Week 9: Homework 1: Chapter 7: Configmap: Signature Project

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

Name: Priyanka, Stu-id: 20179

Step 1: Create MongoDB Using Persistent Volume on GKE and Insert Records

1. **Create a GKE Cluster:**
   gcloud container clusters create kubia --num-nodes=1 --machine-type=e2 micro --region=us-east1-b
   Wait for the cluster creation to finish.

```
priyanka599@cloudshell:~ (clouddemo-441418)$ gcloud container clusters create kubia --num-nodes=1 --machine-type=e2-micro --zone=us-east1-b
Note: The Kubelet readonly port (10255) is now deprecated. Please update your workloads to use the recommended alternatives. See https://cloud.google.com
kubelet-readonly-port for ways to check usage and for migration instructions.
Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most 1008 node(s).
Creating cluster kubia in us-east1-b... Cluster is being health-checked (Kubernetes Control Plane is healthy)...done.
Created [https://container.googleapis.com/v1/projects/clouddemo-441418/zones/us-east1-b/clusters/kubia].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/us-east1-b/kubia?project=clouddemo-441418
kubeconfig entry generated for kubia.
NAME: kubia
LOCATION: us-east1-b
MASTER_VERSION: 1.30.5-gke.1443001
MASTER_IP: 35.227.11.56
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.30.5-gke.1443001
NUM_NODES: 1
STATUS: RUNNING
priyanka599@cloudshell:~ (clouddemo-441418)$
```

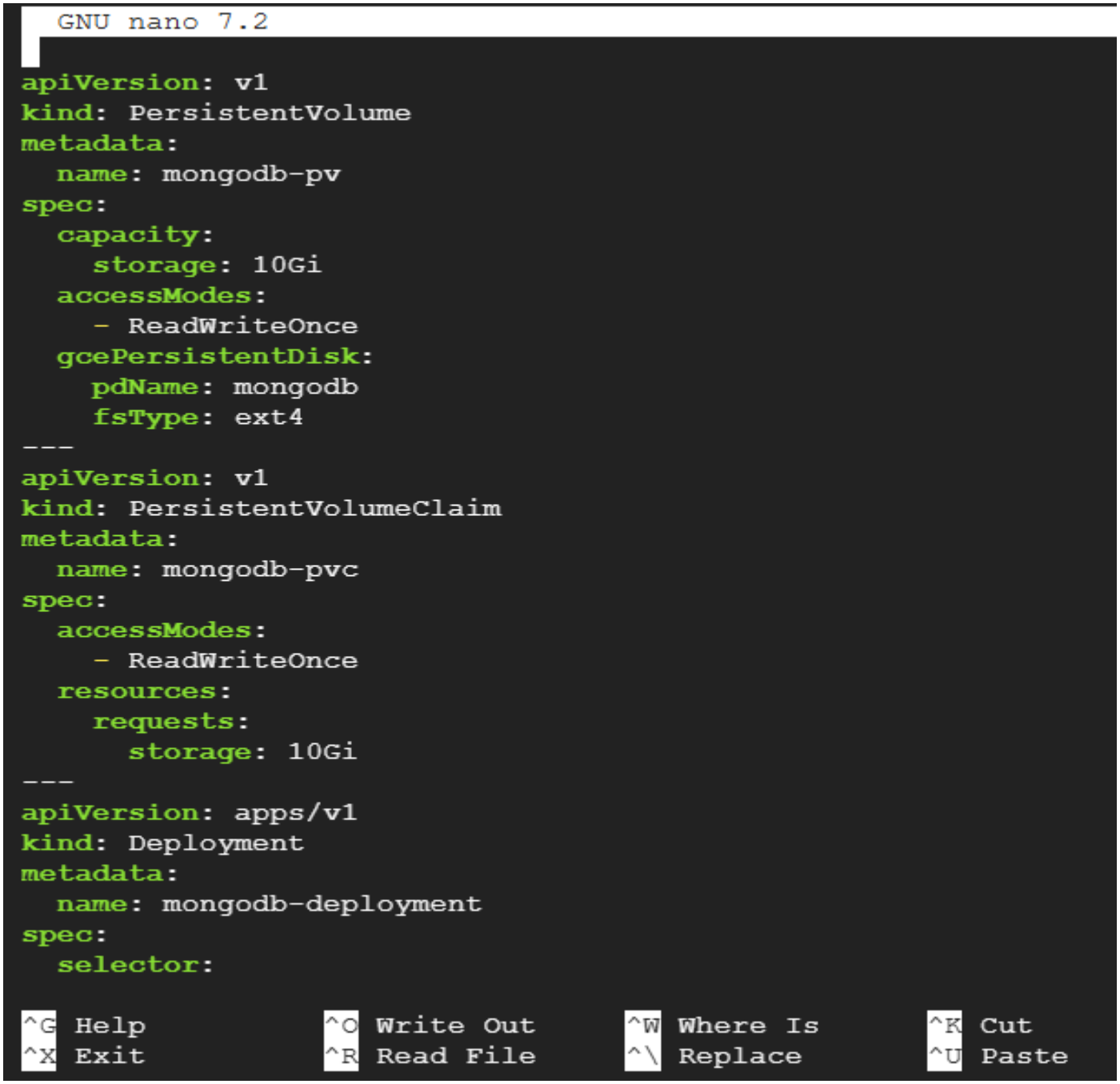
2.Create Persistent Volume:

