Student ID: 19619

CS571 - Cloud computing Infrastructure

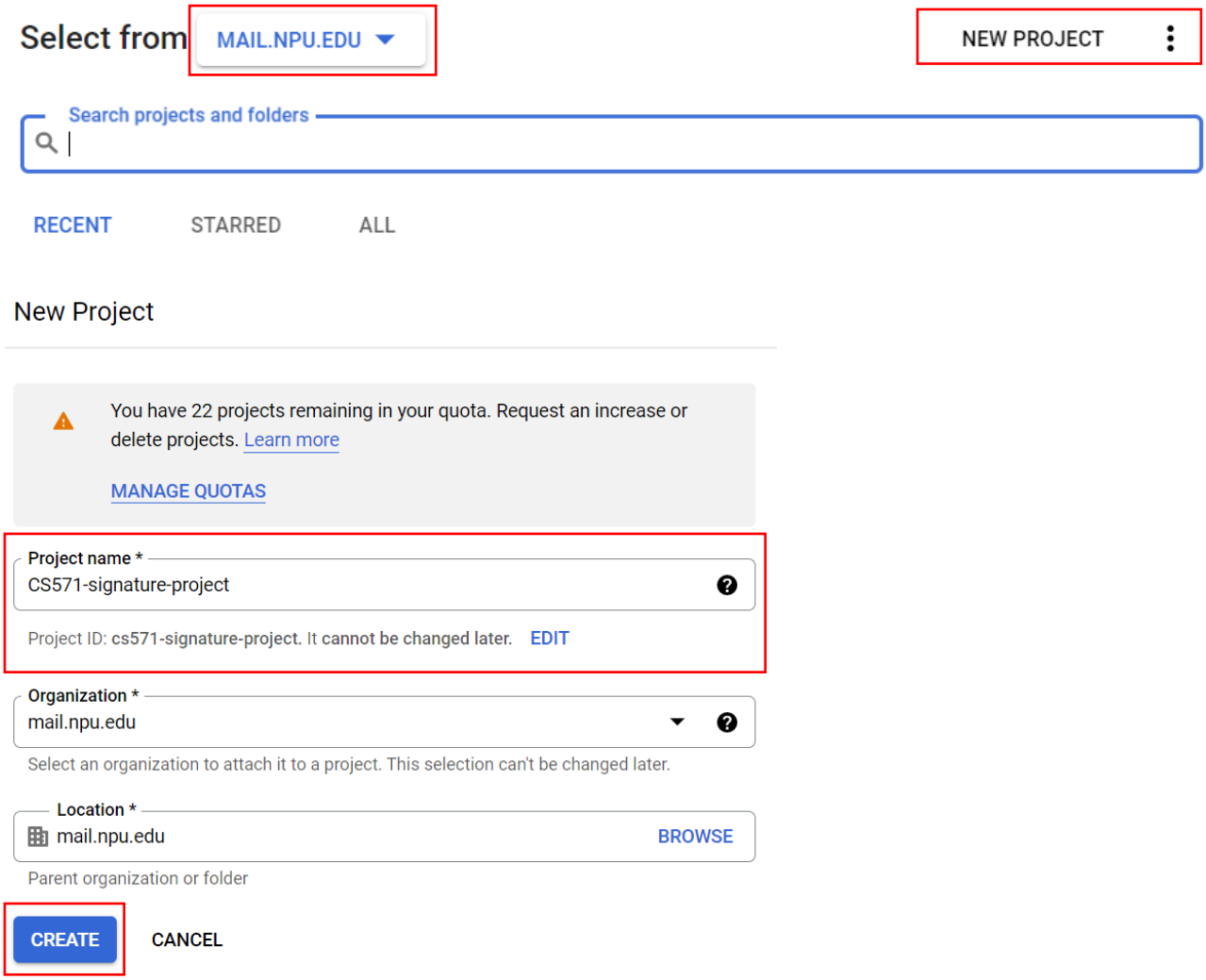
Week - 10 Signature Project

The project has two applications are running, the first **student record** application running with Node.js + MongoDB + GKE and the second one, **bookstore** application using technologies, which are MongoDB + Python Flask Web Framework + REST API + GKE.

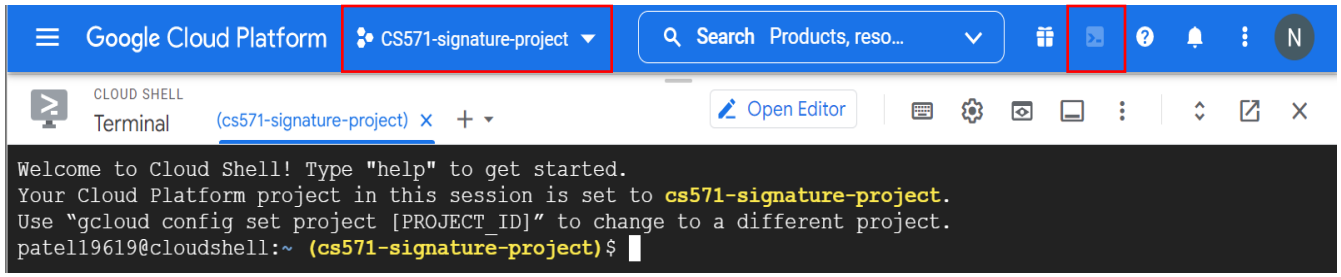
The following are the procedure to configure the GKE cluster to hosting two applications on different pods.

Step 1 Create a new project on GCP for this project work.

