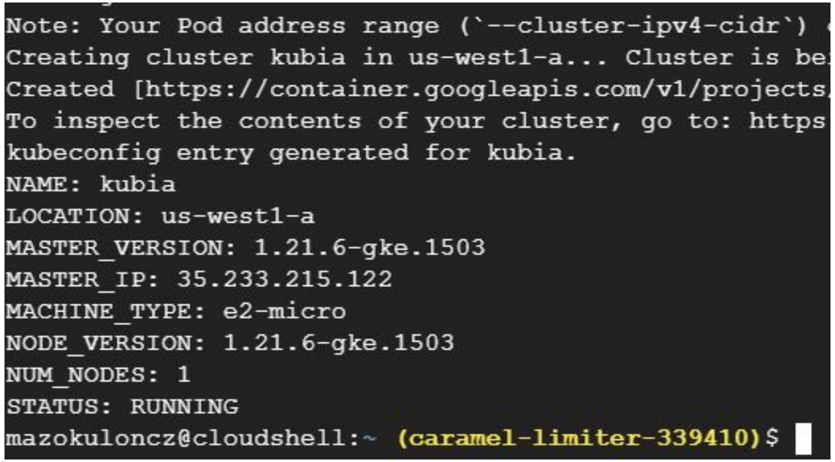
Signature Project: MongoDB + Python Flask Web Framework + REST API + GKE

Long Nguyen Hoang

Create cluster on GKE

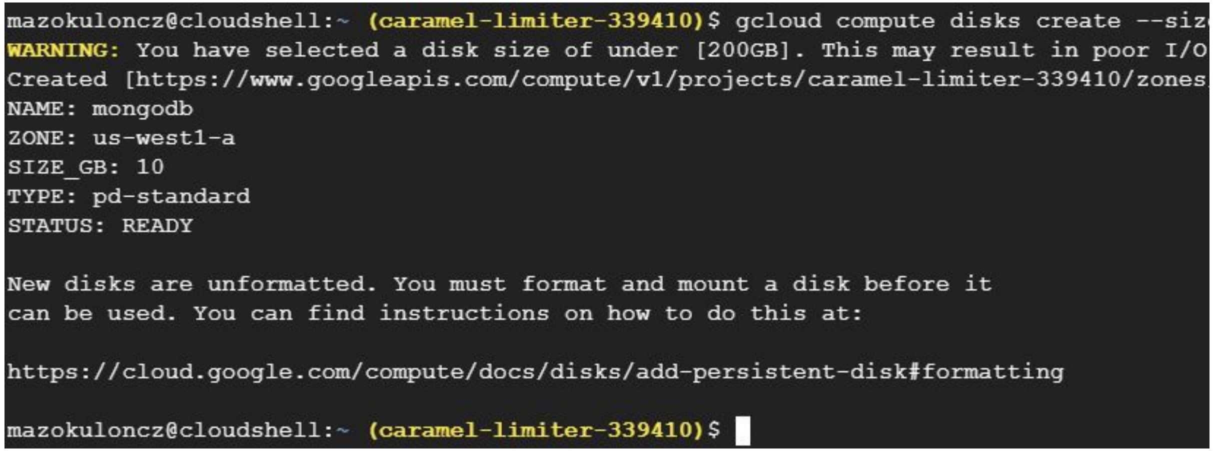
- gcloud container clusters create kubia --num-nodes=1 --machine-type=e2-micro --region=us-west1



```
Note: Your Pod address range (`--cluster-ipv4-cidr`)
Creating cluster kubia in us-west1-a... Cluster is be
Created [https://container.googleapis.com/v1/projects
To inspect the contents of your cluster, go to: https
kubeconfig entry generated for kubia.
NAME: kubia
LOCATION: us-west1-a
MASTER_VERSION: 1.21.6-gke.1503
MASTER_IP: 35.233.215.122
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.21.6-gke.1503
NUM_NODES: 1
STATUS: RUNNING
mazokuloncz@cloudshell:~ (caramel-limiter-339410)$
```

