CS571 Signature Project
Name: Quan Zhou
ID: 19539

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-west1

   Wait for the creation to finish,

```
NAME   LOCATION  MASTER_VERSION    MASTER_IP     MACHINE_TYPE  NODE_VERSION      NUM_NODES  STATUS
kubia  us-west1  1.18.15-gke.1501  35.199.151.6  e2-micro      1.18.15-gke.1501  3          RUNNING
```

2. Let's create a Persistent Volume first, if you have created a persistent volume for the week10's homework, you can skip this one

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

```
zhou19539@cloudshell:~ (cs571-demo-project)$ 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,
see: https://developers.google.com/compute/docs/disks#performance.
Created [https://www.googleapis.com/compute/v1/projects/cs571-demo-project/zones/us-west1-a/disks/mongodb].
NAME     ZONE        SIZE_GB  TYPE         STATUS
mongodb  us-west1-a  10       pd-standard  READY
```

3. Now create a mongodb deployment with this yaml filec

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
spec:
  selector:
    matchLabels:
      app: mongodb
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
        # by default, the image is pulled from docker hub
        - image: mongo
          name: mongo
          ports:
            - containerPort: 27017
          volumeMounts:
            - name: mongodb-data
              mountPath: /data/db
      volumes:
        - name: mongodb-data
          gcePersistentDisk:
            pdName: mongodb
            fsType: ext4
```

kubectl apply -f mongodb-deployment.yaml

```
zhou19539@cloudshell:~/mongodb/yaml (cs571-demo-project)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
```

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

   kubectl get pods

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

```
zhou19539@cloudshell:~/mongodb/yaml (cs571-demo-project)$ kubectl get pods
NAME                                 READY   STATUS    RESTARTS   AGE
mongodb-deployment-554cbb9965-6494p  1/1     Running   0          2m44s
```

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

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

   kubectl apply -f mongodb-service.yaml

```
zhou19539@cloudshell:~/mongodb/yaml (cs571-demo-project)$ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
```

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

   kubectl get svc

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

```
NAME              TYPE           CLUSTER-IP     EXTERNAL-IP      PORT(S)           AGE
kubernetes        ClusterIP      10.3.240.1     <none>           443/TCP           10m
mongodb-service   LoadBalancer   10.3.250.140   35.197.111.141   27017:30359/TCP   39s
```

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

   Now you are inside the mongodb deployment pod

```
zhou19539@cloudshell:~/mongodb/yaml (cs571-demo-project)$ kubectl exec -it mongodb-deployment-554cbb9965-6494p -- bash
root@mongodb-deployment-554cbb9965-6494p:/#
```

   Try

   mongo External-IP

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

```
root@mongodb-deployment-554cbb9965-6494p:/# mongo 35.197.111.141
MongoDB shell version v4.4.4
connecting to: mongodb://35.197.111.141:27017/test?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("b6684190-20f0-40d9-8051-e8bc05d08b6c") }
MongoDB server version: 4.4.4
---
The server generated these startup warnings when booting:
        2021-03-25T00:01:30.075+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engi
ne. See http://dochub.mongodb.org/core/prodnotes-filesystem
        2021-03-25T00:01:31.629+00:00: Access control is not enabled for the database. Read and write access to data and
 configuration is unrestricted
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).

        The monitoring data will be available on a MongoDB website with a unique URL accessible to you
        and anyone you share the URL with. MongoDB may use this information to make product
        improvements and to suggest MongoDB products and deployment options to you.

        To enable free monitoring, run the following command: db.enableFreeMonitoring()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
>
```

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

```
> exit
bye
root@mongodb-deployment-554cbb9965-6494p:/# exit
exit
zhou19539@cloudshell:~/mongodb (cs571-demo-project)$
```

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