1. Create a new project on GCP.



2. Select the project that you created and open the GCP terminal windows.

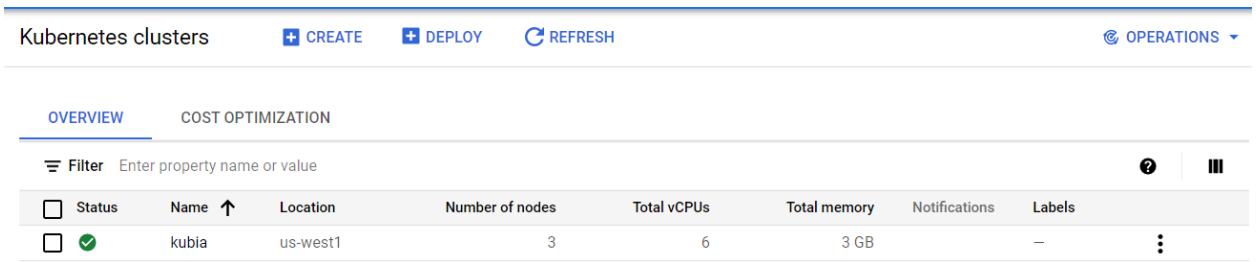


We have organized the space on GCP by creating the new project for this signature project work.

Student record application configuration steps are following:

Step 1 Create and launch the MongoDB database with using Persistent Volume on GKE

1. First, check the **kubia** cluster you have launched in homework 4 is running properly.



If not, launch the **kubia** cluster with following command:

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

```
NAME: kubia
LOCATION: us-west1
MASTER_VERSION: 1.21.9-gke.1002
MASTER_IP: 35.227.137.24
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.21.9-gke.1002
NUM_NODES: 3
STATUS: RUNNING
```

2. Let's check the running node.

kubectl get nodes

```
patel19619@cloudshell:~ (cs571-signature-project)$ kubectl get nodes
NAME                                          STATUS   ROLES    AGE     VERSION
gke-kubia-default-pool-21303fe9-r2w4          Ready    <none>   3m38s   v1.21.9-gke.1002
gke-kubia-default-pool-6e539c26-hm2b          Ready    <none>   3m38s   v1.21.9-gke.1002
gke-kubia-default-pool-839f76f9-vlz5          Ready    <none>   3m37s   v1.21.9-gke.1002
```

3. Create a GCE Persistent Disk **mongodb** with size of 10GiB.

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

```
patel19619@cloudshell:~ (cs571-signature-project)$ gcloud compute disks create --size=10GiB --zone=us-west1-a mon
godb
WARNING: You have selected a disk size of under [200GB]. This may result in poor I/O performance.For more informa
tion, see: https://developers.google.com/compute/docs/disks#performance.
Created [https://www.googleapis.com/compute/v1/projects/cs571-signature-project/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
```

4. Check the **mongodb** disk is created and available.

   gcloud compute disks list

```
patel19619@cloudshell:~ (cs571-signature-project)$ gcloud compute disks list
NAME: gke-kubia-default-pool-21303fe9-r2w4
LOCATION: us-west1-a
LOCATION_SCOPE: zone
SIZE_GB: 100
TYPE: pd-standard
STATUS: READY

NAME: mongodb
LOCATION: us-west1-a
LOCATION_SCOPE: zone
SIZE_GB: 10
TYPE: pd-standard
STATUS: READY

NAME: gke-kubia-default-pool-839f76f9-vlz5
LOCATION: us-west1-b
LOCATION_SCOPE: zone
SIZE_GB: 100
TYPE: pd-standard
STATUS: READY

NAME: gke-kubia-default-pool-6e539c26-hm2b
LOCATION: us-west1-c
LOCATION_SCOPE: zone
SIZE_GB: 100
TYPE: pd-standard
STATUS: READY
```

5. Create a new directory called **mongodb** and go to the created **mongodb** directory.

   mkdir mongodb

   cd mongodb

```
patel19619@cloudshell:~ (cs571-signature-project)$ mkdir mongodb
patel19619@cloudshell:~ (cs571-signature-project)$ cd mongodb
```

6. Again, create a new directory inside in **mongodb** directory called **yaml** and go to **yaml** directory.

   mkdir yaml

   cd yaml

```
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ mkdir yaml
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ cd yaml
patel19619@cloudshell:~/mongodb/yaml (cs571-signature-project)$
```

7. Create a mongodb deployment with mongodb-deployment.yaml file.

   vim mongodb-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
spec:
  selector:
    matchLabels:
      app: mongodb
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
        - image: mongo
          name: mongo
          ports:
            - containerPort: 27017
          volumeMounts:
            - name: mongodb-data
              mountPath: /data/db
      volumes:
        - name: mongodb-data
          gcePersistentDisk:
            pdName: mongodb
            fsType: ext4
```

8. Create the **mongodb-deployment** with using **mongodb-deployment.yaml** file.

   kubectl apply -f mongodb-deployment.yaml

```
patel19619@cloudshell:~/mongodb/yaml (cs571-signature-project)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
```

9. Check the mongodb-deployment pod has been successfully created and running.

   kubectl get pods

```
patel19619@cloudshell:~/mongodb/yaml (cs571-signature-project)$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
mongodb-deployment-57dc68b4bd-ckqp9   1/1     Running   0          3m29s
```

10. Create a **mongodb-service.yaml** for the mongoDB, so it can be access from outside the cluster.

    vim mongodb-service.yaml

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

11. Create the **mongodb-service.**

    kubectl apply -f mongodb-service.yaml

```
patel19619@cloudshell:~/mongodb/yaml (cs571-signature-project)$ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
```

12. Check the **mongodb-service** is created and running.

    kubectl get svc

```
patel19619@cloudshell:~/mongodb/yaml (cs571-signature-project)$ kubectl get svc
NAME              TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)          AGE
kubernetes        ClusterIP      10.36.0.1      <none>          443/TCP          43m
mongodb-service   LoadBalancer   10.36.13.174   34.127.12.117   27017:30652/TCP  35s
```

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

    kubectl exec -it mongodb-deployment-replace-with-your-pod-name -- bash

```
patel19619@cloudshell:~/mongodb/yaml (cs571-signature-project)$ kubectl exec -it mongodb-deployment-57dc68b4bd-ckqp9
 -- bash
root@mongodb-deployment-57dc68b4bd-ckqp9:/#
```

    You are now inside the **mongodb-deployment pod**

14. To run the mongoDB database

    mongo your-External-IP

```
root@mongodb-deployment-57dc68b4bd-ckqp9:/# mongo 34.127.12.117
MongoDB shell version v5.0.6
connecting to: mongodb://34.127.12.117:27017/test?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("e21f0633-82de-4814-9d81-a75711cd9b15") }
MongoDB server version: 5.0.6
================
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
================
Welcome to the MongoDB shell.
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:
        2022-03-22T00:50:50.261+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage
engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
        2022-03-22T00:50:51.108+00:00: Access control is not enabled for the database. Read and write access to data
 and configuration is unrestricted
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).