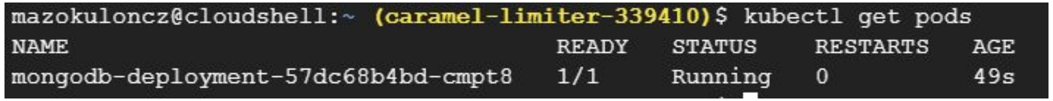
Let’s create a Persistent Volume first

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

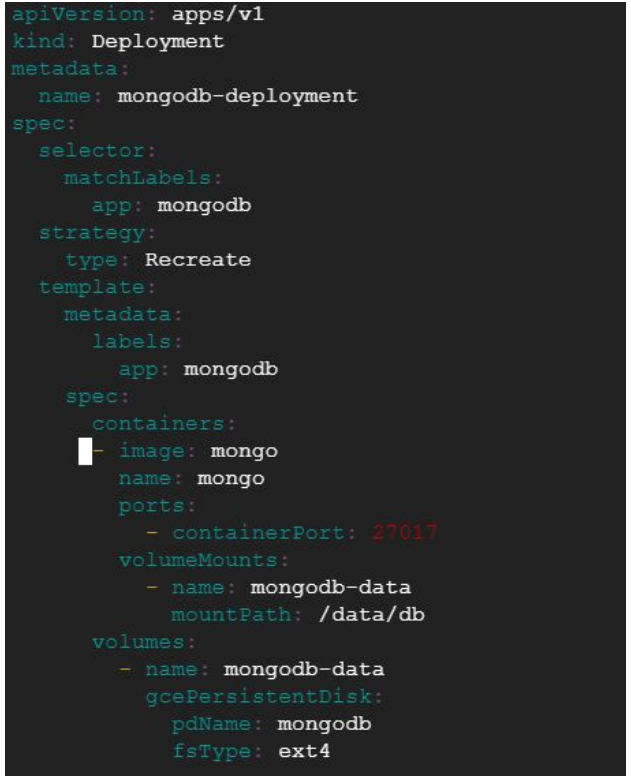


```
mazokuloncz@cloudshell:~ (caramel-limiter-339410)$ gcloud compute disks create --siz
WARNING: You have selected a disk size of under [200GB]. This may result in poor I/O
Created [https://www.googleapis.com/compute/v1/projects/caramel-limiter-339410/zones
NAME: mongodb
ZONE: us-west1-a
SIZE_GB: 10
TYPE: pd-standard
STATUS: READY

New disks are unformatted. You must format and mount a disk before it
can be used. You can find instructions on how to do this at:

https://cloud.google.com/compute/docs/disks/add-persistent-disk#formatting

mazokuloncz@cloudshell:~ (caramel-limiter-339410)$
```

Now create a mongodb deployment with this yaml filec

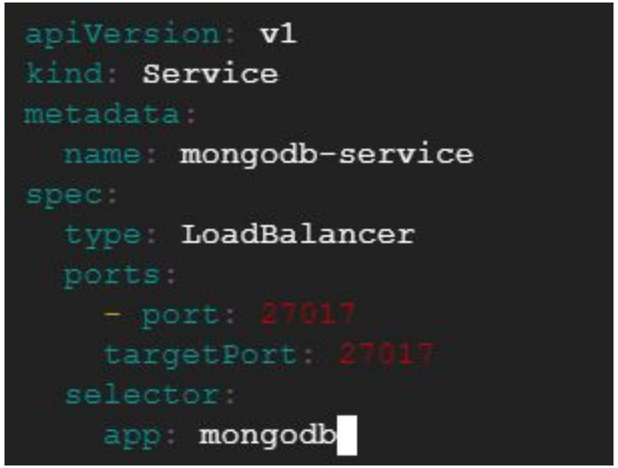


```
mazokuloncz@cloudshell:~ (caramel-limiter-339410)$ kubectl get pods
NAME                                 READY   STATUS    RESTARTS   AGE
mongodb-deployment-57dc68b4bd-cmpt8  1/1     Running   0          49s
```

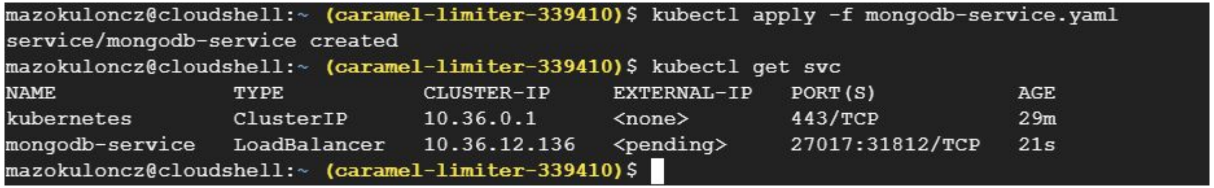


```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
spec:
  selector:
    matchLabels:
      app: mongodb
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
      - image: mongo
        name: mongo
   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

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

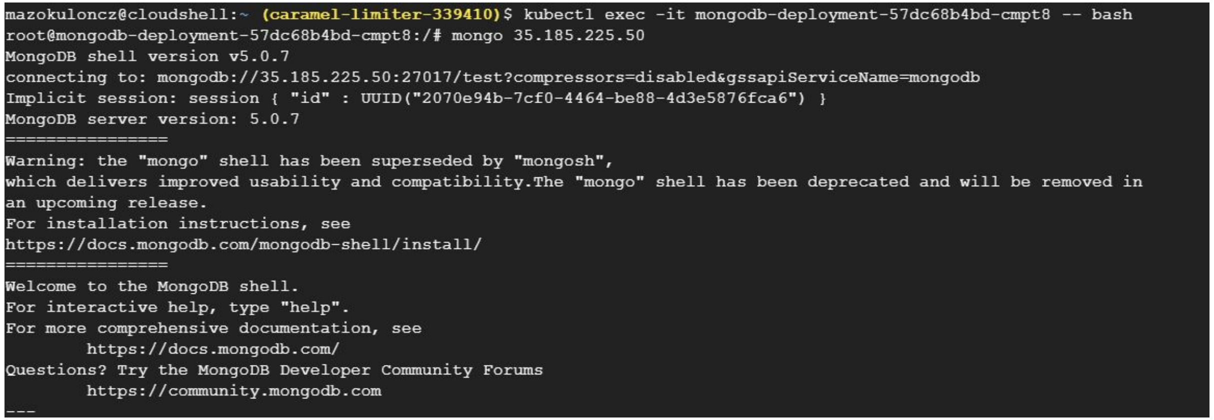


