Deploying MongoDB and Python Flask REST APIs to GKE

1. Create a cluster on GKE

```
gcloud container clusters create kubia --num-nodes=1
--machine-type=e2-micro --region=us-central1-a
```

```
kubeconfig entry generated for kubia.
NAME: kubia
LOCATION: us-central1-a
MASTER_VERSION: 1.21.9-gke.1002
MASTER_IP: 34.133.209.55
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.21.9-gke.1002
NUM_NODES: 1
STATUS: RUNNING
```

2. Create a Persistent Volume

```
gcloud compute disks create --size=10GiB --zone=us-central1-a
mongodb
```

```
dabanoglu19588@cloudshell:~ (cs571-cci)$ gcloud compute disks create --size=10GiB --zone=us-central1-a  mongodb
WARNING: You have selected a disk size of under [200GB]. This may result in poor I/O performance. For more infor
Created [https://www.googleapis.com/compute/v1/projects/cs571-cci/zones/us-central1-a/disks/mongodb].
NAME: mongodb
ZONE: us-central1-a
SIZE_GB: 10
TYPE: pd-standard
STATUS: READY
```

3. Create a mongodb deployment with the yaml file below

```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
```

kubectl apply -f mongodb-deployment.yaml

```
dabanoglu19588@cloudshell:~/mongodb (cs571-cci)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
```

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

gcloud container cluster

```
dabanoglu19588@cloudshell:~/mongodb (cs571-cci)$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
mongodb-deployment-57dc68b4bd-59n7j   1/1     Running   0          2m34s
```

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

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

kubectl apply -f mongodb-service.yaml

```
dabanoglu19588@cloudshell:~/mongodb (cs571-cci)$ 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

```
dabanoglu19588@cloudshell:~/mongodb (cs571-cci)$ kubectl get svc
NAME              TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)           AGE
kubernetes        ClusterIP      10.56.0.1      <none>         443/TCP           37m
mongodb-service   LoadBalancer   10.56.12.144   34.67.12.38    27017:32322/TCP   69s
```

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

kubectl exec -it mongodb-deployment-<your-pod-name> -- bash

Now you are inside the mongodb deployment pod

```
dabanoglu19588@cloudshell:~/mongodb (cs571-cci)$ kubectl exec -it mongodb-deployment-57dc68b4bd-59n7j -- bash
root@mongodb-deployment-57dc68b4bd-59n7j:/#
```

Try

mongo <your-External-IP>

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

```
root@mongodb-deployment-57dc68b4bd-59n7j:/# mongo 34.67.12.38
MongoDB shell version v5.0.6
connecting to: mongodb://34.67.12.38:27017/test?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("bc85458d-ba24-44fd-adab-b24245b583a9") }
MongoDB server version: 5.0.6
================
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.
```

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

```
> exit
bye
root@mongodb-deployment-57dc68b4bd-59n7j:/# exit
exit
dabanoglu19588@cloudshell:~/mongodb (cs571-cci)$
```

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

Enter the following code line by line

```
var MongoClient = require('mongodb').MongoClient; var url =
"mongodb://EXTERNAL-IP/mydb"
// Connect to the db
MongoClient.connect(url,{ useNewUrlParser: true, useUnifiedTopology: true },
function(err, client){
    if (err)
        throw err;
          // create a document to be inserted
    var db = client.db("studentdb");
    const docs = [
            { student_id: 11111, student_name: "Bruce Lee", grade: 84},
            { student_id: 22222, student_name: "Jackie Chen", grade: 93 },
            { student_id: 33333, student_name: "Jet Li", grade: 88}
    ]
    db.collection("students").insertMany(docs, function(err, res){
        if(err) throw err;
        console.log(res.insertedCount);
        client.close();
    });
    db.collection("students").findOne({"student_id": 11111},
    function(err, result){
        console.log(result);
    });
});
```

npm init -y

npm install mongodb

```
dabanoglu19588@cloudshell:~ (cs571-cci)$ npm init -y
Wrote to /home/dabanoglu19588/package.json:

{
  "dependencies": {
    "express": "^4.17.3"
  },
  "name": "dabanoglu19588",
  "version": "1.0.0",
  "main": "index.js",
  "devDependencies": {
    "mocha": "^9.2.2"
  },
  "scripts": {
    "test": "mocha"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}


dabanoglu19588@cloudshell:~ (cs571-cci)$ npm install mongodb

added 2 packages, removed 582 packages, changed 96 packages, and audited 149 packages in 5s

