CS571 Signature Project
Name: Shoumya Singh
ID: 19566

Project: **MongoDB + Python Flask Web Framework + REST API + GKE**

A. Project description

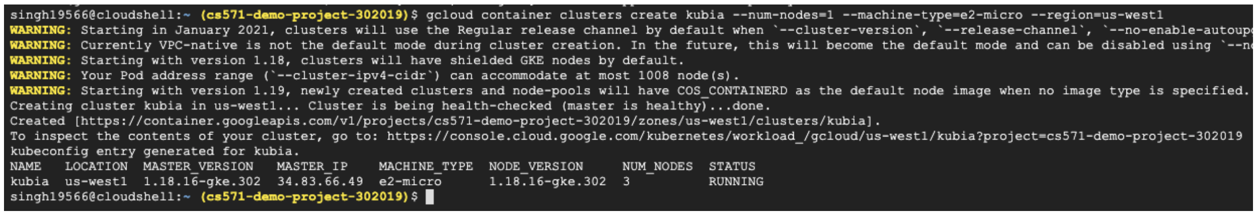
- Pods --- to run 2 applications
  - Pod 1: student records
    - GKE
  - Pod 2: bookstore
    - MongoDB + Python Flask Web Framework + REST API + GKE
- Service --- for outside access the application
- Persistence Volumes --- to store the data with MongoDB
- Ingress ---- to expose both applications under same domain but different path
  - Pod 1: student records
    - curl cs571.project.com/studentserver/api/score?student_id=11111
  - Pod 2: bookstore
    - curl cs571.project.com/bookshelf/books
- ConfigMaps ---- to store MongoDB service address, in case MongoDB is down and restarts with a different service address, and with ConfigMaps, we don't need to build the docker image again with the new address

- We should get these after running this project and add the result:
  1. Access a student's score
  2. List all the books
  3. Add a book
  4. Update a book
  5. Delete a book

<u>Step1</u>: Create MongoDB using Persistent Volume on GKE, and insert records into it

1. Create a cluster as usual on GKE.

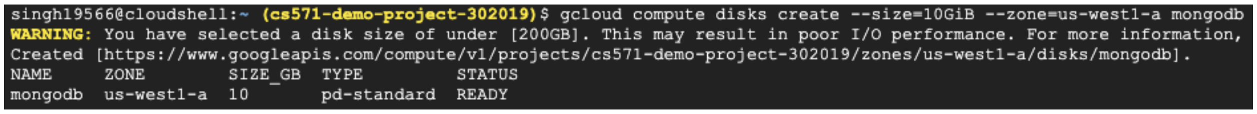
   - gcloud container clusters create kubia –num-nodes=1 - -machine-type=e2-micro –region=us-west1



```
singh19566@cloudshell:~ (cs571-demo-project-302019)$ gcloud container clusters create kubia --num-nodes=1 --machine-type=e2-micro --region=us-west1
WARNING: Starting in January 2021, clusters will use the Regular release channel by default when `--cluster-version`, `--release-channel`, `--no-enable-autoup
WARNING: Currently VPC-native is not the default mode during cluster creation. In the future, this will become the default mode and can be disabled using `--n
WARNING: Starting with version 1.18, clusters will have shielded GKE nodes by default.
WARNING: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most 1008 node(s).
WARNING: Starting with version 1.19, newly created clusters and node-pools will have COS_CONTAINERD as the default node image when no image type is specified.
Creating cluster kubia in us-west1... Cluster is being health-checked (master is healthy)...done.
Created [https://container.googleapis.com/v1/projects/cs571-demo-project-302019/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=cs571-demo-project-302019
kubeconfig entry generated for kubia.
NAME   LOCATION  MASTER_VERSION   MASTER_IP    MACHINE_TYPE  NODE_VERSION     NUM_NODES  STATUS
kubia  us-west1  1.18.16-gke.302  34.83.66.49  e2-micro      1.18.16-gke.302  3          RUNNING
singh19566@cloudshell:~ (cs571-demo-project-302019)$
```

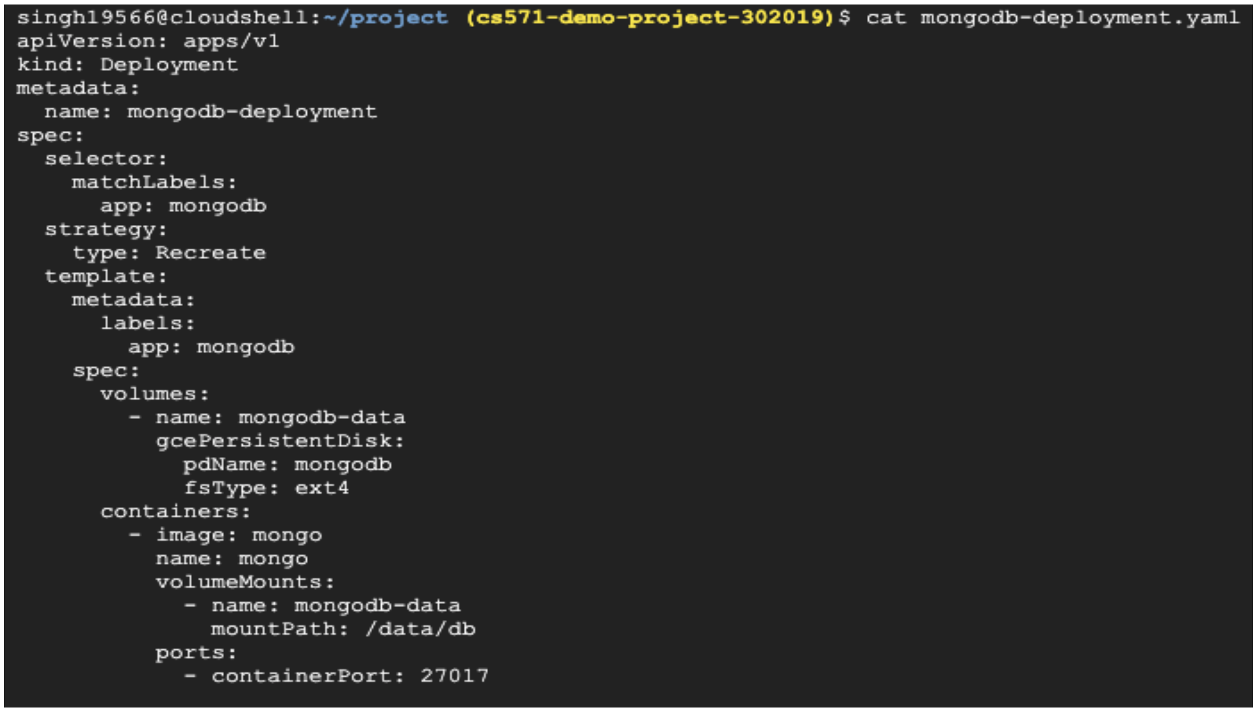
2. Let’s create a Persistent Volume.

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



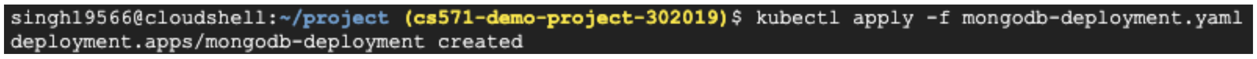
```
singh19566@cloudshell:~ (cs571-demo-project-302019)$ gcloud compute disks create --size=10GiB --zone=us-west1-a mongodb
WARNING: You have selected a disk size of under [200GB]. This may result in poor I/O performance. For more information,
Created [https://www.googleapis.com/compute/v1/projects/cs571-demo-project-302019/zones/us-west1-a/disks/mongodb].
NAME     ZONE        SIZE_GB  TYPE         STATUS
mongodb  us-west1-a  10       pd-standard  READY
```

3. Now create a mongodb deployment with this yaml file.



```
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ cat mongodb-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
spec:
  selector:
    matchLabels:
      app: mongodb
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      volumes:
        - name: mongodb-data
          gcePersistentDisk:
            pdName: mongodb
            fsType: ext4
      containers:
        - image: mongo
          name: mongo
          volumeMounts:
            - name: mongodb-data
              mountPath: /data/db
          ports:
            - containerPort: 27017
```