gcloud compute disks create --size=10GiB --zone=us-east1-b mongodb

```
priyanka599@cloudshell:~ (clouddemo-441418)$ gcloud compute disks create --size=10GiB --zone=us-east1-b mongodb
WARNING: You have selected a disk size of under [200GB]. This may result in poor I/O performance. For more information
Created [https://www.googleapis.com/compute/v1/projects/clouddemo-441418/zones/us-east1-b/disks/mongodb].
NAME: mongodb
ZONE: us-east1-b
SIZE_GB: 10
TYPE: pd-standard
STATUS: READY

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

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

priyanka599@cloudshell:~ (clouddemo-441418)$
```

3.Deploy MongoDB:

Apply the mongodb-deployment.yaml configuration:



apiVersion: v1

kind: PersistentVolume

metadata:

name: mongodb-pv

spec:

capacity:

storage: 10Gi

accessModes:

```
    - ReadWriteOnce
  gcePersistentDisk:
    pdName: mongodb
    fsType: ext4
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mongodb-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
---
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
  persistentVolumeClaim:
    claimName: mongodb-pvc
```

```
priyanka599@cloudshell:~ (clouddemo-441418)$ nano mongodb-deployment.yaml
priyanka599@cloudshell:~ (clouddemo-441418)$ kubectl apply -f mongodb-deployment.yaml
persistentvolume/mongodb-pv created
persistentvolumeclaim/mongodb-pvc created
deployment.apps/mongodb-deployment created
priyanka599@cloudshell:~ (clouddemo-441418)$
```

4. **Check Deployment Status:**
   kubectl get pods

```
priyanka599@cloudshell:~ (clouddemo-441418)$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
mongodb-deployment-658f677f67-c5xgg   1/1     Running   0          5m45s
priyanka599@cloudshell:~ (clouddemo-441418)$
```

Ensure the pod status is Running.

5. **Create MongoDB Service**:
   Apply the mongodb-service.yaml configuration:

```
  GNU nano 7.2
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  type: LoadBalancer
  selector:
    app: mongodb
  ports:
    - port: 27017          # Service port within the cluster
      targetPort: 27017
```

kubectl apply -f mongodb-service.yaml

```
priyanka599@cloudshell:~ (clouddemo-441418)$ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
priyanka599@cloudshell:~ (clouddemo-441418)$
```

6. **Verify Service Status:**
   kubectl get svc
   Wait for the EXTERNAL-IP to be assigned.

```
priyanka599@cloudshell:~ (clouddemo-441418)$ kubectl get svc
NAME              TYPE           CLUSTER-IP       EXTERNAL-IP      PORT(S)           AGE
kubernetes        ClusterIP      34.118.224.1     <none>           443/TCP           41m
mongodb-service   LoadBalancer   34.118.231.193   35.237.145.144   27017:31414/TCP   2m14s
priyanka599@cloudshell:~ (clouddemo-441418)$
```

7. **Test MongoDB Connection: Access the MongoDB pod and connect:**
   kubectl exec -it mongodb-deployment-replace-with-your-pod-name – bash

```
priyanka599@cloudshell:~ (clouddemo-441418)$ ^C
priyanka599@cloudshell:~ (clouddemo-441418)$ kubectl exec -it mongodb-deployment-658f677f67-c5xgg -- bash
root@mongodb-deployment-658f677f67-c5xgg:/#
```

Try

mongo External-IP

You should see something like this, which means your mongoDB is up and can be

accessd using the External-IP

```
root@mongodb-deployment-658f677f67-c5xgg:/# mongosh 35.237.145.144
Current Mongosh Log ID: 67328981e8c8a67a7efe6910
Connecting to:          mongodb://35.237.145.144:27017/?directConnection=true&appName=mongosh+2.3.2
Using MongoDB:          8.0.3
Using Mongosh:          2.3.2

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
   The server generated these startup warnings when booting
   2024-11-11T22:41:47.377+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage en
   2024-11-11T22:42:07.870+00:00: Access control is not enabled for the database. Read and write access to data a
   2024-11-11T22:42:07.870+00:00: For customers running the current memory allocator, we suggest changing the con
   2024-11-11T22:42:07.870+00:00: For customers running the current memory allocator, we suggest changing the con
   2024-11-11T22:42:07.870+00:00: We suggest setting the contents of sysfsFile to 0.
   2024-11-11T22:42:07.870+00:00: Your system has glibc support for rseq built in, which is not yet supported by
onment variable GLIBC_TUNABLES=glibc.pthread.rseq=0
   2024-11-11T22:42:07.870+00:00: vm.max_map_count is too low
------

test>
```

