CS571 Signature Project

Name: Farishta Mahzoz
ID: 19560

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

1. Create a cluster as usual on GKE

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

   If you have already created, check your cluster is running using below command

   $ kubectl cluster-info

```
mahzoz19560@cloudshell:~ (cs571-cloud-computing-infra)$ kubectl cluster-info
Kubernetes control plane is running at https://34.94.169.85
GLBCDefaultBackend is running at https://34.94.169.85/api/v1/namespaces/kube-system/services/default-http-backend:http/proxy
KubeDNS is running at https://34.94.169.85/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
Metrics-server is running at https://34.94.169.85/api/v1/namespaces/kube-system/services/https:metrics-server:/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
mahzoz19560@cloudshell:~ (cs571-cloud-computing-infra)$
```

2. Let's create a Persistent Volume first, if you have created a persistent volume for the week10's homework, you can skip this one

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

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

```
mahzoz19560@cloudshell:~ (cs571-cloud-computing-infra)$ gcloud compute disks create --size=10GiB --zone=us-west2-a mongodb
WARNING: You have selected a disk size of under [200GB]. This may result in poor I/O performance. For more information, see: https://developers.google.com/compute/docs/disks#performance.
Created [https://www.googleapis.com/compute/v1/projects/cs571-cloud-computing-infra/zones/us-west2-a/disks/mongodb].
NAME     ZONE        SIZE_GB  TYPE         STATUS
mongodb  us-west2-a  10       pd-standard  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

mahzoz19560@cloudshell:~ (cs571-cloud-computing-infra)$
```

3- Now create a mongodb deployment with this yaml files

kubectl apply -f mongodb-deployment.yaml

```
mahzoz19560@cloudshell:~ (cs571-cloud-computing-infra)$ 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
              mountPath: /data/db
      volumes:
        - name: mongodb
          gcePersistentDisk:
            pdName: mongodb
            fsType: ext4
mahzoz19560@cloudshell:~ (cs571-cloud-computing-infra)$
mahzoz19560@cloudshell:~ (cs571-cloud-computing-infra)$
mahzoz19560@cloudshell:~ (cs571-cloud-computing-infra)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
mahzoz19560@cloudshell:~ (cs571-cloud-computing-infra)$
```

--

4. Check if the deployment pod has been successfully created and started running kubectl get pods
5. Please wait until you see the STATUS is running, then you can move forward

```
  Type              Status
  Initialized       True
  Ready             False
  ContainersReady   False
  PodScheduled      True
Volumes:
  mongodb:
    Type:       GCEPersistentDisk (a Persistent Disk resource in Google Compute Engine)
    PDName:     mongodb
    FSType:     ext4
    Partition:  0
    ReadOnly:   false
  default-token-5kvv2:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-5kvv2
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason                  Age   From                     Message
  ----    ------                  ----  ----                     -------
  Normal  Scheduled               18s   default-scheduler        Successfully assigned default/mongodb-deployment-ddbb4f557-lfjnb to gke-kubia-default-pool-0af81da0-n72j
  Normal  SuccessfulAttachVolume  12s   attachdetach-controller  AttachVolume.Attach succeeded for volume "mongodb"
  Normal  Pulling                 8s    kubelet                  Pulling image "mongo"
mahzoz19560@cloudshell:~ (cs571-cloud-computing-infra)$ kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
mongodb-deployment-ddbb4f557-lfjnb  1/1     Running   0          36s
mahzoz19560@cloudshell:~ (cs571-cloud-computing-infra)$
```

6. 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:
      # service port in cluster
    - port: 27017
      # port to contact inside container
      targetPort: 27017
  selector:
    app: mongodb
```

kubectl apply -f mongodb-service.yaml

7. Wait couple of minutes, and check if the service is up kubectl get svc

Please wait until you see the external-ip is generated for mongodb-service, then you can move forward

