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

1. Create a cluster as usual on GKE using the command

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

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ gcloud container clusters create kubiaa --num-nodes=1 --machine-type=e2-micro --region=us-west1
Default change: VPC-native is the default mode during cluster creation for versions greater than 1.21.0-gke.1500. To create advanced routes based clusters, please pass
the `--no-enable-ip-alias` flag
Note: The Kubelet readonly port (10255) is now deprecated. Please update your workloads to use the recommended alternatives. See https://cloud.google.com/kubernetes-eng
ine/docs/how-to/disable-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 kubiaa in us-west1... Cluster is being health-checked (master is healthy)...done.
Created [https://container.googleapis.com/v1/projects/cs570-big-data-analytics/zones/us-west1/clusters/kubiaa].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/us-west1/kubiaa?project=cs570-big-data-analytics
kubeconfig entry generated for kubiaa.
NAME: kubiaa
LOCATION: us-west1
MASTER_VERSION: 1.29.6-gke.1254000
MASTER_IP: 34.169.8.151
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.29.6-gke.1254000
NUM_NODES: 3
STATUS: RUNNING
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

2. Create a Persistent Volume first using the command

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

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ gcloud compute disks create mongodb --size=10GiB --zone=us-west1-a
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/cs570-big-data-analytics/zones/us-west1-a/disks/mongodb].
NAME: mongodb
ZONE: us-west1-a
SIZE_GB: 10
TYPE: pd-standard
STATUS: READY

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

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

nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

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:
      - name: mongo
        image: mongo
        ports:
        - containerPort: 27017
        volumeMounts:
        - name: mongodb-data
          mountPath: /data/db
      volumes:
      - name: mongodb-data
        persistentVolumeClaim:
          claimName: mongodb-pvc
```

Here is the detailed breakdown of the YAML file provided above:

1. **PersistentVolume (PV)**: Defines a persistent storage resource in the cluster.
2. **PersistentVolumeClaim (PVC)**: Requests storage resources defined by the PV.
3. **Deployment**: Manages the deployment of the MongoDB container, ensuring the specified number of replicas are running and using the persistent storage.

- *kubectl apply -f mongodb_deployment.yaml*

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl apply -f mongodb_deployment.yaml
persistentvolume/mongodb-pv created
persistentvolumeclaim/mongodb-pvc created
deployment.apps/mongodb-deployment configured
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

This command will create the PersistentVolume, PersistentVolumeClaim, and the MongoDB Deployment, ensuring that the MongoDB instance has persistent storage backed by the Google Cloud persistent disk you created earlier.

4. Check deployment of pods.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
mongodb-deployment-5c589898cb-vdgw6   1/1     Running   0          110s
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

5. Create a mongodb service.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ vi mongodb_service.yaml
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ cat mongodb_service.yaml
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  type: LoadBalancer
  ports:
    # service port in cluster
    - port: 27017
      # port to contact inside container
      targetPort: 27017
  selector:
    app: mongodb

nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl apply -f mongodb_service.yaml
service/mongodb-service created
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

This will create a LoadBalancer service that exposes your MongoDB deployment to the external network, allowing you to connect to MongoDB using the external IP address assigned by your cloud provider.

6. Check service.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl get svc
NAME              TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)           AGE
kubernetes        ClusterIP      34.118.224.1    <none>           443/TCP           110m
mongodb-service   LoadBalancer   34.118.227.17   35.227.133.217   27017:31592/TCP   5m55s
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

7. Test MongoDB connection.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl exec -it mongodb-deployment-5c589898cb-vdgw6 -- bash
root@mongodb-deployment-5c589898cb-vdgw6:/#
```

