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

Friday, 03.31.2022

—

Creator

Jenish Patel | 19549

Agenda

- Step1: Create MongoDB using Persistent Volume on GKE, and insert records into it
- Step2: Modify our studentServer to get records from MongoDB and deploy to GKE
- Step3: Create a python Flask bookshelf REST API and deploy on GKE
- Step4: Create ConfigMap for both applications to store MongoDB URL and MongoDB Name
- Step5: Expose 2 application using ingress with Nginx, so we can put them on the same Domain but different PATH

Notes

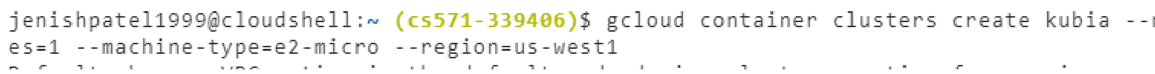
- **Please Install Minikube via**.
- **Install docker via**
  - You must have a docker id

Action Items

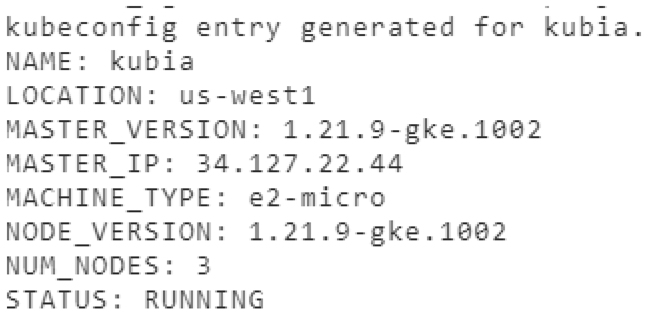
Step1: Create MongoDB using Persistent Volume on GKE, and insert records into it

- Create a cluster as usual on GKE:

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



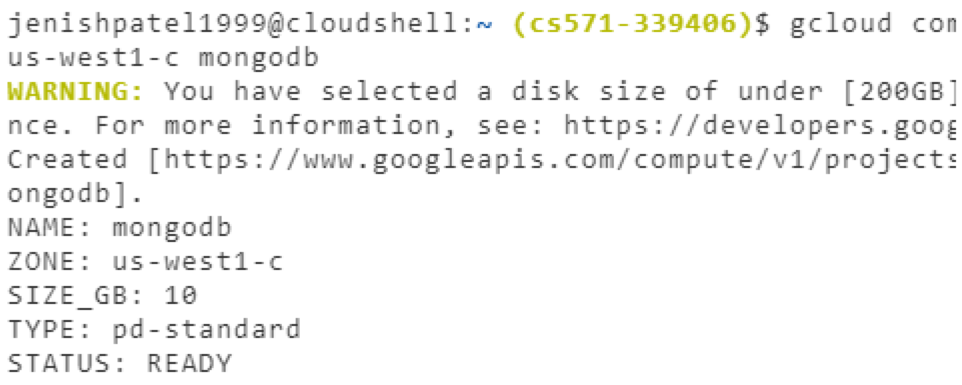
```
jenishpatel1999@cloudshell:~ (cs571-339406)$ gcloud container clusters create kubia --n
es=1 --machine-type=e2-micro --region=us-west1
```



```
kubeconfig entry generated for kubia.
NAME: kubia
LOCATION: us-west1
MASTER_VERSION: 1.21.9-gke.1002
MASTER_IP: 34.127.22.44
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.21.9-gke.1002
NUM_NODES: 3
STATUS: RUNNING
```

- Let's create a Persistent Volume first:

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



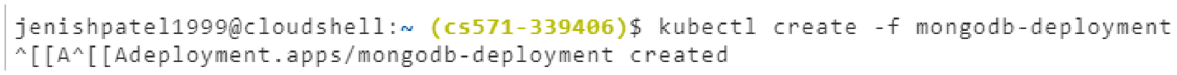
```
jenishpatel1999@cloudshell:~ (cs571-339406)$ gcloud com
us-west1-c mongodb
WARNING: You have selected a disk size of under [200GB]
nce. For more information, see: https://developers.goog
Created [https://www.googleapis.com/compute/v1/projects
ongodb].
NAME: mongodb
ZONE: us-west1-c
SIZE_GB: 10
TYPE: pd-standard
STATUS: READY
```

- Now create a mongodb deployment with this yaml file

```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 defult, 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
```