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

found 0 vulnerabilities
dabanoglu19588@cloudshell:~ (cs571-cci)$
```

node

```
dabanoglu19588@cloudshell:~ (cs571-cci)$ node
Welcome to Node.js v17.6.0.
Type ".help" for more information.
> var MongoClient = require('mongodb').MongoClient;
undefined
> var url = "mongodb://34.67.12.38/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();
.......      });
.....      var query = {"student_id": 11111};
.....      db.collection("students").findOne({"student_id": 11111},function(err, result){
.......          if (err) throw err;
.......          console.log(result);
.......
.......      });
..... });
undefined
> 3
{
  _id: new ObjectId("62442e67d9c24ae1ade44007"),
  student_id: 11111,
  student_name: 'Bruce Lee',
  grade: 84
}
```

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

1. Create studentServer.js

```
var http = require('http');
var url = require('url');
var mongodb = require('mongodb');
const {
MONGO_URL,MONGO_DATABASE
} = process.env;
var MongoClient = mongodb.MongoClient;
var uri = `mongodb://${MONGO_URL}/${MONGO_DATABASE}`;
console.log(uri);
var server = http.createServer(function (req, res) {
 var result;
 var parsedUrl = url.parse(req.url, true);
 var student_id = parseInt(parsedUrl.query.student_id);
 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 inputvector/studentserver .
```

```
dabanoglu19588@cloudshell:~/mongodb (cs571-cci)$ docker build -t inputvector/studentserver .
Sending build context to Docker daemon   7.68kB
Step 1/4 : FROM node:7
7: Pulling from library/node
ad74af05f5a2: Pull complete
2b032b8bbe8b: Pull complete
a9a5b35f6ead: Pull complete
3245b5a1c52c: Pull complete
```

```
Successfully built 597a77b7973a
Successfully tagged inputvector/studentserver:latest
dabanoglu19588@cloudshell:~/mongodb (cs571-cci)$
```

4. Push the docker image

```
docker build -t studentserver .
docker tag studentserver your_username/repo_name
sudo docker login -u="<username>"
docker push your_username/repo_name
```

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

1. Create bookshelf.py

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


app = Flask(__name__)
app.config["MONGO_URI"] =
"mongodb://"+os.getenv("MONGO_URL")+"/"+os.getenv("MONGO_DATABASE")
app.config['JSONIFY_PRETTYPRINT_REGULAR'] = True
mongo = PyMongo(app)
db = mongo.db
```

```
@app.route("/")
def index():
    hostname = socket.gethostname()
    return jsonify(message="Welcome to bookshelf app! I am running inside {}
pod!".format(hostname)
)

@app.route("/books")
def get_all_tasks():
       books = db.bookshelf.find()
       data = []
       for book in books:
                   data.append({
                       "id": str(book["_id"]),
                       "Book Name": book["book_name"],
                       "Book Author": book["book_author"],
                       "ISBN" : book["ISBN"]
                   })
       return jsonify(
data )


@app.route("/book", methods=["POST"])
def add_book():
   book = request.get_json(force=True)
   db.bookshelf.insert_one({
       "book_name": book["book_name"],
       "book_author": book["book_author"],
       "ISBN": book["isbn"]
       })
   return jsonify(
       message="Task saved successfully!"
)

@app.route("/book/<id>", methods=["PUT"])
def update_book(id):
   data = request.get_json(force=True)
   print(data)
   response = db.bookshelf.update_many({"_id": ObjectId(id)}, {"$set":
   {"book_name": data['book_name'], "book_author": data["book_author"], "ISBN":
data["isbn"] }})
```

```
    if response.matched_count:
            message = "Task updated successfully!"
    else:
            message = "No book found!"
    return jsonify(
            message=message)

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

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


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

2. Create a Dockerfile

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

3. Build the bookshelf app into a docker image

see for details: https://docs.docker.com/language/python/build-images/

```
docker build -t inputvector/bookshelf .
```

```
Successfully built 6d50cf2d1933
Successfully tagged inputvector/bookshelf:latest
dabanoglu19588@cloudshell:~/mongodb/bookshelf (cs571-cci)$
```

4. Push the docker image to your dockerhub

```
docker push inputvector/bookshelf
```

```
dabanoglu19588@cloudshell:~/mongodb/bookshelf (cs571-cci)$ docker push inputvector/bookshelf
Using default tag: latest
The push refers to repository [docker.io/inputvector/bookshelf]
46ae4133ec83: Pushed
76d8ebdbae28: 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: sha256:74a7430dfab4154b612698888f45351feba84b1a2f4b34df240c1ca87df85e46 size: 1787
```

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

1. Create a file named studentserver-configmap.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
    MONGO_URL: 34.67.12.38
    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: 34.67.12.38
  MONGO_DATABASE: mydb
```