8. Connect to MongoDB using its shell.

mongosh --host 35.227.133.217 --port 27017

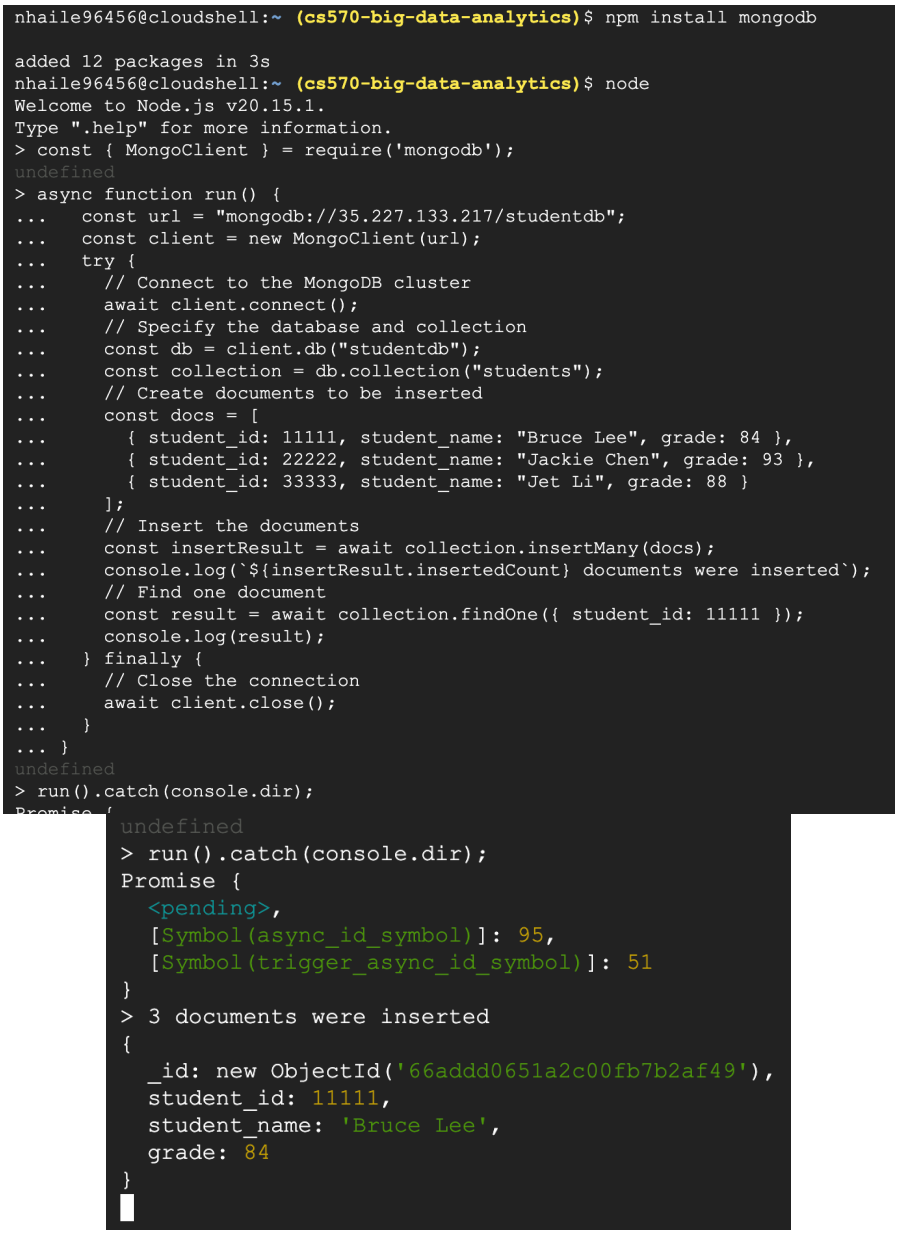
```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl exec -it mongodb-deployment-5c589898cb-vdgw6 -- bash
root@mongodb-deployment-5c589898cb-vdgw6:/# mongosh --host 35.227.133.217 --port 27017
Current Mongosh Log ID: 66addbace96a21036c149f47
Connecting to:          mongodb://35.227.133.217:27017/?directConnection=true&appName=mongosh+2.2.10
Using MongoDB:          7.0.12
Using Mongosh:          2.2.10

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


To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

------
   The server generated these startup warnings when booting
   2024-08-03T07:12:55.717+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-file
system
   2024-08-03T07:12:56.897+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
   2024-08-03T07:12:56.898+00:00: vm.max_map_count is too low
------

test>
```

```
root@mongodb-deployment-5c589898cb-vdgw6:/# mongod --version
db version v7.0.12
Build Info: {
    "version": "7.0.12",
    "gitVersion": "b6513ce0781db6818e24619e8a461eae90bc94fc",
    "openSSLVersion": "OpenSSL 3.0.2 15 Mar 2022",
    "modules": [],
    "allocator": "tcmalloc",
    "environment": {
        "distmod": "ubuntu2204",
        "distarch": "x86_64",
        "target_arch": "x86_64"
    }
}
root@mongodb-deployment-5c589898cb-vdgw6:/#
```