- And then create a deployment for the file

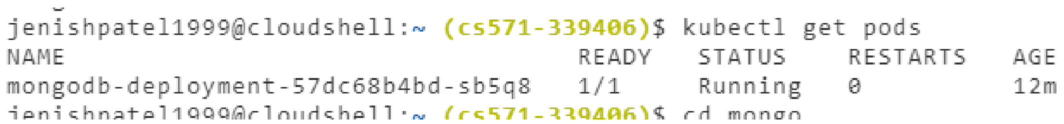
  kubectl apply -f mongodb-deployment.yaml



```
jenishpatel1999@cloudshell:~ (cs571-339406)$ kubectl create -f mongodb-deployment
^[[A^[[Adeployment.apps/mongodb-deployment created
```

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

  `Kubectl get pods`



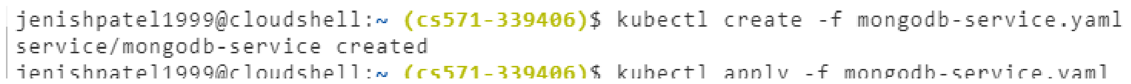
```
jenishpatel1999@cloudshell:~ (cs571-339406)$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
mongodb-deployment-57dc68b4bd-sb5q8   1/1     Running   0          12m
jenishpatel1999@cloudshell:~ (cs571-339406)$ cd mongo
```

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

```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
```

- And then create a deployment for the file

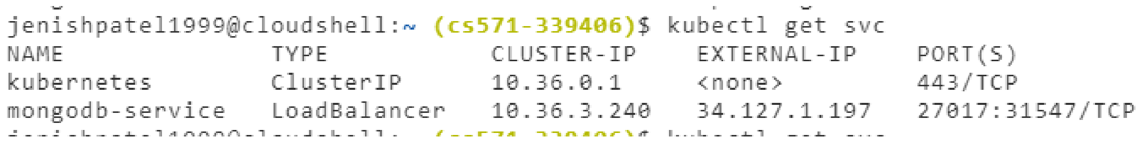
  kubectl create -f mongodb-service.yaml



```
jenishpatel1999@cloudshell:~ (cs571-339406)$ kubectl create -f mongodb-service.yaml
service/mongodb-service created
jenishpatel1999@cloudshell:~ (cs571-339406)$ kubectl apply -f mongodb-service.yaml
```

- Check the ip-address

  Kubectl get svc



```
jenishpatel1999@cloudshell:~ (cs571-339406)$ kubectl get svc
NAME              TYPE           CLUSTER-IP    EXTERNAL-IP    PORT(S)
kubernetes        ClusterIP      10.36.0.1     <none>         443/TCP
mongodb-service   LoadBalancer   10.36.3.240   34.127.1.197   27017:31547/TCP
```

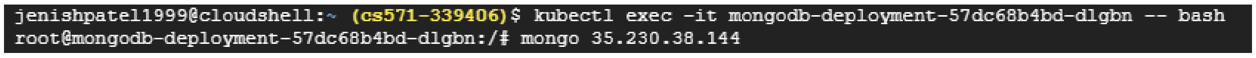
- Now try and see if mongoDB is functioning for connections using the External-IP

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

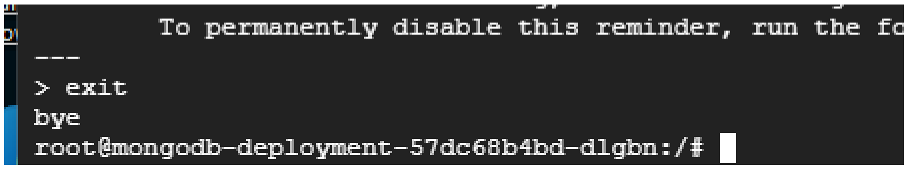
  Try

  mongo External-IP

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



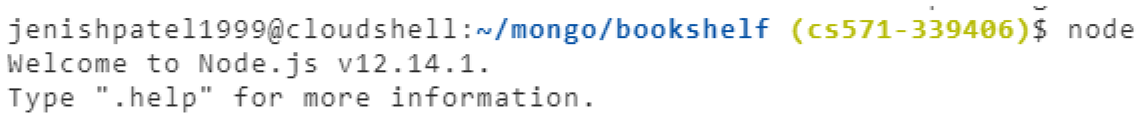
```
jenishpatel1999@cloudshell:~ (cs571-339406)$ kubectl exec -it mongodb-deployment-57dc68b4bd-dlgbn -- bash
root@mongodb-deployment-57dc68b4bd-dlgbn:/# mongo 35.230.38.144
```



```
        To permanently disable this reminder, run the fo
---
> exit
bye
root@mongodb-deployment-57dc68b4bd-dlgbn:/#
```