8. **Type exit to exit from the shell and go back to the cloud shell**

```
test> exit
root@mongodb-deployment-658f677f67-rff99:/# exit
exit
priyanka599@cloudshell:~ (clouddemo-441418)$
```

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

```
priyanka599@cloudshell:~ (clouddemo-441418)$ node
Welcome to Node.js v22.11.0.
Type ".help" for more information.
>
```

Enter the following line by line

const { MongoClient } = require('mongodb');

const url = "mongodb://35.237.145.144:27017/mydb"; // Replace with your MongoDB connection string

```
async function run() {
  const client = new MongoClient(url);

  try {
    // Connect to the MongoDB server
    await client.connect();
    console.log("Connected to MongoDB");

    const db = client.db("studentdb");

    // Documents to insert
    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 }
    ];

    // Insert the documents
    const insertResult = await db.collection("students").insertMany(docs);
    console.log(` ${insertResult.insertedCount} documents inserted `);

    // Retrieve the document with student_id 11111
    const student = await db.collection("students").findOne({ student_id: 11111 });
    console.log("Retrieved document:", student);
  } catch (err) {
    console.error("An error occurred:", err);
  } finally {
    // Close the database connection
    await client.close();
    console.log("Connection closed");
```

```
  }
}

run();
```

```
...         // Documents to insert
...         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 }
...         ];
...
...         // Insert the documents
...         const insertResult = await db.collection("students").insertMany(docs);
...         console.log(`${insertResult.insertedCount} documents inserted`);
...
...         // Retrieve the document with student_id 11111
...         const student = await db.collection("students").findOne({ student_id: 11111 });
...         console.log("Retrieved document:", student);
...     } catch (err) {
...         console.error("An error occurred:", err);
...     } finally {
...         // Close the database connection
...         await client.close();
...         console.log("Connection closed");
...     }
... }
undefined
>
> run();
Promise {
  <pending>,
  [Symbol(async_id_symbol)]: 60,
  [Symbol(trigger_async_id_symbol)]: 6
}
> Connected to MongoDB
3 documents inserted
Retrieved document: {
  _id: new ObjectId('67326df2d18b01219a57c841'),
  student_id: 11111,
  student_name: 'Bruce Lee',
  grade: 84
}
```

Then we can see that 3 rows were inserted into the database and we could retrieve student data with id 11111.

Step 2: Modify StudentServer to Fetch 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;  // Ensure these are set in your environment


var MongoClient = mongodb.MongoClient;

var uri = `mongodb://${MONGO_URL}/${MONGO_DATABASE}`;


console.log(`Connecting to MongoDB at ${uri}`);


var server = http.createServer(function (req, res) {
 var result;


 // Parse the query string from the URL
 var parsedUrl = url.parse(req.url, true);
 var student_id = parseInt(parsedUrl.query.student_id);


 // Match the URL for API endpoint /api/score
 if (/^\/api\/score/.test(req.url)) {
  // Connect to MongoDB
  MongoClient.connect(uri, { useUnifiedTopology: true }, function (err, client) {
   if (err) {
    res.writeHead(500, { 'Content-Type': 'application/json' });
    res.end(JSON.stringify({ error: "Failed to connect to the database" }) + '\n');
    return;
   }
   var db = client.db(MONGO_DATABASE);


   // Fetch the student from the 'students' collection based on the student_id
```

```
    db.collection("students").findOne({ "student_id": student_id }, (err, student) => {
      if (err) {
        res.writeHead(500, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify({ error: "Error fetching student data" }) + '\n');
        return;
      }

      if (student) {
        // Remove the MongoDB _id field before returning the response
        const { _id, ...studentResponse } = student;

        res.writeHead(200, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify(studentResponse) + '\n');
      } else {
        res.writeHead(404, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify({ error: "Student Not Found" }) + '\n');
      }
      client.close();  // Always close the database connection
    });
  });
 } else {
   res.writeHead(404, { 'Content-Type': 'application/json' });
   res.end(JSON.stringify({ error: "Wrong URL, please try again" }) + '\n');
 }
});

server.listen(8080, () => {
  console.log("Server is listening on port 8080");
});
```