```
zhou19539@cloudshell:~/mongodb (cs571-demo-project)$ node
Welcome to Node.js v12.14.1.
Type ".help" for more information.
>
```

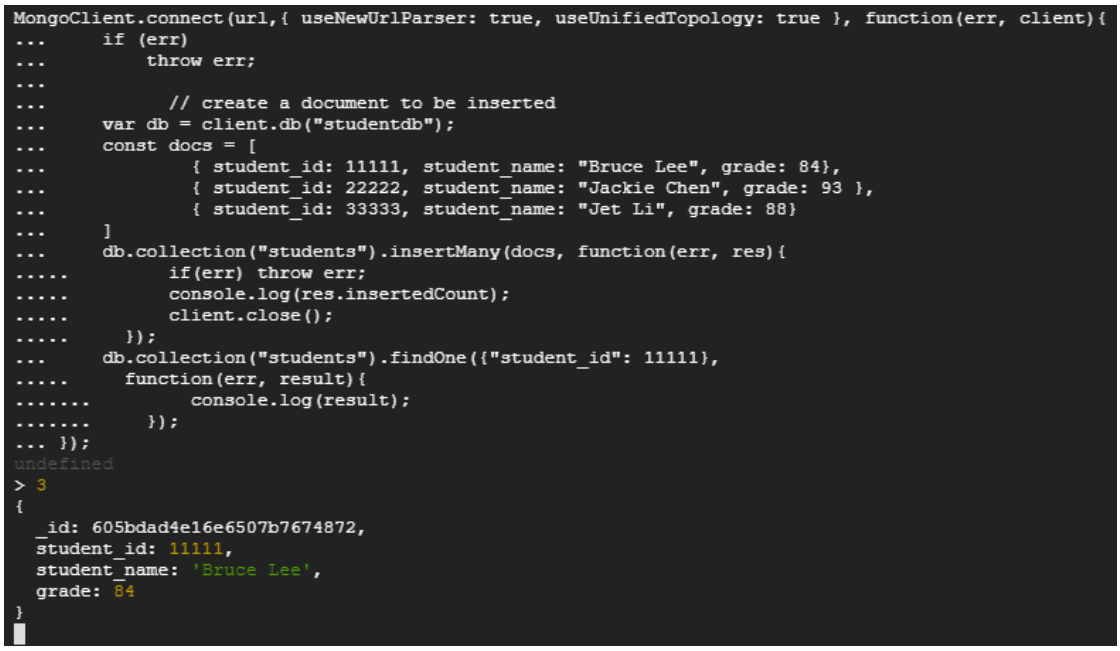
Enter the following line by line

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://EXTERNAL-IP/mydb"
// Connect to the db

MongoClient.connect(url,{ useNewUrlParser: true, useUnifiedTopology: true },
function(err, client){
    if (err)
        throw err;

          // create a document to be inserted
    var db = client.db("studentdb");
    const docs = [
            { student_id: 11111, student_name: "Bruce Lee", grade: 84},
            { student_id: 22222, student_name: "Jackie Chen", grade: 93 },
            { student_id: 33333, student_name: "Jet Li", grade: 88}
    ]
    db.collection("students").insertMany(docs, function(err, res){
        if(err) throw err;
        console.log(res.insertedCount);
        client.close();
    });
    db.collection("students").findOne({"student_id": 11111},
    function(err, result){
        console.log(result);
    });
});
```

If Everything is correct, you should see this,
3 means three records was inserted, and we tried search for student_id=11111



Step2 Modify our studentServer to get records from MongoDB and deploy to GKE

1. Create a studentServer

```
var http = require('http');
var url = require('url');
var mongodb = require('mongodb');
const {
  MONGO_URL,
  MONGO_DATABASE
}  = process.env;
// - Expect the request to contain a query
//   string with a key 'student_id' and a student ID as
//   the value. For example
//     /api/score?student_id=1111
// - The JSON response should contain only 'student_id', 'student_name'
//   and 'student_score' properties. For example:
//
//     {
//       "student_id": 1111,
//       "student_name": Bruce Lee,
//       "student_score": 84
//     }
//

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");
            }
        });
    });
  } else {
  res.writeHead(404);
  res.end("Wrong url, please try again\n");
  }
});
server.listen(8080);
```

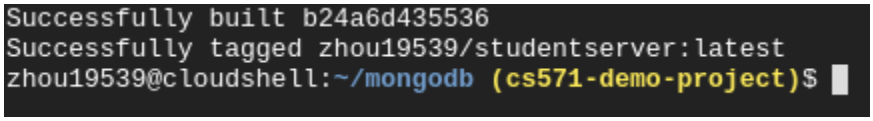
2. Create Dockerfile

```
FROM node:7
ADD studentServer.js /studentServer.js
ENTRYPOINT ["node", "studentServer.js"]
RUN npm install mongodb
```

3. Build the studentserver docker image

   docker build -t yourdockerhubID/studentserver .

