

Aung Phone Kyaw

19930

CS571

Signature project

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

```
Welcome to Cloud Shell! Type "help" to get started.
To set your Cloud Platform project in this session use "gcloud config set project [PROJECT_ID]"
aphonek779@cloudshell:~$ gcloud auth login

You are already authenticated with gcloud when running
inside the Cloud Shell and so do not need to run this
command. Do you wish to proceed anyway?

Do you want to continue (Y/n)? y
```

Login to gcp account using this command “gcloud auth login”

```
aphonek779@cloudshell:~$ gcloud container clusters create firstapp --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: Your Pod address range (--cluster-ipv4-cidr) can accommodate at most 1008 nodes.
Creating cluster firstapp in us-west1... Cluster is being health-checked (master is healthy)...done.
Created [https://console.cloud.google.com/v1/projects/handy-curve-413900/zones/us-west1/clusters/firstapp].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload/_gcloud/us-west1/firstapp?project=handy-curve-413900
kubeconfig entry generated for firstapp.
NAME: firstapp
LOCATION: us-west1
MASTER VERSION: 1.27.5-gke.1047004
MASTER IP: 34.43.189.83
MACHINE TYPE: e2-micro
MIN VER: 1.27.5-gke.1047004
NUM NODES: 3
STATUS: RUNNING
aphonek779@cloudshell:~$
```

Create a cluster using this command “gcloud container clusters create firstapp --num-nodes=1 --machine-type=e2-micro --region=us-west1”

```
aphonek779@cloudshell:~$ gcloud compute disks create --size=10GiB --zone=us-west1-a mongodb
WARNING: You have selected a disk size of under 100GB. 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/handy-curve-413900/zones/us-west1-a/disks/mongodb].
NAME: mongodb
ZONE: us-west1-a
SIZE_GB: 10
TYPE: pd-standard
STATUS: READY

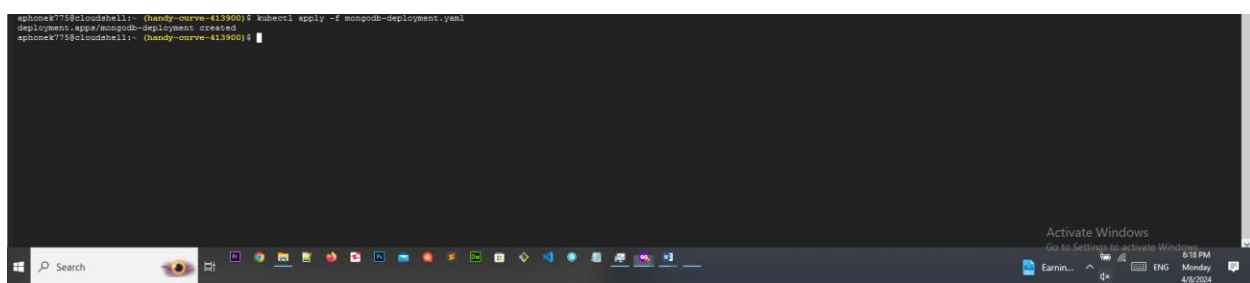
New disks are unformatted. You must format and mount a disk before it
can be used. You can find instructions on how to do this at:
https://cloud.google.com/compute/docs/disks/add-persistent-disk#formatting
aphonek779@cloudshell:~$
```

Create a persistent volume using this command “gcloud compute disks create --size=10GiB --zone=us-west1-a mongodb”



