CS571 Signature Project
Name: Hartina Vonyee Cleon
ID: 20145

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

1. Create a cluster as usual on GKE
gcloud container clusters create kubia --num-nodes=1 --machine-type=e2-micro --region=us-west1
Wait for the creation to finish,

```
NAME: kubia
LOCATION: us-west1
MASTER_VERSION: 1.29.6-gke.1038001
MASTER_IP: 35.247.56.186
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.29.6-gke.1038001
NUM_NODES: 3
STATUS: RUNNING
```

2. Let's create a Persistent Volume first
gcloud compute disks create --size=10GiB --zone=us-west1-a mongodb

```
NAME: mongodb
ZONE: us-west1-a
SIZE_GB: 10
TYPE: pd-standard
STATUS: READY
```

3. Now create a mongodb deployment with this yaml file

```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:
    - name: mongodb
      image: mongo:4.4
      ports:
        - containerPort: 27017
      volumeMounts:
        - name: mongodb-data
          mountPath: /data/db
   volumes:
    - name: mongodb-data
      gcePersistentDisk:
        pdName: mongodb
        fsType: ext4
```

kubectl apply -f mongodb-deployment.yaml

```
hcleon48970@cloudshell:~ (gke-demo-426300)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
```

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

kubectl get pods

Please wait until you see the STATUS is running, then you can move forward

```
NAME                                  READY   STATUS    RESTARTS   AGE
mongodb-deployment-dc97b685f-8vx8n    1/1     Running   0          5h43m
```

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

```yaml
apiVersion: v1
kind: Service
metadata:
 name: mongodb-service
spec:
 type: LoadBalancer
 ports:
  # Service port exposed externally
  - port: 27017
    # Port exposed inside the container
    targetPort: 27017
    # Optional: Port on the node
    # nodePort: 30000  # Only needed for NodePort type services, or if specific node port is required.
 selector:
  app: mongodb
```

kubectl apply -f mongodb-service.yaml

```
hcleon48970@cloudshell:~ (gke-demo-426300)$ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
```

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

kubectl get svc

Please wait until you see the external-ip is generated for mongodb-service, then you can move forward

```
hcleon48970@cloudshell:~ (gke-demo-426300)$ kubectl get svc
NAME              TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)           AGE
kubernetes        ClusterIP      34.118.224.1    <none>           443/TCP           6h22m
mongodb-service   LoadBalancer   34.118.231.16   34.105.116.240   27017:31606/TCP   91s
```

7. 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

Now you are inside the mongodb deployment pod

```
hcleon48970@cloudshell:~ (gke-demo-426300)$ kubectl exec -it mongodb-deployment-dc97b685f-8vx8n -- b
ash
```

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

```
root@mongodb-deployment-dc97b685f-8vx8n:/# mongo 34.105.116.240
MongoDB shell version v4.4.29
connecting to: mongodb://34.105.116.240:27017/test?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("4f52fe8a-8cb7-400b-a209-0862ce07d4f8") }
MongoDB server version: 4.4.29
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:
        2024-07-31T16:59:49.058+00:00: Using the XFS filesystem is strongly recommended with the WiredTig
er storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
        2024-07-31T16:59:49.878+00:00: Access control is not enabled for the database. Read and write acc
ess to data and configuration is unrestricted
```

8. Type exit to exit mongodb and back to our google console

```
> exit
bye
root@mongodb-deployment-dc97b685f-8vx8n:/#
```

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

```
root@mongodb-deployment-dc97b685f-8vx8n:/# node -v
v10.19.0
```

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

If Everything is correct, you should see this,
3 means three records was inserted, and we tried search for student_id=11111

Step2 Modify our studentServer to get records from MongoDB and deploy to GKE

```
var http = require('http');
var url = require('url');
var mongodb = require('mongodb');
const {
MONGO_URL,
MONGO_DATABASE
} = process.env;
// - Expect the request to contain a query
// string with a key 'student_id' and a student ID as
// the value. For example
// /api/score?student_id=1111
// - The JSON response should contain only 'student_id', 'student_name'
// and 'student_score' properties. For example:
//
// {
// "student_id": 1111,
```

```
// "student_name": Bruce Lee,
// "student_score": 84
// }
//
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);
```

1. Create Dockerfile

```
FROM node:7
ADD studentServer.js /studentServer.js
ENTRYPOINT ["node", "studentServer.js"]
RUN npm install mongodb
```