9. Insert records into MongoDB.

```
const { MongoClient } = require('mongodb');
async function run() {
  const url = "mongodb://35.227.133.217/studentdb";
  const client = new MongoClient(url);
  try {
    // Connect to the MongoDB cluster
    await client.connect();
    // Specify the database and collection
    const db = client.db("studentdb");
    const collection = db.collection("students");
    // Create documents to be inserted
    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 collection.insertMany(docs);
    console.log(`${insertResult.insertedCount} documents were inserted`);
    // Find one document
    const result = await collection.findOne({ student_id: 11111 });
    console.log(result);
  } finally {
    // Close the connection
    await client.close();
  }
}
run().catch(console.dir);
```



Step 2: Modify Student Server to Get Records from MongoDB and Deploy to GKE

1. Create studentServer.js

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ vi studentServer.js
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ cat studentServer.js
const http = require('http');
const url = require('url');
const { MongoClient } = require('mongodb');

// Environment variables for MongoDB connection
const { MONGO_URL, MONGO_DATABASE } = process.env;
const uri = `mongodb://${MONGO_URL}/${MONGO_DATABASE}`;
console.log(`MongoDB URI: ${uri}`);

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);

    // Check if the URL matches /api/score and student_id is provided
    if (/^\/api\/score/.test(req.url) && !isNaN(student_id)) {
      // Connect to the database
      const client = new MongoClient(uri);
      try {
```

2. Create a dockerfile.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ vi Dockerfile
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ cat Dockerfile
# Use a smaller base image
FROM node:alpine

# Set the working directory
WORKDIR /app

# Copy package.json and package-lock.json
COPY package*.json ./

# Install dependencies
RUN npm install

# Copy the rest of the application code
COPY . .

# Expose the port your app runs on
EXPOSE 8080

# Use CMD to run your application
CMD ["node", "studentServer.js"]

nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

3. Build docker image.

docker build -t student-server .

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ docker build -t student-server .
[+] Building 10.6s (10/10) FINISHED                                                   docker:default
 => [internal] load build definition from Dockerfile                                            0.0s
 => => transferring dockerfile: 397B                                                            0.0s
 => [internal] load metadata for docker.io/library/node:alpine                                  1.0s
 => [internal] load .dockerignore                                                               0.0s
 => => transferring context: 2B                                                                 0.0s
 => [1/5] FROM docker.io/library/node:alpine@sha256:39005f06b2fae765764d6fdf20ad1c4d0890f5ad3e1f39b56a18768334b8ecd6  4.5s
 => => resolve docker.io/library/node:alpine@sha256:39005f06b2fae765764d6fdf20ad1c4d0890f5ad3e1f39b56a18768334b8ecd6  0.0s
 => => sha256:39005f06b2fae765764d6fdf20ad1c4d0890f5ad3e1f39b56a18768334b8ecd6 6.62kB / 6.62kB  0.0s
 => => sha256:c83e6e8aa2c458cf740b18b7b13e546751fe081d36223aac253b5ec0da2cd89d 1.72kB / 1.72kB  0.0s
 => => sha256:5c4cc5767575c711b99b1b077ad8afb8cfbf407aca174b7cbc998ce6db1e4f93 6.36kB / 6.36kB  0.0s
 => => sha256:c6a83fedfae6ed8a4f5f7cbb6a7b6f1c1ec3d86fea8cb9e5ba2e5e6673fde9f6 3.62MB / 3.62MB  0.4s
 => => sha256:8d90f41c769e0bfd90a1e8456db9f590ae8dc42842ffa098693b6ed4bd44eba3 47.36MB / 47.36MB  1.1s
 => => sha256:c4f54159f74a5dccd97b9af978fab507483785736e822851638f20f275716fc3 1.39MB / 1.39MB  0.2s
