Create MongoDB using persistent volume on GKE and insert records

1. If starting from a new project, enable GKE

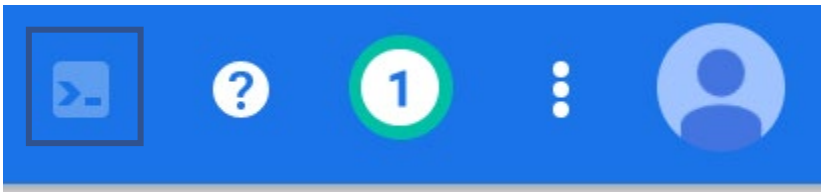
Kubernetes Engine API

Google

Builds and manages container-based applications, powered by the open source Kubernetes technology.

ENABLE TRY THIS API

2. Activate Cloud Shell and create a cluster



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

```
NAME   LOCATION   MASTER_VERSION   MASTER_IP        MACHINE_TYPE   NODE_VERSION      NUM_NODES   STATUS
kubia  us-west1  1.18.16-gke.502  104.196.236.204  e2-micro       1.18.16-gke.502   3           RUNNING
```

3. Create a persistent volume

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

```
NAME     ZONE        SIZE_GB  TYPE         STATUS
mongodb  us-west1-a  10       pd-standard  READY
```

4. Create a yaml file named mongodb-deployment.yaml

```
montaos19518@cloudshell:~ (cs571-new)$ vi mongodb-deployment.yaml
montaos19518@cloudshell:~ (cs571-new)$ 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:
            containers:
                # by default, the image is pulled from docker hub
                - image: mongo
                  name: mongo
                  ports:
                    - containerPort: 27017
                  volumeMounts:
                    - name: mongodb-data
                      mountPath: /data/db
            volumes:
                - name: mongodb-data
                  gcePersistentDisk:
                    pdName: mongodb
                    fsType: ext4
```

5. Create a MongoDB deployment pod: `kubectl apply -f mongodb-deployment.yaml`

```
montaos19518@cloudshell:~ (cs571-new)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
```

6. Check if the pod is running: `kubectl get pods`

```
montaos19518@cloudshell:~ (cs571-new)$ kubectl get pods
NAME                                 READY   STATUS    RESTARTS   AGE
mongodb-deployment-554cbb9965-lh569   1/1     Running   0          57s
```

7. Create a yaml file named mongodb-service.yaml to create a service to access from the outside

```
montaos19518@cloudshell:~ (cs571-new)$ vi mongodb-service.yaml
montaos19518@cloudshell:~ (cs571-new)$ 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
```

8. Create a service for MongoDB: `kubectl apply -f mongodb-service.yaml`

```
montaos19518@cloudshell:~ (cs571-new)$ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
```

9. Check if the service is running: `kubectl get svc`

```
montaos19518@cloudshell:~ (cs571-new)$ kubectl get svc
NAME              TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)           AGE
kubernetes        ClusterIP      10.3.240.1     <none>          443/TCP           22m
mongodb-service   LoadBalancer   10.3.245.57    35.230.71.134   27017:32306/TCP   48s
```

10. Check if MongoDB connection with external IP is working

`kubectl exec -it <mongo-db-deployment pod name> -- bash`

```
montaos19518@cloudshell:~ (cs571-new)$ kubectl exec -it mongodb-deployment-554cbb9965-lh569 -- bash
root@mongodb-deployment-554cbb9965-lh569:/#
```

`mongo <external-ip>`

```
root@mongodb-deployment-554cbb9965-lh569:/# mongo 35.230.71.134
MongoDB shell version v4.4.5
connecting to: mongodb://35.230.71.134:27017/test?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("90ddeb05-8111-4551-8cc0-3ff9611c5f19") }
MongoDB server version: 4.4.5
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:
        2021-04-12T17:07:12.144+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
        2021-04-12T17:07:12.907+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).

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

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

`exit`

```
> exit
bye
root@mongodb-deployment-554cbb9965-lh569:/# exit
exit
montaos19518@cloudshell:~ (cs571-new)$
```

11. Install mongoose: `npm install mongoose`

```
montaos19518@cloudshell:~ (cs571-new)$ npm install mongoose

added 36 packages, and audited 37 packages in 2s

2 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

12. Insert records in MongoDB: node

```
montaos19518@cloudshell:~ (cs571-306821)$ node
Welcome to Node.js v12.14.1.
Type ".help" for more information.
>
```

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://<external-ip>/mydb"
// Connect to 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);
  });
});
```