```
const http = require('http');
const url = require('url');
const { MongoClient } = require('mongodb');
const { MONGO_URL, MONGO_DATABASE } = process.env;

// Connection URI
const uri = mongodb://${MONGO_URL}/${MONGO_DATABASE};
console.log(uri);

// Create a server
const server = http.createServer(async (req, res) => {
  try {
    // Parse the URL and query string
    const parsedUrl = url.parse(req.url, true);
    const student_id = parseInt(parsedUrl.query.student_id);

    // Match req.url with the string /api/score
    if (/^\/api\/score/.test(req.url)) {
      // Connect to the database
      const client = new MongoClient(uri);
      await client.connect();
      const db = client.db("studentdb");

      // Find the student document
      const student = await db.collection("students").findOne({ "student_id": student_id });
      await client.close();

      if (student) {
        // Prepare the response object
        const response = {
          student_id: student.student_id,
          student_name: student.student_name,
          student_score: student.grade
```

2. **Create Dockerfile**

FROM node:7

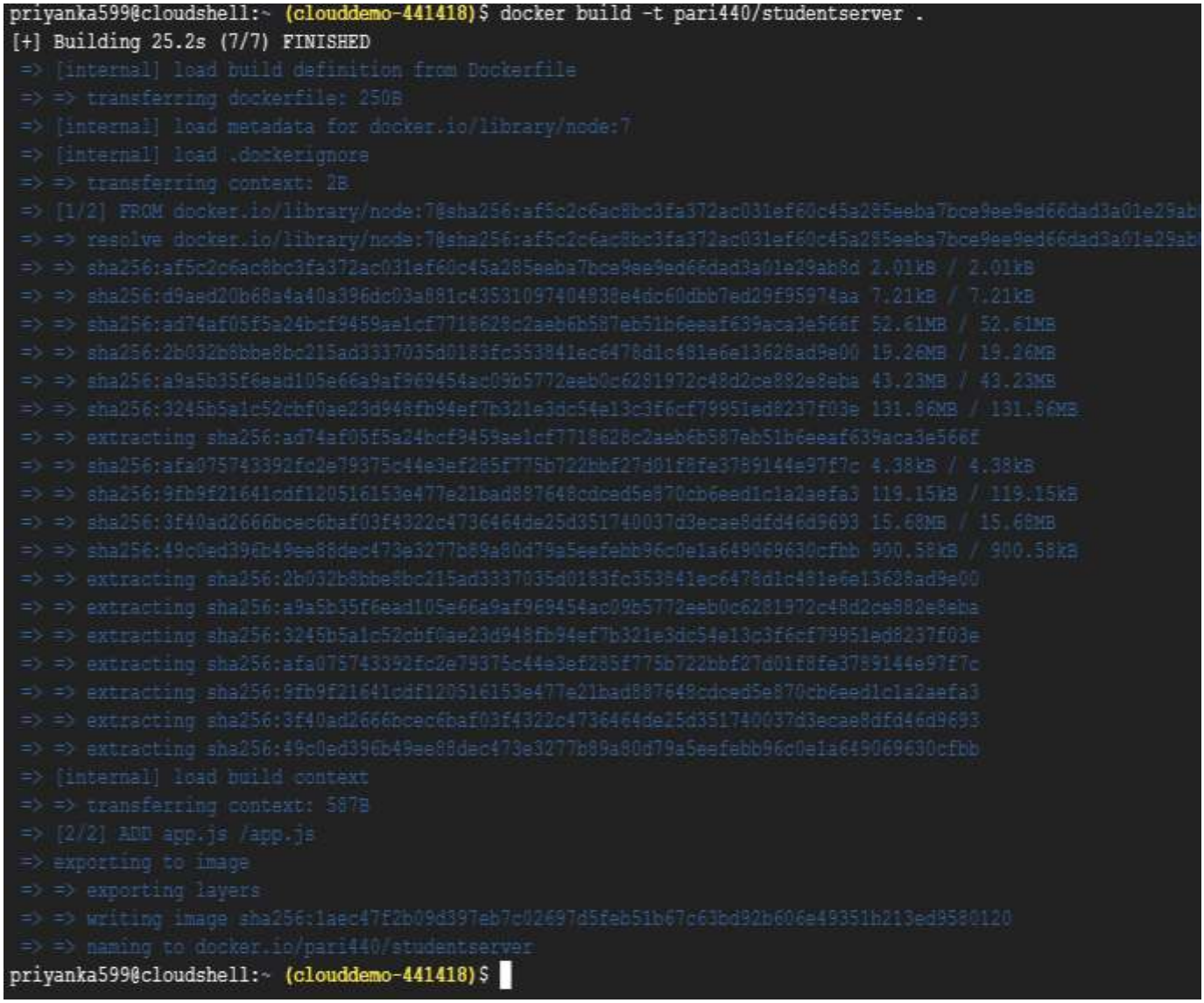
ADD studentServer.js /studentServer.js

ENTRYPOINT ["node", "studentServer.js"]

RUN npm install mongodb

3**.Build the studentserver docker image**

docker build -t yourdockerhubID/studentserver .



4.Push Docker Image to Docker Hub:

docker push yourdockerhubID/studentserver