```yaml
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  type: LoadBalancer
  ports:
    - port: 27017
    targetPort: 27017
  selector:
    app: mongodb
```



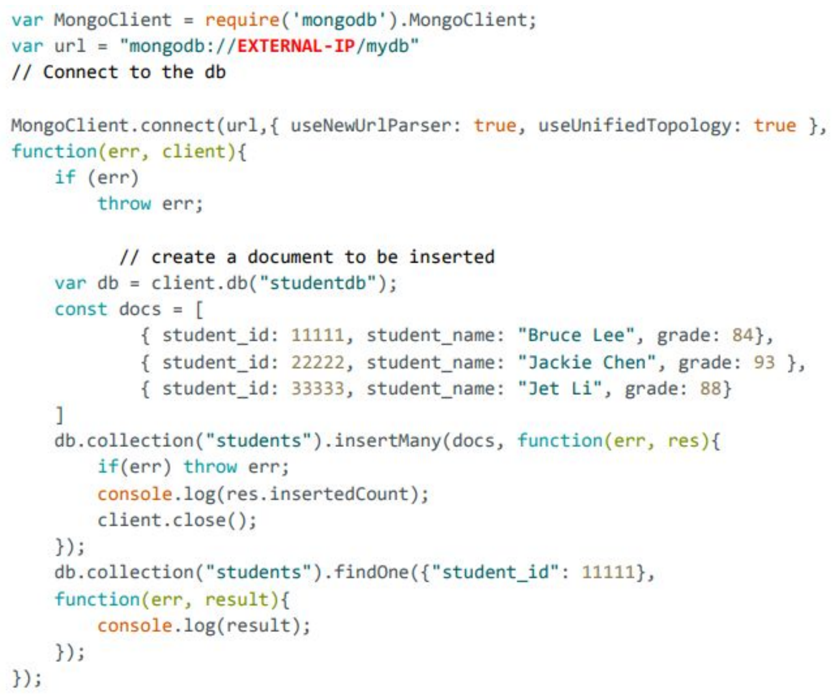
```
mazokuloncz@cloudshell:~ (caramel-limiter-339410)$ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
mazokuloncz@cloudshell:~ (caramel-limiter-339410)$ kubectl get svc
NAME              TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)           AGE
kubernetes        ClusterIP      10.36.0.1      <none>        443/TCP           29m
mongodb-service   LoadBalancer   10.36.12.136   <pending>     27017:31812/TCP   21s
mazokuloncz@cloudshell:~ (caramel-limiter-339410)$
```

Connect mongoDB is functioning for connections using the External-IP



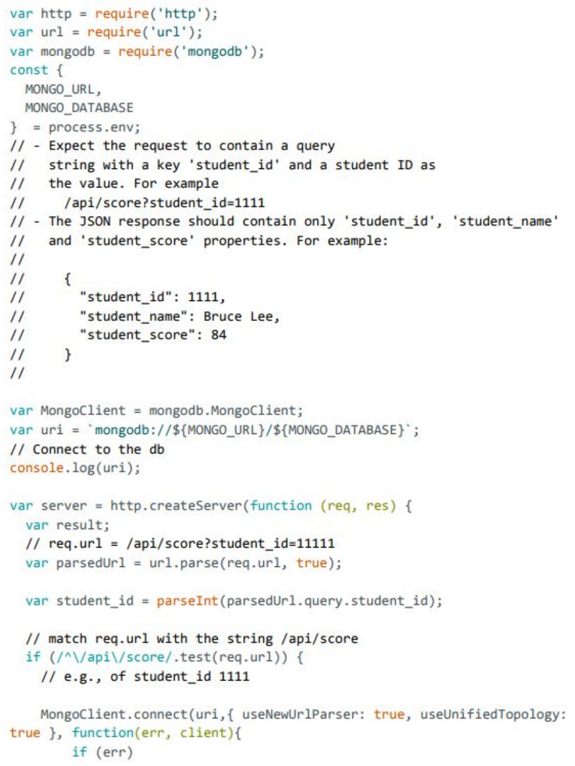
```
mazokuloncz@cloudshell:~ (caramel-limiter-339410)$ kubectl exec -it mongodb-deployment-57dc68b4bd-cmpt8 -- bash
root@mongodb-deployment-57dc68b4bd-cmpt8:/# mongo 35.185.225.50
MongoDB shell version v5.0.7
connecting to: mongodb://35.185.225.50:27017/test?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("2070e94b-7cf0-4464-be88-4d3e5876fca6") }
MongoDB server version: 5.0.7
================
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
================
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
---
```

Insert records into the mongoDB using node

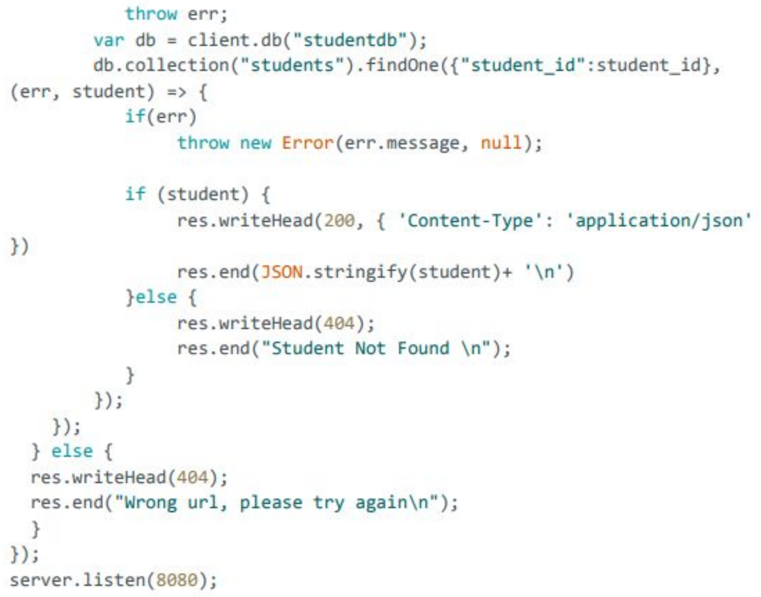