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

  node



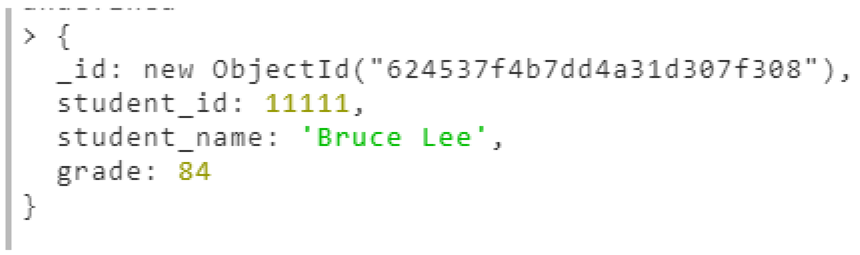
```
jenishpatel1999@cloudshell:~/mongo/bookshelf (cs571-339406)$ node
Welcome to Node.js v12.14.1.
Type ".help" for more information.
```

- To connect with your mongodb try following steps
  - var MongoClient = require('mongodb').MongoClient;
  - var url = "mongodb://EXTERNAL-IP/mydb"

- To add some records inside your database follow the below steps

  -
    ```
    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);
    });
    });
    ```

    - You should get the output like this



```
> {
  _id: new ObjectId("624537f4b7dd4a31d307f308"),
  student_id: 11111,
  student_name: 'Bruce Lee',
  grade: 84
}
```

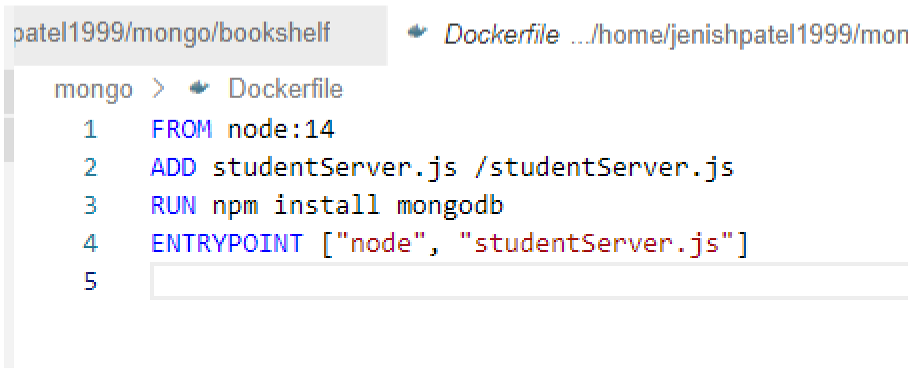
- Step 2: Modify our studentServer to get records from MongoDB and deploy to GKE

    - Create a studentServer.js file

```
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);
```

- Create Dockerfile

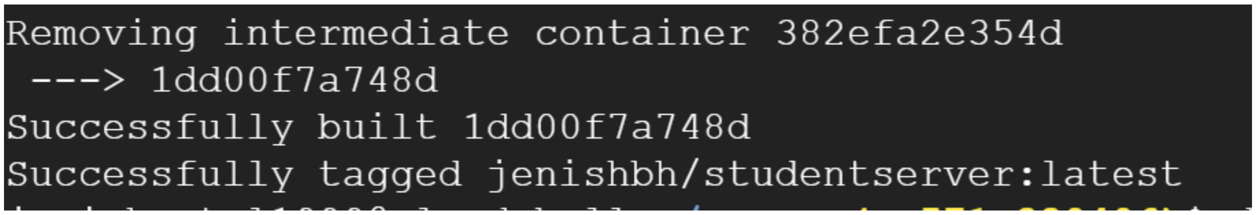
```
FROM node:14
ADD studentServer.js /studentServer.js
RUN npm install mongodb
ENTRYPOINT ["node", "studentServer.js"]
```


patel1999/mongo/bookshelf
Dockerfile .../home/jenishpatel1999/mon
mongo > Dockerfile

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

- Build the studentserver docker image

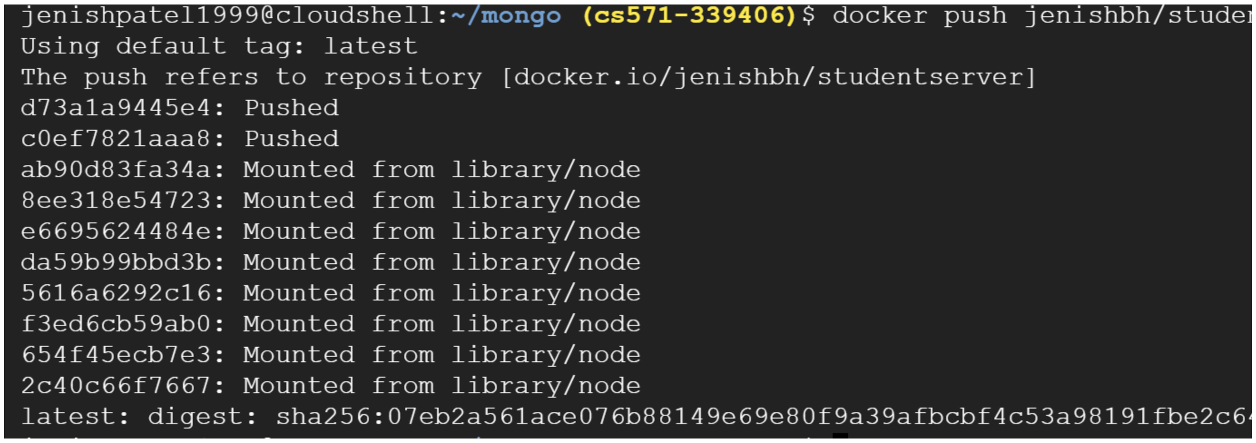
  docker build -t yourdockerhubID/studentserver .



