - path: /bookshelf(/|$)(.*)
            pathType: Prefix
            backend:
              service:
                name: bookshelf-service
                port:
                  number: 5000
```

kubectl get ingress



```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ kubectl get ingress
NAME             CLASS   HOSTS                        ADDRESS        PORTS   AGE
project-server   nginx   cs571.signatureproject.com   192.168.49.2   80      15s
```

Add the ingress service IP Address to **/etc/hosts** file

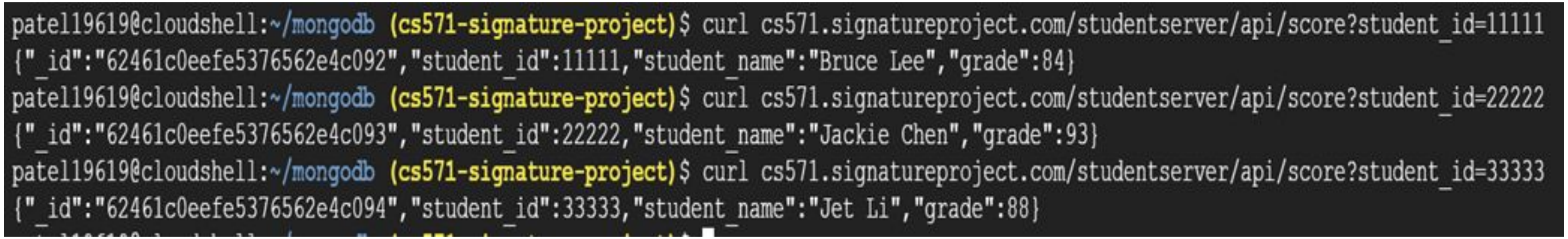


```
# Kubernetes-managed hosts file.
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
fe00::0 ip6-mcastprefix
fe00::1 ip6-allnodes
fe00::2 ip6-allrouters
172.17.0.4      cs-810844977107-default
192.168.49.2    cs571.signatureproject.com
```

Test

To test the student server Node.js application

```
curl cs571.signatureproject.com/studentserver/api/score?student_id=11111/22222/33333
```

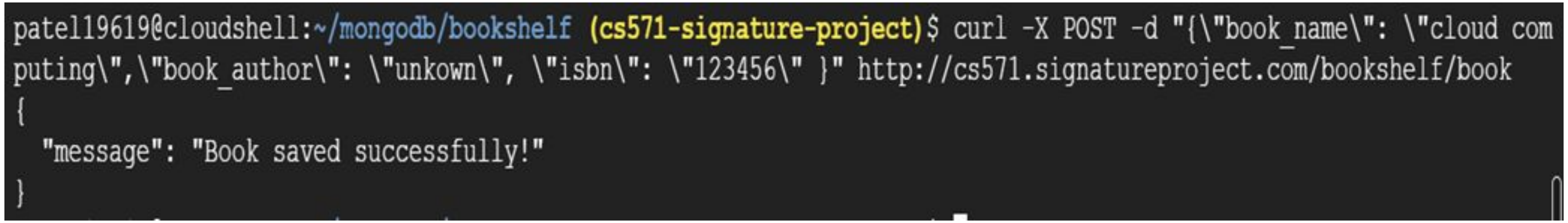


```
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ curl cs571.signatureproject.com/studentserver/api/score?student_id=11111
{"_id":"62461c0eefe5376562e4c092","student_id":11111,"student_name":"Bruce Lee","grade":84}
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ curl cs571.signatureproject.com/studentserver/api/score?student_id=22222
{"_id":"62461c0eefe5376562e4c093","student_id":22222,"student_name":"Jackie Chen","grade":93}
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ curl cs571.signatureproject.com/studentserver/api/score?student_id=33333
{"_id":"62461c0eefe5376562e4c094","student_id":33333,"student_name":"Jet Li","grade":88}
```

Test

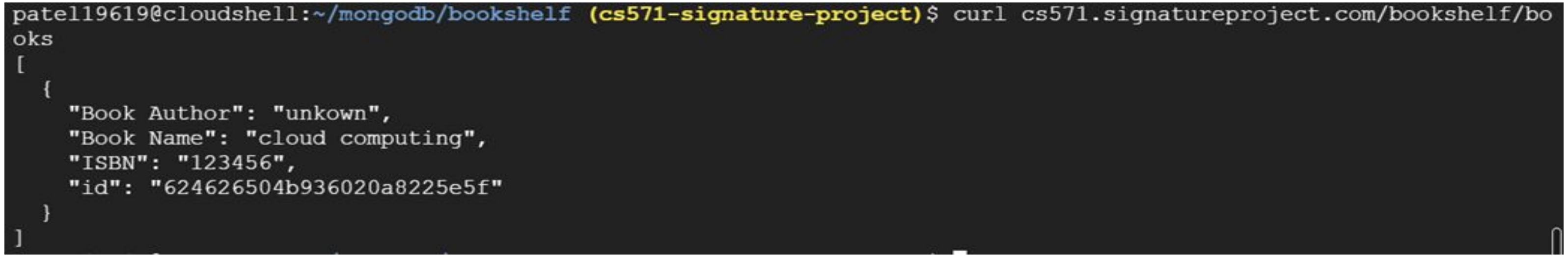
For Bookshelf python + REST API application do following operations

Add Books