```
mahzoz19560@cloudshell:~ (cs571-cloud-computing-infra)$ kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
mongodb-deployment-ddbb4f557-lfjnb   1/1     Running   0          36s
mahzoz19560@cloudshell:~ (cs571-cloud-computing-infra)$ kubectl get svc
NAME              TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)           AGE
kubernetes        ClusterIP      10.3.240.1     <none>          443/TCP           21m
mongodb-service   LoadBalancer   10.3.242.228   35.197.58.159   27017:30652/TCP   14m
mahzoz19560@cloudshell:~ (cs571-cloud-computing-infra)$
```

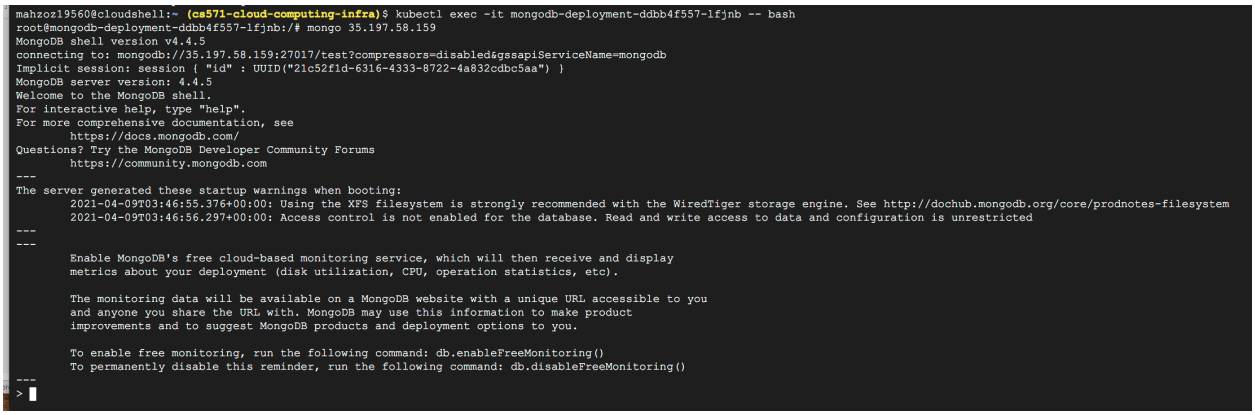
8. Now try and see if mongoDB is functioning for connections using the External-IP
   kubectl exec -it mongodb-deployment-replace-with-your-pod-name -- bash

   Now you are inside the mongodb deployment pod

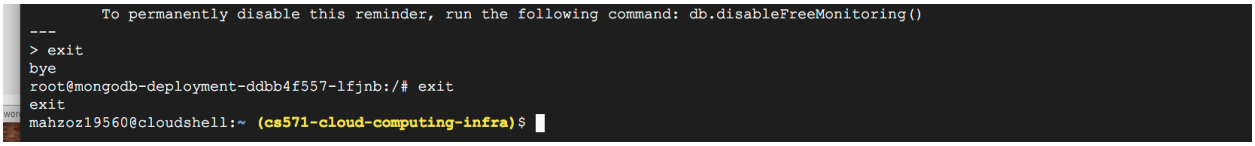
```
mahzoz19560@cloudshell:~ (cs571-cloud-computing-infra)$ kubectl exec -it mongodb-deployment-ddbb4f557-lfjnb -- bash
root@mongodb-deployment-ddbb4f557-lfjnb:/#
```

Try
mongo External-IP

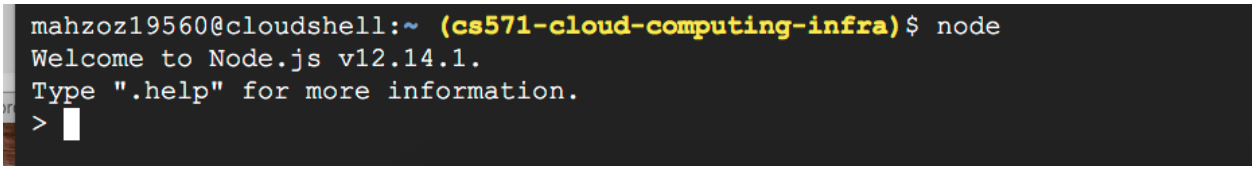
You should see something like this, which means your mongoDB is up and can be accessd using the External-IP



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



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

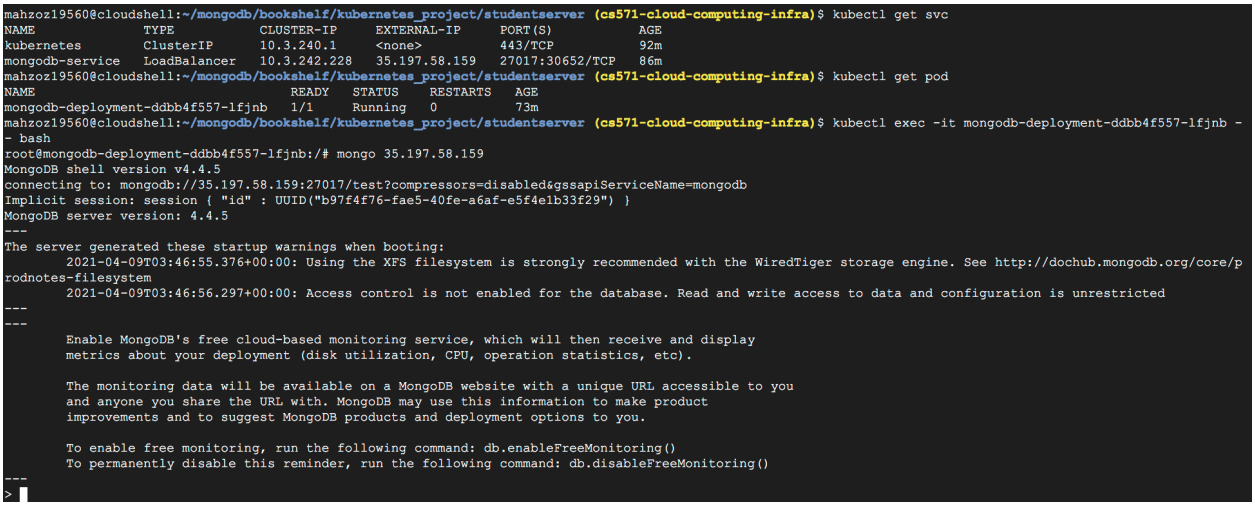


Enter the following line by line

```
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);
    });

});
```

If Everything is correct, you should see this,
3 means three records was inserted, and we tried search for student_id=11111