```
> var MongoClient = require('mongodb').MongoClient;
undefined
> var url = "mongodb://35.230.71.134/mydb"
undefined
> // Connect to 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
> null
3
```

13. Look for student_id=11111

```
> student_id=11111
11111
```

Modify studentServer to get records from MongoDB and deploy to GKE

1. Ctrl + C twice to exit node
2. Create 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=11111
// - The JSON response should contain only 'student_id', 'student_name'
//      and 'student_score' properties. For example:
//           {
//                 "student_id": 11111,
//                 "student_name": Bruce Lee,
//                 "student_score": 84
//           }
var MongoClient = mongodb.MongoClient;
var uri = `mongodb://${MONGO_URL}/${MONGO_DATABASE}`;
// Connect to 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 string /api/score
  if(/^\/api\/score/.test(req.url)) {
    // e.g., of student_id 11111
    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);
```

3. Create Dockerfile

```
montaos19518@cloudshell:~ (cs571-new)$ vi Dockerfile
montaos19518@cloudshell:~ (cs571-new)$ cat Dockerfile
FROM node:7
ADD studentServer.js /studentServer.js
ENTRYPOINT ["node", "studentServer.js"]
RUN npm install mongodb
```

4. Build the studentServer docker image
   `docker build -t <Docker Hub ID>/studentserver .`

```
Successfully built ef8c751ff90d
Successfully tagged mva456/studentserver:latest
```

5. Push the docker image: `docker push <Docker Hub ID>/studentserver`

```
montaos19518@cloudshell:~ (cs571-new)$ docker push mva456/studentserver
Using default tag: latest
The push refers to repository [docker.io/mva456/studentserver]
398489ce523e: Pushed
fb63d709165d: 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: sha256:69f5dbda774de816b0b716d899da2c8ea36cb58213cd51d417f32d5517c68301 size: 2424
```

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 requirements.txt

```
montaos19518@cloudshell:~ (cs571-306821)$ cat requirements.txt
Flask>=1.1.1,<1.2
flask-restplus>=0.13,<0.14
Flask-SSLify>=0.1.5,<0.2
Flask-Admin>=1.5.3,<1.6
gunicorn>=19,<20
Flask-PyMongo
```

3. Create a Dockerfile

```
montaos19518@cloudshell:~ (cs571-306821)$ cat Dockerfile
FROM python:alpine3.7
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
ENV PORT 5000
EXPOSE 5000
ENTRYPOINT ["python3.7"]
CMD ["bookshelf.py"]
```

4. Build the bookshelf app into a docker image
   `docker build -t <Docker Hub ID>/bookshelf .`

```
Successfully built f55034334358
Successfully tagged mva456/bookshelf:latest
```

5. Push the docker image to Docker Hub: `docker push <Docker Hub ID>/bookshelf`

```
montaos19518@cloudshell:~ (cs571-new)$ docker push mva456/bookshelf
Using default tag: latest
The push refers to repository [docker.io/mva456/bookshelf]
74a40ac8f6a8: Pushed
43175facf1b0: Pushed
66b6213aba6c: 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:d70b83c78659d13e55792c7f66f5081087750a32fa9f05ff24afd469f5d6c232 size: 2000
```

Create ConfigMap for both apps and store MongoDB URL and MongoDB name

1. Create a file named studentserver-configmap.yaml (Replace MONGO_URL into your external IP)

```
montaos19518@cloudshell:~ (cs571-new)$ vi studentserver-configmap.yaml
montaos19518@cloudshell:~ (cs571-new)$ cat studentserver-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
    name: studentserver-config
data:
    MONGO_URL: 35.230.71.134
    MONGO_DATABASE: mydb
```

2. Create a file named bookshelf-configmap.yaml (Replace MONGO_URL into your external IP)

```
montaos19518@cloudshell:~ (cs571-new)$ vi bookshelf-configmap.yaml
montaos19518@cloudshell:~ (cs571-new)$ cat bookshelf-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
    name: bookshelf-config
data:
    # SERVICE_NAME.NAMESPACE.svc.cluster.local:SERVICE_PORT
    MONGO_URL: 35.230.71.134
    MONGO_DATABASE: mydb
```

Expose two applications using ingress with Nginx to put both in the same domain but different paths

1. Create a file named studentserver-deployment.yaml (Replace container image into <Docker Hub ID>/studentserver)

```
montaos19518@cloudshell:~ (cs571-new)$ vi studentserver-deployment.yaml
montaos19518@cloudshell:~ (cs571-new)$ 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: mva456/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 a file named bookshelf-deployment.yaml