```
priyanka599@cloudshell:~ (clouddemo-441418)$ docker push pari440/studentserver
Using default tag: latest
The push refers to repository [docker.io/pari440/studentserver]
e7882650c041: 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: sha256:603fb0a083553dcf1fdd7f295ff9e82778fa54f2e21ac21f34637b11adeac09c size: 2213
priyanka599@cloudshell:~ (clouddemo-441418)$
```

Step 3: Create the Flask Application

1. **Create bookshelf.py:**

```
from flask import Flask, request, jsonify

from flask_pymongo import PyMongo

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_books():
  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="Book saved successfully!")

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

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

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

@app.route("/books/delete", methods=["POST"])
def delete_all_books():
    db.bookshelf.delete_many({})
    return jsonify(message="All books deleted!")

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

3.Dockerize the Flask Application

create requirements.txt

```
Flask==2.2.2
Flask-PyMongo==2.3.0
PyMongo>=3.3
```

Create the Dockerfile

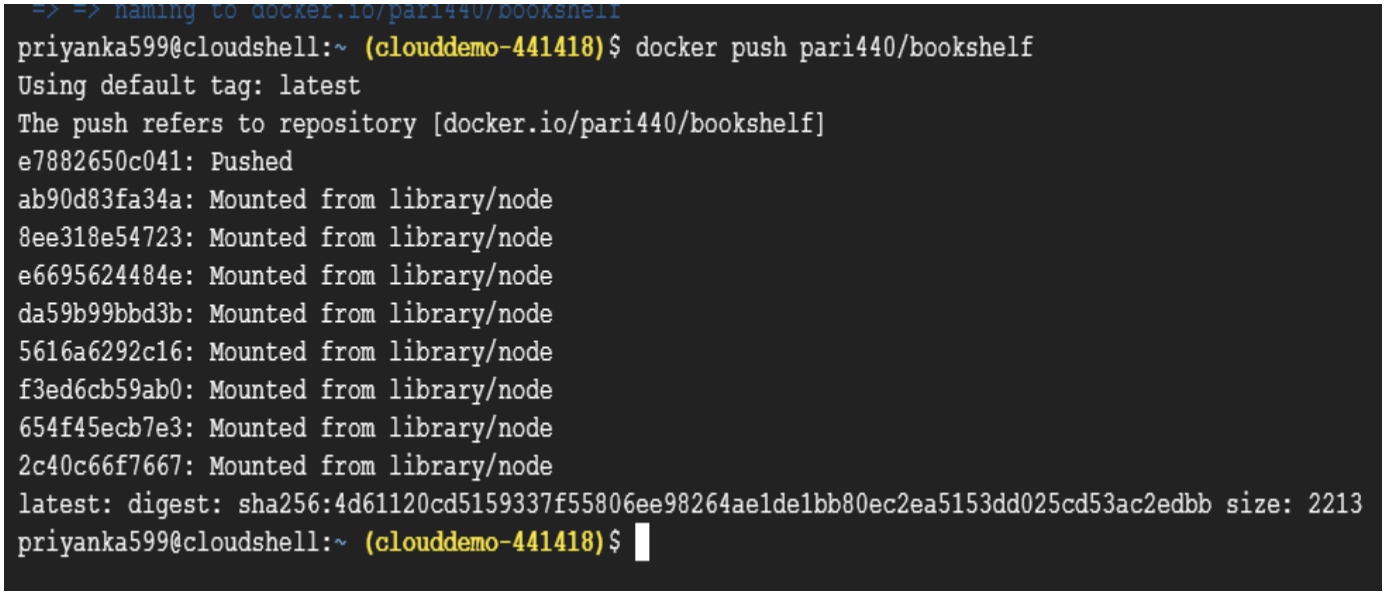
```
FROM python:3.7-alpine
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
ENV PORT 5000
EXPOSE 5000
ENTRYPOINT ["python3"]
CMD ["bookshelf.py"]
```