```
Removing intermediate container 382efa2e354d
 ---> 1dd00f7a748d
Successfully built 1dd00f7a748d
Successfully tagged jenishbh/studentserver:latest
```

- Push the docker image
  - If not then login the docker using
    docker login -u username -p password

  docker push yourdockerhubID/studentserver



```
jenishpatel1999@cloudshell:~/mongo (cs571-339406)$ docker push jenishbh/studen
Using default tag: latest
The push refers to repository [docker.io/jenishbh/studentserver]
d73a1a9445e4: Pushed
c0ef7821aaa8: 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: sha256:07eb2a561ace076b88149e69e80f9a39afbcbf4c53a98191fbe2c6
```

- Step 3 :Create a python Flask bookshelf REST API and deploy on GKE
    - . 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_DATA
BASE")
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)
```

- Create a Dockerfile

```
FROM python:3.8-slim-buster
WORKDIR /app
RUN pip3 install flask
RUN pip3 install flask_pymongo
COPY . /app
ENV PORT 5000
EXPOSE 5000
ENTRYPOINT [ "python3" ]
CMD [ "bookshelf.py" ]
```

  - Build the docker image
    docker build -t jenishbh/bookshelf

    - Push the docker image

    docker push jenishbh/bookshelf

- Step 4 :Create ConfigMap for both applications to store MongoDB URL and MongoDB name

  - 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
```

Step 5: Expose 2 application using ingress with Nginx, so we can put them on the same Domain but different PATH

- 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: jenishbh/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
```

- 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: jenishbh/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
```

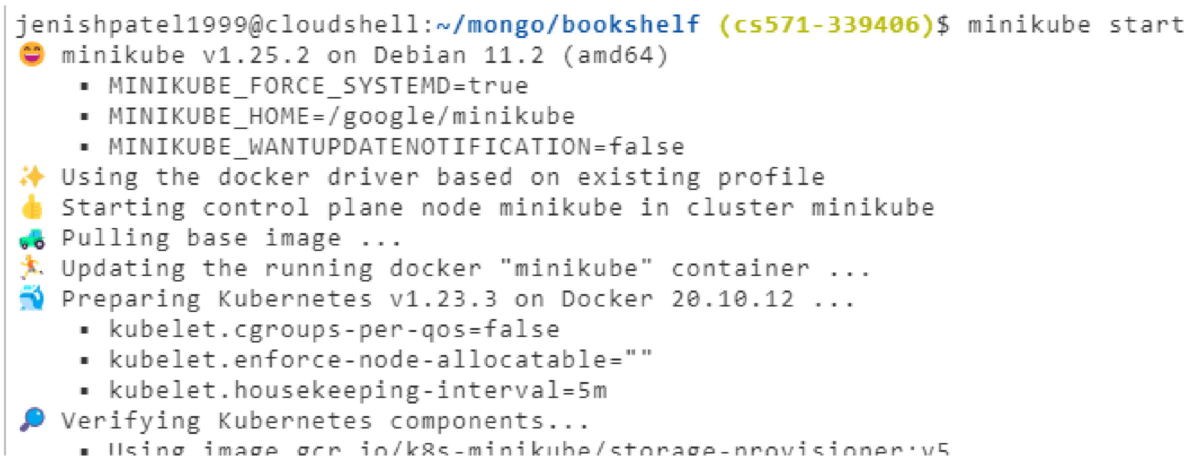
- Create sutdentserver-service.yaml

```
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
```

- . 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
```