        The monitoring data will be available on a MongoDB website with a unique URL accessible to you
        and anyone you share the URL with. MongoDB may use this information to make product
        improvements and to suggest MongoDB products and deployment options to you.

        To enable free monitoring, run the following command: db.enableFreeMonitoring()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
>
```

15. Type exit to exit mongoDB database and back to google console terminal.

exit

```
> exit
bye
root@mongodb-deployment-57dc68b4bd-ckqp9:/# exit
exit
patel19619@cloudshell:~/mongodb/yaml (cs571-signature-project)$
```

16. Go to mongodb directory from yaml directory

cd ..

```
patel19619@cloudshell:~/mongodb/yaml (cs571-signature-project)$ cd ..
patel19619@cloudshell:~/mongodb (cs571-signature-project)$
```

17. To insert the student records into the mongoDB database

First install mongodb module in our configuration environment

npm install mongodb

```
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ npm install mongodb

added 20 packages, and audited 21 packages in 2s

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

found 0 vulnerabilities
npm notice
npm notice New patch version of npm available! 8.5.3 -> 8.5.5
npm notice Changelog: https://github.com/npm/cli/releases/tag/v8.5.5
npm notice Run npm install -g npm@8.5.5 to update!
npm notice
```

Then type **node**

```
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ node
Welcome to Node.js v12.14.1.
Type ".help" for more information.
>
```

18. Enter the following program line by line.

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://EXTERNAL-IP/mydb"
// Connect to the db
MongoClient.connect(url, { useNewUrlParser: true, useUnifiedTopology: true },
    function (err, client) {
        if (err)
```

```
            throw err;
        // create a document to be inserted
        var db = client.db("studentdb");
        const docs = [
            { student_id: 11111, student_name: "Bruce Lee", grade: 84 },
            { student_id: 22222, student_name: "Jackie Chen", grade: 93 },
            { student_id: 33333, student_name: "Jet Li", grade: 88 }
        ]
        db.collection("students").insertMany(docs, function (err, res) {
            if (err) throw err;
            console.log(res.insertedCount);

        });
        db.collection("students").findOne({ "student_id": 11111 },
            function (err, result) {
                if (err) throw err;
                console.log(result);
                client.close();
            });
    });
```

19. If Everything is correct, you should see the following result, in the result 3 means three records was inserted and query result of student_id=11111.

```
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ node
Welcome to Node.js v12.14.1.
Type ".help" for more information.
> var MongoClient = require('mongodb').MongoClient;
undefined
> var url = "mongodb://35.227.156.131/mydb"
undefined
> // Connect to the db
undefined
> MongoClient.connect(url, { useNewUrlParser: true, useUnifiedTopology: true },
...     function (err, client) {
.....         if (err)
.....             throw err;
.....         // create a document to be inserted
.....         var db = client.db("studentdb");
.....         const docs = [
.....             { student_id: 11111, student_name: "Bruce Lee", grade: 84 },
.....             { student_id: 22222, student_name: "Jackie Chen", grade: 93 },
.....             { student_id: 33333, student_name: "Jet Li", grade: 88 }
.....         ]
.....         db.collection("students").insertMany(docs, function (err, res) {
.......             if (err) throw err;
.......             console.log(res.insertedCount);
.......
.......         });
.....         db.collection("students").findOne({ "student_id": 11111 },
.......             function (err, result) {
.........                 if (err) throw err;
.........                 console.log(result);
.........                 client.close();
.........             });
.....     });
undefined
> 3
{
  _id: new ObjectId("62461c0eefe5376562e4c092"),
  student_id: 11111,
  student_name: 'Bruce Lee',
  grade: 84
}
```

Step 2 Modify our studentServer.js to get records from MongoDB database and deploy to GKE

1. In **mongodb** directory create a **studentServer-record.js** with following command:

   vim studentServer-record.js

   Save the above code, which we have ran in the node

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://34.127.12.117:27017/mydb"
// Connect to the db


MongoClient.connect(url,{ useNewUrlParser: true, useUnifiedTopology: true }, function(err, client){
    if (err)
        throw err;

          // create a document to be inserted
    var db = client.db("studentdb");
    const docs = [
            { student_id: 11111, student_name: "Bruce Lee", grade: 84},
            { student_id: 22222, student_name: "Jackie Chen", grade: 93 },
            { student_id: 33333, student_name: "Jet Li", grade: 88}
    ]
    db.collection("students").insertMany(docs, function(err, res){
        if(err) throw err;
        console.log(res.insertedCount);
        client.close();
    });
    db.collection("students").findOne({"student_id": 11111},
    function(err, result){
        console.log(result);
    });

});
```

2. Now, Create a **studentServer.js** with following command:

   vim studentServer.js

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

var MongoClient = mongodb.MongoClient;
var uri = `mongodb://${MONGO_URL}/${MONGO_DATABASE}`;
// Connect to the db
console.log(uri);

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

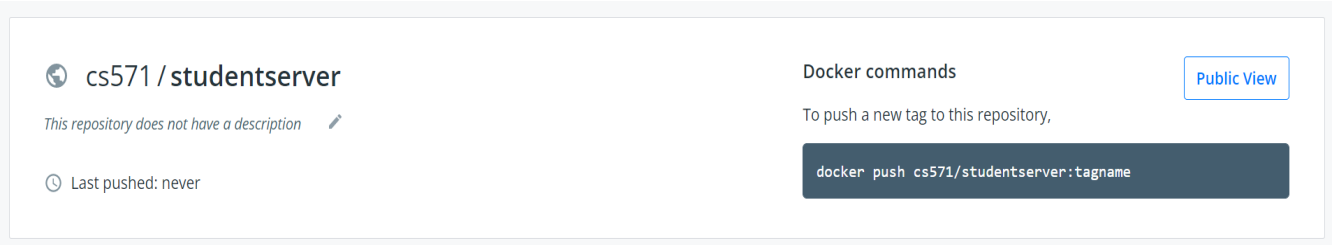
```
  var result;
  // req.url = /api/score?student_id=11111
  var parsedUrl = url.parse(req.url, true);

  var student_id = parseInt(parsedUrl.query.student_id);