1. **Build the Docker Image:**

   docker build -t pari440/bookshelf .

```
priyanka599@cloudshell:~ (clouddemo-441418)$ docker build -t pari440/bookshelf .
[+] Building 23.5s (7/7) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 250B
 => [internal] load metadata for docker.io/library/node:7
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load build context
 => => transferring context: 587B
 => [1/2] FROM docker.io/library/node:7@sha256:af5c2c6ac8bc3fa372ac031ef60c45a285eeba7bce9ee9ed66d
 => => resolve docker.io/library/node:7@sha256:af5c2c6ac8bc3fa372ac031ef60c45a285eeba7bce9ee9ed66d
 => => sha256:ad74af05f5a24bcf9459ae1cf7718628c2aeb6b587eb51b6eeaf639aca3e566f 52.61MB / 52.61MB
 => => sha256:2b032b8bbe8bc215ad3337035d0183fc353841ec6478d1c481e6e13628ad9e00 19.26MB / 19.26MB
 => => sha256:a9a5b35f6ead105e66a9af969454ac09b5772eeb0c6281972c48d2ce882e8eba 43.23MB / 43.23MB
 => => sha256:af5c2c6ac8bc3fa372ac031ef60c45a285eeba7bce9ee9ed66dad3a01e29ab8d 2.01kB / 2.01kB
 => => sha256:d9aed20b68a4a40a396dc03a881c43531097404838e4dc60dbb7ed29f95974aa 7.21kB / 7.21kB
 => => extracting sha256:ad74af05f5a24bcf9459ae1cf7718628c2aeb6b587eb51b6eeaf639aca3e566f
 => => sha256:afa075743392fc2e79375c44e3ef285f775b722bbf27d01f8fe3789144e97f7c 4.38kB / 4.38kB
 => => sha256:3245b5a1c52cbf0ae23d948fb94ef7b321e3dc54e13c3f6cf79951ed8237f03e 131.86MB / 131.86MB
 => => sha256:9fb9f21641cdf120516153e477e21bad887648cdced5e870cb6eed1c1a2aefa3 119.15kB / 119.15kB
 => => sha256:3f40ad2666bcec6baf03f4322c4736464de25d351740037d3ecae8dfd46d9693 15.68MB / 15.68MB
 => => sha256:49c0ed396b49ee88dec473e3277b89a80d79a5eefebb96c0e1a649069630cfbb 900.58kB / 900.58kB
 => => extracting sha256:2b032b8bbe8bc215ad3337035d0183fc353841ec6478d1c481e6e13628ad9e00
 => => extracting sha256:a9a5b35f6ead105e66a9af969454ac09b5772eeb0c6281972c48d2ce882e8eba
 => => extracting sha256:3245b5a1c52cbf0ae23d948fb94ef7b321e3dc54e13c3f6cf79951ed8237f03e
 => => extracting sha256:afa075743392fc2e79375c44e3ef285f775b722bbf27d01f8fe3789144e97f7c
 => => extracting sha256:9fb9f21641cdf120516153e477e21bad887648cdced5e870cb6eed1c1a2aefa3
 => => extracting sha256:3f40ad2666bcec6baf03f4322c4736464de25d351740037d3ecae8dfd46d9693
 => => extracting sha256:49c0ed396b49ee88dec473e3277b89a80d79a5eefebb96c0e1a649069630cfbb
 => [2/2] ADD app.js /app.js
 => exporting to image
 => => exporting layers
 => => writing image sha256:21a0c049a6cf0c2af976265741b493e340d85fce29ef38e14a18a52ca75ba9ab
 => => naming to docker.io/pari440/bookshelf
priyanka599@cloudshell:~ (clouddemo-441418)$
```

2. **Push the Docker Image to Docker Hub:**
   docker push pari440/bookshelf



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

1. Create a ConfigMap for studentServer

Create a file named studentserver-configmap.yaml with the following content:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  MONGO_URL: <Your-MongoDB-EXTERNAL-IP>
  MONGO_DATABASE: mydb
```

2. Create a ConfigMap for bookshelf

Create a file named bookshelf-configmap.yaml with the following content:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: bookshelf-config
data:
  MONGO_URL: 35.237.145.144  # MongoDB external IP
  MONGO_DATABASE: mydb
```

Notice:

• The reason for creating these two ConfigMaps is to avoid re-building the Docker image again if the MongoDB pod restarts with a different External-IP.