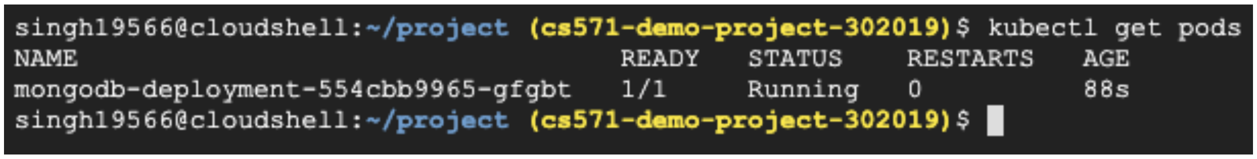
- kubectl apply -f mongodb-deployment.yaml



```
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
```

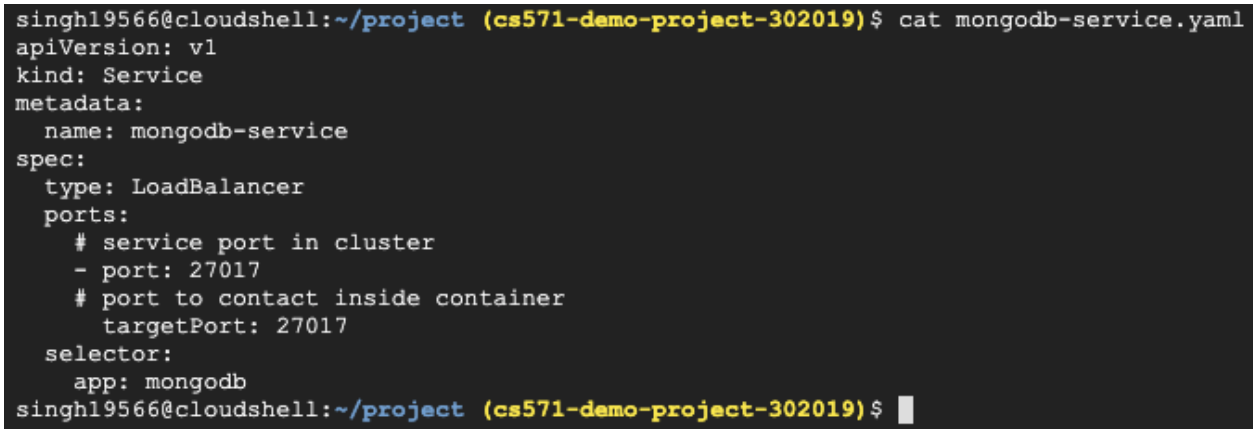
4. Check if the deployment pod has been successfully created and started running .

    - kubectl get pods



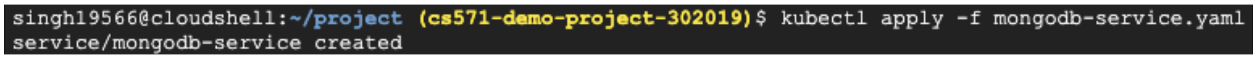
```
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
mongodb-deployment-554cbb9965-gfgbt   1/1     Running   0          88s
singh19566@cloudshell:~/project (cs571-demo-project-302019)$
```

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



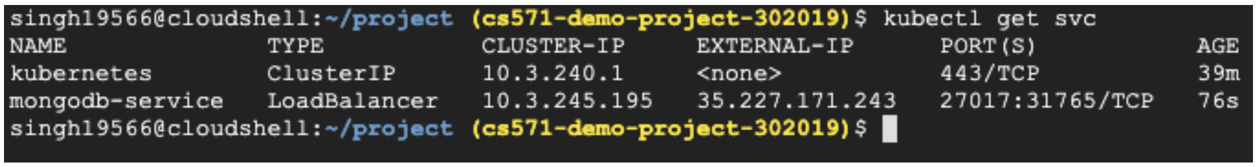
```
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ cat mongodb-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  type: LoadBalancer
  ports:
    # service port in cluster
    - port: 27017
    # port to contact inside container
      targetPort: 27017
  selector:
    app: mongodb
singh19566@cloudshell:~/project (cs571-demo-project-302019)$
```

- kubectl apply -f mongodb-service.yaml



```
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ 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

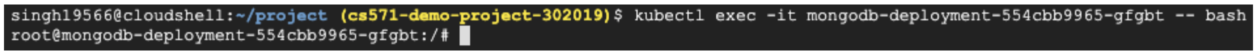


