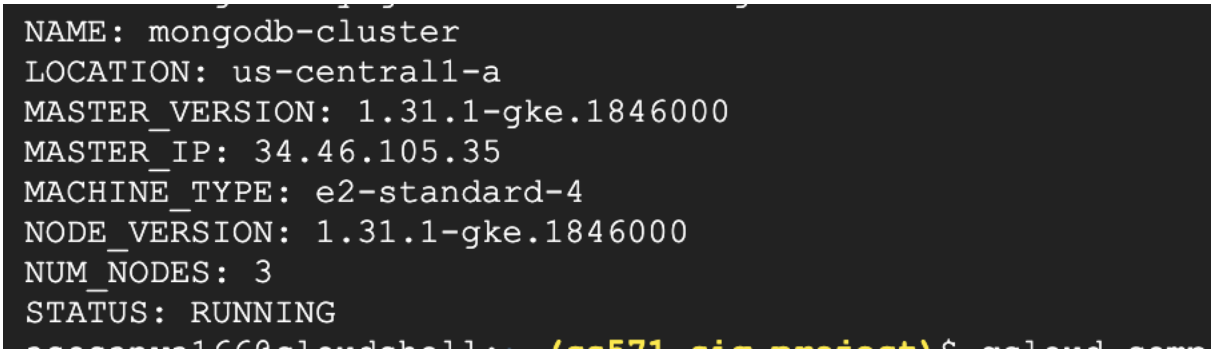
Sosanya Adebayo **19944** **CS571**

Github: https://github.com/Junehuit/Cloud-Computing/releases/tag/v1.0.0 or https://github.com/Junehuit/Cloud-Computing/blob/main/MongoDB%20%2B%20Python%20Flask%20Web%20Framework%20%2B%20REST%20API%20%2B%20GKE

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

Create a cluster as usual on GKE
gcloud container clusters create kubia --num-nodes=1 --machine-type=e2-micro --zone=us-west1-b



Let's create a Persistent Volume first

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

Now create a mongodb deployment with this yaml filec

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
      - name: mongodb
        image: mongo:latest
        ports:
        - containerPort: 27017
        volumeMounts:
        - name: mongo-storage
          mountPath: /data/db
      volumes:
      - name: mongo-storage
        persistentVolumeClaim:
          claimName: mongo-pvc
```

kubectl apply -f mongodb-deployment.yaml

```
asosanya166@cloudshell:~ (cs571-sig-project)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
```

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

kubectl get pods

```
asosanya166@cloudshell:~ (cs571-sig-project)$ kubectl get pods
NAME                                 READY   STATUS    RESTARTS   AGE
mongodb-deployment-5fd5555667-wvwfz   0/1     Pending   0          2m27s
```

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

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

kubectl apply -f mongodb-service.yaml

```
asosanya166@cloudshell:~ (cs571-sig-project)$ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
```

Wait couple of minutes, and check if the service is up

kubectl get svc

```
asosanya166@cloudshell:~ (cs571-sig-project)$ kubectl get svc
NAME              TYPE           CLUSTER-IP       EXTERNAL-IP     PORT(S)           AGE
kubernetes        ClusterIP      34.118.224.1     <none>          443/TCP           31m
mongodb-service   LoadBalancer   34.118.236.198   34.29.128.228   27017:30789/TCP   49s
```

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

kubectl exec -it mongodb-deployment-6675d4fb5-slgkq -- bashNow you are inside the

```
asosanya166@cloudshell:~ (cs571-sig-project)$ kubectl exec -it mongodb-deployment-5fd5555667-wvwfz -- bash
root@mongodb-deployment-5fd5555667-wvwfz:/#
```

mongodb deployment pod

```
asosanya166@cloudshell:~ (cs571-sig-project)$ kubectl apply -f mongo-pvc.yaml
persistentvolumeclaim/mongo-pvc created
```

Try

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

External-IP