```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://EXTERNAL-IP/mydb"
// Connect to the db

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);
    });
});
```

Create a studentServer.js

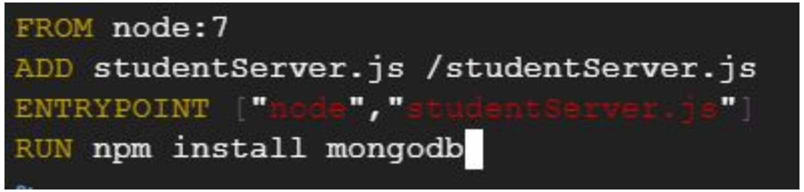


```
var http = require('http');
var url = require('url');
var mongodb = require('mongodb');
const {
  MONGO_URL,
  MONGO_DATABASE
}  = process.env;
// - Expect the request to contain a query
//   string with a key 'student_id' and a student ID as
//   the value. For example
//     /api/score?student_id=1111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

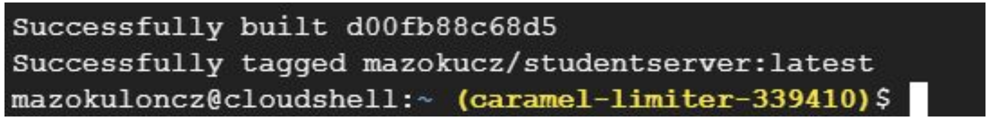


```
               throw err;
           var db = client.db("studentdb");
           db.collection("students").findOne({"student_id":student_id},
(err, student) => {
               if(err)
                   throw new Error(err.message, null);

               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);
```

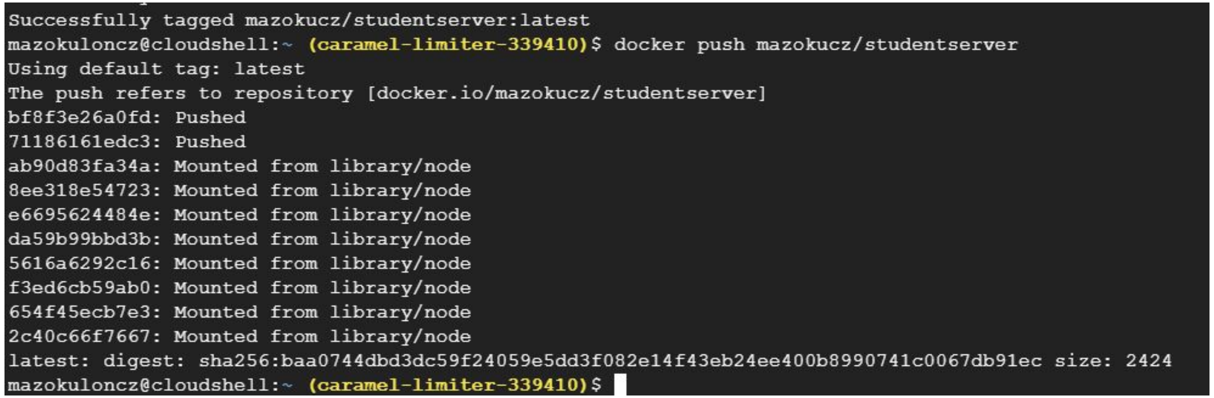
Build the studentserver docker image



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

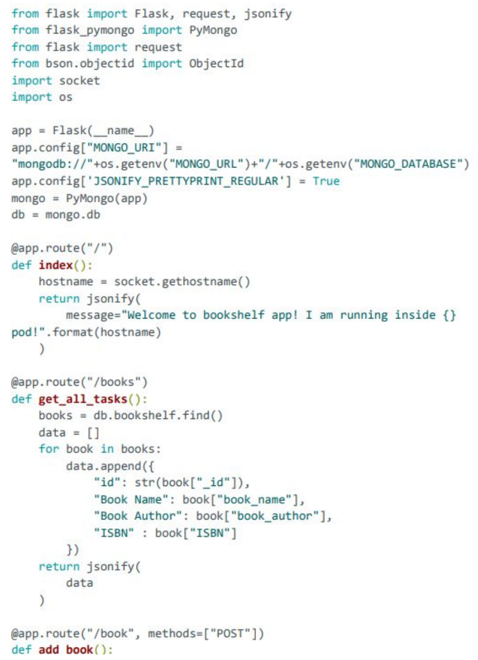


```
Successfully built d00fb88c68d5
Successfully tagged mazokucz/studentserver:latest
mazokuloncz@cloudshell:~ (caramel-limiter-339410)$
```