  // match req.url with the string /api/score
  if (/^\/api\/score/.test(req.url)) {
    // e.g., of student_id 1111

    MongoClient.connect(uri,{ useNewUrlParser: true, useUnifiedTopology: true },
function(err, client){
      if (err)
           throw err;
      var db = client.db("studentdb");
      db.collection("students").findOne({"student_id":student_id}, (err, student)
=> {
     if(err)
    throw new Error(err.message, null);

     if (student) {
      res.writeHead(200, { 'Content-Type': 'application/json' })
      res.end(JSON.stringify(student)+ '\n')
       }else {
      res.writeHead(404);
    res.end("Student Not Found \n");
     }
  });
    });
  } else {
  res.writeHead(404);
  res.end("Wrong url, please try again\n");
  }
});
server.listen(8080);
```

3. Create a Dockerfile.

vim Dockerfile

```
FROM node:14
ADD studentServer.js /studentServer.js
RUN npm install mongodb
ENTRYPOINT ["node", "studentServer.js"]
```

4. Create the docker hub **studentServer** repository for this student record application here



5. Build the **studentserver** docker image

docker build -t yourdockerhubID/studentserver .

```
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ docker build -t cs571/studentserver .
Sending build context to Docker daemon  9.291MB
Step 1/4 : FROM node:14
14: Pulling from library/node
d172ccdc78ea: Pull complete
4f105b790398: Pull complete
01ae525c10a7: Pull complete
72215048f783: Pull complete
8cfa9c61a0ab: Pull complete
7f6b791c006f: Pull complete
6138e26477a6: Pull complete
62d956a575b5: Pull complete
8ac0f5d179f7: Pull complete
Digest: sha256:7d38b5ed42b2ac006c3a79ef8ad9f1e912bde6cb4cb4243c188689d5aa1aa437
Status: Downloaded newer image for node:14
 ---> 903c2c873ea4
Step 2/4 : ADD studentServer.js /studentServer.js
 ---> 447819bb7785
Step 3/4 : RUN npm install mongodb
 ---> Running in 55a35c988b04
npm WARN saveError ENOENT: no such file or directory, open '/package.json'
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN enoent ENOENT: no such file or directory, open '/package.json'
npm WARN !invalid#1 No description
npm WARN !invalid#1 No repository field.
npm WARN !invalid#1 No README data
npm WARN !invalid#1 No license field.

+ mongodb@4.4.1
added 20 packages from 59 contributors and audited 20 packages in 2.308s

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

found 0 vulnerabilities

Removing intermediate container 55a35c988b04
 ---> aedfc7d7cd9f
Step 4/4 : ENTRYPOINT ["node", "studentServer.js"]
 ---> Running in 34d616bda905
Removing intermediate container 34d616bda905
 ---> a43e3412ef6c
Successfully built a43e3412ef6c
Successfully tagged cs571/studentserver:latest
```

6. Push the **studentserver** docker image to docker hub repository.

   docker push yourdockerhubID/studentserver

```
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ docker push cs571/studentserver
Using default tag: latest
The push refers to repository [docker.io/cs571/studentserver]
11a95f317cf9: Pushed
4350e07da86d: 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:a2e1699f3edbe5493e80ef2411c1c974e48cd21ed80bcdc76dae70b2f59cf1df size: 2424
```

Step 3 Start the single instance cluster to configure the application to test

1. Start minikube

   minikube start

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ minikube start
* minikube v1.25.2 on Debian 11.2 (amd64)
  - MINIKUBE_FORCE_SYSTEMD=true
  - MINIKUBE_HOME=/google/minikube
  - MINIKUBE_WANTUPDATENOTIFICATION=false
* Automatically selected the docker driver. Other choices: none, ssh
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Downloading Kubernetes v1.23.3 preload ...
    > preloaded-images-k8s-v17-v1...: 505.68 MiB / 505.68 MiB  100.00% 77.73 Mi
* Creating docker container (CPUs=2, Memory=4000MB) ...
* Preparing Kubernetes v1.23.3 on Docker 20.10.12 ...
  - kubelet.cgroups-per-qos=false
  - kubelet.enforce-node-allocatable=""
  - kubelet.housekeeping-interval=5m
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

2. Enable Ingress service on minikube

   minikube addons enable ingress

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ minikube addons enable ingress
  - Using image k8s.gcr.io/ingress-nginx/controller:v1.1.1
  - Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
  - Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
* Verifying ingress addon...
* The 'ingress' addon is enabled
```