```
MongoDB shell version v4.4.29
connecting to: mongodb://34.29.128.228:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("c78a1e09-fcbf-4ecd-aa9c-deaeaa464e03") }
MongoDB server version: 8.0.3
WARNING: shell and server versions do not match
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        https://docs.mongodb.com/
Questions? Try the MongoDB Developer Community Forums
        https://community.mongodb.com
---
The server generated these startup warnings when booting:
        2024-11-15T02:44:30.794+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-f
stem
        2024-11-15T02:44:31.504+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
        2024-11-15T02:44:31.505+00:00: For customers running the current memory allocator, we suggest changing the contents of the following sysfsFile
        2024-11-15T02:44:31.505+00:00:         allocator: tcmalloc-google
        2024-11-15T02:44:31.505+00:00:         sysfsFile: /sys/kernel/mm/transparent_hugepage/enabled
        2024-11-15T02:44:31.505+00:00:         currentValue: never
        2024-11-15T02:44:31.505+00:00:         desiredValue: always
        2024-11-15T02:44:31.505+00:00: For customers running the current memory allocator, we suggest changing the contents of the following sysfsFile
        2024-11-15T02:44:31.505+00:00:         allocator: tcmalloc-google
        2024-11-15T02:44:31.505+00:00:         sysfsFile: /sys/kernel/mm/transparent_hugepage/defrag
        2024-11-15T02:44:31.505+00:00:         currentValue: madvise
        2024-11-15T02:44:31.505+00:00:         desiredValue: defer+madvise
        2024-11-15T02:44:31.505+00:00: We suggest setting the contents of sysfsFile to 0.
        2024-11-15T02:44:31.505+00:00:         sysfsFile: /sys/kernel/mm/transparent_hugepage/khugepaged/max_ptes_none
        2024-11-15T02:44:31.505+00:00:         currentValue: 511
        2024-11-15T02:44:31.505+00:00: Your system has glibc support for rseq built in, which is not yet supported by tcmalloc-google and has critical performance implica
 Please set the environment variable GLIBC_TUNABLES=glibc.pthread.rseq=0
        2024-11-15T02:44:31.505+00:00: vm.max_map_count is too low
        2024-11-15T02:44:31.505+00:00:         currentValue: 65530
        2024-11-15T02:44:31.505+00:00:         recommendedMinimum: 1677720
        2024-11-15T02:44:31.505+00:00:         maxConns: 838860
---
>
```

Type exit to exit mongodb and back to our google console

```
> exit
bye
asosanya166@cloudshell:~ (cs571-sig-project)$
```

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

```
  GNU nano 7.2                                                  mongo_insert.js *
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://34.29.128.228:27017/mydb"; // Make sure to replace with your actual external IP and database name

// 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");  // Make sure to replace "studentdb" with your actual database name
  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 documents
  db.collection("students").insertMany(docs, function(err, res) {
    if (err) throw err;
    console.log(res.insertedCount);
    client.close();
  });

  // Query for one document
  db.collection("students").findOne({ "student_id": 11111 }, function(err, result) {
    if (err) throw err;
    console.log(result);
  });
});
^C
asosanya166@cloudshell:~ (cs571-sig-project)$ nano mongo_insert.js
asosanya166@cloudshell:~ (cs571-sig-project)$ node mongo_insert.js
```

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

var http = require('http');

var url = require('url');

var mongodb = require('mongodb');

// Use environment variables for MongoDB connection details

const {

MONGO_URL = '34.29.128.228', // MongoDB external IP (replace with your IP)

MONGO_DATABASE = 'studentdb'

```
} = process.env;

var MongoClient = mongodb.MongoClient;
var uri = `mongodb://${MONGO_URL}:27017/${MONGO_DATABASE}`;  // MongoDB URI

// Log the connection string for debugging
console.log(`MongoDB URI: ${uri}`);

// Create the HTTP server
var server = http.createServer(function (req, res) {
  var parsedUrl = url.parse(req.url, true);  // Parse the incoming request URL
  var student_id = parseInt(parsedUrl.query.student_id);  // Extract the student_id from the query string

  if (/^\/api\/score/.test(req.url)) {
    // Connect to the MongoDB server
    MongoClient.connect(uri, { useNewUrlParser: true, useUnifiedTopology: true }, function(err, client) {
      if (err) {
        console.error('MongoDB connection error:', err);
        res.writeHead(500);
        res.end('Database connection failed\n');
        return;
      }

      var db = client.db(MONGO_DATABASE);  // Access the database
      db.collection("students").findOne({ "student_id": student_id }, function(err, student) {
```