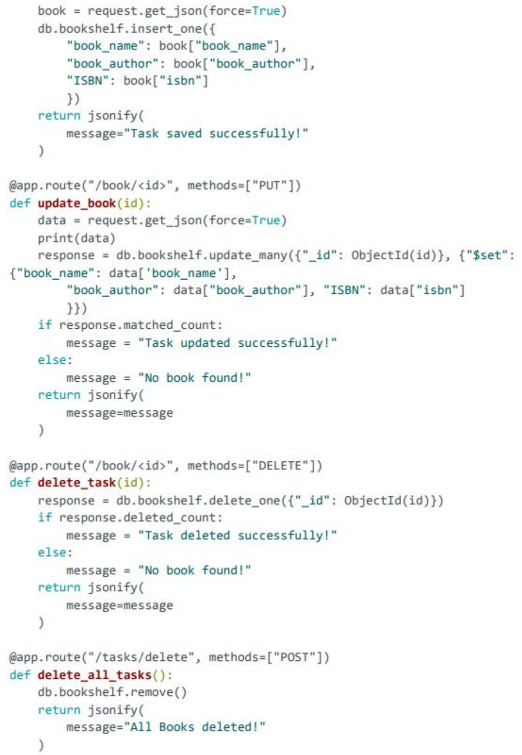


```
Successfully tagged mazokucz/studentserver:latest
mazokuloncz@cloudshell:~ (caramel-limiter-339410)$ docker push mazokucz/studentserver
Using default tag: latest
The push refers to repository [docker.io/mazokucz/studentserver]
bf8f3e26a0fd: Pushed
71186161edc3: Pushed
ab90d83fa34a: Mounted from library/node
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:baa0744dbd3dc59f24059e5dd3f082e14f43eb24ee400b8990741c0067db91ec size: 2424
mazokuloncz@cloudshell:~ (caramel-limiter-339410)$
```

Create a python Flask bookshelf REST API and deploy on GKE

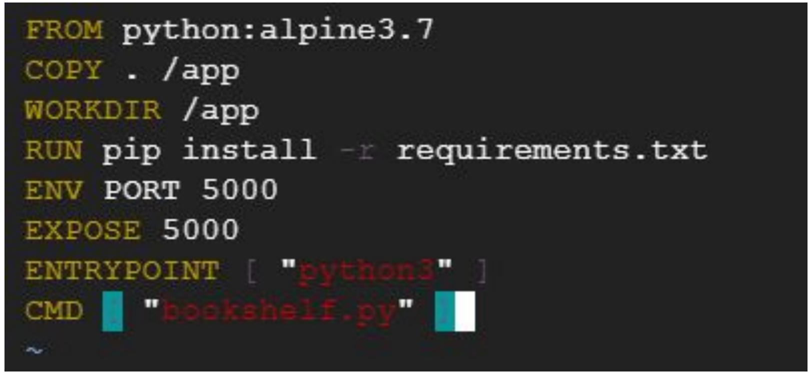


```
from flask import Flask, request, jsonify
from flask_pymongo import PyMongo
from flask import request
from bson.objectid import ObjectId
import socket
import os

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():
```

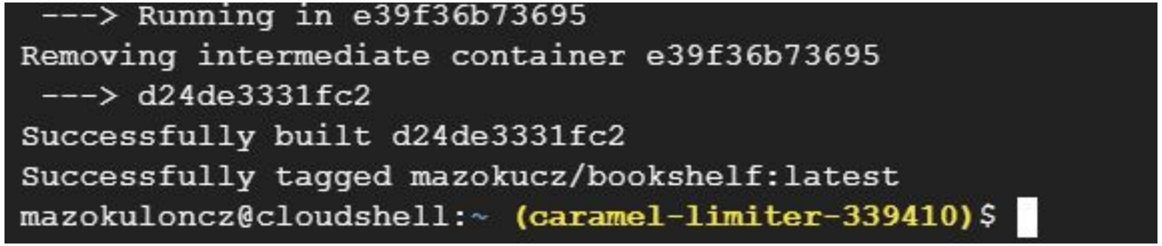


```
    book = request.get_json(force=True)
    db.bookshelf.insert_one({
        "book_name": book["book_name"],
        "book_author": book["book_author"],
        "ISBN": book["isbn"]
        })
    return jsonify(
        message="Task saved successfully!"
    )

@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!"
    )
```

Create a bookshelf Dockerfile

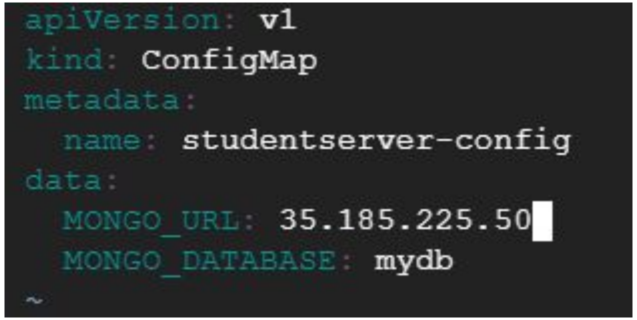


```
FROM python:alpine3.7
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
ENV PORT 5000
EXPOSE 5000
ENTRYPOINT [ "python3" ]
CMD [ "bookshelf.py" ]
~
```