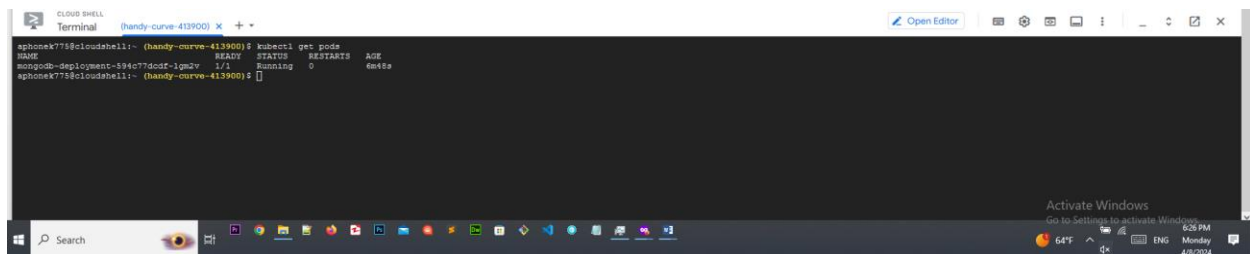
```
GNU nano 3.0 mongodb-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
spec:
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
        - image: mongo
          name: mongo
          ports:
            - containerPort: 27017
          volumeMounts:
            - name: mongodb-data
              mountPath: /data/db
      volumes:
        - name: mongodb-data
          mountPath: /data/db
```

Create a file name called mongodb-deployment.yaml using this command “nano mongodb-deployment.yaml” and enter the configuration



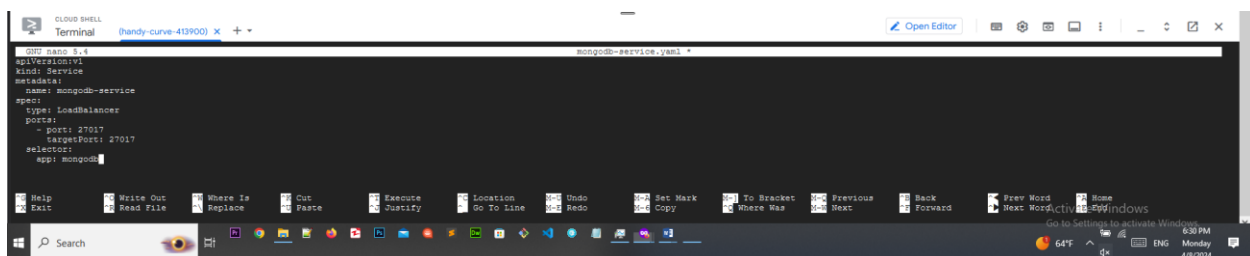
```
sphoek773@cloudshell: (handy-curve-413900) kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
sphoek773@cloudshell: (handy-curve-413900) █
```

Create mongodb deployment using this command “kubectl apply -f mongodb-deployment.yaml”



```
sphoek773@cloudshell: (handy-curve-413900) kubectl get pods
NAME                                READY    STATUS    RESTARTS   AGE
mongodb-deployment-194c77d0sf-1gm2v  1/1      Running   0           6s
```

Check if the pod is created by using this command “kubectl get pods”



```
GNU nano 3.0 mongodb-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  type: LoadBalancer
  ports:
    - port: 27017
      targetPort: 27017
  selector:
    app: mongodb
```

Create a file mongodb-service.yaml for a service to access mongodb from outside by creating service configuration file

```
CLOUD SHELL
Terminal (handy-curve-413900) X +
sphonek775@cloudshell:~ (handy-curve-413900) $ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
sphonek775@cloudshell:~ (handy-curve-413900) $
```

Create mongodb service using this command “kubectl apply -f mongodb-service.yaml”

```
CLOUD SHELL
Terminal (handy-curve-413900) X +
sphonek775@cloudshell:~ (handy-curve-413900) $ clear
sphonek775@cloudshell:~ (handy-curve-413900) $ nano mongodb-deployment.yaml
sphonek775@cloudshell:~ (handy-curve-413900) $ nano mongodb-service.yaml
sphonek775@cloudshell:~ (handy-curve-413900) $ clear
sphonek775@cloudshell:~ (handy-curve-413900) $ kubectl apply -f mongodb-service.yaml
error: error parsing mongodb-service.yaml: error converting YAML to JSON: yaml: line 2: mapping values are not allowed in this context
sphonek775@cloudshell:~ (handy-curve-413900) $ clear
sphonek775@cloudshell:~ (handy-curve-413900) $ nano mongodb-service.yaml
sphonek775@cloudshell:~ (handy-curve-413900) $ clear
sphonek775@cloudshell:~ (handy-curve-413900) $ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
sphonek775@cloudshell:~ (handy-curve-413900) $ kubectl get svc
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes ClusterIP   10.0.0.1         <none>            443/TCP      80m
mongodb-service LoadBalancer 10.0.0.1         10.203.142.164   27017:30113/TCP 8m1s
sphonek775@cloudshell:~ (handy-curve-413900) $
```