```
mongodb-deployment-ddbb4f557-lfjnb   1/1     Running   0          73m
mahzoz19560@cloudshell:~/mongodb/bookshelf/kubernetes_project/studentserver (cs571-cloud-computing-infra)$ kubectl exec -it mongodb-deployment-ddbb4f557-lfjnb -
- bash
root@mongodb-deployment-ddbb4f557-lfjnb:/# mongo 35.197.58.159
MongoDB shell version v4.4.5
connecting to: mongodb://35.197.58.159:27017/test?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("b97f4f76-fae5-40fe-a6af-e5f4e1b33f29") }
MongoDB server version: 4.4.5
---
The server generated these startup warnings when booting:
        2021-04-09T03:46:55.376+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/p
rodnotes-filesystem
        2021-04-09T03:46:56.297+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()
---
> use mydb
switched to db mydb
> show collections
> db.students.find()
> use studentdb
switched to db studentdb
> show collections
students
> db.students.find()
{ "_id" : ObjectId("606fdecac818c70a851a2f78"), "student_id" : 11111, "student_name" : "Bruce Lee", "grade" : 84 }
{ "_id" : ObjectId("606fdecac818c70a851a2f79"), "student_id" : 22222, "student_name" : "Jackie Chen", "grade" : 93 }
{ "_id" : ObjectId("606fdecac818c70a851a2f7a"), "student_id" : 33333, "student_name" : "Jet Li", "grade" : 88 }
>
```

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

1. Create a studentServer

```
    var http = require('http');
    var url = require('url');
    var mongodb = require('mongodb');
    const {

MONGO_URL,

      MONGO_DATABASE
    }  = process.env;
    // - Expect the request to contain a query
       //    string with a key 'student_id' and a student ID as



       //    the value. For example
       //     /api/score?student_id=1111
       // - The JSON response should contain only 'student_id',
   'student_name'



    //   and 'student_score' properties. For example:
    //
```

```
    //     {
       //         "student_id": 1111,



       //         "student_name": Bruce Lee,



       //         "student_score": 84
       //     }
       //
       var MongoClient = mongodb.MongoClient;
       var uri = `mongodb://${MONGO_URL}/${MONGO_DATABASE}`;
       // Connect to the db
       console.log(uri);
       var server = http.createServer(function (req, res) {
         var result;
         // req.url = /api/score?student_id=11111
         var parsedUrl = url.parse(req.url, true);
         var student_id = parseInt(parsedUrl.query.student_id);
         // match req.url with the string /api/score
         if (/^\/api\/score/.test(req.url)) {
           // e.g., of student_id 1111
           MongoClient.connect(uri,{ useNewUrlParser: true,
   useUnifiedTopology:
       true }, function(err, client){

   if (err)

throw err;
        var db = client.db("studentdb");
        db.collection("students").findOne({"student_id":student_id},
(err, student) => {
           if(err)
     throw new Error(err.message, null);
if (student) {
     res.writeHead(200, { 'Content-Type': 'application/json'
     res.end(JSON.stringify(student)+ '\n')
}else {
     res.writeHead(404);
     res.end("Student Not Found \n");
}

})