```
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ kubectl get svc
NAME              TYPE           CLUSTER-IP     EXTERNAL-IP      PORT(S)           AGE
kubernetes        ClusterIP      10.3.240.1     <none>           443/TCP           39m
mongodb-service   LoadBalancer   10.3.245.195   35.227.171.243   27017:31765/TCP   76s
singh19566@cloudshell:~/project (cs571-demo-project-302019)$
```

7. Let's try and see if mongodb is functioning for connections using the External-IP

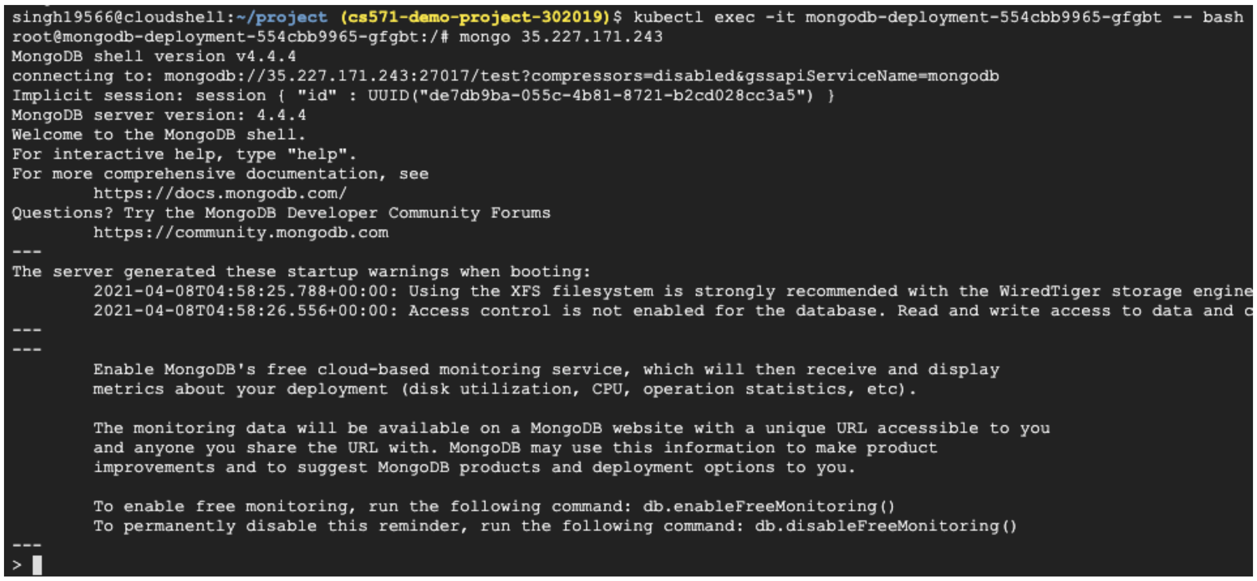
   - kubectl exec -it mongodb-deployment-replace-with-your-pod-name -- bash

   Now we are inside the mongodb deployment pod



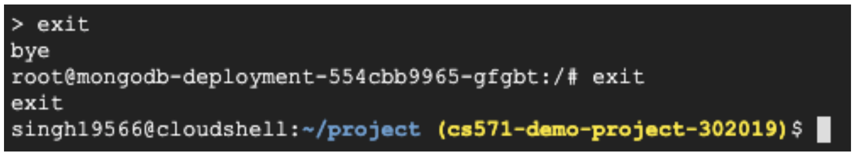
```
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ kubectl exec -it mongodb-deployment-554cbb9965-gfgbt -- bash
root@mongodb-deployment-554cbb9965-gfgbt:/#
```

   - mongo External-IP



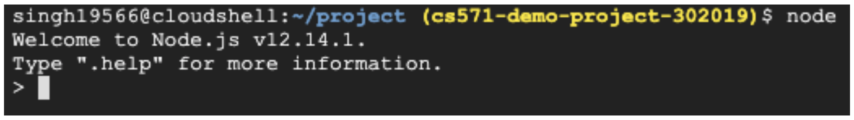
```
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ kubectl exec -it mongodb-deployment-554cbb9965-gfgbt -- bash
root@mongodb-deployment-554cbb9965-gfgbt:/# mongo 35.227.171.243
MongoDB shell version v4.4.4
connecting to: mongodb://35.227.171.243:27017/test?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("de7db9ba-055c-4b81-8721-b2cd028cc3a5") }
MongoDB server version: 4.4.4
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        https://docs.mongodb.com/
Questions? Try the MongoDB Developer Community Forums
        https://community.mongodb.com
---
The server generated these startup warnings when booting:
        2021-04-08T04:58:25.788+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
        2021-04-08T04:58:26.556+00:00: Access control is not enabled for the database. Read and write access to data and c
---
---
        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-gfgbt:/# exit
exit
singh19566@cloudshell:~/project (cs571-demo-project-302019)$
```

9. We need to insert some records into the mongodb for later use
   Type - node

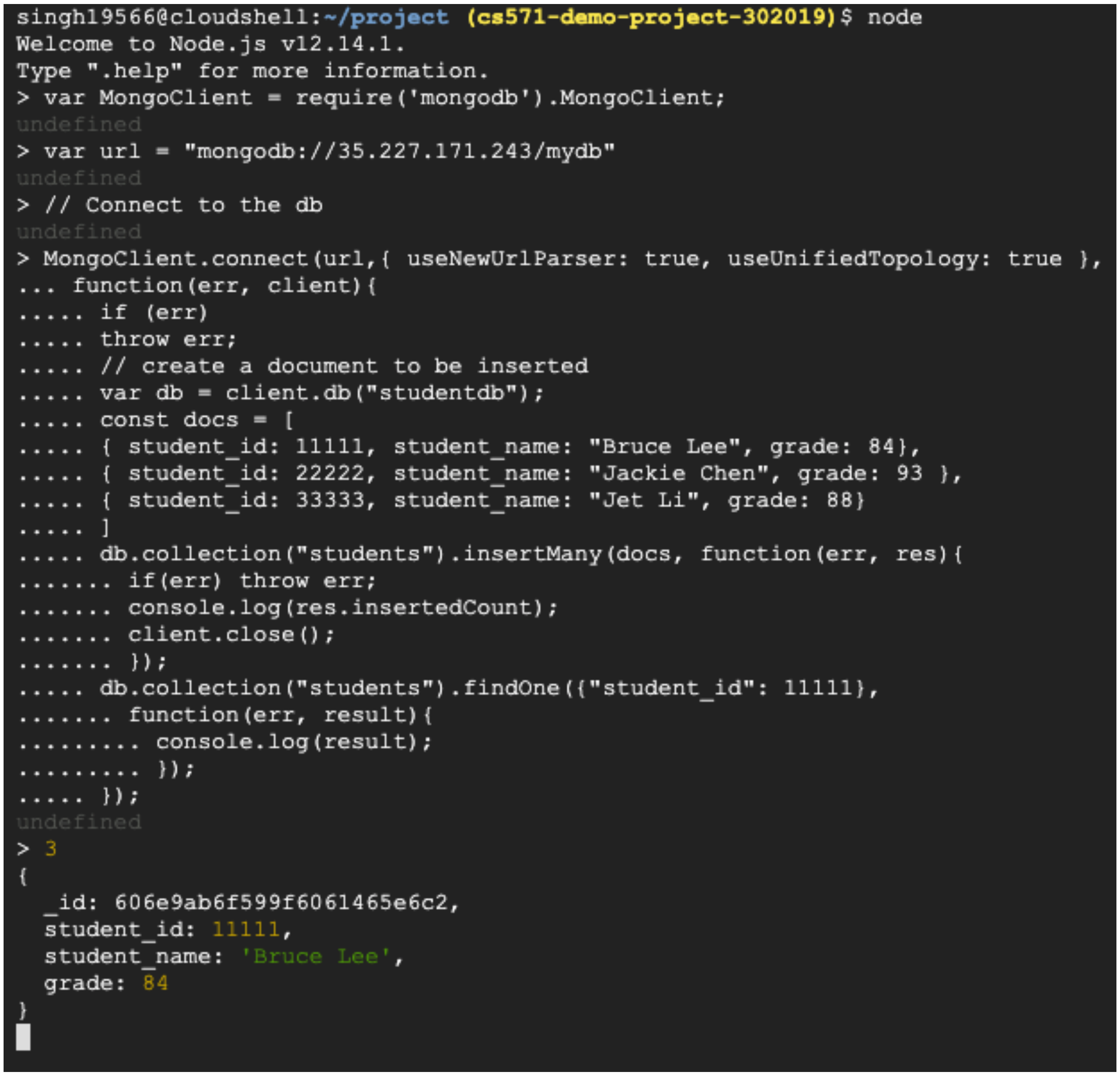


```
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ node
Welcome to Node.js v12.14.1.
Type ".help" for more information.
>
```

- Enter the following line by line

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://your 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



```
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ node
Welcome to Node.js v12.14.1.
Type ".help" for more information.
> var MongoClient = require('mongodb').MongoClient;
undefined
> var url = "mongodb://35.227.171.243/mydb"
undefined
> // Connect to the db
undefined
> 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);
......... });
..... });
undefined
> 3
{
  _id: 606e9ab6f599f6061465e6c2,
  student_id: 11111,
  student_name: 'Bruce Lee',
  grade: 84
}
```

<u>Step2</u>: Modify our studentServer to get records from MongoDB and deploy to GKE

1. Create a studentServer

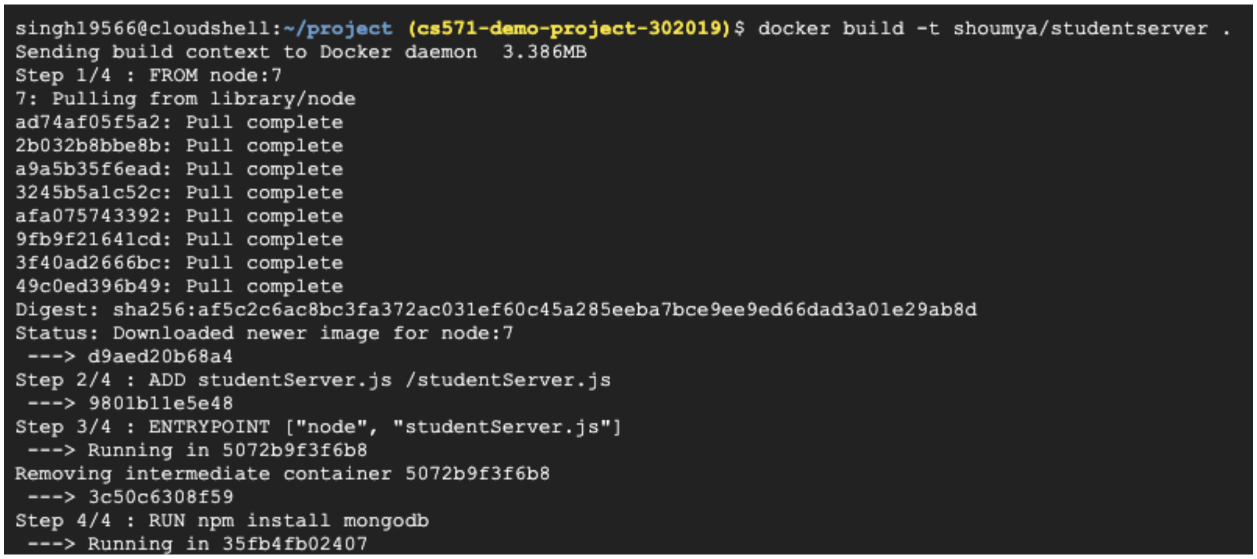
```
var http = require('http');
var url = require('url');
var mongodb = require('mongodb');
const {
    MONGO_URL,
    MONGO_DATABASE
} = process.env;
// - Expect the request to contain a query
// string with a key 'student_id' and a student ID as
// the value. For example
// /api/score?student_id=1111
// - 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

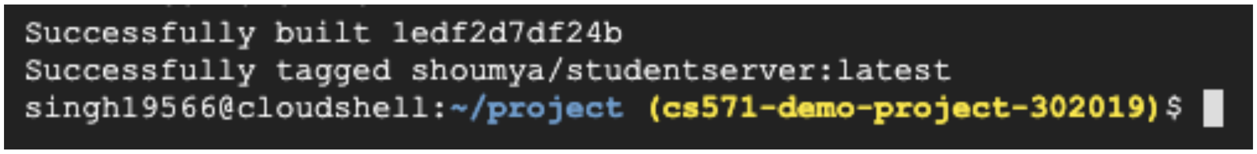
```
FROM node:7
ADD studentServer.js /studentServer.js
ENTRYPOINT ["node", "studentServer.js"]
RUN npm install mongodb
```

3. Build the studentserver docker image.

   - docker build -t yourdockerhubID/studentserver .