Check if the service is running by using this command “kubectl get svc”

```
sphonek775@cloudshell:~ (handy-curve-413900) $ kubectl exec -it mongodb-deployment-894c77dndf-lgm2v -- bash
root@mongodb-deployment-894c77dndf-lgm2v:/$
```

Use created pod using this command “kubectl exec -it mongodb-deployment-pod_name -- bash”

Use exit to stop accessing mongo db pod

```
sphonek775@cloudshell:~ (handy-curve-413900) $ node
Welcome to Node.js v20.11.1.
Type ".help" for more information.
> |
```

Use node by typing node to insert data into mongo db

```
aphonek78@cloudshell:~ (bandy-curve-413900) $ node
Welcome to Node.js v20.11.1.
Type ".help" for more information.
> var MongoClient = require('mongodb').MongoClient;
>
> var url = "mongodb://104.196.109.186/mydb"
>
> 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);
...   });
...   });
Promise {
  [Symbol(key_id_symbol)]: 119,
  [Symbol(trigger_key_id_symbol)]: 4
}
> (node:5895) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
(Use 'node --inspect=0.0.0.0' to show where the warning was created)
(node:5895) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
```

Add data into mogo db using these code

```
var MongoClient = require('mongodb').MongoClient;
```

```
var url = "mongodb://EXTERNAL-IP/mydb"
```

```
// Connect to the db
```

```
MongoClient.connect(url,{ useNewUrlParser: true, useUnifiedTopology: true },
```

```
function(err, client){
```

```
if (err)
```

```
throw err;
```

```
// create a document to be inserted
```

```
var db = client.db("studentdb");
```

```
const docs = [
```

```
{ student_id: 11111, student_name: "Bruce Lee", grade: 84},
```

```
{ student_id: 22222, student_name: "Jackie Chen", grade: 93 },
```

```
{ student_id: 33333, student_name: "Jet Li", grade: 88}
```

```
]
```

```
db.collection("students").insertMany(docs, function(err, res){
```

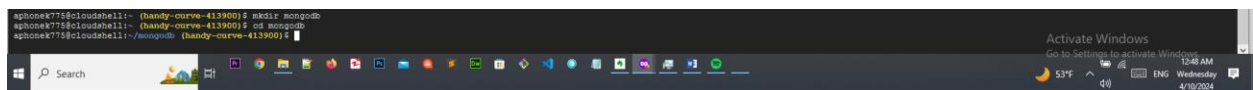
```
if(err) throw err;
```

```
console.log(res.insertedCount);
```

```
client.close();
```