2. Build the studentserver docker image docker build -t yourdockerhubID/studentserver .
Make sure there is no error

```
 => => sha256:5f32ed3c3f278edda4fc571c880b5277355a29ae8f52b52cdf865f058378a590 35.24MB / 35.24MB
 => => sha256:0c8cc2f24a4dcb64e602e086fc9446b0a541e8acd9ad72d2e90df3ba22f158b3 2.29MB / 2.29MB
 => => sha256:0d27a8e861329007574c6766fba946d48e20d2c8e964e873de352603f22c4ceb 450B / 450B
 => => extracting sha256:b253aeafeaa7e0671bb60008df01de101a38a045ff7bc656e3b0fbfc7c05cca5
 => => extracting sha256:3d2201bd995cccf12851a50820de03d34a17011dcbb9ac9fdf3a50c952cbb131
 => => extracting sha256:1de76e268b103d05fa8960e0f77951ff54b912b63429c34f5d6adfd09f5f9ee2
 => => extracting sha256:d9a8df5894511ce28a05e2925a75e8a4acbd0634c39ad734fdfba8e23d1b1569
 => => extracting sha256:6f51ee005deac0d99898e41b8ce60ebf250ebe1a31a0b03f613aec6bbc9b83d8
 => => extracting sha256:5f32ed3c3f278edda4fc571c880b5277355a29ae8f52b52cdf865f058378a590
 => => extracting sha256:0c8cc2f24a4dcb64e602e086fc9446b0a541e8acd9ad72d2e90df3ba22f158b3
 => => extracting sha256:0d27a8e861329007574c6766fba946d48e20d2c8e964e873de352603f22c4ceb
 => [2/5] WORKDIR /usr/src/app
 => [3/5] COPY package*.json ./
 => [4/5] RUN npm install
 => [5/5] COPY . .
 => exporting to image
 => => exporting layers
 => => writing image sha256:2edd2c0c4bfee1780a39f158b18f83449e4eda1ff27e026256a5a4fea4816334
```

3. Push the docker image docker push yourdockerhubID/studentserver

```
hc1eon48970@cloudshell:~ (gke-demo-426300)$ docker push hartina/studentserver
Using default tag: latest
The push refers to repository [docker.io/hartina/studentserver]
f20d2e32e90e: Pushed
9a3da50b978d: Pushed
8ea25a8296c5: Pushed
b491e8e689f9: 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: sha256:3fcb3923bf42de735b9915efcab21d8120da4be4ef910233cb067d31dd2c6baa size: 3053
```

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

1. Create bookshelf.py

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

app = Flask(__name__)
```

```
# Configure MongoDB URI
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.get("book_name"),
            "Book Author": book.get("book_author"),
```

```
            "ISBN": book.get("ISBN")
        })
    return jsonify(data)
```

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

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

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

2. Create a 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" ]
```

3. Build the bookshelf app into a docker image

docker build -t zhou19539/bookshelf .

Make sure this step build successfully

```
hcleon48970@cloudshell:~ (gke-demo-426300)$ docker build -t hartina/bookshelf .
[+] Building 8.3s (10/10) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 625B
 => [internal] load metadata for docker.io/library/python:alpine3.7
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/5] FROM docker.io/library/python:alpine3.7@sha256:35f6f83ab08f98c727dbefd53738e3b3174a48b4571ccb1910bae480dcdba847
 => [internal] load build context
 => => transferring context: 181.36kB
 => CACHED [2/5] WORKDIR /app
 => [3/5] COPY . /app
 => [4/5] RUN pip install --upgrade pip
 => [5/5] RUN pip install -r requirements.txt
 => exporting to image
 => => exporting layers
 => => writing image sha256:23ba1ac4f58ef38fb6fbdd369a49bf08c9dcc5eee4edd17edc0cf0fb201deaaa
 => => naming to docker.io/hartina/bookshelf
hcleon48970@cloudshell:~ (gke-demo-426300)$
```

4. Push the docker image to your dockerhub

docker push yourdockerhubID/bookshelf

```
hcleon48970@cloudshell:~ (gke-demo-426300)$ docker push  hartina/bookshelf
Using default tag: latest
The push refers to repository [docker.io/hartina/bookshelf]
89ca2eb72d62: Pushed
00cfd917993b: Pushed
41a3bc5e9e2e: Pushed
b1f20c7dc3cd: Pushed
5fa31f02caa8: Mounted from library/python
88e61e328a3c: Mounted from library/python
9b77965e1d3f: Mounted from library/python
50f8b07e9421: Mounted from library/python
629164d914fc: Mounted from library/python
latest: digest: sha256:49bc7bffa3f96936b44d70e34444f7f0ec00da598a283686040bd58bca3652de size: 2207
```