 => => sha256:6ecb2bd0d8e8f8628fe4a7cf14404c59d67a5547c6e5faa4a507f4b232bd316e 449B / 449B  0.4s
 => => extracting sha256:c6a83fedfae6ed8a4f5f7cbb6a7b6f1c1ec3d86fea8cb9e5ba2e5e6673fde9f6  0.2s
 => => extracting sha256:8d90f41c769e0bfd90a1e8456db9f590ae8dc42842ffa098693b6ed4bd44eba3  3.0s
 => => extracting sha256:c4f54159f74a5dccd97b9af978fab507483785736e822851638f20f275716fc3  0.1s
 => => extracting sha256:6ecb2bd0d8e8f8628fe4a7cf14404c59d67a5547c6e5faa4a507f4b232bd316e  0.0s
 => [internal] load build context                                                               2.0s
 => => transferring context: 78.86MB                                                            2.0s
 => [2/5] WORKDIR /app                                                                          1.3s
 => [3/5] COPY package*.json ./                                                                 0.1s
 => [4/5] RUN npm install                                                                       2.0s
 => [5/5] COPY . .                                                                              0.8s
 => exporting to image                                                                          0.7s
 => => exporting layers                                                                         0.7s
 => => writing image sha256:3723e13ed5333418c52a2fda143aa2649c654bd958be22068bc1e4d169dbc6db    0.0s
 => => naming to docker.io/library/student-server                                               0.0s
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

4. Push Docker image by tagging it first.

   docker tag student-server:latest nhaile/mydb:latest

   docker push nhaile/mydb:latest

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ docker tag student-server:latest nhaile/mydb:latest
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ docker push nhaile/mydb:latest
The push refers to repository [docker.io/nhaile/mydb]
3607cfb1d8e8: Pushed
b619adf55f4d: Pushed
b681a67fda5b: Pushed
3bf41f1e9a4c: Pushed
a64fa369054d: Mounted from library/node
1a944437090a: Mounted from library/node
4b76468bfe06: Mounted from library/node
78561cef0761: Mounted from library/node
latest: digest: sha256:49a2342772d9ba0cce059d7e123d5a6fb85b996e5a775a2352809d5bad5ee4d6 size: 1995
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

Step 3: Create Python Flask Bookshelf REST API and Deploy on GKE

1. Create bookshelf.py.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ vi bookshelf.py
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ cat 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"] = f"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=f"Welcome to bookshelf app! I am running inside {hostname} pod!")

@app.route("/books", methods=["GET"])
def get_all_books():
    try:
        books = db.bookshelf.find()
        data = []
        for book in books:
```

2. Create Dockerfile.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ vi Dockerfile
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ cat Dockerfile
# Use a specific tag for the Python base image to ensure consistency
FROM python:3.7-alpine

# Set the working directory
WORKDIR /app

# Copy only the requirements file first for better caching
COPY requirements.txt ./

# Install dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Copy the rest of the application code
COPY . .

# Set environment variables
ENV PORT 5000

# Expose the port
EXPOSE 5000

# Set the command to run the application
ENTRYPOINT ["python3"]
CMD ["bookshelf.py"]

nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

3. Create requirements.txt.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ vi requirements.txt
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ cat requirements.txt
Flask==2.0.1
flask_pymongo==2.3.0
pymongo==3.11.4
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

4. Build docker image

docker build -t nhaile/bookshelf .