3. Check the ingress is available on minikube

   minikube addons list

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ minikube addons list
|-----------------------------|----------|--------------|--------------------------------|
|         ADDON NAME          | PROFILE  |    STATUS    |           MAINTAINER           |
|-----------------------------|----------|--------------|--------------------------------|
| ambassador                  | minikube | disabled     | third-party (ambassador)       |
| auto-pause                  | minikube | disabled     | google                         |
| csi-hostpath-driver         | minikube | disabled     | kubernetes                     |
| dashboard                   | minikube | disabled     | kubernetes                     |
| default-storageclass        | minikube | enabled ✅   | kubernetes                     |
| efk                         | minikube | disabled     | third-party (elastic)          |
| freshpod                    | minikube | disabled     | google                         |
| gcp-auth                    | minikube | disabled     | google                         |
| gvisor                      | minikube | disabled     | google                         |
| helm-tiller                 | minikube | disabled     | third-party (helm)             |
| ingress                     | minikube | enabled ✅   | unknown (third-party)          |
| ingress-dns                 | minikube | disabled     | google                         |
| istio                       | minikube | disabled     | third-party (istio)            |
| istio-provisioner           | minikube | disabled     | third-party (istio)            |
| kong                        | minikube | disabled     | third-party (Kong HQ)          |
| kubevirt                    | minikube | disabled     | third-party (kubevirt)         |
| logviewer                   | minikube | disabled     | unknown (third-party)          |
| metallb                     | minikube | disabled     | third-party (metallb)          |
| metrics-server              | minikube | disabled     | kubernetes                     |
| nvidia-driver-installer     | minikube | disabled     | google                         |
| nvidia-gpu-device-plugin    | minikube | disabled     | third-party (nvidia)           |
| olm                         | minikube | disabled     | third-party (operator          |
|                             |          |              | framework)                     |
| pod-security-policy         | minikube | disabled     | unknown (third-party)          |
| portainer                   | minikube | disabled     | portainer.io                   |
| registry                    | minikube | disabled     | google                         |
| registry-aliases            | minikube | disabled     | unknown (third-party)          |
| registry-creds              | minikube | disabled     | third-party (upmc enterprises) |
| storage-provisioner         | minikube | enabled ✅   | google                         |
| storage-provisioner-gluster | minikube | disabled     | unknown (third-party)          |
| volumesnapshots             | minikube | disabled     | kubernetes                     |
|-----------------------------|----------|--------------|--------------------------------|
```

Step 4 Create ConfigMap to store mongodb configuration to avoid re-building docker image if the mongoDB pod restarts.

1. Create studentserver-configmap.yaml.

   vim studentserver-configmap.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  # SERVICE_N                                    l:SERVICE_PORT
  MONGO_URL:  Your_mongodb_service_externalIP
  MONGO_DATABASE: mydb
```

2. Create the **studentserver-config** with using newly created studentserver-configmap.yaml

   kubectl apply -f studentserver-configmap.yaml

```
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ kubectl apply -f studentserver-configmap.yaml
configmap/studentserver-config created
```

3. Check the studentserver-config is running.

   kubectl get configmap

```
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ kubectl get configmap
NAME                  DATA   AGE
kube-root-ca.crt      1      7h52m
studentserver-config  2      30s
```

Step 5 Expose the student record application using ingress with Nginx to assign domain name

1. Create studentserver-deployment.yaml

   vim studentserver-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: studentserver-deployment
  labels:
    app: studentserver-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: studentserver-deployment
  template:
    metadata:
      labels:
        app: studentserver-deployment
    spec:
      containers:
        - image: cs571/studentserver
          imagePullPolicy: Always
          name: studentserver-deployment
          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 **studentserver-deploy** with using studentserver- deployment.yaml

   kubectl apply -f studentServer-deployment.yaml

```
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ kubectl apply -f studentServer-deployment.yaml
deployment.apps/studentserver-deployment created
```

3. Check the newly created studentserver-deploy is running.

   kubectl get deployments

```
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ kubectl get deployments
NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
mongodb-deployment        1/1     1            1           65m
studentserver-deployment  1/1     1            1           48m
```

Step 6 Create the service to access the content from outside the cluster

1. Create studentserver-service.yaml

   vim studentserver-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: studentserver-service
spec:
  type: LoadBalancer
  ports:
      # service port in cluster
    - port: 8080
      # port to contact inside container
      targetPort: 8080
  selector:
    app: studentserver-deployment
```

2. Create the **studentserver-service** with using the studentserver-service.yaml

   kubectl apply -f studentserver-service.yaml

```
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ kubectl apply -f studentserver-service.yaml
service/studentserver-service created
```

3. Check the created studentserver-service

   kubectl get svc

```
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ kubectl get svc
NAME                  TYPE           CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
kubernetes            ClusterIP      10.36.0.1     <none>         443/TCP          25m
mongodb-service       LoadBalancer   10.36.3.133   34.145.21.88   27017:31337/TCP  20m
studentserver-service LoadBalancer   10.36.1.37    35.247.18.41   8080:31228/TCP   68s
```

Bookshelf application configuration procedure steps are following:

Step 1 Create a python Flask bookshelf REST API and deploy on GKE

1. Make a new **bookshelf** directory and go to bookshelf directory.

   mkdir bookshelf

   cd bookshelf

```
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ mkdir bookshelf
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ cd bookshelf
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$
```

2. Create bookshelf.py with adding following code.

   vim 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_tasks():
    books = db.bookshelf.find()
    data = []
    for book in books:
        data.append({
```

```
            "id": str(book["_id"]),
            "Book Name": book["book_name"],
            "Book Author": book["book_author"],
            "ISBN": book["ISBN"]
        })
    return jsonify(
        data
    )

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

@app.route("/book/<id>", methods=["PUT"])
def update_book(id):
    data = request.get_json(force=True)
    print(data)
    response = db.bookshelf.update_many({"_id": ObjectId(id)}, {"$set":
{"book_name": data['book_name'],
        "book_author": data["book_author"], "ISBN": data["isbn"]
        }})
    if response.matched_count:
        message = "Book updated successfully!"
    else:
        message = "No book found!"
    return jsonify(
        message=message
    )

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

```
        message=message
    )

@app.route("/tasks/delete", methods=["POST"])
def delete_all_tasks():
    db.bookshelf.remove()
    return jsonify(
        message="All Books deleted!"
    )
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