   Make sure there is no error



4. Push the docker image

   docker push yourdockerhubID/studentserver

```
zhou19539@cloudshell:~/mongodb (cs571-demo-project)$ docker push zhou19539/studentserver
Using default tag: latest
The push refers to repository [docker.io/zhou19539/studentserver]
e532d718feb7: Pushed
1002594c636e: Pushed
ab90d83fa34a: Layer already exists
8ee318e54723: Layer already exists
e6695624484e: Layer already exists
da59b99bbd3b: Layer already exists
5616a6292c16: Layer already exists
f3ed6cb59ab0: Layer already exists
654f45ecb7e3: Layer already exists
2c40c66f7667: Layer already exists
latest: digest: sha256:ff96070378d962a965a23a32940137ea5db5a7ae6d3eef26c3f6c804f83b5995 size: 2424
```

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

1. Create bookshelf.py

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

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

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

@app.route("/tasks/delete", methods=["POST"])
def delete_all_tasks():
    db.bookshelf.remove()
    return jsonify(
        message="All Books deleted!"
    )
```

```
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

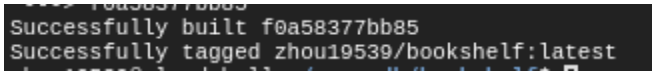
2. Create a Dockerfile

```
FROM python:alpine3.7
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
ENV PORT 5000
EXPOSE 5000
ENTRYPOINT [ "python3" ]
CMD [ "bookshelf.py" ]
```

3. Build the bookshelf app into a docker image

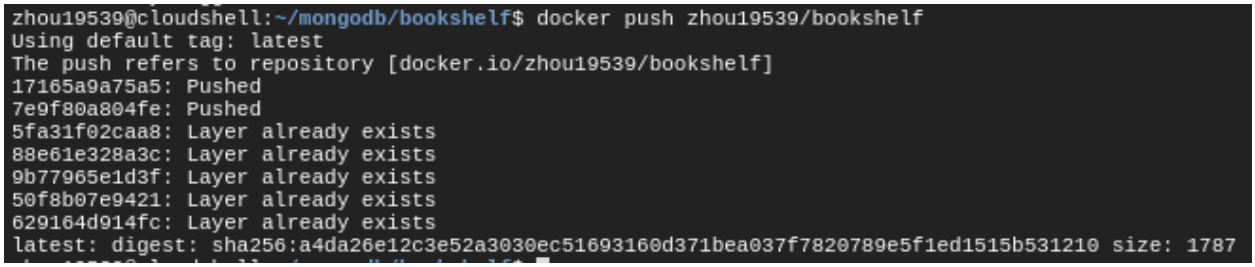
   docker build -t zhou19539/bookshelf .

   Make sure this step build successfully



4. Push the docker image to your dockerhub

   docker push yourdockerhubID/bookshelf



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

1. Create a file named studentserver-configmap.yaml

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

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

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

1. Create studentserver-deployment.yaml

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

```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 sutdentserver-service.yaml

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

4. Create bookshelf-service.yaml

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

5. Start minikube

   minikube start

```
zhou19539@cloudshell:~/mongodb/bookshelf (cs571-demo-project)$ minikube start
😄  minikube v1.18.1 on Debian 10.8 (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 v1.20.2 preload ...
    > preloaded-images-k8s-v9-v1....: 491.22 MiB / 491.22 MiB  100.00% 158.34 M
🔥  Creating docker container (CPUs=2, Memory=4000MB) ...
🐳  Preparing Kubernetes v1.20.2 on Docker 20.10.3 ...
    ▪ Generating certificates and keys ...
    ▪ Booting up control plane ...
    ▪ Configuring RBAC rules ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v4
🌟  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

```
zhou19539@cloudshell:~/mongodb/bookshelf (cs571-demo-project)$ minikube addons enable ingress
    ▪ Using image us.gcr.io/k8s-artifacts-prod/ingress-nginx/controller:v0.40.2
    ▪ Using image jettech/kube-webhook-certgen:v1.2.2
    ▪ Using image jettech/kube-webhook-certgen:v1.3.0
🔎  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

```
zhou19539@cloudshell:~/mongodb/bookshelf (cs571-demo-project)$ kubectl apply -f ../studentserver-deployment.yaml
deployment.apps/web created
zhou19539@cloudshell:~/mongodb/bookshelf (cs571-demo-project)$ kubectl apply -f ../studentserver-configmap.yaml
configmap/studentserver-config created
zhou19539@cloudshell:~/mongodb/bookshelf (cs571-demo-project)$ kubectl apply -f ../studentserver-service.yaml
service/web created
```

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

   kubectl apply -f bookshelf-deployment.yaml

   kubectl apply -f bookshelf-configmap.yaml

   kubectl apply -f bookshelf-service.yaml