```
      if (err) {
        console.error('Error fetching student:', err);
        res.writeHead(500);
        res.end('Error fetching student data\n');
        return;
      }

      if (student) {
        // If the student is found, return their data
        res.writeHead(200, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify({
          student_id: student.student_id,
          student_name: student.student_name,
          student_score: student.grade
        }) + '\n');
      } else {
        // If the student is not found
        res.writeHead(404);
        res.end("Student Not Found\n");
      }
    });
  });
} else {
  // Handle invalid URLs
  res.writeHead(404);
  res.end("Wrong URL, please try again\n");
```

```
  }
});


// Start the server on port 8080
server.listen(8080, () => {
  console.log('Server is running at http://localhost:8080');
});
```

```
asosanya166@cloudshell:~ (cs571-sig-project)$ nano studentServer.js
asosanya166@cloudshell:~ (cs571-sig-project)$ node studentServer.js
MongoDB URI: mongodb://34.29.128.228:27017/studentdb
Server is running at http://localhost:8080
```

Create Dockerfile

```
# Use Node.js version 7 as base image
FROM node14

# Copy the studentServer.js file into the container
ADD studentServer.js /studentServer.js

# Install MongoDB driver
RUN npm install mongodb

# Run the studentServer.js script when the container starts
ENTRYPOINT ["node", "/studentServer.js"]
```

Build the studentserver docker image

docker build -t yourdockerhubID/studentserver .

Make sure there is no error

```
asosanya166@cloudshell:~ (cs571-sig-project)$                junehuit/studentserver .

[+] Building 0.8s (9/9) FINISHED                                 docker:default
 => [internal] load build definition from Dockerfile                      0.0s
 => => transferring dockerfile: 334B                                      0.0s
 => [internal] load metadata for docker.io/library/node:14                0.6s
 => [auth] library/node:pull token for registry-1.docker.io               0.0s
 => [internal] load .dockerignore                                         0.0s
 => => transferring context: 2B                                           0.0s
 => [internal] load build context                                         0.0s
 => => transferring context: 38B                                          0.0s
 => [1/3] FROM docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c59  0.0s
 => CACHED [2/3] ADD studentServer.js /studentServer.js                   0.0s
 => CACHED [3/3] RUN npm install mongodb                                  0.0s
 => exporting to image                                                    0.0s
 => => exporting layers                                                   0.0s
 => => writing image sha256:88f399944c7514297ffde42aa7506a9fde83d43193919  0.0s
```

Push the docker image

docker push yourdockerhubID/studentserver

```
asosanya166@cloudshell:~ (cs571-sig-project)$             junehuit/studentserver:latest

The push refers to repository [docker.io/junehuit/studentserver]
2d0b9beb7758: Pushed
ec01d36b311c: Pushed
0d5f5a015e5d: Mounted from library/node
3c777d951de2: Mounted from library/node
f8a91dd5fc84: Mounted from library/node
cb81227abde5: Mounted from library/node
e01a454893a9: Mounted from library/node
c45660adde37: Mounted from library/node
fe0fb3ab4a0f: Mounted from library/node
f1186e5061f2: Mounted from library/node
b2dba7477754: Mounted from library/node
latest: digest: sha256:ec2326d69359eba70a2b22e3c828a9f58ca3b3fb5d343b4dfccfeb5981958f23 size: 2634
asosanya166@cloudshell:~ (cs571-sig-project)$
```

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

from flask import Flask, request, jsonify

from flask_pymongo import PyMongo

from bson.objectid import ObjectId

import socket

import os

Initialize Flask application

```
app = Flask(__name__)


# MongoDB URI setup with environment variables
app.config["MONGO_URI"] = "mongodb://" + os.getenv("MONGO_URL") + "/" +
os.getenv("MONGO_DATABASE")
app.config['JSONIFY_PRETTYPRINT_REGULAR'] = True