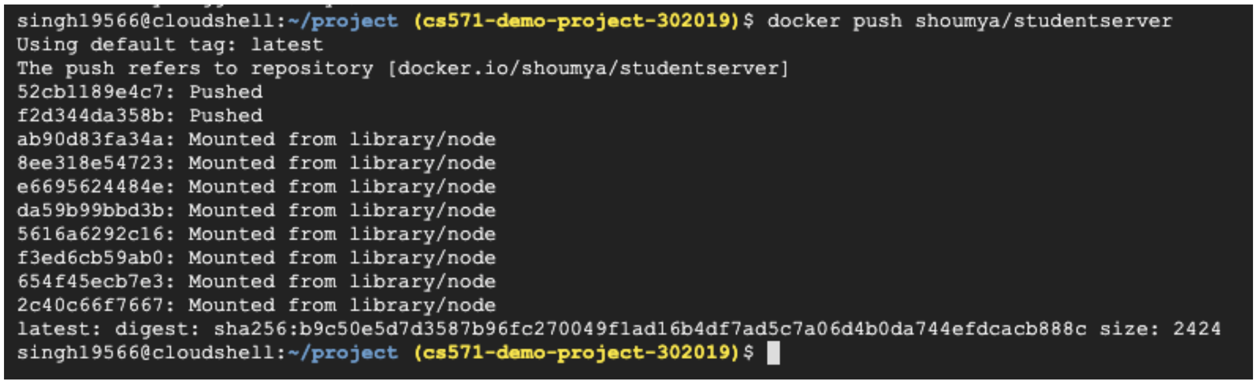
```
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ docker build -t shoumya/studentserver .
Sending build context to Docker daemon  3.386MB
Step 1/4 : FROM node:7
7: Pulling from library/node
ad74af05f5a2: Pull complete
2b032b8bbe8b: Pull complete
a9a5b35f6ead: Pull complete
3245b5a1c52c: Pull complete
afa075743392: Pull complete
9fb9f21641cd: Pull complete
3f40ad2666bc: Pull complete
49c0ed396b49: Pull complete
Digest: sha256:af5c2c6ac8bc3fa372ac031ef60c45a285eeba7bce9ee9ed66dad3a01e29ab8d
Status: Downloaded newer image for node:7
 ---> d9aed20b68a4
Step 2/4 : ADD studentServer.js /studentServer.js
 ---> 9801b11e5e48
Step 3/4 : ENTRYPOINT ["node", "studentServer.js"]
 ---> Running in 5072b9f3f6b8
Removing intermediate container 5072b9f3f6b8
 ---> 3c50c6308f59
Step 4/4 : RUN npm install mongodb
 ---> Running in 35fb4fb02407
```



```
Successfully built 1edf2d7df24b
Successfully tagged shoumya/studentserver:latest
singh19566@cloudshell:~/project (cs571-demo-project-302019)$
```

4. Push the docker image

   - docker push yourdockerhubID/studentserver



```
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ docker push shoumya/studentserver
Using default tag: latest
The push refers to repository [docker.io/shoumya/studentserver]
52cb1189e4c7: Pushed
f2d344da358b: Pushed
ab90d83fa34a: Mounted from library/node
8ee318e54723: Mounted from library/node
e6695624484e: Mounted from library/node
da59b99bbd3b: Mounted from library/node
5616a6292c16: Mounted from library/node
f3ed6cb59ab0: Mounted from library/node
654f45ecb7e3: Mounted from library/node
2c40c66f7667: Mounted from library/node
latest: digest: sha256:b9c50e5d7d3587b96fc270049f1ad16b4df7ad5c7a06d4b0da744efdcacb888c size: 2424
singh19566@cloudshell:~/project (cs571-demo-project-302019)$
```

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

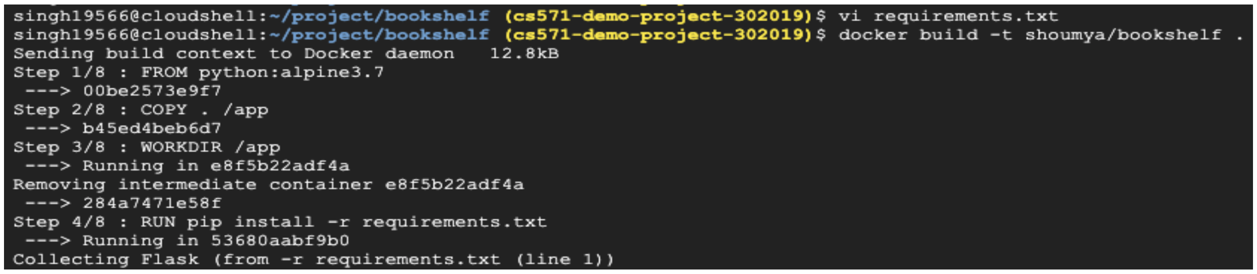
```
FROM python:alpine3.7
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
ENV PORT 5000
EXPOSE 5000
ENTRYPOINT [ "python3" ]
CMD [ "bookshelf.py" ]
```

3. Create a requirements.txt

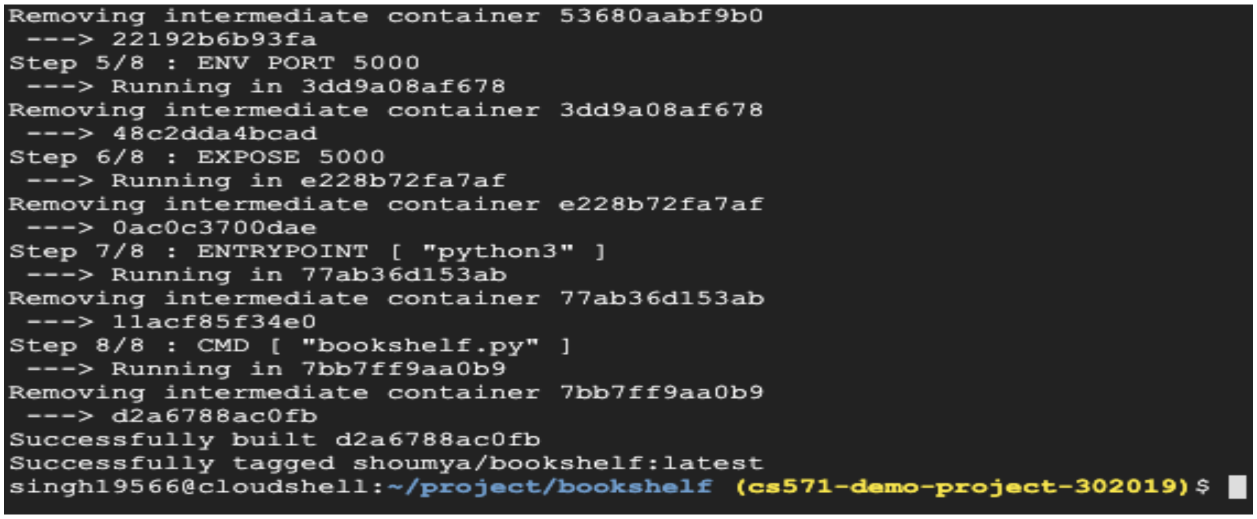
```
Flask
Flask-PyMongo
```

4. Build the bookshelf app into a docker image

   - docker build -t shoumya/bookshelf .