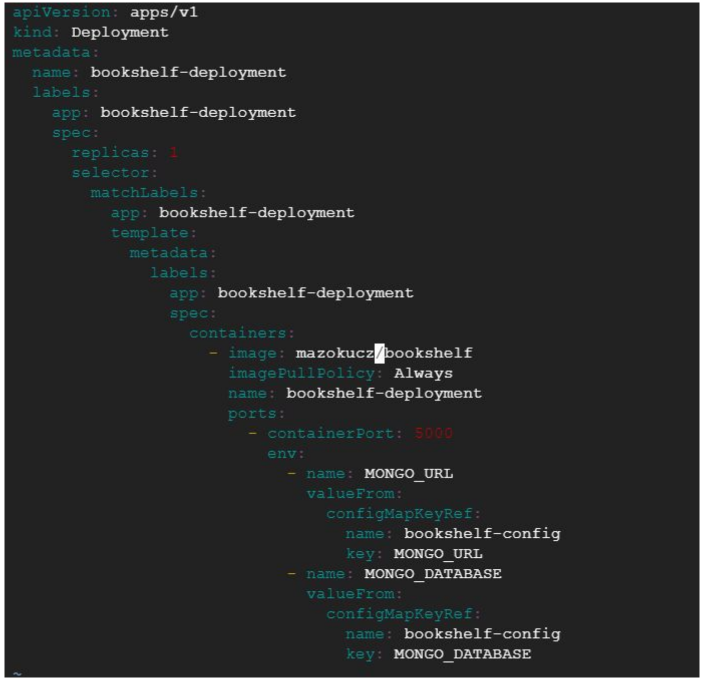
```
 ---> Running in e39f36b73695
Removing intermediate container e39f36b73695
 ---> d24de3331fc2
Successfully built d24de3331fc2
Successfully tagged mazokucz/bookshelf:latest
mazokuloncz@cloudshell:~ (caramel-limiter-339410)$
```

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

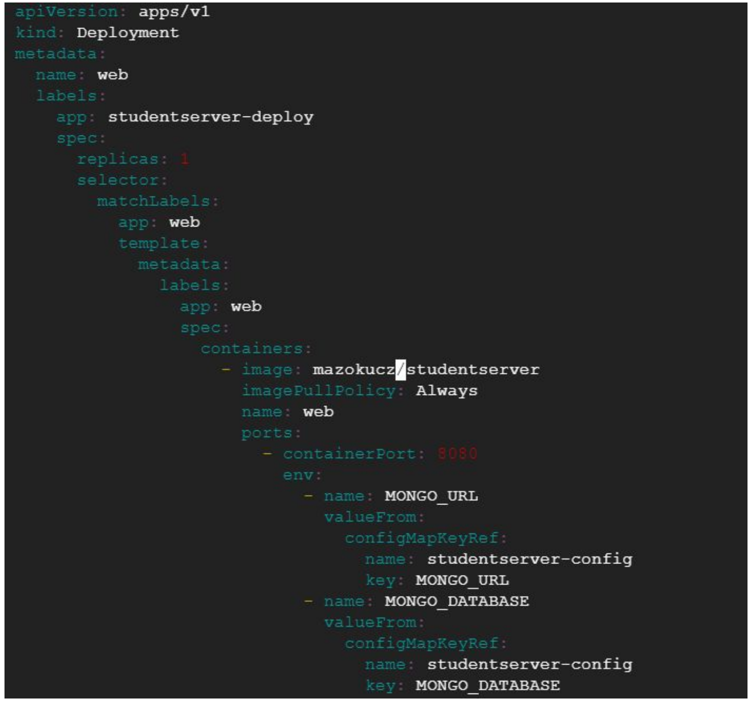


```
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  MONGO_URL: 35.185.225.50
  MONGO_DATABASE: mydb
~
```