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

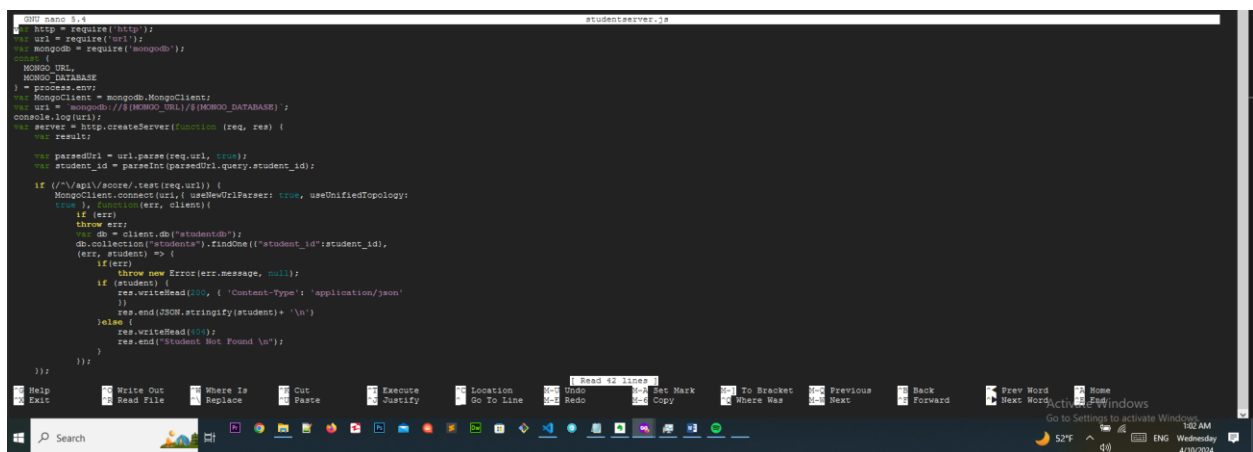
Make sure to enter them line by line.



Create a folder called mongodb using this command “mkdir mongodb”

Go to folder path using this command “cd mongodb”


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



Create a file called studentserver.js and enter these line of codes

Create a python Flask bookshelf REST API and deploy on GKE

```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to handy-curve-413900.
Use 'gcloud config set project [PROJECT_ID]' to change to a different project.
sphonek77@cloudshell: (handy-curve-413900) $ mkdir bookshelf
sphonek77@cloudshell: (handy-curve-413900) $ cd bookshelf
sphonek77@cloudshell: /bookshelf (handy-curve-413900) $
```



Create a folder called bookshelf using this command “mkdir bookshelf” and go to bookshelf folder directory using this “cd bookshelf”

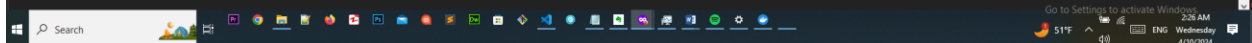
```
GMP nano 5.4.0 bookshelf.py
from flask import Flask, request, jsonify
from flask_pymongo import PyMongo
from flask import request
from bson.objectid import ObjectId
import socket
import os

app = Flask(__name__)
app.config["MONGO_URI"] = "mongodb://" + os.getenv("MONGO_URL") + "/" + os.getenv("MONGO_DATABASE")
app.config["JSONIFY_PRETTYPRINT_REGULAR"] = True
mongo = PyMongo(app)
db = mongo.db

@app.route("/")
def index():
    hostname = socket.gethostname()
    return jsonify(
        message="Welcome to bookshelf app! I am running inside {} pod!".format(hostname)
    )

@app.route("/books")
def get_all_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"]
        }
    )
```



Create a file called bookshelf.py and add these following codes

```
from flask import Flask, request, jsonify
from flask_pymongo import PyMongo
from flask import request
from bson.objectid import ObjectId
import socket
import os

app = Flask(__name__)
app.config["MONGO_URI"] = "mongodb://" + os.getenv("MONGO_URL") +
"/"+os.getenv("MONGO_DATABASE")
app.config['JSONIFY_PRETTYPRINT_REGULAR'] = True
```

```

mongo = PyMongo(app)
db = mongo.db

@app.route("/")
def index():
    hostname = socket.gethostname()
    return jsonify(
        message="Welcome to bookshelf app! I am running inside {}
pod!".format(hostname)
    )

@app.route("/books")
def get_all_tasks():
    books = db.bookshelf.find()
    data = []
    for book in books:
        data.append({
            "id": str(book["_id"]),
            "Book Name": book["book_name"],
            "Book Author": book["book_author"],
            "ISBN" : book["ISBN"]
        })
    return jsonify(
        data
    )

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

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