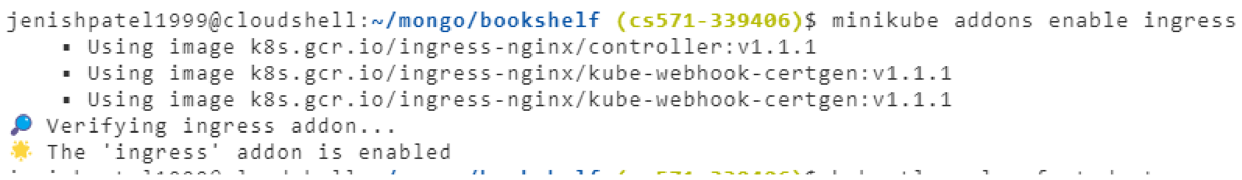
- Start minikube



```
jenishpatel1999@cloudshell:~/mongo/bookshelf (cs571-339406)$ minikube start
😄  minikube v1.25.2 on Debian 11.2 (amd64)
    ▪ MINIKUBE_FORCE_SYSTEMD=true
    ▪ MINIKUBE_HOME=/google/minikube
    ▪ MINIKUBE_WANTUPDATENOTIFICATION=false
✨  Using the docker driver based on existing profile
👍  Starting control plane node minikube in cluster minikube
🚜  Pulling base image ...
🏃  Updating the running docker "minikube" container ...
🐳  Preparing Kubernetes v1.23.3 on Docker 20.10.12 ...
    ▪ kubelet.cgroups-per-qos=false
    ▪ kubelet.enforce-node-allocatable=""
    ▪ kubelet.housekeeping-interval=5m
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
```

- Start Ingress

  minikube addons enable ingress



```
jenishpatel1999@cloudshell:~/mongo/bookshelf (cs571-339406)$ 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
```

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

  kubectl apply -f studentserver-deployment.yaml

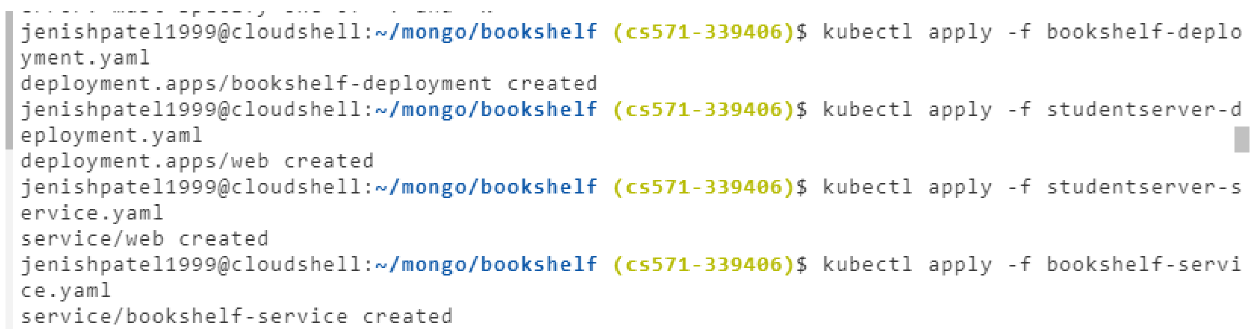
  kubectl apply -f studentserver-configmap.yaml

  kubectl apply -f studentserver-service.yaml

  kubectl apply -f bookshelf-deployment.yaml

  kubectl apply -f bookshelf-configmap.yaml

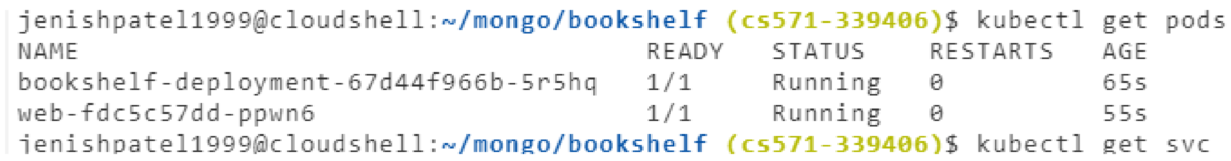
  kubectl apply -f bookshelf-service.yaml



```
jenishpatel1999@cloudshell:~/mongo/bookshelf (cs571-339406)$ kubectl apply -f bookshelf-deplo
yment.yaml
deployment.apps/bookshelf-deployment created
jenishpatel1999@cloudshell:~/mongo/bookshelf (cs571-339406)$ kubectl apply -f studentserver-d
eployment.yaml
deployment.apps/web created
jenishpatel1999@cloudshell:~/mongo/bookshelf (cs571-339406)$ kubectl apply -f studentserver-s
ervice.yaml
service/web created
jenishpatel1999@cloudshell:~/mongo/bookshelf (cs571-339406)$ kubectl apply -f bookshelf-servi
ce.yaml
service/bookshelf-service created
```