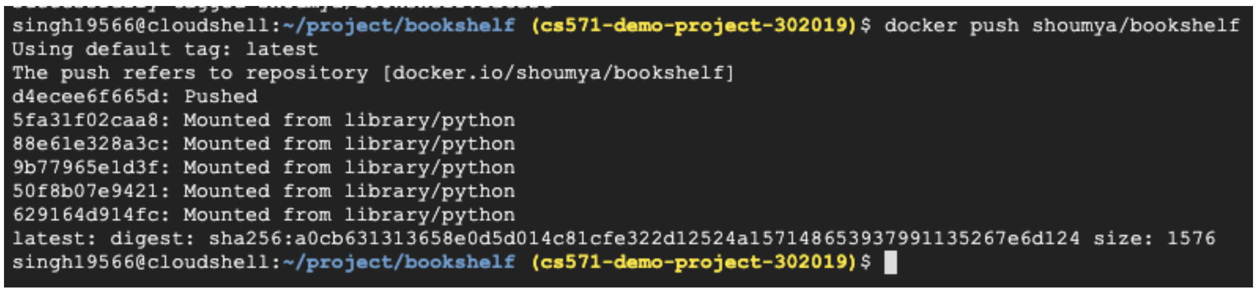
```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ vi requirements.txt
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ docker build -t shoumya/bookshelf .
Sending build context to Docker daemon   12.8kB
Step 1/8 : FROM python:alpine3.7
 ---> 00be2573e9f7
Step 2/8 : COPY . /app
 ---> b45ed4beb6d7
Step 3/8 : WORKDIR /app
 ---> Running in e8f5b22adf4a
Removing intermediate container e8f5b22adf4a
 ---> 284a7471e58f
Step 4/8 : RUN pip install -r requirements.txt
 ---> Running in 53680aabf9b0
Collecting Flask (from -r requirements.txt (line 1))
```



```
Removing intermediate container 53680aabf9b0
 ---> 22192b6b93fa
Step 5/8 : ENV PORT 5000
 ---> Running in 3dd9a08af678
Removing intermediate container 3dd9a08af678
 ---> 48c2dda4bcad
Step 6/8 : EXPOSE 5000
 ---> Running in e228b72fa7af
Removing intermediate container e228b72fa7af
 ---> 0ac0c3700dae
Step 7/8 : ENTRYPOINT [ "python3" ]
 ---> Running in 77ab36d153ab
Removing intermediate container 77ab36d153ab
 ---> 11acf85f34e0
Step 8/8 : CMD [ "bookshelf.py" ]
 ---> Running in 7bb7ff9aa0b9
Removing intermediate container 7bb7ff9aa0b9
 ---> d2a6788ac0fb
Successfully built d2a6788ac0fb
Successfully tagged shoumya/bookshelf:latest
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$
```

5. Push the docker image to your dockerhub

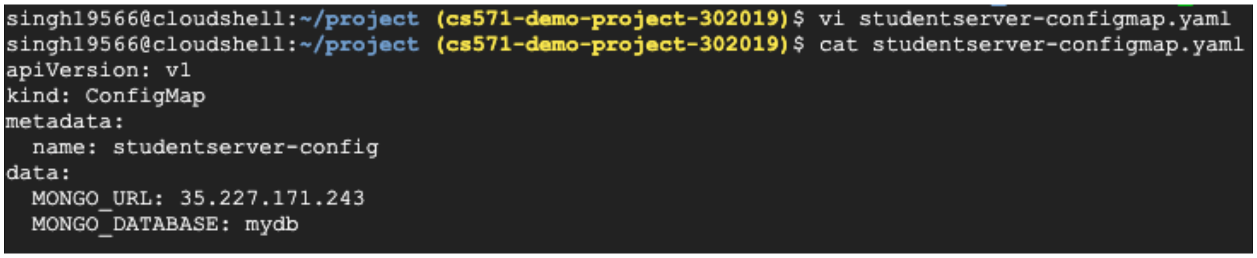
    - docker push yourdockerhubID/bookshelf



```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ docker push shoumya/bookshelf
Using default tag: latest
The push refers to repository [docker.io/shoumya/bookshelf]
d4ecee6f665d: 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:a0cb631313658e0d5d014c81cfe322d12524a15714865393799113 5267e6d124 size: 1576
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ 
```

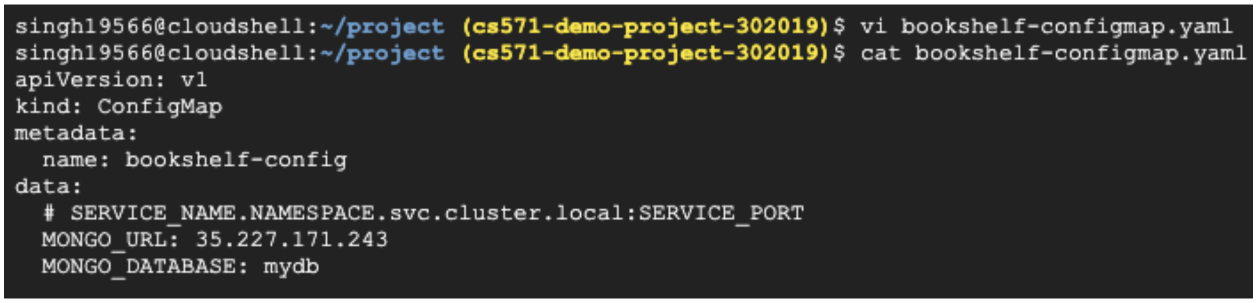
Step4 : Create ConfigMap for both applications to store Mongodb URL and Mongodb name

1. Create a file named studentserver-configmap.yaml



```
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ vi studentserver-configmap.yaml
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ cat studentserver-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  MONGO_URL: 35.227.171.243
  MONGO_DATABASE: mydb
```

2. Create a file named bookshelf-configmap.yaml



```
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ vi bookshelf-configmap.yaml
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ cat bookshelf-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: bookshelf-config
data:
  # SERVICE_NAME.NAMESPACE.svc.cluster.local:SERVICE_PORT
  MONGO_URL: 35.227.171.243
  MONGO_DATABASE: mydb
```

3. Change this in the code

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

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

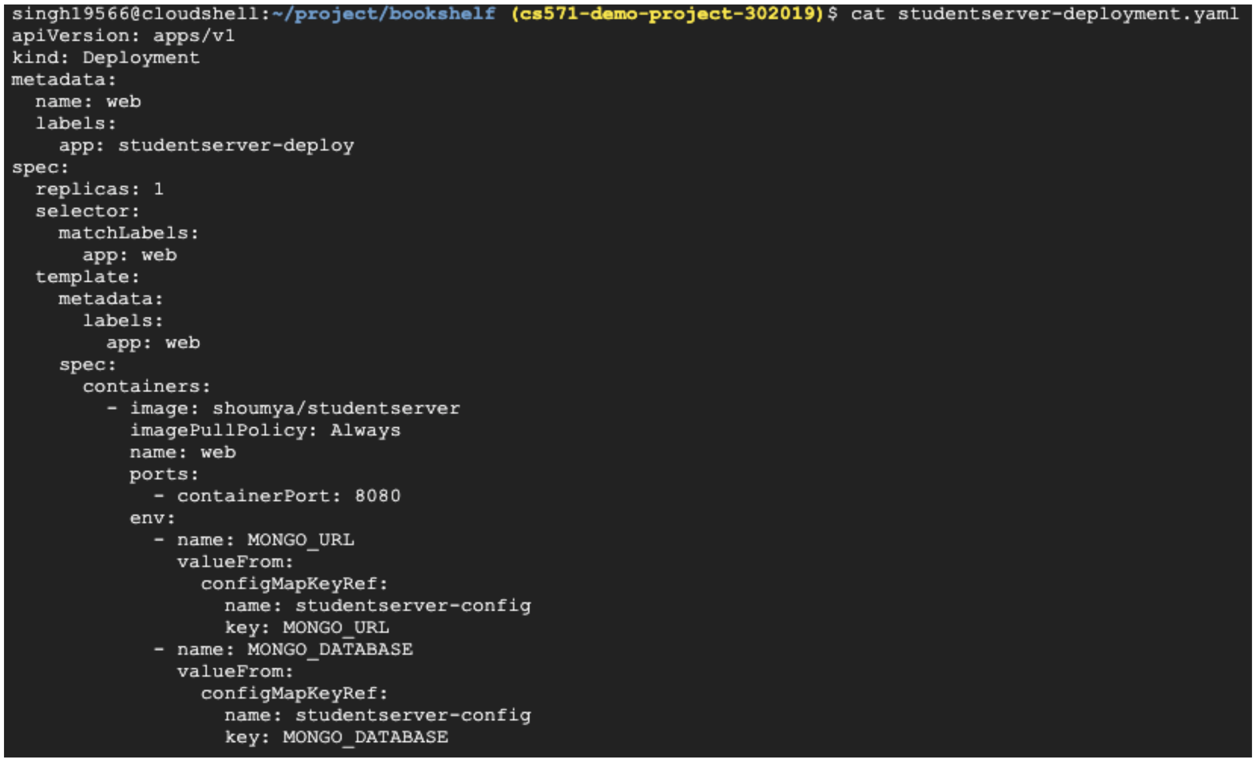
<u>Step5 :</u> Expose 2 application using ingress with Nginx, so we can put them on the same Domain but different PATH