```



```

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

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

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

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

```

The screenshot shows a Windows terminal window with the title 'GNU nano 5.4'. The terminal content includes the following lines:

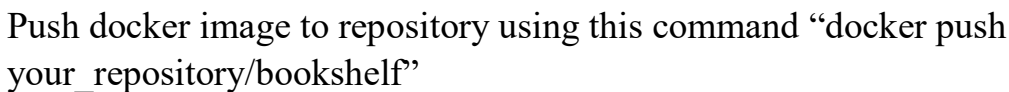
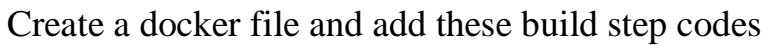
```

werkzeug==1.0.1
flask==1.1.2
jinja2==2.11.3

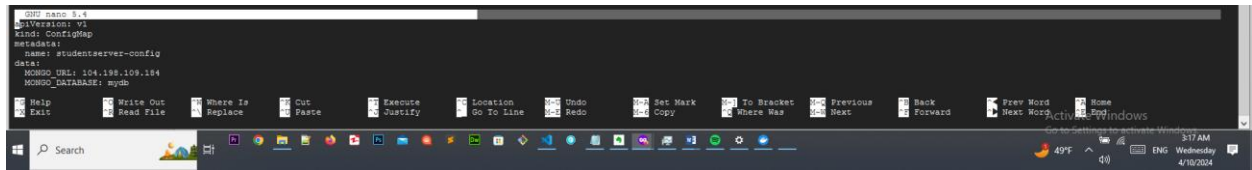
```

On the right side of the terminal window, there is a large, semi-transparent watermark that reads 'Activate Windows' and 'Go to Settings to activate Windows.' Below this, the system tray shows the date and time as '2:53 AM' on 'Wednesday 4/10/2024', along with icons for network, volume, and a notification bubble.

Create requirement.txt file and add these modules

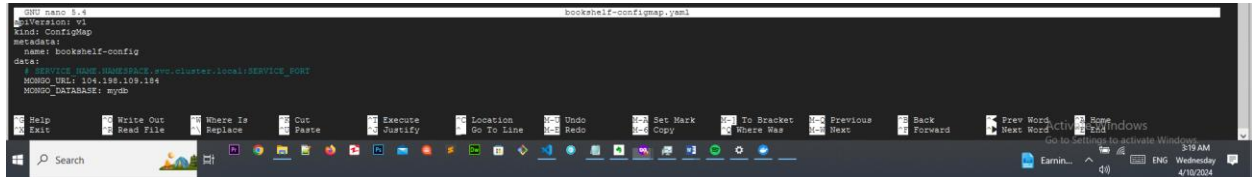


Create ConfigMap for both applications to store MongoDB URL and MongoDB name



```
GNU nano 3.4
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  MONGO_URL: 104.198.109.184
  MONGO_DATABASE: mydb
```