3. Before creating the Dockerfile, first add the **requirement.txt** file in your current working directory.

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ vim requirements.txt
```

And add two dependencies **Flask** and **Flask-PyMongo** for Bookstore application

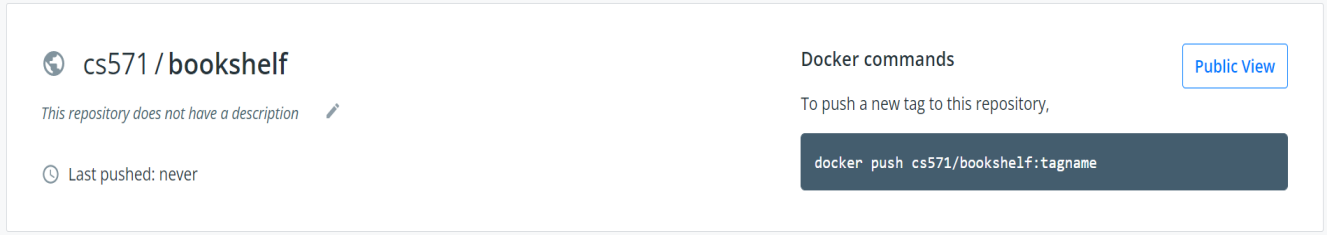
```
Flask
Flask-PyMongo
```

4. Create a DockerFile

vim Dockerfile

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

5. Create the docker hub **bookshelf** repository for this student record application [here]



6. Build the bookshelf docker image

docker build -t yourdockerhubID/bookshelf .

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ docker build -t cs571/bookshelf .
Sending build context to Docker daemon  23.04kB
Step 1/8 : FROM python:alpine3.7
alpine3.7: Pulling from library/python
48ecbb6b270e: Already exists
692f29ee68fa: Already exists
6439819450d1: Already exists
3c7be240f7bf: Already exists
ca4b349df8ed: Already exists
Digest: sha256:35f6f83ab08f98c727dbefd53738e3b3174a48b4571ccb1910bae480dcdba847
Status: Downloaded newer image for python:alpine3.7
 ---> 00be2573e9f7
Step 2/8 : COPY . /app
 ---> 1bd8320c6109
Step 3/8 : WORKDIR /app
 ---> Running in 035caa46a0c0
Removing intermediate container 035caa46a0c0
 ---> dfd022bca666
Step 4/8 : RUN pip freeze > requirements.txt && pip install -r requirements.txt
 ---> Running in 6cfc5de1c0bc
You are using pip version 19.0.1, however version 22.0.4 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
Removing intermediate container 6cfc5de1c0bc
 ---> c2a2edbdae00
Step 5/8 : ENV PORT 5000
 ---> Running in 58e33e2b8d0f
Removing intermediate container 58e33e2b8d0f
 ---> 8b2eb90430d0
Step 6/8 : EXPOSE 5000
 ---> Running in 29f69facc72c
Removing intermediate container 29f69facc72c
 ---> fc87bd29b603
Step 7/8 : ENTRYPOINT [ "python3" ]
 ---> Running in c68456ed1615
Removing intermediate container c68456ed1615
 ---> 8fe4f4bb84cf
Step 8/8 : CMD [ "bookshelf.py" ]
 ---> Running in 7f438cef50c9
Removing intermediate container 7f438cef50c9
 ---> c266bb5af386
Successfully built c266bb5af386
Successfully tagged cs571/bookshelf:latest
```

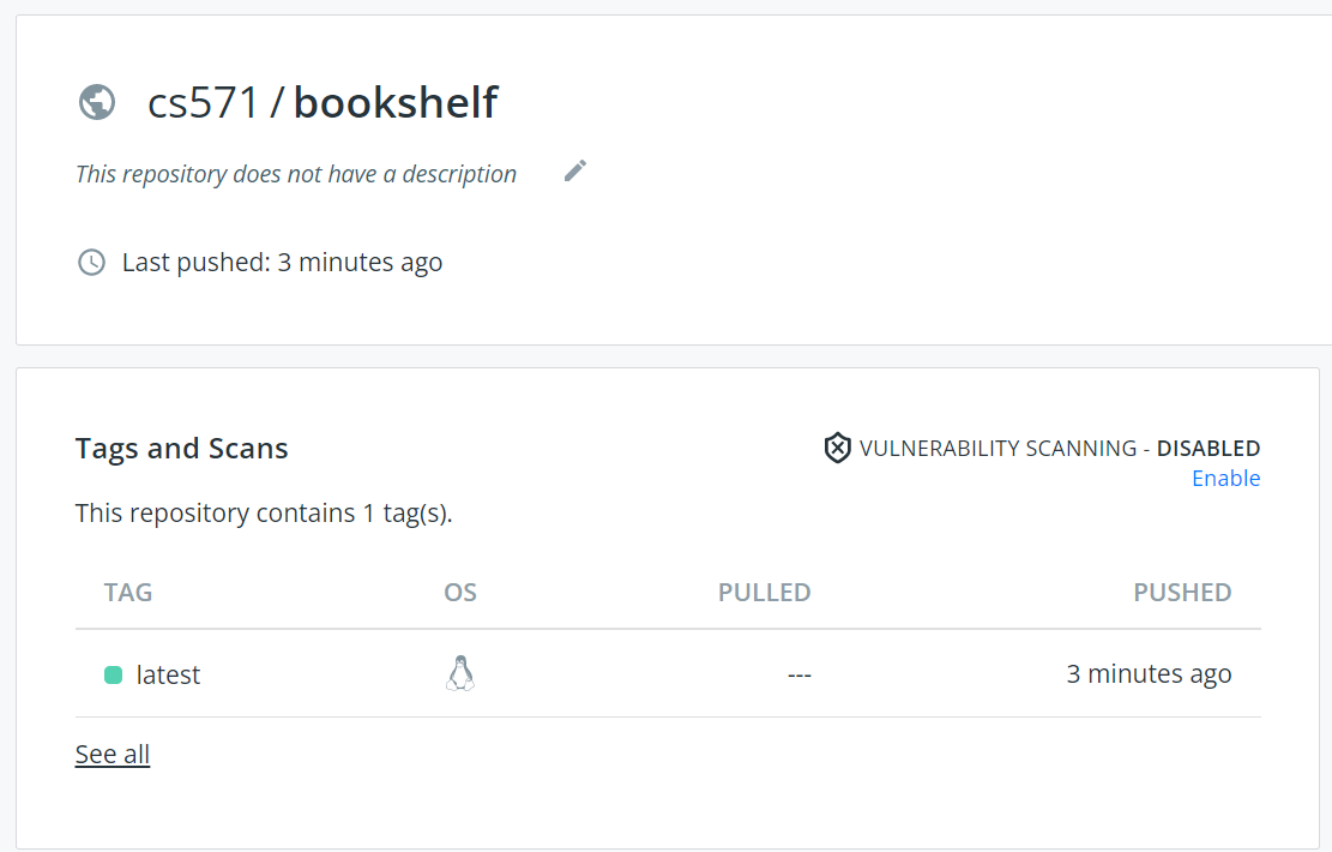
7. Push the bookshelf docker image to docker hub bookshelf repository