# Initialize PyMongo
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_many({"_id": ObjectId(id)}, {"$set": {
    "book_name": data['book_name'],
    "book_author": data["book_author"],
    "ISBN": data["isbn"]
  }})

  if response.matched_count:
    message = "Book updated successfully!"
```

```
    else:
        message = "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)})

    if response.deleted_count:
        message = "Book deleted successfully!"
    else:
        message = "No book found!"

    return jsonify(message=message)

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

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

Create a Dockerfile

```
FROM python:3.9-slim

# Install dependencies

RUN pip install --upgrade pip

COPY requirements.txt /app/requirements.txt

RUN pip install -r /app/requirements.txt

# Add the Flask application code

COPY bookshelf.py /app/bookshelf.py

# Set environment variables

ENV MONGO_URL=<your_mongo_url>

ENV MONGO_DATABASE=<your_mongo_database>

# Expose the port the app runs on

EXPOSE 5000

# Run the Flask app

CMD ["python", "/app/bookshelf.py"]
```

Build the bookshelf app into a docker image

docker build -t junehuit/studentserver

```
asosanya166@cloudshell:~ (cs571-sig-project)$ docker build -t junehuit/studentserver .
[+] Building 14.7s (10/10) FINISHED                                 docker:default
 => [internal] load build definition from Dockerfile                          0.0s
 => => transferring dockerfile: 639B                                          0.0s
 => [internal] load metadata for docker.io/library/python:3.8-slim            1.0s
 => [auth] library/python:pull token for registry-1.docker.io                 0.0s
 => [internal] load .dockerignore                                             0.0s
 => => transferring context: 2B                                               0.0s
 => [1/4] FROM docker.io/library/python:3.8-slim@sha256:1d52838af602b4b5a     4.0s
 => => resolve docker.io/library/python:3.8-slim@sha256:1d52838af602b4b5a     0.0s
 => => sha256:1d52838af602b4b5a831beb13a0e4d073280665ea 10.41kB / 10.41kB     0.0s
 => => sha256:314bc2fb0714b7807bf5699c98f0c73817e579799f2 1.75kB / 1.75kB     0.0s
 => => sha256:b5f62925bd0f63f48cc8acd5e87d0c3a07e2f229cd2 5.25kB / 5.25kB     0.0s
 => => sha256:302e3ee498053a7b5332ac79e8efebec16e900289 29.13MB / 29.13MB     0.5s
 => => sha256:030d7bdc20a63e3d22192b292d006a69fa3333949f5 3.51MB / 3.51MB     0.3s
 => => sha256:a3f1dfe736c5f959143f23d75ab522a60be2da902 14.53MB / 14.53MB     0.6s
 => => sha256:3971691a363796c39467aae4cdce6ef773273fe6bfc6715 248B / 248B     0.4s
 => => extracting sha256:302e3ee498053a7b5332ac79e8efebec16e900289fc1ecd1     1.9s
 => => extracting sha256:030d7bdc20a63e3d22192b292d006a69fa3333949f536d62     0.2s
 => => extracting sha256:a3f1dfe736c5f959143f23d75ab522a60be2da902efac236     0.9s
 => => extracting sha256:3971691a363796c39467aae4cdce6ef773273fe6bfc67154     0.0s
 => [internal] load build context                                             1.2s
 => => transferring context: 46.90MB                                          1.2s
 => [2/4] WORKDIR /app                                                        1.0s
 => [3/4] COPY . /app                                                         0.5s
 => [4/4] RUN pip install --no-cache-dir -r requirements.txt                  7.6s
 => exporting to image                                                        0.5s
 => => exporting layers                                                       0.5s
 => => writing image sha256:1a187fa949070c23e631222bce75e69fbf998b443e3db     0.0s
 => => naming to docker.io/junehuit/studentserver                             0.0s
asosanya166@cloudshell:~ (cs571-sig-project)$
```