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ docker build -t nhaile/bookshelf .
[+] Building 13.2s (10/10) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 544B
 => [internal] load metadata for docker.io/library/python:3.7-alpine
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/5] FROM docker.io/library/python:3.7-alpine@sha256:f3d31c8677d03f0b3c724446077f229a6ce9d3ac430f5c08cd7dff00292048c3
 => => resolve docker.io/library/python:3.7-alpine@sha256:f3d31c8677d03f0b3c724446077f229a6ce9d3ac430f5c08cd7dff00292048c3
 => => sha256:1bac8ae77e4af0b868b62a75115616a20e025e0451eeed05d94a4cfc4523e58a 6.87kB / 6.87kB
 => => sha256:96526aa774ef0126ad0fe9e9a95764c5fc37f409ab9e97021e7b4775d82bf6fa 3.40MB / 3.40MB
 => => sha256:9875af95546db78168a6761b7fa205ed1cd0c153cd89356c1512e551c12b2d5c 622.29kB / 622.29kB
 => => sha256:4819c95424fc4a94767c9329b02238ebcce0bc682384cb671379bc1fb8a12b55 10.94MB / 10.94MB
 => => sha256:f3d31c8677d03f0b3c724446077f229a6ce9d3ac430f5c08cd7dff00292048c3 1.65kB / 1.65kB
 => => sha256:e6da3ee9bb64dd12b98fa609487f112fe1e365522e6e8345309db15c22a80a51 1.37kB / 1.37kB
 => => extracting sha256:96526aa774ef0126ad0fe9e9a95764c5fc37f409ab9e97021e7b4775d82bf6fa
 => => sha256:148762f75a1f92cc9857e9c488bf95d5aac61e9905ec47a7408025b2dd5c3b7a 240B / 240B
 => => extracting sha256:9875af95546db78168a6761b7fa205ed1cd0c153cd89356c1512e551c12b2d5c
 => => sha256:ea1518237b3753b3fe40ee773d77651704178d9baa72ae5012e13a992cfa6c63 2.85MB / 2.85MB
 => => extracting sha256:4819c95424fc4a94767c9329b02238ebcce0bc682384cb671379bc1fb8a12b55
 => => extracting sha256:148762f75a1f92cc9857e9c488bf95d5aac61e9905ec47a7408025b2dd5c3b7a
 => => extracting sha256:ea1518237b3753b3fe40ee773d77651704178d9baa72ae5012e13a992cfa6c63
 => [internal] load build context
 => => transferring context: 190.58kB
 => [2/5] WORKDIR /app
 => [3/5] COPY requirements.txt ./
 => [4/5] RUN pip install --no-cache-dir -r requirements.txt
 => [5/5] COPY . .
 => exporting to image
 => => exporting layers
 => => writing image sha256:44f97b046cd5034663c08225c5f4f6c8915ca23cf0170f86676141ac97204de4
 => => naming to docker.io/nhaile/bookshelf

 1 warning found (use --debug to expand):
 - LegacyKeyValueFormat: "ENV key=value" should be used instead of legacy "ENV key value" format (line 17)
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

5. Push docker image.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ docker push nhaile/bookshelf
Using default tag: latest
The push refers to repository [docker.io/nhaile/bookshelf]
cb5bb6af2508: Pushed
1bb97eff5a2f: Pushed
8ffe78acfbcf: Pushed
4e51a863c261: Pushed
ae2ed3079163: Mounted from library/python
aa3a591fc84e: Mounted from library/python
7f29b11ef9dd: Mounted from library/python
a1c2f058ec5f: Mounted from library/python
cc2447e1835a: Mounted from library/python
latest: digest: sha256:68e4e756782d0a61588623c57fadc181a7a1a9265279dbb24cb34cc0ce6da6a4 size: 2204
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