1. Create studentserver-deployment.yaml

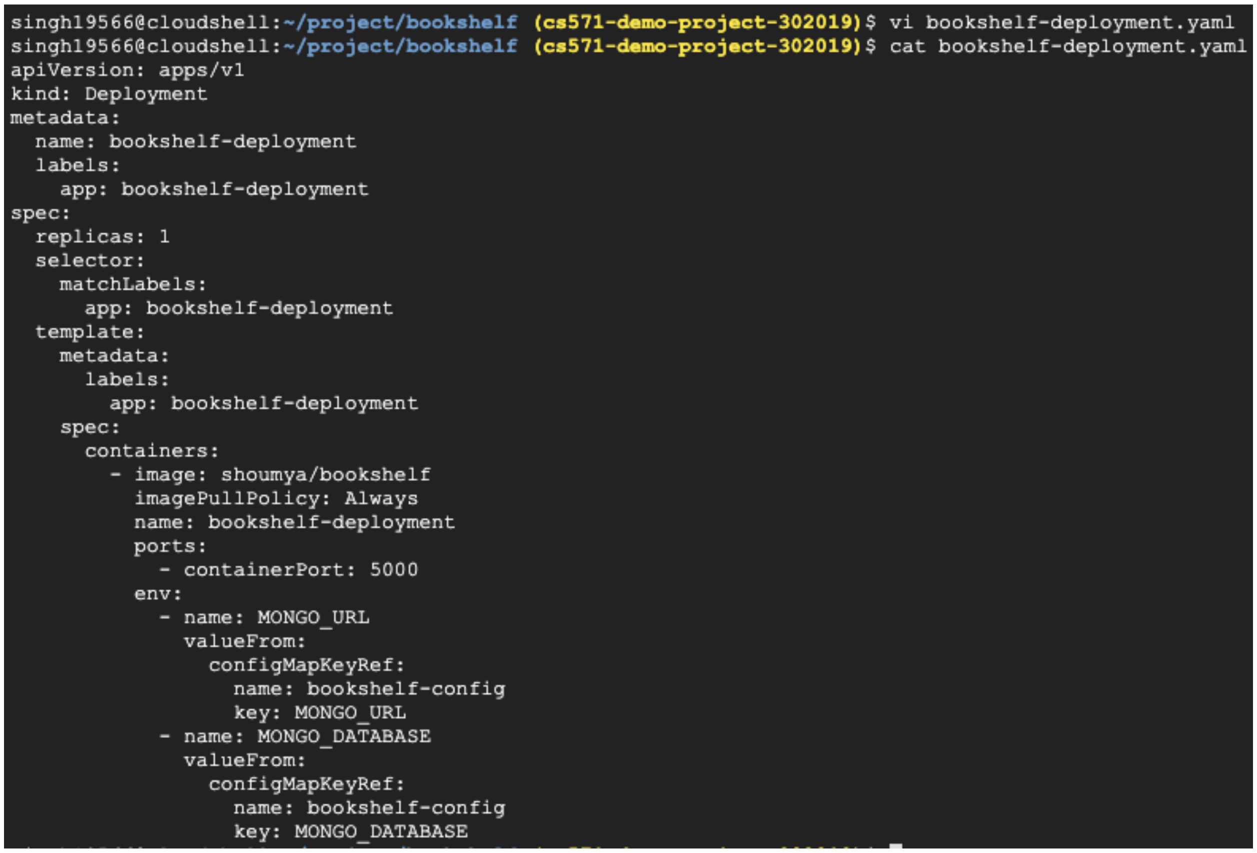
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
  labels:
    app: studentserver-deploy
spec:
  replicas: 1
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
        - image: shoumya/studentserver
         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



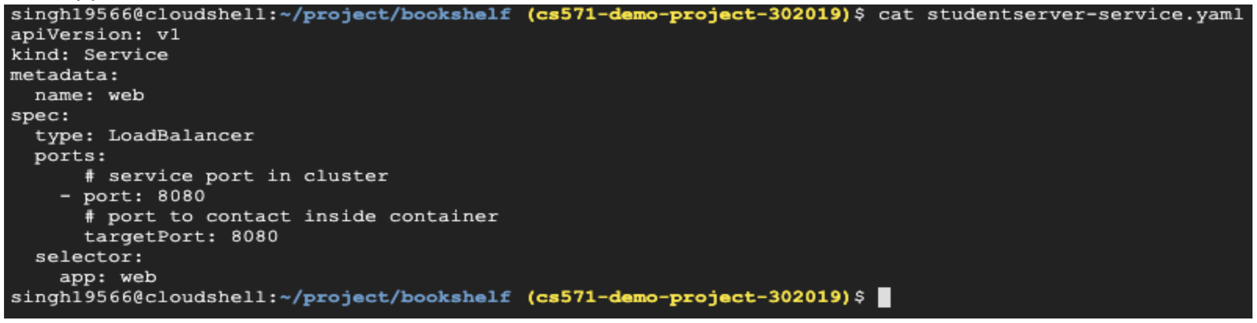
```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ cat studentserver-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
  labels:
    app: studentserver-deploy
spec:
  replicas: 1
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
        - image: shoumya/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
```



```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ vi bookshelf-deployment.yaml
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ cat 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: shoumya/bookshelf
          imagePullPolicy: Always
          name: bookshelf-deployment
          ports:
            - containerPort: 5000
          env:
            - name: MONGO_URL
              valueFrom:
                configMapKeyRef:
                  name: bookshelf-config
                  key: MONGO_URL
            - name: MONGO_DATABASE
              valueFrom:
                configMapKeyRef:
                  name: bookshelf-config
                  key: MONGO_DATABASE
```

3. Create sutdentserver-service.yaml

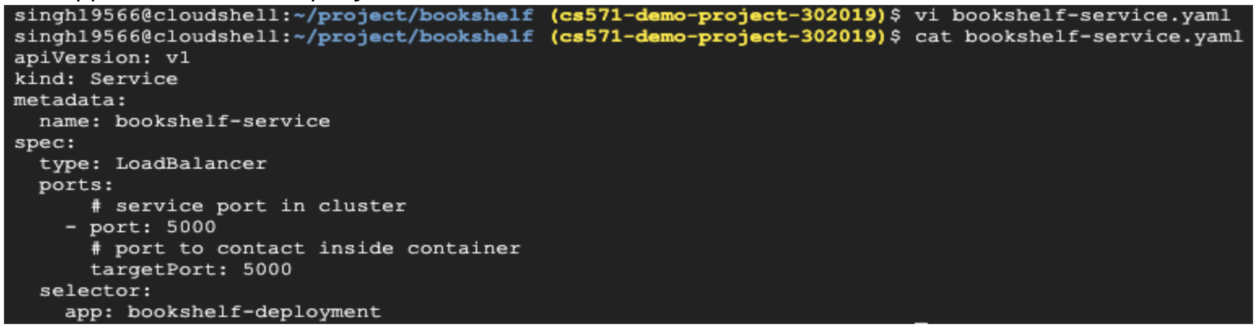
```
apiVersion: v1
kind: Service
metadata:
  name: web
spec:
  type: LoadBalancer
  ports:
      # service port in cluster
    - port: 8080
      # port to contact inside container
      targetPort: 8080
  selector:
    app: web
```



```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ cat studentserver-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: web
spec:
  type: LoadBalancer
  ports:
      # service port in cluster
    - port: 8080
      # port to contact inside container
      targetPort: 8080
  selector:
    app: web
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$
```