Step 5: Expose Two Applications Using Ingress with Nginx

1. **Create studentserver-deployment.yaml**

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: studentserver
 labels:
  app: studentserver-deploy
spec:
 replicas: 1
 selector:
  matchLabels:
   app: studentserver
 template:
  metadata:
   labels:
    app: studentserver
  spec:
   containers:
   - image: pari440/studentserver
    name: studentserver
    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
```

```
 labels:
   app: bookshelf-deploy
spec:
 replicas: 1
 selector:
  matchLabels:
   app: bookshelf
 template:
  metadata:
   labels:
    app: bookshelf
  spec:
   containers:
   - image: pari440/bookshelf
     name: bookshelf
     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. studentserver-service.yaml

```
apiVersion: v1
kind: Service
```

```
metadata:
  name: studentserver-service
spec:
  type: LoadBalancer
  ports:
    - port: 8080
      targetPort: 8080
  selector:
    app: studentserver
```

4. bookshelf-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: bookshelf-service
spec:
  type: LoadBalancer
  ports:
    - port: 5000
      targetPort: 5000
  selector:
    app: bookshelf
```

5.Start Minikube

```
minikube start
```

```
priyanka599@cloudshell:~ (clouddemo-441418)$ minikube start
* minikube v1.34.0 on Ubuntu 24.04 (amd64)
  - MINIKUBE_FORCE_SYSTEMD=true
  - MINIKUBE_HOME=/google/minikube
  - MINIKUBE_WANTUPDATENOTIFICATION=false
* Automatically selected the docker driver. Other choices: ssh, none
* Using Docker driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.45 ...
* Downloading Kubernetes v1.31.0 preload ...
    > preloaded-images-k8s-v18-v1...:  326.69 MiB / 326.69 MiB  100.00% 212.79
    > gcr.io/k8s-minikube/kicbase...:  487.90 MiB / 487.90 MiB  100.00% 91.09 M
* Creating docker container (CPUs=2, Memory=4000MB) ...
* Preparing Kubernetes v1.31.0 on Docker 27.2.0 ...
  - kubelet.cgroups-per-qos=false
  - kubelet.enforce-node-allocatable=""
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
* 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
priyanka599@cloudshell:~ (clouddemo-441418)$
```

6.Start Ingress

minikube addons enable ingress

```
priyanka599@cloudshell:~ (clouddemo-441418)$ minikube addons enable ingress
* ingress is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
  - Using image registry.k8s.io/ingress-nginx/controller:v1.11.2
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.3
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.3
* Verifying ingress addon...
* The 'ingress' addon is enabled
priyanka599@cloudshell:~ (clouddemo-441418)$
```

7.Create studentserver Related Pods and Start Service

kubectl apply -f studentserver-deployment.yaml

kubectl apply -f studentserver-configmap.yaml

kubectl apply -f studentserver-service.yaml

```
priyanka599@cloudshell:~ (clouddemo-441418)$ nano studentserver-deployment.yaml
priyanka599@cloudshell:~ (clouddemo-441418)$ kubectl apply -f studentserver-deployment.yaml
deployment.apps/web created
priyanka599@cloudshell:~ (clouddemo-441418)$
```

9.Check if All Pods are Running Correctly

```
priyanka599@cloudshell:~ (clouddemo-441418)$ kubectl apply -f studentserver-deployment.yaml
kubectl apply -f studentserver-configmap.yaml
kubectl apply -f studentserver-service.yaml
deployment.apps/web unchanged
configmap/studentserver-config created
```

8.Create bookshelf Related Pods and Start Service

kubectl apply -f bookshelf-deployment.yaml

kubectl apply -f bookshelf-configmap.yaml

kubectl apply -f bookshelf-service.yaml

```
priyanka599@cloudshell:~ (clouddemo-441418)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment unchanged
priyanka599@cloudshell:~ (clouddemo-441418)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config unchanged
priyanka599@cloudshell:~ (clouddemo-441418)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service unchanged
priyanka599@cloudshell:~ (clouddemo-441418)$
```

9.kubectl get pods

```
priyanka599@cloudshell:~ (clouddemo-441418)$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
bookshelf-deployment-c9c787b8d-rmnfh  1/1     Running   0          4m47s
web-86db8df96b-lz9jh                  1/1     Running   0          10m
priyanka599@cloudshell:~ (clouddemo-441418)$
```

10.Create Ingress Service YAML File

Create a file named studentservermongoIngress.yaml with the following content:

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: server

annotations:

nginx.ingress.kubernetes.io/rewrite-target: /$2

```
spec:
 rules:
 - host: cs571.project.com  # This is the domain name to access the services
   http:
    paths:
    - path: /studentserver(/|$)(.*)  # This routes requests starting with /studentserver to the studentserver service
      pathType: Prefix
      backend:
       service:
         name: web  # Replace with your studentserver service name
         port:
          number: 8080
    - path: /bookshelf(/|$)(.*)  # This routes requests starting with /bookshelf to the bookshelf service
      pathType: Prefix
      backend:
       service:
         name: bookshelf-service  # Replace with your bookshelf service name
         port:
          number: 5000
```

11.Create the Ingress Service

kubectl apply -f studentservermongoIngress.yaml

```
priyanka599@cloudshell:~ (clouddemo-441418)$ kubectl apply -f studentservermongoIngress.yaml
Warning: path /studentserver(/|$)(.*) cannot be used with pathType Prefix
Warning: path /bookshelf(/|$)(.*) cannot be used with pathType Prefix
ingress.networking.k8s.io/server created
priyanka599@cloudshell:~ (clouddemo-441418)$
```

Check if Ingress is Running

kubectl get ingress

Wait until you see the Address, then move forward.