   } else {
   res.writeHead(404);
   res.end("Wrong url, please try again\n");
```

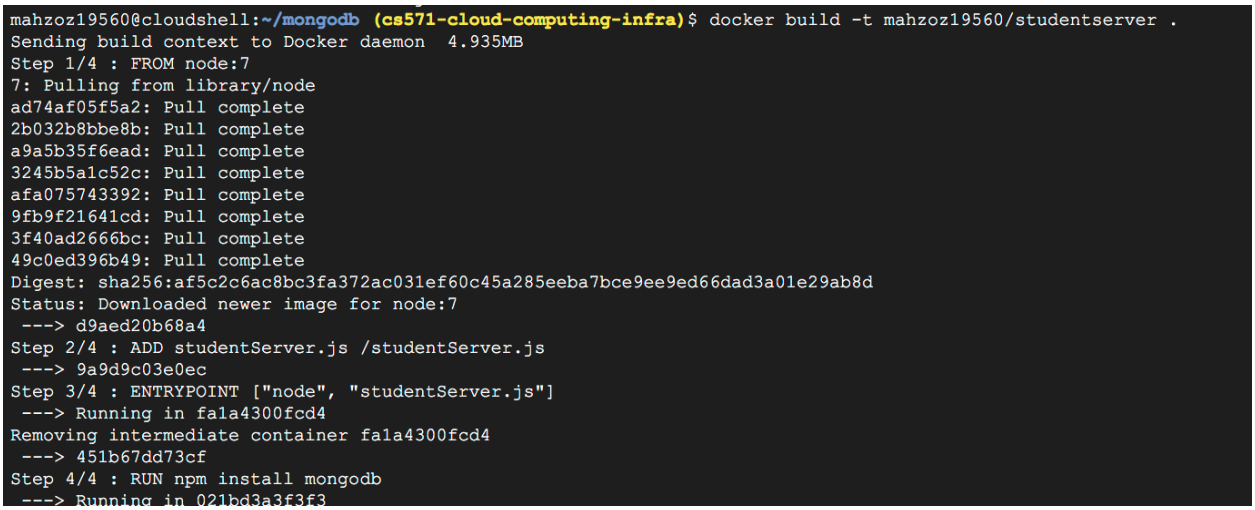
```
  }
});
server.listen(8080);
```

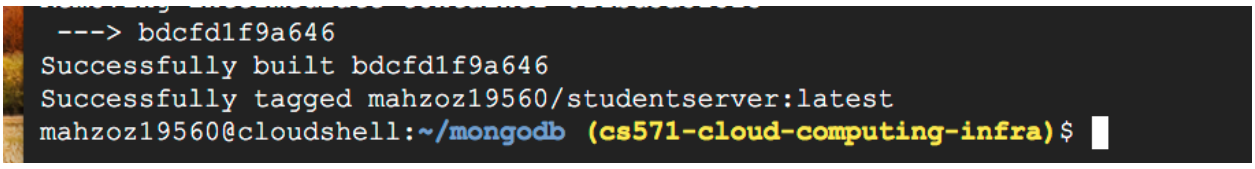
2. Create Dockerfile

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

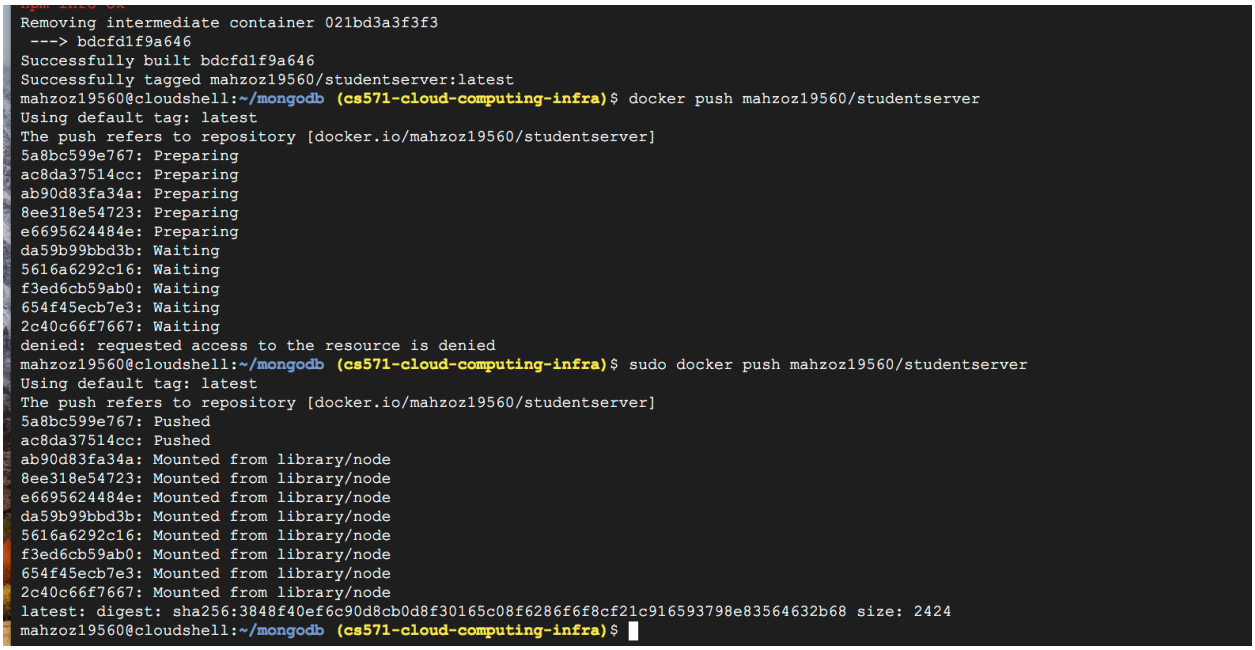
3. Build the studentserver docker image
docker build -t yourdockerhubID/studentserver .

Make sure there is no error





4. Push the docker image
docker push yourdockerhubID/studentserver



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 docker build -t zhou19539/bookshelf .

   Make sure this step build successfully

```
mahzoz19560@cloudshell:~/mongodb/bookshelf/kubernetes_project/bookshelf (cs571-cloud-computing-infra)$ ls
bookshelf-configmap.yaml  bookshelf-deployment.yaml  bookshelf.py  bookshelf-service.yaml  Dockerfile  requirements.txt
mahzoz19560@cloudshell:~/mongodb/bookshelf/kubernetes_project/bookshelf (cs571-cloud-computing-infra)$ docker build -t mahzoz19560/bookshelf .
Sending build context to Docker daemon  22.53kB
Step 1/8 : FROM python:alpine3.7
alpine3.7: Pulling from library/python
48ecbb6b270e: Pull complete
692f29ee68fa: Pull complete
6439819450d1: Pull complete
3c7be240f7bf: Pull complete
ca4b349df8ed: Pull complete
Digest: sha256:35f6f83ab08f98c727dbefd53738e3b3174a48b4571ccb1910bae480dcdba847
Status: Downloaded newer image for python:alpine3.7
 ---> 00be2573e9f7
Step 2/8 : COPY . /app
 ---> ce276478ef8d
Step 3/8 : WORKDIR /app
 ---> Running in 0dde538c24f9
Removing intermediate container 0dde538c24f9
 ---> abf4a40d9295
Step 4/8 : RUN pip install -r requirements.txt
 ---> Running in 86fec8a62db0
Collecting Flask (from -r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/f2/28/2a03252dfb9ebf377f40fba6a7841b47083260bf8bd8e737b0c6952df83f/Flask-1.1.2-py2.py3-none-any.whl (94kB)
Collecting Flask-PyMongo (from -r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/67/b8/0322016b9ce09a64fba9018211e7c35fd51380527ffd9ea248744f389239/Flask_PyMongo-2.3.0-py2.py3-none-any.whl
Collecting Jinja2>=2.10.1 (from Flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/7e/c2/1eece8c95ddbc9b1aeb64f5783a9e07a286de42191b7204d67b7496ddf35/Jinja2-2.11.3-py2.py3-none-any.whl (125kB
)
Collecting itsdangerous>=0.24 (from Flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/76/ae/44b03b253d6fade317f32c24d100b3b35c2239807046a4c953c7b89fa49e/itsdangerous-1.1.0-py2.py3-none-any.whl
Collecting Werkzeug>=0.15 (from Flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/cc/94/5f7079a0e00bd6863ef8f1da638721e9da21e5bacee597595b318f71d62e/Werkzeug-1.0.1-py2.py3-none-any.whl (298k
```