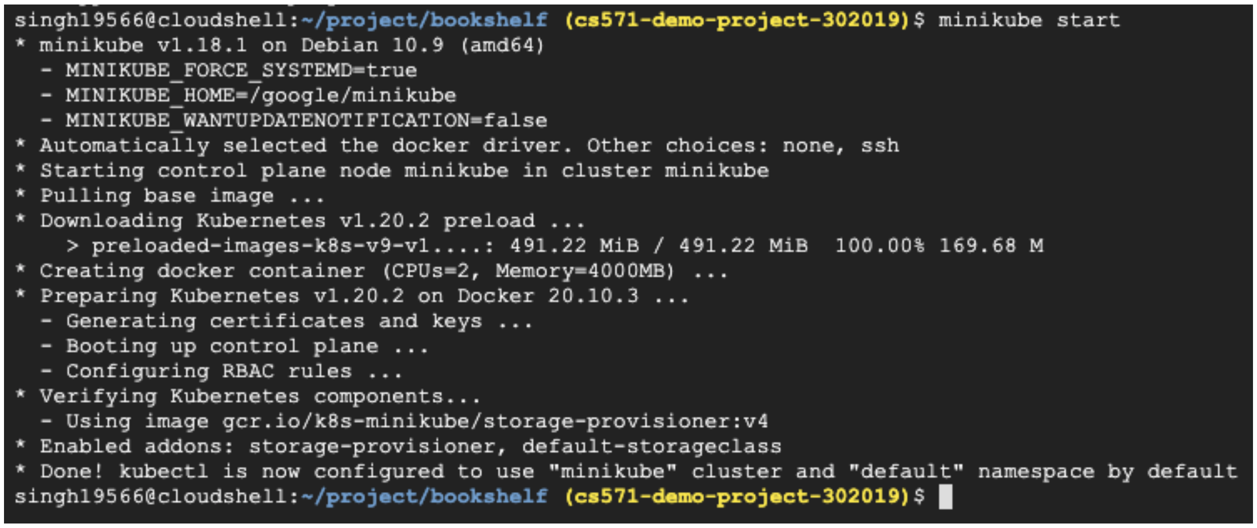
4. Create bookshelf-service.yaml

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



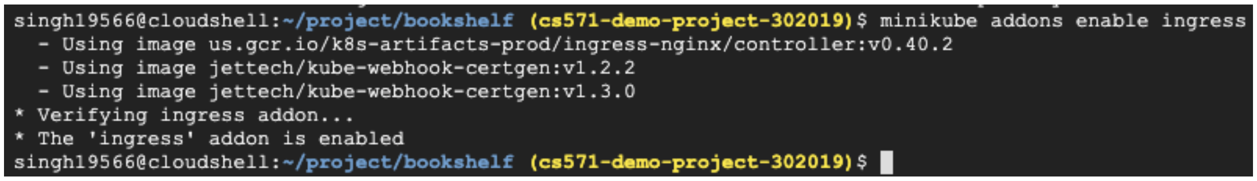
```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ vi bookshelf-service.yaml
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ cat bookshelf-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: bookshelf-service
spec:
  type: LoadBalancer
  ports:
      # service port in cluster
    - port: 5000
      # port to contact inside container
      targetPort: 5000
  selector:
    app: bookshelf-deployment
```

5. Start minikube
   - minikube start



```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ minikube start
* minikube v1.18.1 on Debian 10.9 (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% 169.68 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
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$
```

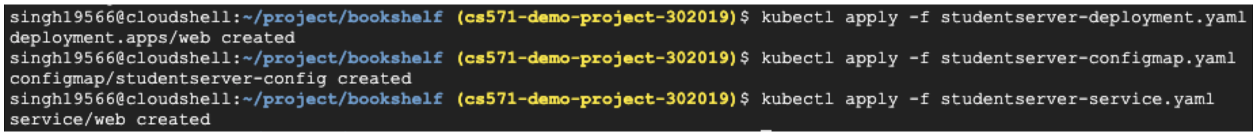
6. Start Ingress
   - minikube addons enable ingress



```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ 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
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$
```

7. Create studentserver related pods and start service using the above yaml file

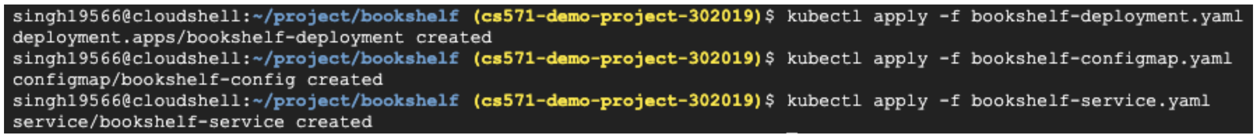
   - kubectl apply -f studentserver-deployment.yaml
   - kubectl apply -f studentserver-configmap.yaml
   - kubectl apply -f studentserver-service.yaml



```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ kubectl apply -f studentserver-deployment.yaml
deployment.apps/web created
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ kubectl apply -f studentserver-configmap.yaml
configmap/studentserver-config created
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ kubectl apply -f studentserver-service.yaml
service/web created
```

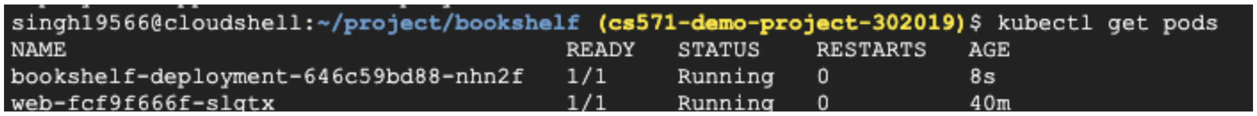
8. Create bookshelf related pods and start service using the above yaml file

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


```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
```

9. Check if all the pods are running correctly

   - kubectl get pods

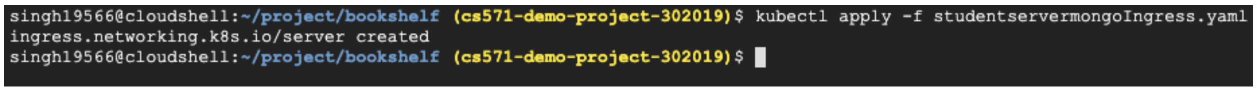

```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ kubectl get pods
NAME                                   READY   STATUS    RESTARTS   AGE
bookshelf-deployment-646c59bd88-nhn2f  1/1     Running   0          8s
web-fcf9f666f-slqtx                    1/1     Running   0          40m
```

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

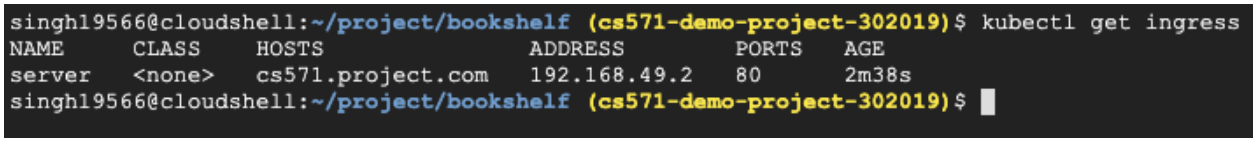
```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: server
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
  rules:
    - host: cs571.project.com
      http:
        paths:
          - path: /studentserver(/|$)(.*)
            pathType: Prefix
            backend: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


```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ kubectl apply -f studentservermongoIngress.yaml
ingress.networking.k8s.io/server created
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$
```

12. Check if ingress is running
    - kubectl get ingress

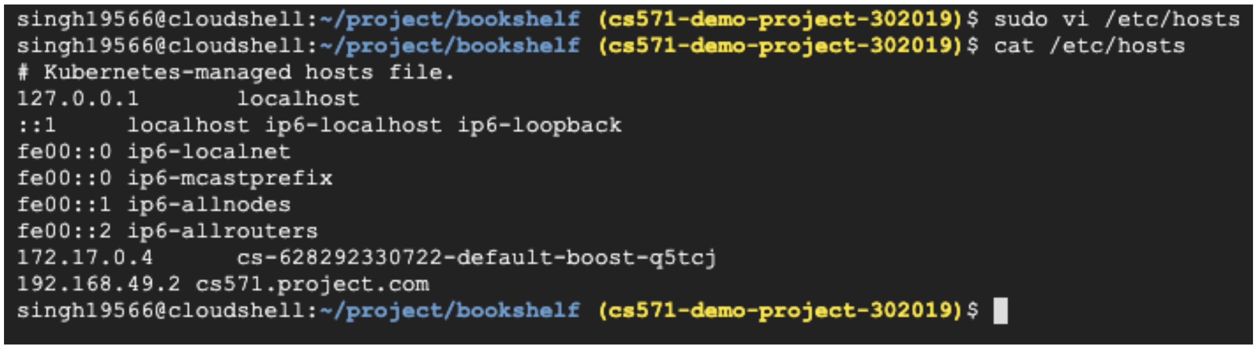