   docker push yourdockerhubID/bookshelf

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ docker push cs571/bookshelf
Using default tag: latest
The push refers to repository [docker.io/cs571/bookshelf]
58d81f6818c0: Pushed
c8601ba7350f: Pushed
e571d2d3c73c: Mounted from library/python
da7b0a80a4f2: Mounted from library/python
ceee8816bb96: Mounted from library/python
47458fb45d99: Mounted from library/python
46829331b1e4: Mounted from library/python
d35c5bda4793: Mounted from library/python
a3c1026c6bcc: Mounted from library/python
f1d420c2af1a: Mounted from library/python
461719022993: Mounted from library/python
latest: digest: sha256:e2d67589ded050ad5c6cf345097023e2a10ec22e0a1b3ce569d541e3bed5b60d size: 2644
```

8. Check the uploaded bookshelf docker image on docker hub

   Go to docker hub -> select bookshelf repository



Step 2 Create ConfigMap to avoid re-building docker image if the mongoDB pod restarts.

1. Create bookshelf-configmap.yaml.

   vim bookshelf-configmap.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: bookshelf-config
data:
  # SERVICE_N                                    ocal:SERVICE_PORT
  MONGO_URL:  Your_mongodb_service_externalIP
  MONGO_DATABASE: mydb
```

2. Create the bookshelf-config with using newly created bookshelf-configmap.yaml

   kubectl apply -f bookshelf-configmap.yaml

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
```

3. Check the bookshelf-configmap is running.

   kubectl get configmap

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ kubectl get configmap
NAME                  DATA   AGE
bookshelf-config      2      26s
kube-root-ca.crt      1      8h
studentserver-config  2      39m
```

Step 3 Expose the bookshelf application using ingress with Nginx to assign domain name

1. Create bookshelf-deployment.yaml

   vim 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: cs571/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
```

2. Create **bookshelf-deployment** with using bookshelf-deployment.yaml

   kubectl apply -f bookshelf-deployment.yaml

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created
```

3. Check the newly created bookshelf-deployment

   kubectl get deployments

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ kubectl get deployments
NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
bookshelf-deployment      1/1     1            1           28s
mongodb-deployment        1/1     1            1           8h
studentserver-deployment  1/1     1            1           8h
```

Step 4 Create the service to access the content from outside the cluster

1. Create a bookshelf-service.yaml

   vim bookshelf-service.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: bookshelf-service
spec:
  type: LoadBalancer
  ports:
      # service port in cluster
    - port: 5000
      # port to contact inside container
      targetPort: 5000
  selector:
    app: bookshelf-deployment
```

2. Create the **bookshelf-service** with using the bookshelf-service.yaml

   kubectl apply -f bookshelf-service.yaml

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
```

3. Check the **bookshelf-service**

   kubectl get svc

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ kubectl get svc
NAME                  TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
bookshelf-service     LoadBalancer   10.98.79.237   <pending>     5000:30585/TCP   5s
kubernetes            ClusterIP      10.96.0.1      <none>        443/TCP          44m
studentserver-service LoadBalancer   10.98.124.44   <pending>     8080:31894/TCP   98s
```

Step 5 Assign the domain name to both applications with using Ingress service

1. Check the full cluster configuration is running

   kubectl get all

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ kubectl get all
NAME                                          READY   STATUS    RESTARTS   AGE
pod/bookshelf-deployment-b975f5fb4-9vmgf      1/1     Running   0          4m30s
pod/studentserver-deployment-8457c48674-ntprq 1/1     Running   0          6m3s

NAME                          TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
service/bookshelf-service     LoadBalancer   10.98.79.237   <pending>     5000:30585/TCP   4m15s
service/kubernetes            ClusterIP      10.96.0.1      <none>        443/TCP          48m
service/studentserver-service LoadBalancer   10.98.124.44   <pending>     8080:31894/TCP   5m48s

NAME                                     READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/bookshelf-deployment     1/1     1            1           4m30s
deployment.apps/studentserver-deployment 1/1     1            1           6m3s

NAME                                                DESIRED   CURRENT   READY   AGE
replicaset.apps/bookshelf-deployment-b975f5fb4      1         1         1       4m30s
replicaset.apps/studentserver-deployment-8457c48674 1         1         1       6m3s
```

2. Create a signature-project-ingress.yaml file for this project ingress service.

   vim signature-project-ingress.yaml

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: project-server
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
  rules:
    - host: cs571.signatureproject.com
      http:
        paths:
          - path: /studentserver(/|$)(.*)
            pathType: Prefix
            backend:
              service:
                name: studentserver-service
                port:
                  number: 8080
          - path: /bookshelf(/|$)(.*)
            pathType: Prefix
            backend:
              service:
                name: bookshelf-service
                port:
                  number: 5000
```

3. Create the **project-server** ingress service with using signature-project-ingress.yaml

   kubectl apply -f signature-project-ingress.yaml

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ kubectl apply -f signature-project-ingress.yaml
ingress.networking.k8s.io/project-server created
```

4. Check the project-server ingress service is running.

   kubectl get ingress

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ kubectl get ingress
NAME             CLASS   HOSTS                        ADDRESS        PORTS   AGE
project-server   nginx   cs571.signatureproject.com   192.168.49.2   80      15s
```

5. Add ADDRESS to /etc/hosts

   vi /etc/hosts

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

   Your-ADDRESS **cs571.signatureproject.com**

```
# Kubernetes-managed hosts file.
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
fe00::0 ip6-mcastprefix
fe00::1 ip6-allnodes
fe00::2 ip6-allrouters
172.17.0.4      cs-810844977107-default
192.168.49.2    cs571.signatureproject.com
```