  - Check if all the pods are running correctly

    kubectl get pods



```
jenishpatel1999@cloudshell:~/mongo/bookshelf (cs571-339406)$ kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
bookshelf-deployment-67d44f966b-5r5hq   1/1     Running   0          65s
web-fdc5c57dd-ppwn6                     1/1     Running   0          55s
jenishpatel1999@cloudshell:~/mongo/bookshelf (cs571-339406)$ kubectl get svc
```

- 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
```

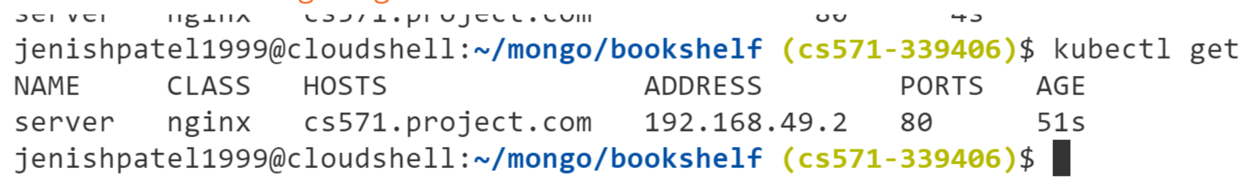
```
http:
  paths:
    - path: /studentserver(/|$)(.*)
      pathType: Prefix
      backend:
        service:
          name: web
          port:
            number: 8080
    - path: /bookshelf(/|$)(.*)
      pathType: Prefix
      backend:
        service:
          name: bookshelf-service
          port:
            number: 5000
```

- Create the ingress service using the above yaml file

  kubectl apply -f studentservermongoIngress.yaml

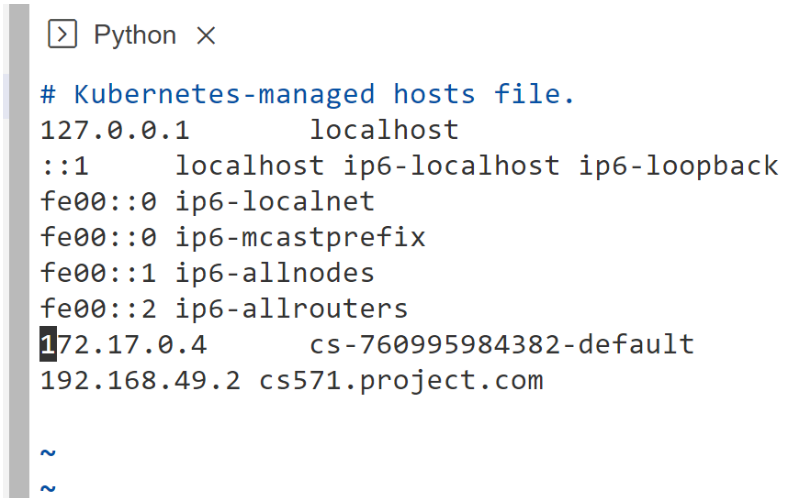
- . Check if ingress is running

  kubectl get ingress



```
jenishpatel1999@cloudshell:~/mongo/bookshelf (cs571-339406)$ kubectl get
NAME     CLASS   HOSTS               ADDRESS        PORTS   AGE
server   nginx   cs571.project.com   192.168.49.2   80      51s
jenishpatel1999@cloudshell:~/mongo/bookshelf (cs571-339406)$
```

- 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

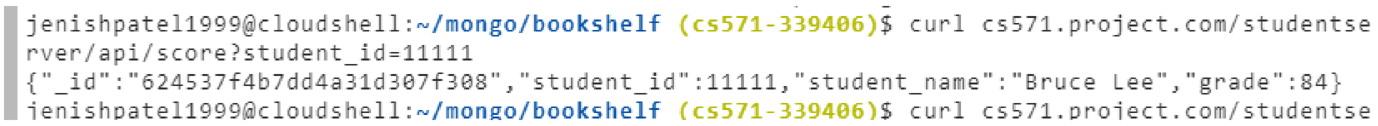