```
Removing intermediate container 86fec8a62db0
 ---> f97ed5777456
Step 5/8 : ENV PORT 5000
 ---> Running in a5140140507a
Removing intermediate container a5140140507a
 ---> a839f6d57417
Step 6/8 : EXPOSE 5000
 ---> Running in 047ee9c3182b
Removing intermediate container 047ee9c3182b
 ---> feaa2a3bc157
Step 7/8 : ENTRYPOINT [ "python3" ]
 ---> Running in 0740e1396a85
Removing intermediate container 0740e1396a85
 ---> 4503cf2c312e
Step 8/8 : CMD [ "bookshelf.py" ]
 ---> Running in 78d110263465
Removing intermediate container 78d110263465
 ---> ce336e51405a
Successfully built ce336e51405a
Successfully tagged mahzoz19560/bookshelf:latest
mahzoz19560@cloudshell:~/mongodb/bookshelf/kubernetes_project/bookshelf (cs571-cloud-computing-infra)$
```

3. Push the docker image to your dockerhub

docker push yourdockerhubID/bookshelf

```
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ sudo docker push mahzoz19560/bookshelf
Using default tag: latest
The push refers to repository [docker.io/mahzoz19560/bookshelf]
1190d22239da: Pushed
d799b108173a: 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:ce922ef479c97bd4dedcc777411d1c165219f65b7c2db0e93d1cdc82721fe84f size: 1787
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$
```

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

Create a file named studentserver-configmap.yaml

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  MONGO_URL: Change-this-to-your-mongoDB-EXTERNAL-IP
  MONGO_DATABASE: mydb
```

Create a file named bookshelf-configmap.yaml

```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
```

Notice: the reason of creating those two ConfigMap is to avoid re-building docker image again if the mongoDB pod restarts with a different External-IP

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

```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: mahzoz19560/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: mahzoz19560/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

```
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ start minikube
-bash: start: command not found
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ 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% 171.61 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: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$
```

6. Start Ingress
   minikube addons enable ingress

```
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ 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
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$
```

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

```
* The 'ingress' addon is enabled
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ kubectl apply -f ../studentserver-deploym
ent.yaml
deployment.apps/web created
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ kubectl apply -f ../studentserver-configm
ap.yaml
configmap/studentserver-config created
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ kubectl apply -f ../studentserver-service
.yaml
service/web created
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$
```

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

```
* 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: ssh, none
* 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% 167.62 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
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ 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
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ kubectl apply -f ../studentserver-deployment.yaml
deployment.apps/web created
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ kubectl apply -f ../studentserver-configmap.yaml
configmap/studentserver-config created
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ kubectl apply -f ../studentserver-service.yaml
service/web created
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ kubectl get po
NAME                                  READY   STATUS    RESTARTS   AGE
bookshelf-deployment-578757999f-q7m4j   1/1     Running   0          43s
web-59b45585db-fl4d2                  1/1     Running   0          98s
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$
```

kubectl apply -f bookshelf-deployment.yaml

kubectl apply -f bookshelf-configmap.yaml

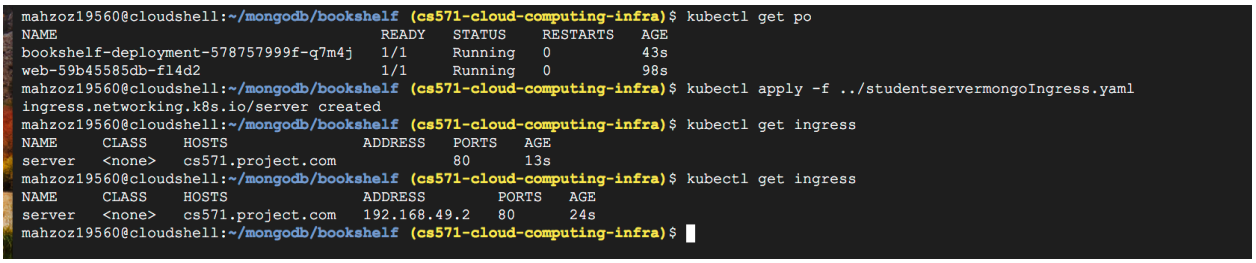
kubectl apply -f bookshelf-service.yaml

```
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ kubectl apply -f bookshelf-deployment.yam
l
deployment.apps/bookshelf-deployment created
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ kubectl apply -f bookshelf-configmap.yaml

configmap/bookshelf-config created
```

```
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$
```

9. Check if all the pods are running correctly

   kubectl get pods



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