Create file studentserver-configmap.yaml and add these codes

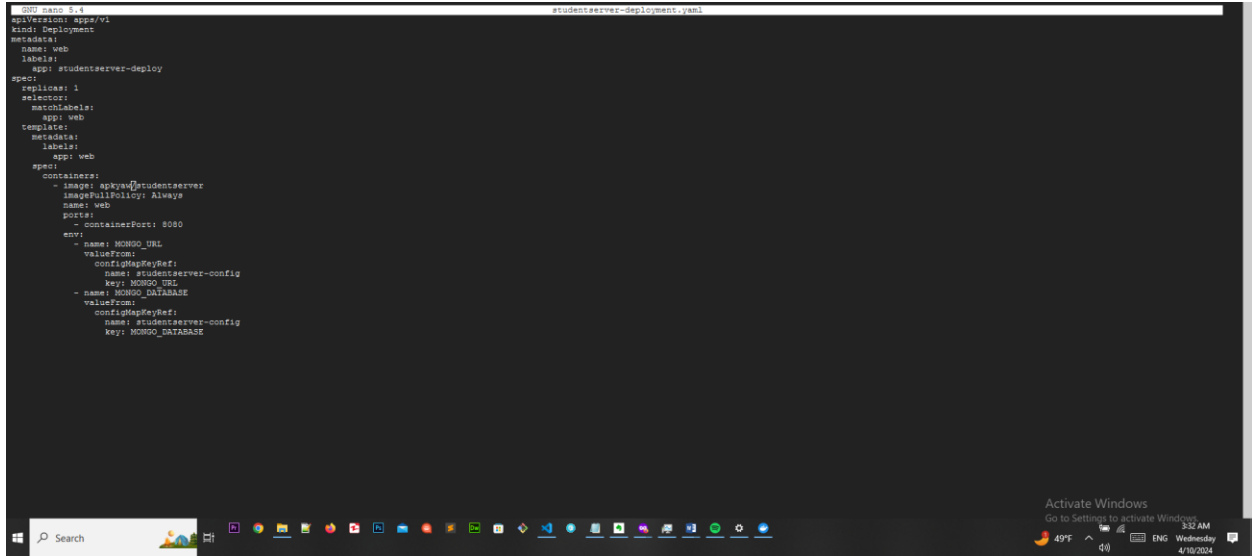


```
GNU nano 3.4 bookshelf-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: bookshelf-config
data:
  MONGO_URL: 104.198.109.184
  MONGO_DATABASE: mydb
```

Create file bookshelf-configmap.yaml and add these codes

The reason for these two config files are to avoid rebuilding docker image again if the mongoDB pod restarts with a different External-IP

Expose 2 applications using ingress with Nginx, so we can put them on the same Domain but different PATH



```
GNU nano 3.4 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:
        - image: apkyra/studentserver
          imagePullPolicy: Always
          name: web
          ports:
            - containerPort: 8080
          env:
            - name: MONGO_URL
              valueFrom:
                configMapKeyRef:
                  name: studentserver-config
                  key: MONGO_URL
            - name: MONGO_DATABASE
              valueFrom:
                configMapKeyRef:
                  name: studentserver-config
                  key: MONGO_DATABASE
```

Create a file studentserver-deployment.yaml

```
GNU nano 2.8 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: apkyaw/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 a file bookshelf-deployment.yaml

```
GNU nano 2.8 studentserver-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: web
spec:
  type: LoadBalancer
  ports:
    - port: 8080
      targetPort: 8080
  selector:
    app: web
```

Create a file studentserver-service.yaml

```
GNU nano 3.4 bookshelf-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: bookshelf-service
spec:
  type: LoadBalancer
  ports:
    - port: 5000
      targetPort: 5000
  selector:
    app: bookshelf-deployment
```

Create a file bookshelf-service.yaml

```
aphonek779@cloudshell:~/bookshelf: minikube start
* minikube v1.32.0 on Debian 11.9 (amd64)
- MINIKUBE_FORCE_DRIVER=true
- MINIKUBE_WOBBLE=/google/minikube
- MINIKUBE_WANTUPDATENOTIFICATION=false
* Automatically selected the docker driver. Other choices: ssh, none
* Using Docker driver with root privileges
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Downloading Kubernetes v1.28.3 preload ...
  > preloaded-images/k8s-v18-v1... 403.35 MiB / 403.35 MiB 100.00% 167.34
  > gcr.io/k8s-minikube/kicbase... 453.90 MiB / 453.90 MiB 100.00% 100.90
* Creating docker container (CPUs=2, Memory=4000MB) ...
X Docker is nearly out of disk space, which may cause deployments to fail! (96% of capacity). You can pass '--force' to skip this check.
* Suggestion:
  Try one or more of the following to free up space on the device:
  1. Run "docker system prune" to remove unused Docker data (optionally with "-a")
  2. Increase the storage allocated to Docker for Desktop By clicking on:
     Docker icon > Preferences > Resources > Disk Image Size
  3. Run "minikube ssh -- docker system prune" if using the Docker container runtime
* Related issue: https://github.com/kubernetes/minikube/issues/9024
* Preparing Kubernetes v1.28.3 on Docker 24.0.7 ...
- kubelet.cgroup-per-qos=false
- kubelet.enforce-node-allocatable=""
- Generating certificates and keys ...
- Booting up control plane ...
- Configuring kube rules ...
- Configuring bridge CNI (Container Networking Interface) ...
- Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Verifying Kubernetes components...
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
aphonek779@cloudshell:~/bookshelf (handy-curve-413900) $
```

Start minikube