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

Step 6 Test both applications

1. Now, we are able to access both applications with one HOST URL.

   curl cs571.signatureproject.com/studentserver/api/score?student_id=11111/22222/33333

```
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ curl cs571.signatureproject.com/studentserver/api/score?student_id=11111
{"_id":"62461c0eefe5376562e4c092","student_id":11111,"student_name":"Bruce Lee","grade":84}
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ curl cs571.signatureproject.com/studentserver/api/score?student_id=22222
{"_id":"62461c0eefe5376562e4c093","student_id":22222,"student_name":"Jackie Chen","grade":93}
patel19619@cloudshell:~/mongodb (cs571-signature-project)$ curl cs571.signatureproject.com/studentserver/api/score?student_id=33333
{"_id":"62461c0eefe5376562e4c094","student_id":33333,"student_name":"Jet Li","grade":88}
```

2. On another path, you should be able to use the REST API with bookshelf application i.e list all books

   curl cs571.signatureproject.com/bookshelf/books

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ curl cs571.signatureproject.com/bookshelf/bo
oks
[]
```

   You will get the empty array result because mongoDB database doesnot have any book record till now. So now we are going to add new book to database.

3. To add a book in MongoDB database

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

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ curl -X POST -d "{\"book_name\": \"cloud com
puting\",\"book_author\": \"unkown\", \"isbn\": \"123456\" }" http://cs571.signatureproject.com/bookshelf/book
{
  "message": "Book saved successfully!"
}
```

4. Check the added book in the MongoDB database.

   curl cs571.signatureproject.com/bookshelf/books

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ curl cs571.signatureproject.com/bookshelf/bo
oks
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "624626504b936020a8225e5f"
  }
]
```

5. To update a book in the MongoDB database

curl -X PUT -d "{\"book_name\": \"123\",\"book_author\": \"test\", \"isbn\": \"123updated\" }"

http://cs571.signatureproject.com/bookshelf/book/yourbook_id

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ curl -X PUT -d "{\"book_name\": \"123\",\"bo
ok_author\": \"test\", \"isbn\": \"123updated\" }" http://cs571.signatureproject.com/bookshelf/book/624626504b936
020a8225e5f
{
  "message": "Book updated successfully!"
}
```

6. Check the updated book in the MongoDB database.

curl cs571.signatureproject.com/bookshelf/books

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ curl cs571.signatureproject.com/bookshelf/bo
oks
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123updated",
    "id": "624626504b936020a8225e5f"
  }
]
```

7. To Delete a book into MongoDB database

curl -X DELETE cs571.signatureproject.com/bookshelf/book/yourbook-id

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ curl -X DELETE cs571.signatureproject.com/bo
okshelf/book/624626504b936020a8225e5f
{
  "message": "Book deleted successfully!"
}
```

8. Check the updated book in the MongoDB database.

curl cs571.signatureproject.com/bookshelf/books

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ curl cs571.signatureproject.com/bookshelf/bo
oks
[]
```

Again, you will get the empty array output because the book we added first then updated that book and after that we deleted so now out database does not have any book information.

Step 7 Secure the Ingress with TLS (Transport Layer Security) traffic

1. First delete the project-server ingress from minikube cluster.

   kubectl delete ingress project-server

2. Configure ingress to handle TLS traffic, update the **signature-project-ingress.yaml** file.

   vim signature-project-ingress.yaml

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
 name: project-ingress-tls
 annotations:
   nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
  tls:
  - hosts:
    - cs571.signatureproject.com
    secretName: tls-secret
  rules:
    - host: cs571.signatureproject.com
      http:
        paths:
          - path: /studentserver(/|$)(.*)
            pathType: Prefix
            backend:
              service:
                name: studentserver-service
                port:
                  number: 8080
          - path: /bookshelf(/|$)(.*)
            pathType: Prefix
            backend:
              service:
                name: bookshelf-service
                port:
                  number: 5000
```

3. Create TLS certificates, public and private keys

   openssl genrsa -out tls.key 2048

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ openssl genrsa -out tls.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....................................................+++++
.........................................+++++
e is 65537 (0x010001)
```

4. Create your first certificate signing request

   openssl req -new -x509 -key tls.key -out tls.cert -days 360 -subj /CN=cs571.signatureproject.com

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ openssl req -new -x509
-key tls.key -out tls.cert -days 360 -subj /CN=cs571.signatureproject.com
```

5. Create secret that holds first certificate and keys

   kubectl create secret tls tls-secret --cert=tls.cert --key=tls.key

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ kubectl create secret
tls tls-secret --cert=tls.cert --key=tls.key
secret/tls-secret created
```

6. Create the TLS ingress with signature-project-ingress.yaml

   kubectl create -f signature-project-ingress.yaml

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ kubectl create -f
project-ingress-tls.yaml
ingress.networking.k8s.io/project-ingress-tls created
```

7. Let's try to access both applications

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

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ curl cs571.signatureproject.com
/studentserver/api/score?student_id=11111
<html>
<head><title>308 Permanent Redirect</title></head>
<body>
<center><h1>308 Permanent Redirect</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

8. Access the book list from mongodb Database

   curl cs571.signatureproject.com/bookshelf/books

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ curl cs571.signatureproject.com
/bookshelf/books
<html>
<head><title>308 Permanent Redirect</title></head>
<body>
<center><h1>308 Permanent Redirect</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

You should get this error because you don't have the trusted certificated to access the applications.

9. We can access the applications

curl cs571.signatureproject.com/studentserver/api/score?student_id=11111 -kL

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ curl cs571.signatureproject.com/
studentserver/api/score?student_id=11111 -kL
{"_id":"62465c1cf08fde77d99c324f","student_id":11111,"student_name":"Bruce Lee","grade":84}
```

curl cs571.signatureproject.com/bookshelf/books -kL

```
patel19619@cloudshell:~/mongodb/bookshelf (cs571-signature-project)$ curl cs571.signatureproject.com/
bookshelf/books -kL
[]
```