```
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$ nano studentservermongoIngress.yaml
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$ kubectl apply -f studentservermongoIngress.yaml
ingress.networking.k8s.io/server created
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$ kubectl get ingress
NAME     CLASS    HOSTS              ADDRESS        PORTS   AGE
server   <none>   cs571.project.com   192.168.49.2   80      55s
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$
```

12. Check if ingress is running kubectl get ingress

Please wait until you see the Address, then move forward

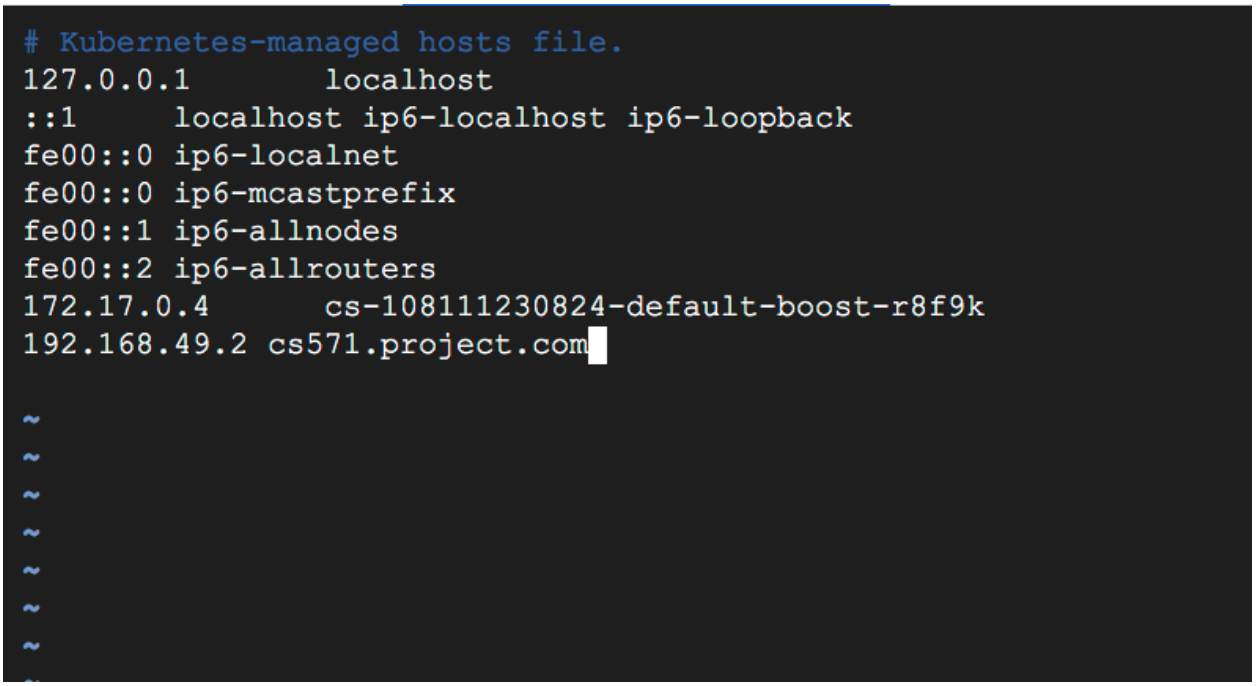

```
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$ nano studentservermongoIngress.yaml
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$ kubectl apply -f studentservermongoIngress.yaml
ingress.networking.k8s.io/server created
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$ kubectl get ingress
NAME     CLASS    HOSTS              ADDRESS        PORTS   AGE
server   <none>   cs571.project.com   192.168.49.2   80      55s
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$
```


13. Add Addreee to /etc/hosts vi /etc/hosts

    Add the address you got from above step to the end of the file

    ```
    Your-address cs571.project.com
    ```

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



14. Build the image studentserver

    docker build -t mahzoz19560/studentserver .

```
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$ docker build -t mahzoz19560/studentserver .
Sending build context to Docker daemon  3.389MB
Step 1/4 : FROM node:7
7: Pulling from library/node
ad74af05f5a2: Pull complete
2b032b8bbe8b: Pull complete
a9a5b35f6ead: Pull complete
3245b5a1c52c: Pull complete
afa075743392: Pull complete
9fb9f21641cd: Pull complete
3f40ad2666bc: Pull complete
49c0ed396b49: Pull complete
Digest: sha256:af5c2c6ac8bc3fa372ac031ef60c45a285eeba7bce9ee9ed66dad3a01e29ab8d
Status: Downloaded newer image for node:7
 ---> d9aed20b68a4
Step 2/4 : ADD studentServer.js /studentServer.js
 ---> e830b85bb650
Step 3/4 : ENTRYPOINT ["node", "studentServer.js"]
 ---> Running in 4ec08d59ce6e
Removing intermediate container 4ec08d59ce6e
 ---> fb5b99e81be7
Step 4/4 : RUN npm install mongodb
 ---> Running in 95bec28c644a
npm info it worked if it ends with ok
npm info using npm@4.2.0
npm info using node@v7.10.1
npm info attempt registry request try #1 at 7:15:04 PM
npm http request GET https://registry.npmjs.org/mongodb
npm http 200 https://registry.npmjs.org/mongodb
npm info addNameTag [ 'mongodb', 'latest' ]
npm info retry fetch attempt 1 at 7:15:04 PM
npm info attempt registry request try #1 at 7:15:04 PM
npm http fetch GET https://registry.npmjs.org/mongodb/-/mongodb-3.6.6.tgz
npm http fetch 200 https://registry.npmjs.org/mongodb/-/mongodb-3.6.6.tgz
npm info attempt registry request try #1 at 7:15:04 PM
npm http request GET https://registry.npmjs.org/bl
npm info attempt registry request try #1 at 7:15:04 PM
npm http request GET https://registry.npmjs.org/bson
npm info attempt registry request try #1 at 7:15:04 PM
npm http request GET https://registry.npmjs.org/denque
npm info attempt registry request try #1 at 7:15:04 PM
npm http request GET https://registry.npmjs.org/optional-require
npm info attempt registry request try #1 at 7:15:04 PM
npm http request GET https://registry.npmjs.org/safe-buffer
npm info attempt registry request try #1 at 7:15:04 PM
npm http request GET https://registry.npmjs.org/saslprep
npm http 200 https://registry.npmjs.org/bl
```