```
zhou19539@cloudshell:~/mongodb/bookshelf (cs571-demo-project)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created
zhou19539@cloudshell:~/mongodb/bookshelf (cs571-demo-project)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
zhou19539@cloudshell:~/mongodb/bookshelf (cs571-demo-project)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
```

9. Check if all the pods are running correctly

   kubectl get pods

```
zhou19539@cloudshell:~/mongodb/bookshelf$ kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
bookshelf-deployment-646c59bd88-grrtn   1/1     Running   0          64m
web-59b45585db-cnslb                    1/1     Running   0          61m
```

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

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

11. Create the ingress service using the above yaml file

    kubectl apply -f ../studentservermongoIngress.yaml

```
zhou19539@cloudshell:~/mongodb/bookshelf (cs571-demo-project)$ kubectl apply -f ../studentservermongoIngress.yaml
ingress.networking.k8s.io/server created
```

12. Check if ingress is running

    kubectl get ingress

    Please wait until you see the Address, then move forward

```
zhou19539@cloudshell:~/mongodb/bookshelf (cs571-demo-project)$ kubectl get ingress
NAME     CLASS    HOSTS              ADDRESS        PORTS   AGE
server   <none>   cs571.project.com  192.168.49.2   80      10m
```

13. Add Addreee 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
    ```

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

```
# 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-738046022024-default-boost-dlp9q
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

```
zhou19539@cloudshell:~/mongodb/bookshelf$ curl cs571.project.com/studentserver/api/score?student_id=11111
{"_id":"605a6b49c3a15527de9d0f9b","student_id":11111,"student_name":"Bruce Lee","grade":84}
zhou19539@cloudshell:~/mongodb/bookshelf$ curl cs571.project.com/studentserver/api/score?student_id=22222
{"_id":"605a6b49c3a15527de9d0f9c","student_id":22222,"student_name":"Jackie Chen","grade":93}
zhou19539@cloudshell:~/mongodb/bookshelf$ curl cs571.project.com/studentserver/api/score?student_id=33333
{"_id":"605a6b49c3a15527de9d0f9d","student_id":33333,"student_name":"Jet Li","grade":88}
```

    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

```
zhou19539@cloudshell:~/mongodb/bookshelf$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123",
    "id": "605d1ba7d40f50a395651765"
  }
]
```

Add a book

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

```
zhou19539@cloudshell:~/mongodb/bookshelf$ curl -X POST -d "{\"book_name\": \"cloud computing\",\"book_author\": \"unkown\", \"isbn\": \"123456\" }" http://cs571.project.com/bookshelf/book
{
  "message": "Task saved successfully!"
}
```

```
zhou19539@cloudshell:~/mongodb/bookshelf$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123",
    "id": "605d1ba7d40f50a395651765"
  },
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "605d2fffbd09c0d7f8cf1f93"
  }
]
```

Update a book

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

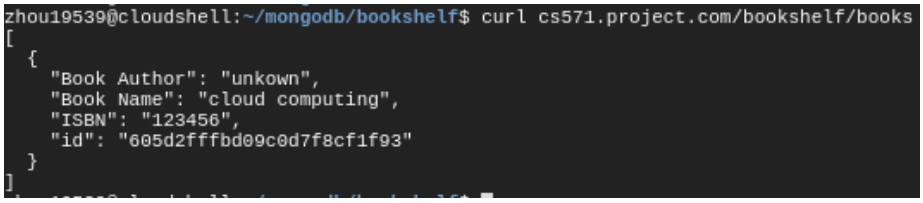
```
zhou19539@cloudshell:~/mongodb/bookshelf$ curl -X PUT -d "{\"book_name\": \"123\",\"book_author\": \"test\", \"isbn\": \"123updated\" }" http://cs571.project.com/bookshelf/book/605d1ba7d40f50a395651765
{
  "message": "Task updated successfully!"
}
```

```
zhou19539@cloudshell:~/mongodb/bookshelf$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123updated",
    "id": "605d1ba7d40f50a395651765"
  },
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "605d2fffbd09c0d7f8cf1f93"
  }
]
```

Delete a book

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

```
zhou19539@cloudshell:~/mongodb/bookshelf$ curl -X DELETE cs571.project.com/bookshelf/book/605d1ba7d40f50a395651765
{
  "message": "Task deleted successfully!"
}
```


zhou19539@cloudshell:~/mongodb/bookshelf$ curl cs571.project.com/bookshelf/books
[
{
"Book Author": "unkown",
"Book Name": "cloud computing",
"ISBN": "123456",
"id": "605d2fffbd09c0d7f8cf1f93"
}
]