```
Python ×

# 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-760995984382-default
192.168.49.2 cs571.project.com

~
~
```

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

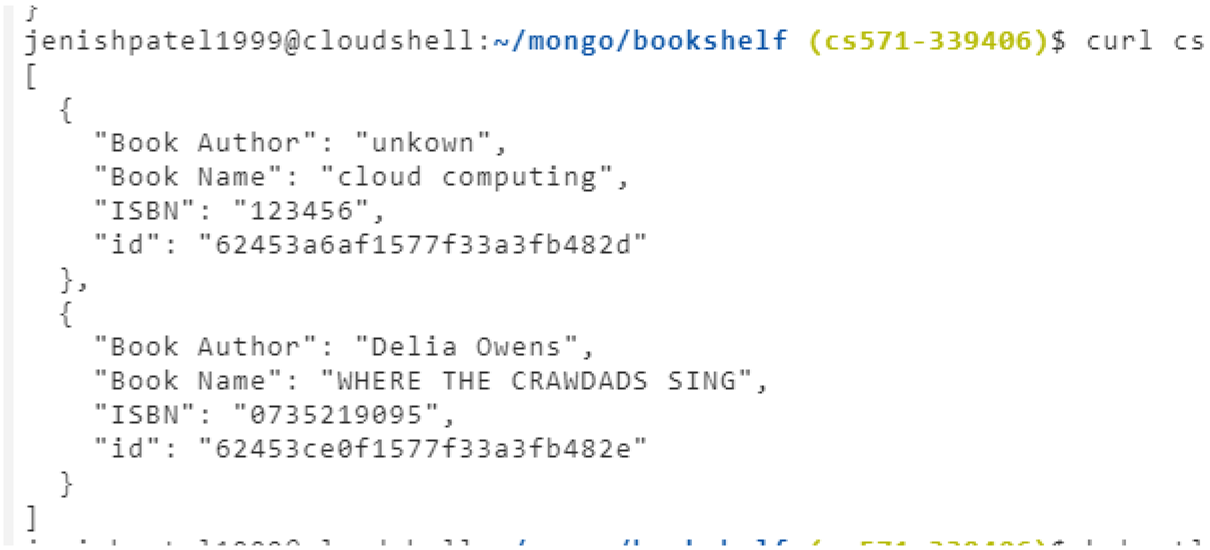


```
jenishpatel1999@cloudshell:~/mongo/bookshelf (cs571-339406)$ curl cs571.project.com/studentse
rver/api/score?student_id=11111
{"_id":"624537f4b7dd4a31d307f308","student_id":11111,"student_name":"Bruce Lee","grade":84}
jenishpatel1999@cloudshell:~/mongo/bookshelf (cs571-339406)$ curl cs571.project.com/studentse
```

- On another path, you should be able to use the REST API with bookshelf application

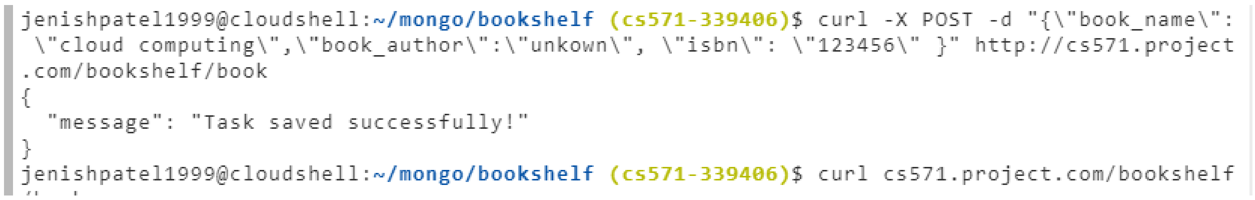
I.e list all books

curl cs571.project.com/bookshelf/books