```
/
`-- mongodb@3.6.6
  +-- bl@2.2.1
  | `-- readable-stream@2.3.7
  |   +-- core-util-is@1.0.2
  |   +-- inherits@2.0.4
  |   +-- isarray@1.0.0
  |   +-- process-nextick-args@2.0.1
  |   +-- safe-buffer@5.1.2
  |   +-- string_decoder@1.1.1
  |   | `-- safe-buffer@5.1.2
  |   `-- util-deprecate@1.0.2
  +-- bson@1.1.6
  +-- denque@1.5.0
  +-- optional-require@1.0.3
  +-- safe-buffer@5.2.1
  `-- saslprep@1.0.3
    `-- sparse-bitfield@3.0.3
      `-- memory-pager@1.5.0

npm WARN enoent ENOENT: no such file or directory, open '/package.json'
npm WARN !invalid#1 No description
npm WARN !invalid#1 No repository field.
npm WARN !invalid#1 No README data
npm WARN !invalid#1 No license field.
npm info ok
Removing intermediate container 95bec28c644a
 ---> b05b5678bb15
Successfully built b05b5678bb15
Successfully tagged mahzoz19560/studentserver:latest
```

15. Make sure to push the image to the docker

```
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$ docker push mahzoz19560/studentserver
Using default tag: latest
The push refers to repository [docker.io/mahzoz19560/studentserver]
85d44b4dfcf5: Preparing
c57ed629fd84: Preparing
ab90d83fa34a: Preparing
8ee318e54723: Preparing
e6695624484e: Preparing
da59b99bbd3b: Waiting
5616a6292c16: Waiting
f3ed6cb59ab0: Waiting
654f45ecb7e3: Waiting
2c40c66f7667: Waiting
denied: requested access to the resource is denied
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: mahzoz19560
Password:
WARNING! Your password will be stored unencrypted in /home/mahzoz19560/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$ docker push mahzoz19560/studentserver
Using default tag: latest
The push refers to repository [docker.io/mahzoz19560/studentserver]
85d44b4dfcf5: Pushed
c57ed629fd84: Pushed
ab90d83fa34a: Layer already exists
8ee318e54723: Layer already exists
e6695624484e: Layer already exists
da59b99bbd3b: Layer already exists
5616a6292c16: Layer already exists
f3ed6cb59ab0: Layer already exists
654f45ecb7e3: Layer already exists
2c40c66f7667: Layer already exists
latest: digest: sha256:1c5d0a1e598d736f35c368b65bfe8d25fa3b25ec57d14db281f90e917232719a size: 2424
```

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

```
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$ kubectl apply -f student
studentserver-configmap.yaml     studentserver-deployment.yaml   studentServer.js              studentservermongoIngress.yaml  studentserver-service.yaml
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$ kubectl apply -f studentserver-deployment.yaml
deployment.apps/web created
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$ kubectl apply -f studentserver-service.yaml
service/web created
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$ kubectl get pods
NAME                   READY   STATUS    RESTARTS   AGE
web-776d5d9db5-mm7d2   1/1     Running   0          29s
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$ curl 192.168.49.2:31642/api/score?student_id=11111
curl: (7) Failed to connect to 192.168.49.2 port 31642: Connection refused
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$ kubectl get svc
NAME         TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes   ClusterIP      10.96.0.1       <none>        443/TCP          12m
web          LoadBalancer   10.110.62.168   <pending>     8080:31526/TCP   75s
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$ curl 192.168.49.2:31642/api/score?student_id=11111
curl: (7) Failed to connect to 192.168.49.2 port 31642: Connection refused
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$ minikube service web --url
http://192.168.49.2:31526
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$ curl 192.168.49.2:31526/api/score?student_id=11111
{"_id":"60748ca328ad6f355d660cb1","student_id":11111,"student_name":"Bruce Lee","grade":84}
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$ kubectl get ingress
NAME     CLASS    HOSTS              ADDRESS        PORTS   AGE
server   <none>   cs571.project.com   192.168.49.2   80      89m
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$ 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-108111230824-default-boost-7gcrx
192.168.49.2 cs571.project.com
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$ curl cs571.project.com/studentserver/api/score?student_id=11111
{"_id":"60748ca328ad6f355d660cb1","student_id":11111,"student_name":"Bruce Lee","grade":84}
mahzoz19560@cloudshell:~/mongodb/studentserver (cs571-cloud-computing-infra)$ cd ../bookshelf/
```

17. Same build the image for bookshelf and push it to the docker