Push the docker image to your dockerhub

docker push yourdockerhubID/studentserver

```
asosanya166@cloudshell:~ (cs571-sig-project)$ docker push yourdockerhubID/studentserver
Using default tag: latest(cs571-sig-project)$ docker push yourdockerhubID/studentserver
The push refers to repository [yourdockerhubID/studentserver]
                                                                        r
Using default tag: latest
The push refers to repository [docker.io/junehuit/studentserver]
868467782454: Pushed
38abc81965d5: Pushed
c56844d5ba03: Pushed
d2a2207b52a4: Mounted from library/python
5d2d143f3d7f: Mounted from library/python
c3772b569c3a: Mounted from library/python
8d853c8add5d: Mounted from library/python
latest: digest: sha256:482078411887f69097554d4c8a379dc9224de5a6d6b6c092d80f449a0c7a3e1d size: 1787
asosanya166@cloudshell:~ (cs571-sig-project)$
```

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

Create a file named studentserver-configmap.yaml

```
  GNU nano 7.2                  studentserver-configmap.yaml *
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  MONGO_URL: "34.29.128.228"
  MONGO_DATABASE: "mydb"
```

Create a file named bookshelf-configmap.yaml

```
  GNU nano 7.2                          bookshelf-cc
apiVersion: v1
kind: ConfigMap
metadata:
  name: bookshelf-config
data:
  MONGO_URL: "34.29.128.228"
  MONGO_DATABASE: "mydb"
```

Verify

```
asosanya166@cloudshell:~ (cs571-sig-project)$ nano studentserver-configmap.yaml
asosanya166@cloudshell:~ (cs571-sig-project)$ nano bookshelf-configmap.yaml
asosanya166@cloudshell:~ (cs571-sig-project)$ kubectl apply -f studentserver-configmap.yaml
                                              kubectl apply -f studentserver-configmap.yaml
asosanya166@cloudshell:~ (cs571-sig-project)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created-project)$ kubectl apply -f bookshelf-configmap.yaml
asosanya166@cloudshell:~ (cs571-sig-project)$ kubectl get configmap
NAME                 DATA   AGE
bookshelf-config     2      26s
kube-root-ca.crt     1      4h58m
studentserver-config 2      41s
```

Expose 2 application using ingress with Nginx, so we can put them on the same

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: zhou19539/studentserver  # Replace with your DockerHub image
        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
```

Create bookshelf-deployment.yaml

```
  GNU nano 7.2                                                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: adebayo19944/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
```

Create sutdentserver-service.yaml

```
  GNU nano 7.2
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
```

Create bookshelf-service.yaml

```
  GNU nano 7.2
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
```

Start minikube

minikube start

```
asosanya166@cloudshell:~ (cs571-sig-project)$ 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% 296.12
    > gcr.io/k8s-minikube/kicbase...:  487.90 MiB / 487.90 MiB  100.00% 104.74
* 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
```

Start Ingress

minikube addons enable ingress

```
asosanya166@cloudshell:~ (cs571-sig-project)$ 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/kube-webhook-certgen:v1.4.3
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.3
  - Using image registry.k8s.io/ingress-nginx/controller:v1.11.2
* Verifying ingress addon...
* The 'ingress' addon is enabled
```

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

```
asosanya166@cloudshell:~ (cs571-sig-project)$ kubectl apply -f studentserver-deployment.yaml
deployment.apps/web created
asosanya166@cloudshell:~ (cs571-sig-project)$ kubectl apply -f studentserver-configmap.yaml
configmap/studentserver-config created
```

```
asosanya166@cloudshell:~ (cs571-sig-project)$ kubectl apply -f sutdentserver-service.yaml
service/web created
```

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

kubectl apply -f bookshelf-deployment.yaml

kubectl apply -f bookshelf-configmap.yaml

kubectl apply -f bookshelf-service.yaml

```
asosanya166@cloudshell:~ (cs571-sig-project)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created
asosanya166@cloudshell:~ (cs571-sig-project)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
asosanya166@cloudshell:~ (cs571-sig-project)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
```

Check if all the pods are running correctly

kubectl get pods