```
montaos19518@cloudshell:~ (cs571-new)$ vi bookshelf-deployment.yaml
montaos19518@cloudshell:~ (cs571-new)$ 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: mva456/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 a file named studentserver-service.yaml

```
montaos19518@cloudshell:~ (cs571-new)$ vi studentserver-service.yaml
montaos19518@cloudshell:~ (cs571-new)$ 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
```

4. Create a file named bookshelf-service.yaml

```
montaos19518@cloudshell:~ (cs571-new)$ vi bookshelf-service.yaml
montaos19518@cloudshell:~ (cs571-new)$ 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`

```
montaos19518@cloudshell:~ (cs571-new)$ 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% 168.90 M
* Creating docker container (CPUs=2, Memory=4000MB) ...
* Preparing Kubernetes v1.20.2 on Docker 20.10.3 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v4
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

6. Start ingress: `minikube addons enable ingress`

```
montaos19518@cloudshell:~ (cs571-new)$ minikube addons enable ingress
  - Using image us.gcr.io/k8s-artifacts-prod/ingress-nginx/controller:v0.40.2
  - Using image jettech/kube-webhook-certgen:v1.2.2
  - Using image jettech/kube-webhook-certgen:v1.3.0
* Verifying ingress addon...
* The 'ingress' addon is enabled
```

7. Create studentserver pods and service using previously created YAML files

`kubectl apply -f studentserver-deployment.yaml`

```
montaos19518@cloudshell:~ (cs571-new)$ kubectl apply -f studentserver-deployment.yaml
deployment.apps/web created
```

`kubectl apply -f studentserver-configmap.yaml`

```
montaos19518@cloudshell:~ (cs571-new)$ kubectl apply -f studentserver-configmap.yaml
configmap/studentserver-config created
```

`kubectl apply -f studentserver-service.yaml`

```
montaos19518@cloudshell:~ (cs571-new)$ kubectl apply -f studentserver-service.yaml
service/web created
```

8. Create bookshelf pods and service using previously create YAML files

`kubectl apply -f bookshelf-deployment.yaml`

```
montaos19518@cloudshell:~ (cs571-new)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created
```

`kubectl apply -f bookshelf-configmap.yaml`

```
montaos19518@cloudshell:~ (cs571-new)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
```

`kubectl apply -f bookshelf-service.yaml`

```
montaos19518@cloudshell:~ (cs571-new)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
```

9. Check if the pods are running: `kubectl get pods`

```
montaos19518@cloudshell:~ (cs571-306821)$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
bookshelf-deployment-7bcfdfc44b-xl7kl   1/1     Running   7          8m17s
web-7cc57c5b74-4n4kd                  1/1     Running   6          120m
```

10. Create a file named studentServerMongoIngress.yaml

```
montaos19518@cloudshell:~ (cs571-306821)$ vi studentServerMongoIngress.yaml
montaos19518@cloudshell:~ (cs571-306821)$ cat 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 an ingress service: `kubectl apply -f studentServerMongoIngress.yaml`

```
montaos19518@cloudshell:~ (cs571-306821)$ kubectl apply -f studentServerMongoIngress.yaml
ingress.networking.k8s.io/server created
```

12. Get the ingress' address: `kubectl get ingress`

```
montaos19518@cloudshell:~ (cs571-306821)$ kubectl get ingress
NAME     CLASS    HOSTS               ADDRESS        PORTS   AGE
server   <none>   cs571.project.com   192.168.49.2   80      45s
```

13. Enter the command `sudo vi /etc/hosts`
14. In the last line, add `<address> cs571.project.com`

```
montaos19518@cloudshell:~ (cs571-306821)$ 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-824671623995-default-boost-2mvvh
192.168.49.2 cs571.project.com
```

15. Test the results by running the following commands
    a. Display student's score
    curl cs571.project.com/studentserver/api/score?student_id=11111

```
montaos19518@cloudshell:~ (cs571-306821)$ curl cs571.project.com/studentserver/api/score?student_id=11111
{"_id":"6074b903888c4370acfc8890","student_id":11111,"student_name":"Bruce Lee","grade":84}
```

    b. Display books: curl cs571.project.com/bookshelf/books

```
montaos19518@cloudshell:~ (cs571-306821)$ curl cs571.project.com/bookshelf/books
[]
```

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

```
{
  "message": "Task saved successfully!"
}
```

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

    e. Delete book
    curl -X DELETE cs571.project.com/bookshelf/book/**<id>**

```
{
  "message": "Task deleted successfully!"
}
```