```
  MONGO_DATABASE: mydb
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ docker build -t mahzoz19560/bookshelf .
Sending build context to Docker daemon  4.939MB
Step 1/8 : FROM python:alpine3.7
alpine3.7: Pulling from library/python
48ecbb6b270e: Pull complete
692f29ee68fa: Pull complete
6439819450d1: Pull complete
3c7be240f7bf: Pull complete
ca4b349df8ed: Pull complete
Digest: sha256:35f6f83ab08f98c727dbefd53738e3b3174a48b4571ccb1910bae480dcdba847
Status: Downloaded newer image for python:alpine3.7
 ---> 00be2573e9f7
Step 2/8 : COPY . /app
 ---> 9ee42b1b534c
Step 3/8 : WORKDIR /app
 ---> Running in 1219319ee323
Removing intermediate container 1219319ee323
 ---> b1148fc8be64
Step 4/8 : RUN pip install -r requirements.txt
 ---> Running in fd8962d6fb02
Collecting Flask (from -r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/f2/28/2a03252dfb9ebf377f40fba6a7841b47083260bf8bd8e737b0c6952df83f/Flask-1.1.2-py2.py3-none-any.whl (94kB)
Collecting Flask-PyMongo (from -r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/67/b8/0322016b9ce09a64fba9018211e7c35fd51380527ffd9ea248744f389239/Flask_PyMongo-2.3.0-py2.py3-none-any.whl
Collecting Jinja2>=2.10.1 (from Flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/7e/c2/1eece8c95ddbc9b1aeb64f5783a9e07a286de42191b7204d67b7496ddf35/Jinja2-2.11.3-py2.py3-none-any.whl (125kB)
Collecting Werkzeug>=0.15 (from Flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/cc/94/5f7079a0e00bd6863ef8f1da638721e9da21e5bacee597595b318f71d62e/Werkzeug-1.0.1-py2.py3-none-any.whl (298kB)
Collecting itsdangerous>=0.24 (from Flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/76/ae/44b03b253d6fade317f32c24d100b3b35c2239807046a4c953c7b89fa49e/itsdangerous-1.1.0-py2.py3-none-any.whl
Collecting click>=5.1 (from Flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/d2/3d/fa76db83bf75c4f8d338c2fd15c8d33fdd7ad23a9b5e57eb6c5de26b430e/click-7.1.2-py2.py3-none-any.whl (82kB)
Collecting PyMongo>=3.3 (from Flask-PyMongo->-r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/72/82/e7196f2f69318dd206db26db68fcfa0ff821d88fbca6d0f0c7b678ba0353/pymongo-3.11.3.tar.gz (777kB)
Collecting MarkupSafe>=0.23 (from Jinja2>=2.10.1->Flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/b9/2e/64db92e53b86efccfaea71321f597fa2e1b2bd3853d8ce658568f7a13094/MarkupSafe-1.1.1.tar.gz
```

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

```
 ---> Running in 498abf21c82f
Removing intermediate container 498abf21c82f
 ---> 082258e1c460
Successfully built 082258e1c460
Successfully tagged mahzoz19560/bookshelf:latest
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ docker push mahzoz19560/bookshelf
Using default tag: latest
The push refers to repository [docker.io/mahzoz19560/bookshelf]
cb620600a241: Pushed
d164f564bd52: Pushed
5fa31f02caa8: Layer already exists
88e61e328a3c: Layer already exists
9b77965e1d3f: Layer already exists
50f8b07e9421: Layer already exists
629164d914fc: Layer already exists
latest: digest: sha256:d4ee314f4f38137f81c3b6fe310c09db4179c59ae778c1b4db5390e768002ff4 size: 1790
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ kubectl apply -f bookshelf-c
error: the path "bookshelf-c" does not exist
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ kubcetl apply -f bookshelf-service.yaml
-bash: kubcetl: command not found
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
bookshelf-deployment-578757999f-lvkc6   1/1     Running   0          65s
web-776d5d9db5-mm7d2                  1/1     Running   0          9m59s
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ curl cs571.project.com/bookshelf/books
[]
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ curl -X POST -d "{\"book_name\": \"cloud computing\",\"book_author\":
> \"unkown\", \"isbn\": \"123456\" }" http://cs571.project.com/bookshelf/book
{
  "message": "Task saved successfully!"
}
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "6074a1810947d9f953e16689"
  }
]
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$
```

```
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ url -X PUT -d "{\"book_name\": \"123\",\"book_author\": \"test\", \"isbn\":
> \"123updated\" }" http://cs571.project.com/bookshelf/book/id
-bash: url: command not found
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ curl -X PUT -d "{\"book_name\": \"123\",\"book_author\": \"test\", \"isbn\":
> \"123updated\" }" http://cs571.project.com/bookshelf/book/6074a1810947d9f953e16689
{
  "message": "Task updated successfully!"
}
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123updated",
    "id": "6074a1810947d9f953e16689"
  }
]
mahzoz19560@cloudshell:~/mongodb/bookshelf (cs571-cloud-computing-infra)$
```