Step 4: Create ConfigMap for Both Applications

1. Create studentserver_config.yaml:

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ vi studentServer_configmap.yaml
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ cat studentServer_configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  MONGO_URL: "35.227.133.217"
  MONGO_DATABASE: "studentdb"

nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

2. Create bookshelf_configmap.yaml.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ vi bookshelf_configmap.yaml
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ cat bookshelf_configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: bookshelf-config
data:
  MONGO_URL: "35.227.133.217"
  MONGO_DATABASE: "studentdb"

nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

3. Create studentServer_deployment.yaml.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ vi studentServer_deployment.yaml
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ cat studentServer_deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
  labels:
    app: studentserver-deploy
spec:
  replicas: 1
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
      - name: web
        image: nhaile/mydb
        imagePullPolicy: Always
        ports:
        - containerPort: 8080
```

4. Create bookshelf_deployment.yaml.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ vi bookshelf_deployment.yaml
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ cat bookshelf_deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: bookshelf-deployment
  labels:
    app: bookshelf-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: bookshelf-deployment
  template:
    metadata:
      labels:
        app: bookshelf-deployment
    spec:
      containers:
      - name: bookshelf
        image: nhaile/bookshelf
        imagePullPolicy: Always
        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

nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

5. Create studentServer_service.yaml.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ vi studentServer_service.yaml
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ cat studentServer_service.yaml
apiVersion: v1
kind: Service
metadata:
  name: web
spec:
  type: LoadBalancer
  ports:
  - port: 8080
    targetPort: 8080
  selector:
    app: web

nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

6. Create bookshelf_service.yaml.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ vi bookshelf_service.yaml
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ cat bookshelf_service.yaml
apiVersion: v1
kind: Service
metadata:
  name: bookshelf-service
spec:
  type: LoadBalancer
  ports:
  - port: 5000
    targetPort: 5000
  selector:
    app: bookshelf_deployment

nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

7. Start minikube.

minikube start

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ minikube start
* minikube v1.33.1 on Ubuntu 22.04 (amd64)
  - MINIKUBE_FORCE_SYSTEMD=true
  - MINIKUBE_HOME=/google/minikube
  - MINIKUBE_WANTUPDATENOTIFICATION=false
* Automatically selected the docker driver. Other choices: none, ssh
* Using Docker driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.44 ...
* Downloading Kubernetes v1.30.0 preload ...
    > preloaded-images-k8s-v18-v1...:  342.90 MiB / 342.90 MiB  100.00% 233.31
    > gcr.io/k8s-minikube/kicbase...:  481.58 MiB / 481.58 MiB  100.00% 96.10 M
* Creating docker container (CPUs=2, Memory=4000MB) ...
* Preparing Kubernetes v1.30.0 on Docker 26.1.1 ...
  - 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
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

8. Start Ingress.

This command enables the Nginx ingress controller in Minikube.

minikube addons enable ingress

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ 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:v1.10.1
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.1
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.1
* Verifying ingress addon...
* The 'ingress' addon is enabled
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

Enabling the ingress addon allows you to manage inbound traffic to your Kubernetes services using the Nginx ingress controller.

9. Apply Student Server Deployment and Service

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl apply -f studentServer_configmap.yaml
configmap/studentserver-config created
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl apply -f studentServer_deployment.yaml
deployment.apps/web created
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl apply -f studentServer_service.yaml
service/web created
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

10. Apply bookshelf deployment and service.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl apply -f bookshelf_configmap.yaml
configmap/bookshelf-config created
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl apply -f bookshelf_deployment.yaml
deployment.apps/bookshelf-deployment created
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl apply -f bookshelf_service.yaml
service/bookshelf-service created
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

11. Check pods' status.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl get pods
NAME                                   READY   STATUS    RESTARTS   AGE
bookshelf-deployment-654cb4dc68-vf7xm   1/1     Running   0          40s
web-5f87797c4d-tgpcd                   1/1     Running   0          2m2s
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

12. Create ingress configuration.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ vi studentservermongoIngress.yaml
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ cat studentservermongoIngress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: server
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
  rules:
  - host: cs571.project.com
    http:
      paths:
      - path: /studentserver(/|$)(.*)
        pathType: ImplementationSpecific
        backend:
          service:
            name: web
            port:
              number: 8080
      - path: /bookshelf(/|$)(.*)
        pathType: ImplementationSpecific
        backend:
          service:
            name: bookshelf-service
            port:
              number: 5000

nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

13. Apply ingress configuration.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl apply -f studentservermongoIngress.yaml
ingress.networking.k8s.io/server created
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

14. Check for ingress status.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl get ingress
NAME     CLASS   HOSTS               ADDRESS   PORTS   AGE
server   nginx   cs571.project.com             80      32s
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

15. Update /etc/hosts

This step updates the /etc/hosts file on your local machine to map the ingress address to the domain name cs571.project.com.

Before applying lets do some checkups: get the Minikube IP.

minikube ip

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ minikube ip
192.168.49.2
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

16. In the file /etc/hosts append the following line.

192.168.49.2 cs571.project.com

17. Access the application.

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

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ curl cs571.project.com/studentserver/api/score?student_id=11111
{"student_id":11111,"student_name":"Bruce Lee","student_score":84}nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
Processing request
```