```
priyanka599@cloudshell:~ (clouddemo-441418)$ kubectl get ingress
NAME     CLASS   HOSTS               ADDRESS        PORTS   AGE
server   nginx   cs571.project.com   192.168.49.2   80      76s
priyanka599@cloudshell:~ (clouddemo-441418)$
```

12.Add Address to /etc/hosts

sudo vi /etc/hosts

```
priyanka599@cloudshell:~ (clouddemo-441418)$ sudo vi /etc/hosts
priyanka599@cloudshell:~ (clouddemo-441418)$
```

Add the address you got from the previous 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:

192.168.49.2 cs571.project.com

```
# consulted at all; see /etc/host.conf for the resolution order.
#

# IPv4 and IPv6 localhost aliases
127.0.0.1       localhost
::1             localhost

#
# Imaginary network.
#10.0.0.2               myname
#10.0.0.3               myfriend
#
# According to RFC 1918, you can use the following IP networks for private
# nets which will never be connected to the Internet:
#
#       10.0.0.0        -   10.255.255.255
#       172.16.0.0      -   172.31.255.255
#       192.168.0.0     -   192.168.255.255
#
# In case you want to be able to connect directly to the Internet (i.e. not
# behind a NAT, ADSL router, etc...), you need real official assigned
# numbers.  Do not try to invent your own network numbers but instead get one
# from your network provider (if any) or from your regional registry (ARIN,
# APNIC, LACNIC, RIPE NCC, or AfriNIC.)
#
169.254.169.254 metadata.google.internal metadata

10.88.0.5 cs-368271150013-default
192.168.49.2 cs571.project.com
~
~
~
~
-- INSERT --
```

13.Test Your Applications

- Retrieve all students from server: If everything goes smoothly,

you should be able to access your applications

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

```
priyanka599@cloudshell:~ (clouddemo-441418)$ sudo vi /etc/hosts
priyanka599@cloudshell:~ (clouddemo-441418)$ curl cs571.project.com/studentserver/api/score?student_id=11111
{"_id":"67326df2d18b01219a57c841","student_id":11111,"student_name":"Bruce Lee","grade":84}
priyanka599@cloudshell:~ (clouddemo-441418)$ curl cs571.project.com/studentserver/api/score?student_id=22222
{"_id":"67326df2d18b01219a57c842","student_id":22222,"student_name":"Jackie Chen","grade":93}
priyanka599@cloudshell:~ (clouddemo-441418)$ curl cs571.project.com/studentserver/api/score?student_id=33333
{"_id":"67326df2d18b01219a57c843","student_id":33333,"student_name":"Jet Li","grade":88}
priyanka599@cloudshell:~ (clouddemo-441418)$
```

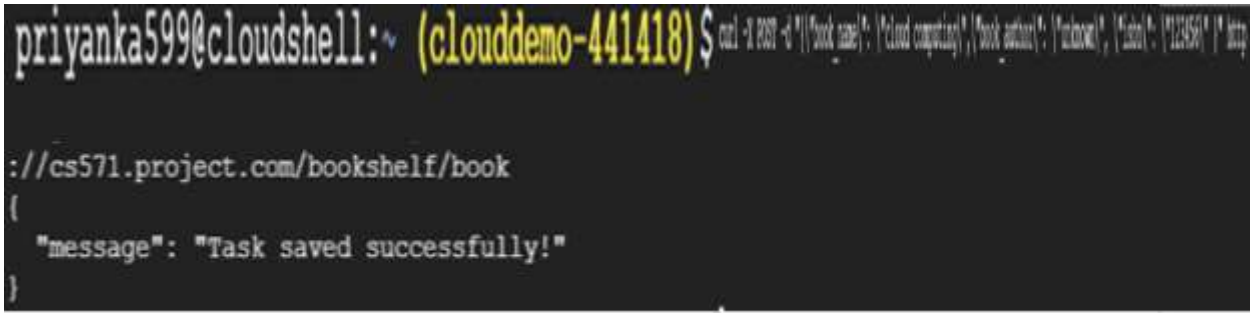
- On another path, you should be able to use the REST API with bookshelf application

i.e. list all books

curl cs571.project.com/bookshelf/books

- Add a book:

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



- Update a book:

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

- Delete a book:

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