```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ kubectl get ingress
NAME     CLASS    HOSTS               ADDRESS        PORTS   AGE
server   <none>   cs571.project.com   192.168.49.2   80      2m38s
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ 
```

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

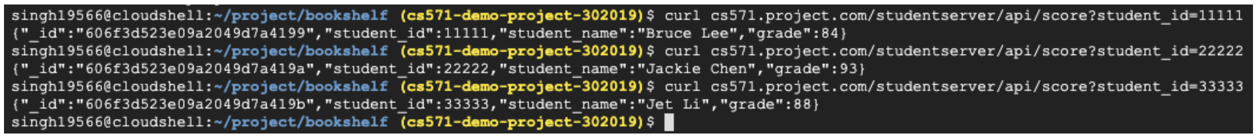
    Your /etc/hosts file should look something like this after adding the line, but your address should be different from mine



```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ sudo vi /etc/hosts
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ cat /etc/hosts
# 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-628292330722-default-boost-q5tcj
192.168.49.2 cs571.project.com
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ 
```

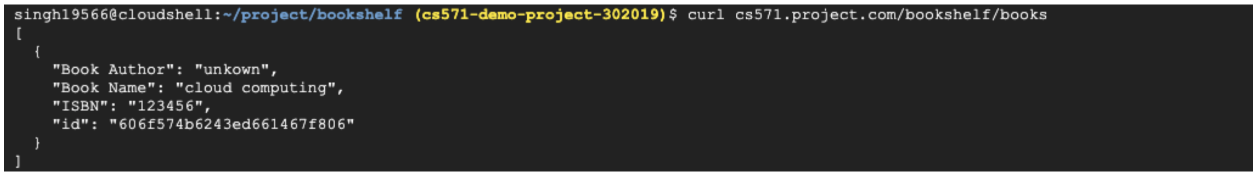
Results

14. If everything goes smoothly, you should be able to access your applications
    - curl cs571.project.com/studentserver/api/score?student_id=11111



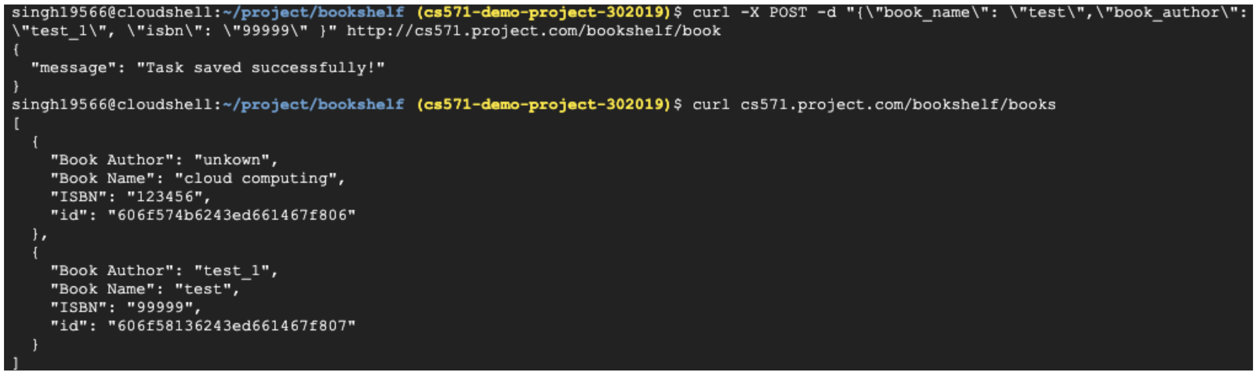
```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ curl cs571.project.com/studentserver/api/score?student_id=11111
{"_id":"606f3d523e09a2049d7a4199","student_id":11111,"student_name":"Bruce Lee","grade":84}
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ curl cs571.project.com/studentserver/api/score?student_id=22222
{"_id":"606f3d523e09a2049d7a419a","student_id":22222,"student_name":"Jackie Chen","grade":93}
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ curl cs571.project.com/studentserver/api/score?student_id=33333
{"_id":"606f3d523e09a2049d7a419b","student_id":33333,"student_name":"Jet Li","grade":88}
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ 
```

15. On another path, we should be able to use the REST API with bookshelf application I.e list all books
    - curl cs571.project.com/bookshelf/books



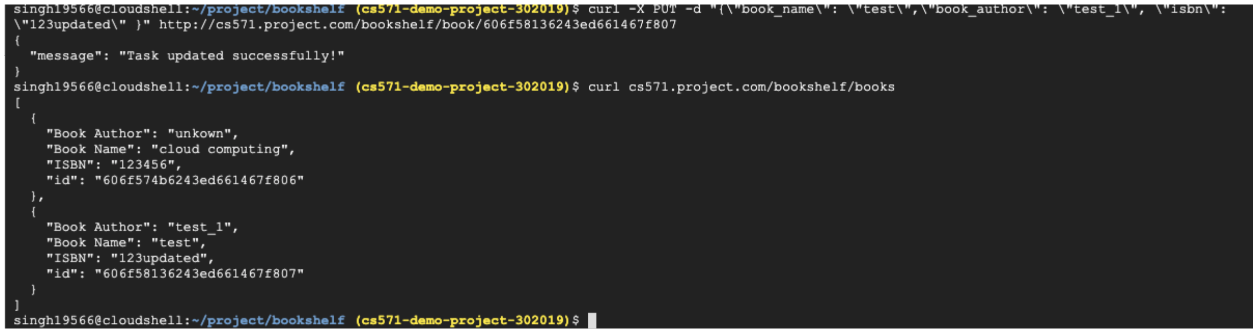
```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "606f574b6243ed661467f806"
  }
]
```

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



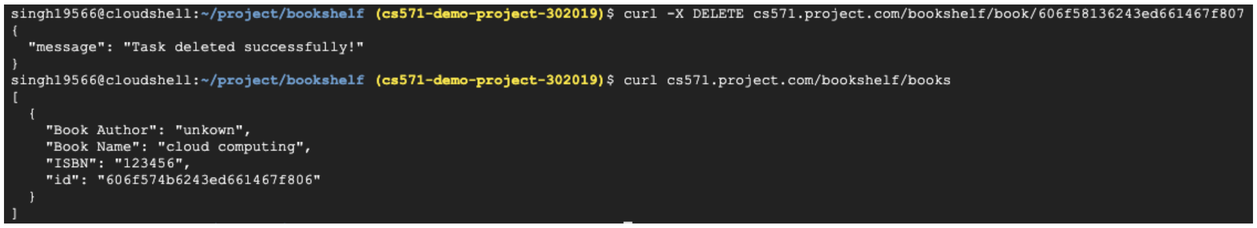
```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ curl -X POST -d "{\"book_name\": \"test\",\"book_author\":
\"test_1\", \"isbn\": \"99999\" }" http://cs571.project.com/bookshelf/book
{
  "message": "Task saved successfully!"
}
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "606f574b6243ed661467f806"
  },
  {
    "Book Author": "test_1",
    "Book Name": "test",
    "ISBN": "99999",
    "id": "606f58136243ed661467f807"
  }
]
```

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



```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ curl -X PUT -d "{\"book_name\": \"test\",\"book_author\": \"test_1\", \"isbn\":
\"123updated\" }" http://cs571.project.com/bookshelf/book/606f58136243ed661467f807
{
  "message": "Task updated successfully!"
}
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "606f574b6243ed661467f806"
  },
  {
    "Book Author": "test_1",
    "Book Name": "test",
    "ISBN": "123updated",
    "id": "606f58136243ed661467f807"
  }
]
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$
```

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



```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ curl -X DELETE cs571.project.com/bookshelf/book/606f58136243ed661467f807
{
  "message": "Task deleted successfully!"
}
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "606f574b6243ed661467f806"
  }
]
```