18. List all the books in the bookshelf.

curl http://cs571.project.com/bookshelf/books

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ curl http://cs571.project.com/bookshelf/books
[]
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

19. Add a book

curl -X POST -H "Content-Type: application/json" -d '{"book_name": "cloud computing", "book_author": "unknown", "isbn": "123456"}' http://cs571.project.com/bookshelf/book

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ curl -X POST -H "Content-Type: application/json" -d '{"book_name": "cloud computing", "book_author": "unknown", "isbn": "123456"}' http://cs571.project.com/bookshelf/book
{
  "message": "Book saved successfully!"
}
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
[haile96456@cloudshell:~ (cs570-big-data-analytics)$ curl http://cs571.project.com/bookshelf/books
  {
    "Book Author": "unknown",
    "Book Name": "cloud computing",
    "ISBN": null,
    "id": "66adee3fe1b2a7796453241e"
  }
]
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

20. Update a book

curl -X PUT -H "Content-Type: application/json" -d '{"book_name": "UpdatedBook Name", "book_author": "Updated Author", "isbn": "Updated ISBN"}'
http://cs571.project.com/bookshelf/book/66adee3fe1b2a7796453241e

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ curl -X PUT -H "Content-Type: application/json" -d '{"book_name": "UpdatedBook Name", "book_author": "Updated Autho
r", "isbn": "Updated ISBN"}' http://cs571.project.com/bookshelf/book/66adee3fe1b2a7796453241e
{
  "message": "Book updated successfully!"
}
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ curl http://cs571.project.com/bookshelf/books
[
  {
    "Book Author": "Updated Author",
    "Book Name": "UpdatedBook Name",
    "ISBN": null,
    "id": "66adee3fe1b2a7796453241e"
  }
]
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

21. Delete a book

curl -X DELETE
http://cs571.project.com/bookshelf/book/66adee3fe1b2a7796453241e

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ curl -X DELETE http://cs571.project.com/bookshelf/book/66adee3fe1b2a7796453241e
{
  "message": "Book deleted successfully!"
}
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ curl http://cs571.project.com/bookshelf/books
[]
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

These commands test the functionality of the deployed applications by performing various CRUD operations via the REST API endpoints. By following these steps, you can deploy and expose your Student Server and Bookshelf applications on the same domain with different paths using Kubernetes and Nginx Ingress.