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

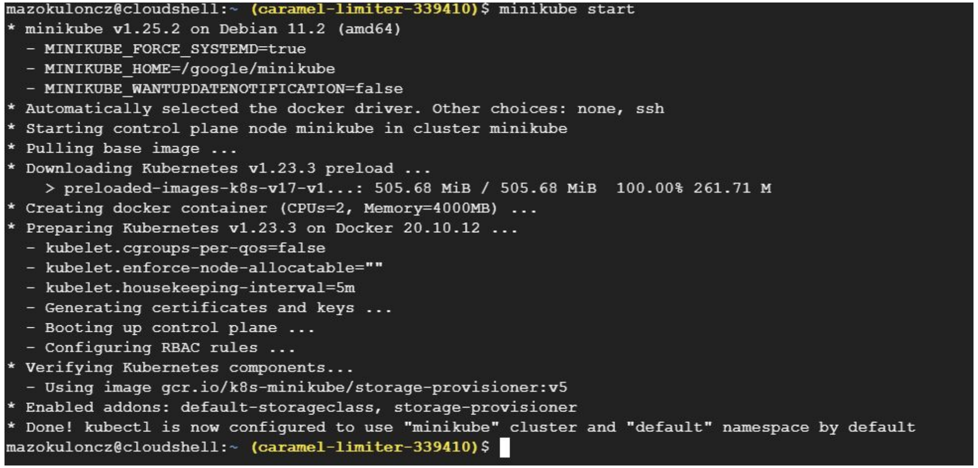


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: bookshelf-deployment
  labels:
    app: bookshelf-deployment
    spec:
      replicas: 1
      selector:
        matchLabels:
          app: bookshelf-deployment
          template:
            metadata:
              labels:
                app: bookshelf-deployment
                spec:
                  containers:
                    - image: mazokucz/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
~
```

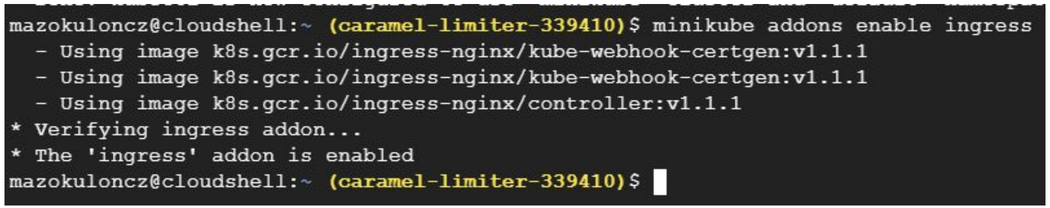


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
  labels:
    app: studentserver-deploy
    spec:
      replicas: 1
      selector:
        matchLabels:
          app: web
          template:
            metadata:
              labels:
                app: web
                spec:
  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

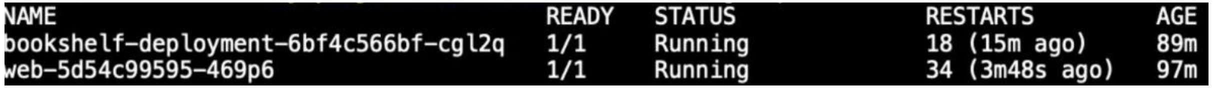
Start minikube and ingress



```
mazokuloncz@cloudshell:~ (caramel-limiter-339410)$ minikube start
* minikube v1.25.2 on Debian 11.2 (amd64)
  - MINIKUBE_FORCE_SYSTEMD=true
  - MINIKUBE_HOME=/google/minikube
  - MINIKUBE_WANTUPDATENOTIFICATION=false
* Automatically selected the docker driver. Other choices: none, ssh
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Downloading Kubernetes v1.23.3 preload ...
    > preloaded-images-k8s-v17-v1...: 505.68 MiB / 505.68 MiB  100.00% 261.71 M
* Creating docker container (CPUs=2, Memory=4000MB) ...
* Preparing Kubernetes v1.23.3 on Docker 20.10.12 ...
  - kubelet.cgroups-per-qos=false
  - kubelet.enforce-node-allocatable=""
  - kubelet.housekeeping-interval=5m
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
mazokuloncz@cloudshell:~ (caramel-limiter-339410)$
```



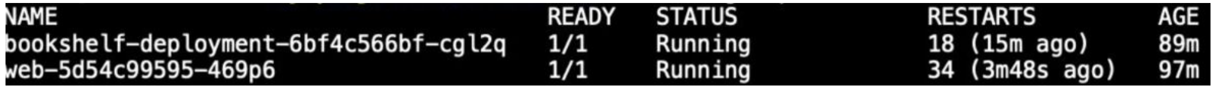
```
mazokuloncz@cloudshell:~ (caramel-limiter-339410)$ minikube addons enable ingress
  - Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
  - Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
  - Using image k8s.gcr.io/ingress-nginx/controller:v1.1.1
* Verifying ingress addon...
* The 'ingress' addon is enabled
mazokuloncz@cloudshell:~ (caramel-limiter-339410)$
```



```
NAME                                   READY   STATUS    RESTARTS        AGE
bookshelf-deployment-6bf4c566bf-cgl2q  1/1     Running   18 (15m ago)    89m
web-5d54c99595-469p6                   1/1     Running   34 (3m48s ago)  97m
```

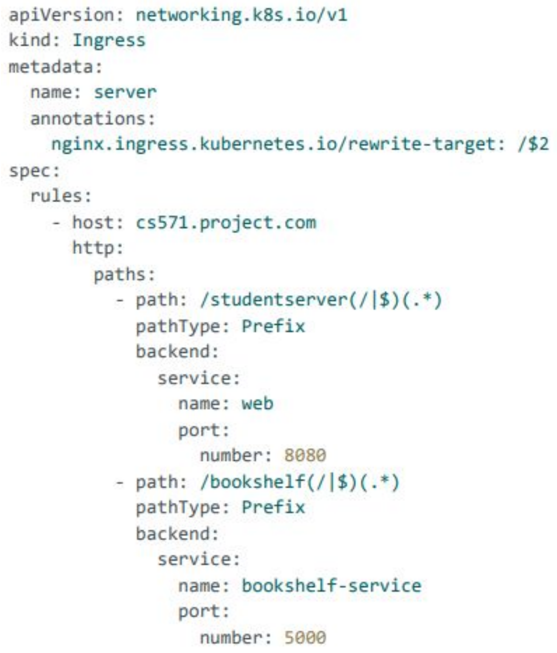
Create studentserver related pods and start service using the above yaml file

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