```
jenishpatel1999@cloudshell:~/mongo/bookshelf (cs571-339406)$ curl cs
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "62453a6af1577f33a3fb482d"
  },
  {
    "Book Author": "Delia Owens",
    "Book Name": "WHERE THE CRAWDADS SING",
    "ISBN": "0735219095",
    "id": "62453ce0f1577f33a3fb482e"
  }
]
```

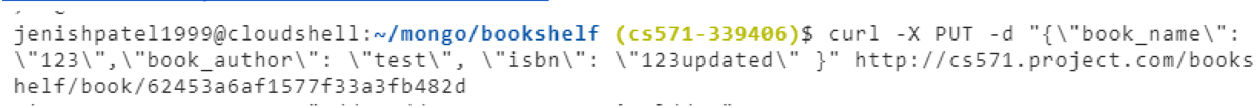
- Add a book
  curl -X POST -d "{\"book_name\": \"cloud computing\",\"book_author\":\"unkown\", \"isbn\": \"123456\" }" http://cs571.project.com/bookshelf/book



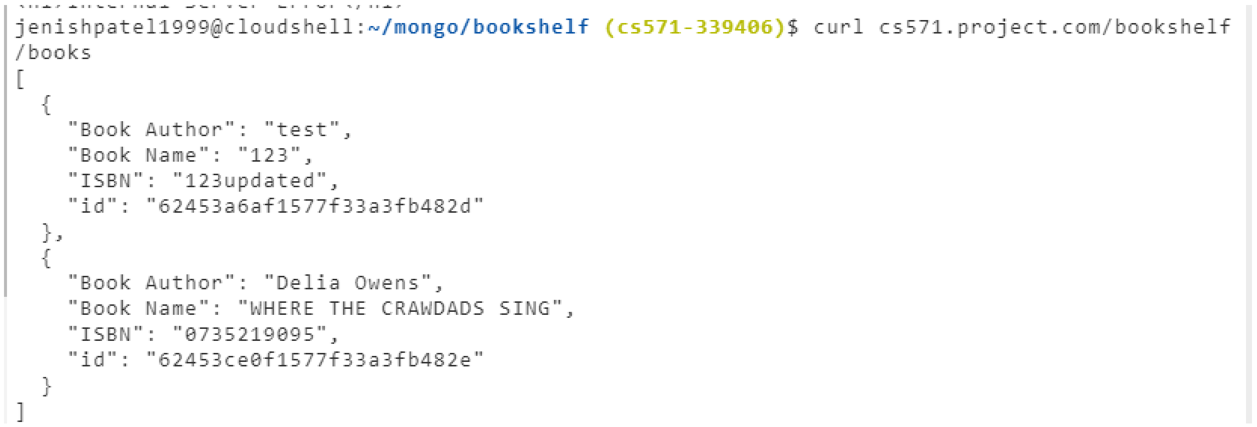
```
jenishpatel1999@cloudshell:~/mongo/bookshelf (cs571-339406)$ curl -X POST -d "{\"book_name\":
 \"cloud computing\",\"book_author\":\"unkown\", \"isbn\": \"123456\" }" http://cs571.project
.com/bookshelf/book
{
  "message": "Task saved successfully!"
}
jenishpatel1999@cloudshell:~/mongo/bookshelf (cs571-339406)$ curl cs571.project.com/bookshelf
```

- Update a book

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

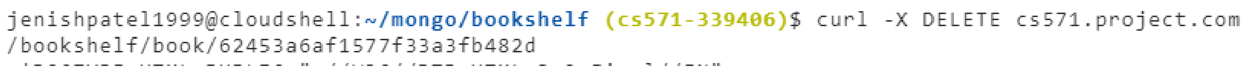


```
jenishpatel1999@cloudshell:~/mongo/bookshelf (cs571-339406)$ curl -X PUT -d "{\"book_name\":
\"123\",\"book_author\": \"test\", \"isbn\": \"123updated\" }" http://cs571.project.com/books
helf/book/62453a6af1577f33a3fb482d
```

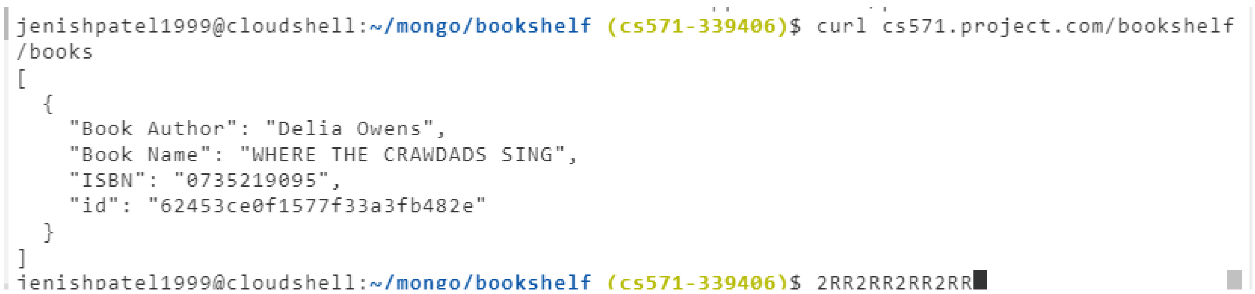


```
jenishpatel1999@cloudshell:~/mongo/bookshelf (cs571-339406)$ curl cs571.project.com/bookshelf
/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123updated",
    "id": "62453a6af1577f33a3fb482d"
  },
  {
    "Book Author": "Delia Owens",
    "Book Name": "WHERE THE CRAWDADS SING",
    "ISBN": "0735219095",
    "id": "62453ce0f1577f33a3fb482e"
  }
]
```

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



```
jenishpatel1999@cloudshell:~/mongo/bookshelf (cs571-339406)$ curl -X DELETE cs571.project.com
/bookshelf/book/62453a6af1577f33a3fb482d
```



```
jenishpatel1999@cloudshell:~/mongo/bookshelf (cs571-339406)$ curl cs571.project.com/bookshelf
/books
[
  {
    "Book Author": "Delia Owens",
    "Book Name": "WHERE THE CRAWDADS SING",
    "ISBN": "0735219095",
    "id": "62453ce0f1577f33a3fb482e"
  }
]
jenishpatel1999@cloudshell:~/mongo/bookshelf (cs571-339406)$ 2RR2RR2RR2RR
```

Next Meeting Agenda Items

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.