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

1.Create studentserver-deployment.yaml

```
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: inputvector/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

```
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: inputvector/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
```

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
```

5. Start minikube

```
minikube start
```

```
dabanoglu19588@cloudshell:~/mongodb/bookshelf (cs571-cci)$ 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
* 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
```

```
dabanoglu19588@cloudshell:~/mongodb/bookshelf (cs571-cci)$ 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
```

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
```

```
dabanoglu19588@cloudshell:~/mongodb/bookshelf (cs571-cci)$ kubectl apply -f studentserver-deployment.yaml
deployment.apps/web created
dabanoglu19588@cloudshell:~/mongodb/bookshelf (cs571-cci)$ kubectl apply -f studentserver-configmap.yaml
configmap/studentserver-config created
```

```
dabanoglu19588@cloudshell:~/mongodb/bookshelf (cs571-cci)$ kubectl apply -f sutdentserver-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
```

```
dabanoglu19588@cloudshell:~/mongodb/bookshelf (cs571-cci)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created
dabanoglu19588@cloudshell:~/mongodb/bookshelf (cs571-cci)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
dabanoglu19588@cloudshell:~/mongodb/bookshelf (cs571-cci)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
```

9. Check if all the pods are running correctly

```
kubectl get pods
```

```
dabanoglu19588@cloudshell:~/mongodb/bookshelf (cs571-cci)$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
bookshelf-deployment-58c778b8bb-v8xb6   1/1     Running   0          92m
web-5df7b4448d-d62sg                  1/1     Running   0          8s
```

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

```
dabanoglu19588@cloudshell:~/mongodb/bookshelf (cs571-cci)$ kubectl apply -f studentservermongoIngress.yaml
ingress.networking.k8s.io/server created
```

12. Check if ingress is running

kubectl get ingress

```
dabanoglu19588@cloudshell:~/mongodb/bookshelf (cs571-cci)$ kubectl get ingress
NAME     CLASS   HOSTS               ADDRESS        PORTS   AGE
server   nginx   cs571.project.com   192.168.49.2   80      2m1s
```

13. Add Address to /etc/hosts

sudo vim /etc/hosts

<your-address> cs571.project.com

```
# 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-162682137539-default
192.168.49.2    cs571.project.com
```

14. If everything goes smoothly, you should be able to access your applications

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

```
dabanoglu19588@cloudshell:~/mongodb/bookshelf (cs571-cci)$ curl cs571.project.com/studentserver/api/score?student_id=11111
{"_id":"62445be174ce74725fd5b5e1","student_id":11111,"student_name":"Bruce Lee","grade":84}
dabanoglu19588@cloudshell:~/mongodb/bookshelf (cs571-cci)$ curl cs571.project.com/studentserver/api/score?student_id=22222
{"_id":"62445be174ce74725fd5b5e2","student_id":22222,"student_name":"Jackie Chen","grade":93}
dabanoglu19588@cloudshell:~/mongodb/bookshelf (cs571-cci)$
```

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
```

```
dabanoglu19588@cloudshell:~/mongodb/bookshelf (cs571-cci)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "test_book",
    "ISBN": "11111116",
    "id": "624467f9d1b618865ec3ee96"
  }
]
```

Add a book

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

```
dabanoglu19588@cloudshell:~/mongodb/bookshelf (cs571-cci)$ curl -X POST -d "{\"book_name\": \"cloud computing\",\"book_author\":
\"unkown\", \"isbn\": \"123456\" }" http://cs571.project.com/bookshelf/book
{
  "message": "Task saved successfully!"
}
```

Update a book

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

```
dabanoglu19588@cloudshell:~/mongodb/bookshelf (cs571-cci)$ curl -X PUT -d "{\"book_name\": \"123\",\"book_author\": \"test\", \"isbn\":
\"123updated\" }" http://cs571.project.com/bookshelf/book/624467f9d1b618865ec3ee96
{
  "message": "Task updated successfully!"
}
```

```
dabanoglu19588@cloudshell:~/mongodb/bookshelf (cs571-cci)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123updated",
    "id": "624467f9d1b618865ec3ee96"
  },
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "6244687dd1b618865ec3ee97"
  }
]
```

Delete a book

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

```
dabanoglu19588@cloudshell:~/mongodb/bookshelf (cs571-cci)$ curl -X DELETE cs571.project.com/bookshelf/book/624467f9d1b618865ec3ee96
{
  "message": "Task deleted successfully!"
}
```

```
dabanoglu19588@cloudshell:~/mongodb/bookshelf (cs571-cci)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "6244687dd1b618865ec3ee97"
  }
]
```