```
aphonek779@cloudshell:~/bookshelf (handy-curve-413900) $ 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:v1.9.4
- Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20231011-8b53cabe0
* Verifying ingress addon...
* The 'ingress' addon is enabled
aphonek779@cloudshell:~/bookshelf (handy-curve-413900) $
```

Start ingress

```
aphonek779@cloudshell:~/bookshelf (handy-curve-413900) $ kubectl apply -f studentserver-deployment.yaml
deployment.apps/web created
aphonek779@cloudshell:~/bookshelf (handy-curve-413900) $ kubectl apply -f studentserver-configmap.yaml
configmap/studentserver-config created
aphonek779@cloudshell:~/bookshelf (handy-curve-413900) $ kubectl apply -f studentserver-service.yaml
service/web created
aphonek779@cloudshell:~/bookshelf (handy-curve-413900) $
```

Create student server related pods and start services

```
aphonek778@cloudshell:~/bookshelf (handy-curve-413900)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created
aphonek778@cloudshell:~/bookshelf (handy-curve-413900)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
aphonek778@cloudshell:~/bookshelf (handy-curve-413900)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
aphonek778@cloudshell:~/bookshelf (handy-curve-413900)$
```

Create book shelf related pods and start services

```
aphonek778@cloudshell:~/bookshelf (handy-curve-413900)$ kubectl get pods
NAME                                STATUS    RESTARTS   AGE
bookshelf-deployment-587fcd7999-qvbc 0/1      CrashLoopBackOff   3 (38s ago)    104s
web-2d9f2ff8b9-ednqr                 0/1      CrashLoopBackOff   4 (57s ago)    3ms
```

Check the pod using kubectl get pods

```
GNU nano 3.4 studentservermongoIngress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: server
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
  rules:
    - host: cs571.project.com
      http:
        paths:
          - path: /studentserver(/|/)(.*)
            pathType: Prefix
            backend:
              service:
                name: web
                port:
                  number: 8080
          - path: /bookshelf(/|/)(.*)
            pathType: Prefix
            backend:
              service:
                name: bookshelf-service
                port:
                  number: 5000
```

Create file studentservermongoIngress.yaml

```
aphonek778@cloudshell:~/bookshelf (handy-curve-413900)$ nano studentservermongoIngress.yaml
aphonek778@cloudshell:~/bookshelf (handy-curve-413900)$ kubectl apply -f studentservermongoIngress.yaml
Warning: path /studentserver(/|/)(.*) cannot be used with pathType Prefix
Warning: path /bookshelf(/|/)(.*) cannot be used with pathType Prefix
ingress.networking.k8s.io/server created
```

Create ingress service using config file

```
aphonek778@cloudshell:~/bookshelf (handy-curve-413900)$ kubectl get ingress
NAME CLASS ADDRESS PORTS AGE
server nginx cs571.project.com 192.168.49.2 80 2m1s
aphonek778@cloudshell:~/bookshelf (handy-curve-413900)$
```

Check created ingress service