```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ curl -X POST -d "{\"book_name\": \"cloud com
puting\",\"book_author\": \"unkown\", \"isbn\": \"123456\" }" http://cs571.signatureproject.com/bookshelf/book
{
  "message": "Book saved successfully!"
}
```

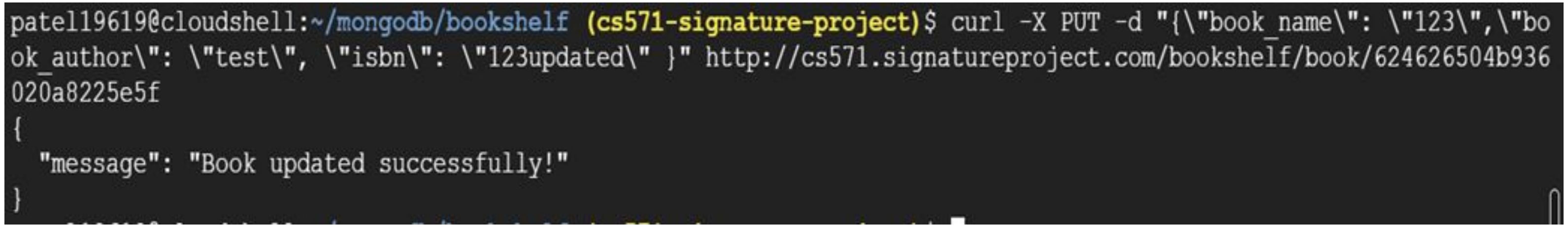
Get Book list



```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ curl cs571.signatureproject.com/bookshelf/bo
oks
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "624626504b936020a8225e5f"
  }
]
```

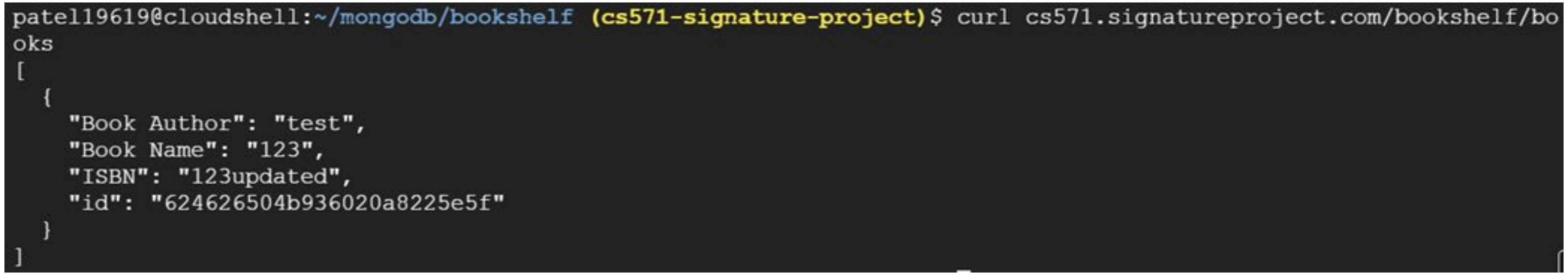
Test

Update Book



```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ curl -X PUT -d "{\"book_name\": \"123\",\"bo
ok_author\": \"test\", \"isbn\": \"123updated\" }" http://cs571.signatureproject.com/bookshelf/book/624626504b936
020a8225e5f
{
  "message": "Book updated successfully!"
}
```

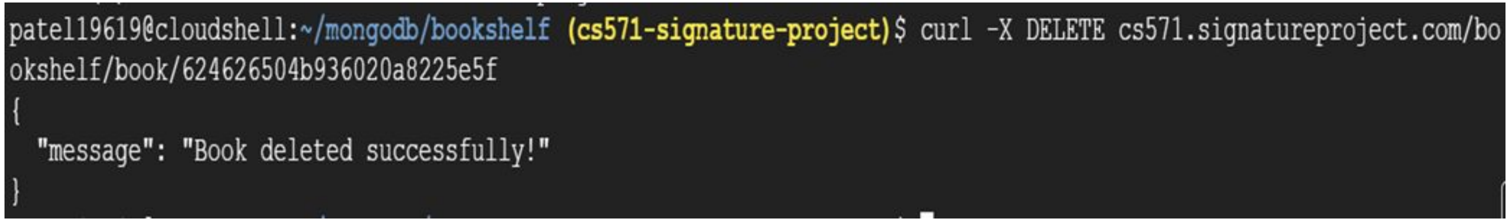
Get Updated Book



```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ curl cs571.signatureproject.com/bookshelf/bo
oks
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123updated",
    "id": "624626504b936020a8225e5f"
  }
]
```

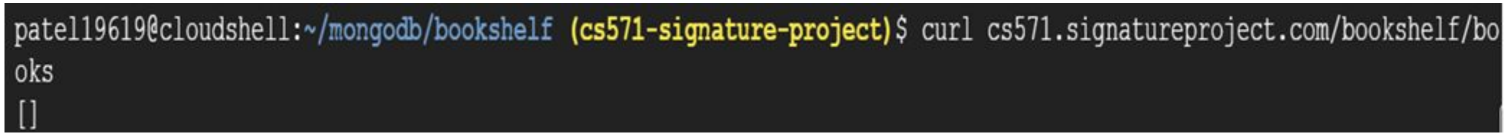
Test

Delete Book



```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ curl -X DELETE cs571.signatureproject.com/bo
okshelf/book/624626504b936020a8225e5f
{
  "message": "Book deleted successfully!"
}
```

Book list



```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ curl cs571.signatureproject.com/bookshelf/bo
oks
[]
```

Enhancement Ideas

- Try getting a real domain name, and host both applications under it for public test
- Adding TLS to MongoDB to secure the database to prevent malicious access

Conclusion

- Kubernetes useful for scale resources and applications in real time
- Orchestrate containers on multiple hosts
- Control and automate deployments and updates
- Multiple applications can be hosted on single domain name (i.e. cs.signatureproject.com)
- Save money by optimizing infrastructural resources with more efficient use of hardware