```
asosanya166@cloudshell:~ (cs571-sig-project)$ kubectl get pods
NAME                                    READY   STATUS             RESTARTS        AGE
bookshelf-deployment-c49fd46fb-fvb72    0/1     CrashLoopBackOff   9  (71s ago)    22m
web-77b8979857-7cbw8                    1/1     Running            0               50m
```

Create an ingress service yaml file called studentservermongoIngress.yaml

```
  GNU nano 7.2                                                studentservermongoIngress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: server
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
  rules:
  - host: cs571.project.com  # Replace with your desired domain
    http:
      paths:
      - path: /studentserver(/|$)(.*)
        pathType: Prefix
        backend:
          service:
            name: web  # Name of the studentserver service
            port:
              number: 8080  # Port to which studentserver is exposed
      - path: /bookshelf(/|$)(.*)
        pathType: Prefix
        backend:
          service:
            name: bookshelf-service  # Name of the bookshelf service
            port:
              number: 5000  # Port to which bookshelf is exposed
```

Create the ingress service using the above yaml file

```
asosanya166@cloudshell:~ (cs571-sig-project)$ nano studentservermongoIngress.yaml
asosanya166@cloudshell:~ (cs571-sig-project)$ kubectl apply -f studentservermongoIngress.yaml
ingress.networking.k8s.io/server configured
```

Check if ingress is running

kubectl get ingress

Please wait until you see the Address, then move forward

```
asosanya166@cloudshell:~ (cs571-sig-project)$ kubectl get ingress
NAME     CLASS   HOSTS               ADDRESS        PORTS   AGE
server   nginx   cs571.project.com   192.168.49.2   80      16m
```

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

```
#
# 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-178264220635-default
34.29.128.228 cs571.project.com
```

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

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

```
acosanyal66@cloudshell:~ (cs571-sig-project)$ curl cs571.project.com/studentserver/api/score?student_id=11111
{"_id":"605a6b49c3a15527de9d0f9b","student_id":11111,"student_name":"Bruce Lee","grade":84}
acosanyal66@cloudshell:~ (cs571-sig-project)$ curl cs571.project.com/studentserver/api/score?student_id=22222
{"_id":"605a6b49c3a15527de9d0f9c","student_id":22222,"student_name":"Jackie Chen","grade":93}
acosanyal66@cloudshell:~ (cs571-sig-project)$ curl cs571.project.com/studentserver/api/score?student_id=33333
{"_id":"605a6b49c3a15527de9d0f9d","student_id":33333,"student_name":"Jet Li","grade":88}
```

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

```
acosanyal66@cloudshell:~ (cs571-sig-project)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123",
    "id": "605d1ba7d40f50a395651765"
  }
]
```

Add a book

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

```
escsanya166@cloudshell:  (cs571-sig-project)$ curl -X POST -d "{\"book_name\": \"cloud computing\",\"book_author\": \"unkown\", \"isbn\": \"12345
6\" }" http://cs571.project.com/bookshelf/book
{
  "message": "Task saved successfully!"
}
```

```
escsanya166@cloudshell:  (cs571-sig-project)$curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123",
    "id": "605d1ba7d40f50a395651765"
  },
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "605d2fffbd09c0d7f8cf1f93"
```

Update a book

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

```
escsanya166@cloudshell:  (cs571-sig-project)$ curl -X PUT -d "{\"book_name\": \"123\",\"book_author\": \"test\", \"isbn\": \"123updated\" }"
://cs571.project.com/bookshelf/book/605d1ba7d40f50a395651765
{
  "message": "Task updated successfully!"
}
```

```
escsanya166@cloudshell:  (cs571-sig-project)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123updated",
    "id": "605d1ba7d40f50a395651765"
  },
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "605d2fffbd09c0d7f8cf1f93"
  }
]
```

Delete a book

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

```
escsanya166@cloudshell:  (cs571-sig-project)$ curl -X DELETE cs571.project.com/bookshelf/book/605d1ba7d40f50a395651765
{
  "message": "Task deleted successfully!"
}
```

```
escsanya166@cloudshell:  (cs571-sig-project)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "605d2fffbd09c0d7f8cf1f93"
  }
]
```