```
# /etc/hosts: Local Host Database
#
# This file describes a number of aliases-to-address mappings for the for
# local hosts that share this file.
#
# In the presence of the domain name service or NIS, this file may not be
# consulted at all; see /etc/nss.conf for the resolution order.
#
# IPv4 and IPv6 localhost aliases
127.0.0.1    localhost
::1         localhost
#
# Emulated network.
#10.0.0.1    myname
#10.0.0.3    myfriend
#
# According to RFC 1918, you can use the following IP networks for private
# nets which will never be connected to the Internet:
#
# 10.0.0.0      - 10.255.255.255
# 172.16.0.0    - 172.31.255.255
# 192.168.0.0   - 192.168.255.255
#
# In case you want to be able to connect directly to the Internet (i.e. not
# behind a NAT, ADSL router, etc...), you need real official assigned
# numbers. Do not try to invent your own network numbers but instead get one
# from your network provider (if any) or from your regional registry (ARIN,
# APNIC, LACNIC, RIPE NCC, or AfriNIC).
#
169.254.169.254 metadata.google.internal metadata
10.88.0.4 cs-771275969317-default
192.168.49.2 cs571.project.com
```

Enter `sudo vi /etc/hosts` to add the address of ingress service

Type your_address cs571.project.com and save the file

Get student info

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

List all the books

curl cs571.project.com/bookshelf/books

Add the books

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

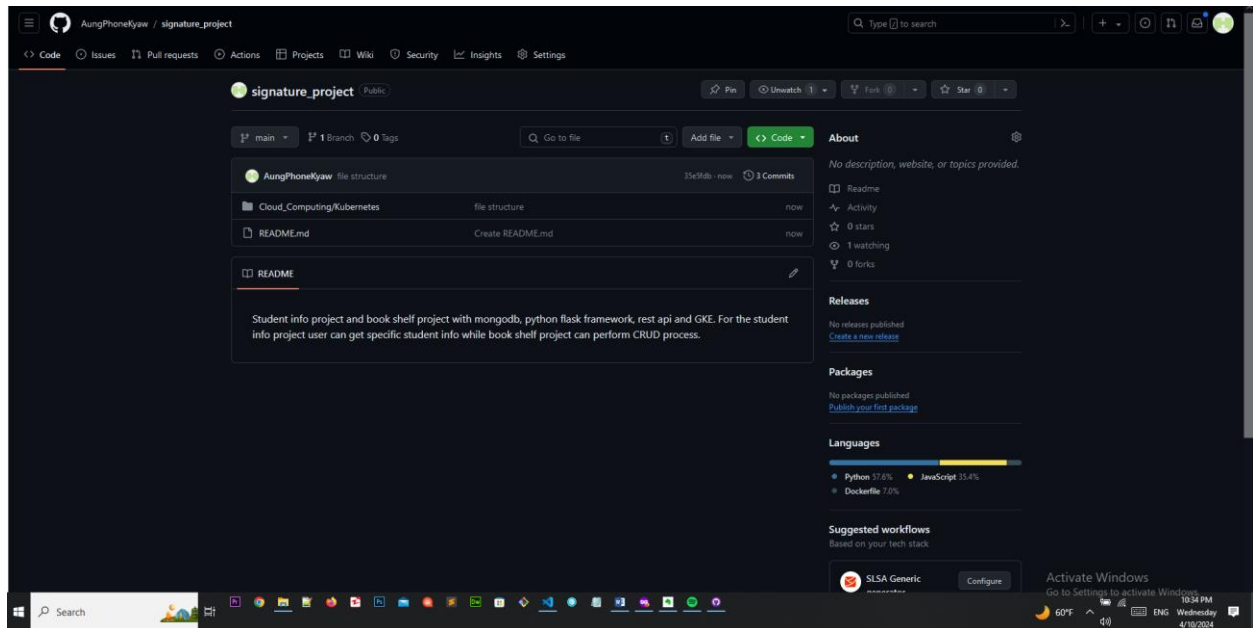
Update a book

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

Delete a book

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

Update your pofolio



Can find the project in this github repository

https://github.com/AungPhoneKyaw/signature_project