```
NAME                                  READY   STATUS    RESTARTS        AGE
bookshelf-deployment-6bf4c566bf-cgl2q 1/1     Running   18 (15m ago)    89m
web-5d54c99595-469p6                  1/1     Running   34 (3m48s ago)  97m
```

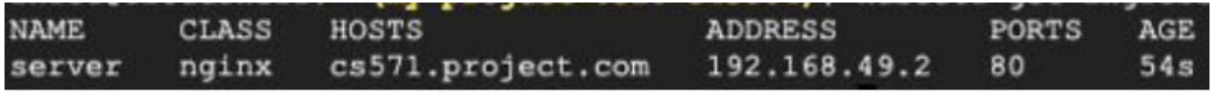
Create an ingress service yaml file called studentservermongoIngress.yaml



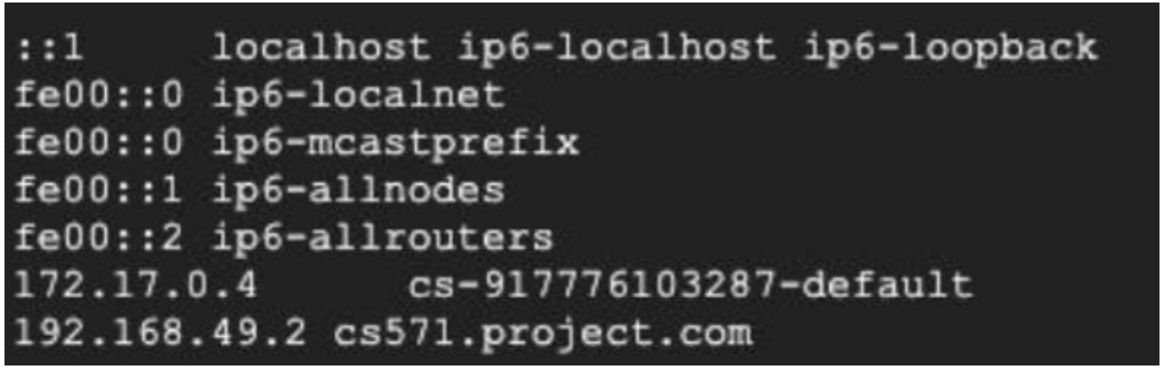
```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: server
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
  rules:
    - host: cs571.project.com
      http:
        paths:
          - path: /studentserver(/|$)(.*)
            pathType: Prefix
          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

Create the ingress service using the above yaml file

- kubectl apply -f ../studentservermongoIngress.yaml
- kubectl get ingress



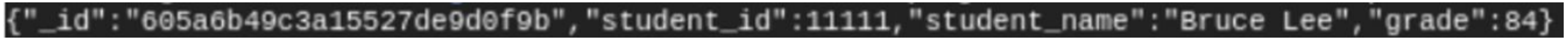
```
NAME     CLASS   HOSTS              ADDRESS        PORTS   AGE
server   nginx   cs571.project.com  192.168.49.2   80      54s
```



```
::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-917776103287-default
192.168.49.2 cs571.project.com
```

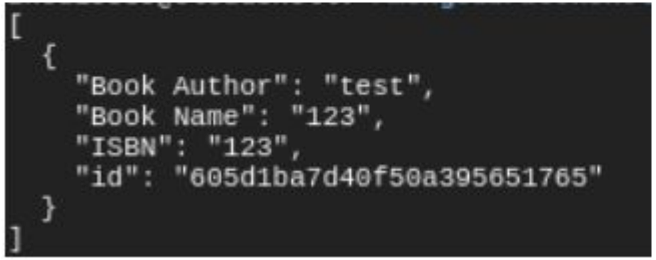
Demo

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



```
{"_id":"605a6b49c3a15527de9d0f9b","student_id":11111,"student_name":"Bruce Lee","grade":84}
```

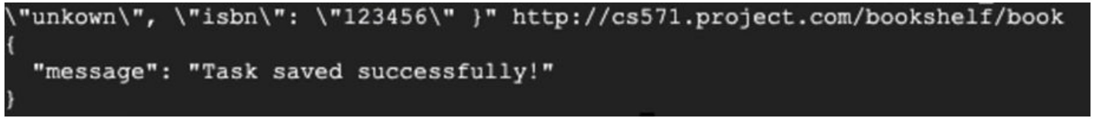
- curl cs571.project.com/bookshelf/books



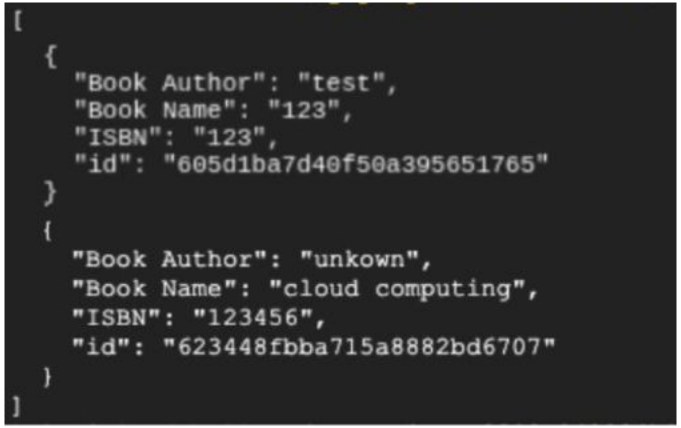
```
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123",
    "id": "605d1ba7d40f50a395651765"
  }
]
```

Demo

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



```
\"unkown\", \"isbn\": \"123456\" }" http://cs571.project.com/bookshelf/book
{
  "message": "Task saved successfully!"
}
```



```
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123",
    "id": "605d1ba7d40f50a395651765"
  }
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "623448fbba715a8882bd6707"
  }
]
```

References

- https://hc.labnet.sfbu.edu/~henry/npu/classes/kubernetes_in_action/configmap/slide/exercise_configmap.html