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

1. Create a file named studentserver-configmap.yaml

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
name: studentserver-config
data:
MONGO_URL: Change-this-to-your-mongoDB-EXTERNAL-IP
MONGO_DATABASE: mydb
```

2. Create a file named bookshelf-configmap.yaml

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
name: bookshelf-config
data:
# SERVICE_NAME.NAMESPACE.svc.cluster.local:SERVICE_PORT
MONGO_URL: Change-this-to-your-mongoDB-EXTERNAL-IP
MONGO_DATABASE: mydb
```

Notice: the reason of creating those two ConfigMap is to avoid re-building docker image again if the mongoDB pod restarts with a different External-IP

Step5 Expose 2 application using ingress with Nginx, so we can put them on the same Domain but different PATH

1. 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
       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
```

3. Create bookshelf-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: bookshelf-deployment
 labels:
   app: bookshelf
spec:
 replicas: 1
 selector:
   matchLabels:
     app: bookshelf
 template:
   metadata:
     labels:
       app: bookshelf
   spec:
     containers:
     - name: bookshelf-container
       image: hartina/bookshelf
       imagePullPolicy: Always
       ports:
       - containerPort: 5000
       env:
       - name: MONGO_URL
         valueFrom:
           configMapKeyRef:
             name: bookshelf-config
             key: MONGO_URL
       - name: MONGO_DATABASE
         valueFrom:
```

```
        configMapKeyRef:
          name: bookshelf-config
          key: MONGO_DATABASE
```

4. Create sutdentserver-service.yaml

```
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
```

5. Create bookshelf-service.yaml

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

6. Start minikube

minikube start

```
hcleon48970@cloudshell:~ (gke-demo-426300)$ minikube start
* minikube v1.33.1 on Ubuntu 22.04 (amd64)
  - MINIKUBE_FORCE_SYSTEMD=true
  - MINIKUBE_HOME=/google/minikube
  - MINIKUBE_WANTUPDATENOTIFICATION=false
* Automatically selected the docker driver. Other choices: none, ssh
* Using Docker driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.44 ...
* Downloading Kubernetes v1.30.0 preload ...
    > preloaded-images-k8s-v18-v1...:  342.90 MiB / 342.90 MiB  100.00% 41.77 M
    > gcr.io/k8s-minikube/kicbase...:  481.58 MiB / 481.58 MiB  100.00% 39.35 M
* Creating docker container (CPUs=2, Memory=4000MB) ...
* Preparing Kubernetes v1.30.0 on Docker 26.1.1 ...
  - kubelet.cgroups-per-qos=false
  - kubelet.enforce-node-allocatable=""
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
hcleon48970@cloudshell:~ (gke-demo-426300)$
```

7. Start Ingress

minikube addons enable ingress

```
hcleon48970@cloudshell:~ (gke-demo-426300)$ minikube addons enable ingress
* ingress is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
  - Using image registry.k8s.io/ingress-nginx/controller:v1.10.1
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.1
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.1
* Verifying ingress addon...
* The 'ingress' addon is enabled
```

8. 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

```
hcleon48970@cloudshell:~ (gke-demo-426300)$ kubectl apply -f studentserver-deployment.yaml
kubectl apply -f studentserver-configmap.yaml
kubectl apply -f studentserver-service.yaml
deployment.apps/web created
configmap/studentserver-config created
```

```
hcleon48970@cloudshell:~ (gke-demo-426300)$ kubectl apply -f sutdentserver-service.yaml
deployment.apps/web configured
```

9. 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

```
hcleon48970@cloudshell:~ (gke-demo-426300)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
```

```
deployment.apps/bookshelf-deployment created
service/bookshelf-service created
```

10. Check if all the pods are running correctly

kubectl get pods

```
hcleon48970@cloudshell:~ (gke-demo-426300)$ kubectl get pods
NAME                                  READY   STATUS             RESTARTS         AGE
bookshelf-deployment-75cfc977d6-mhxmv   0/1     CrashLoopBackOff   8 (4m34s ago)    20m
web-5dd5844b66-bbrq7                  0/1     CrashLoopBackOff   11 (4m52s ago)   36m
```

11. Create an ingress service yaml file called studentservermongoIngress.yaml

```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
```

12. Create the ingress service using the above yaml file

kubectl apply -f ../studentservermongoIngress.yaml

```
hcleon48970@cloudshell:~ (gke-demo-426300)$ 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
```

13. Check if ingress is running

kubectl get ingress

Please wait until you see the Address, then move forward

```
hcleon48970@cloudshell:~ (gke-demo-426300)$ kubectl get ingress
NAME     CLASS   HOSTS                ADDRESS        PORTS   AGE
server   nginx   cs571.project.com    192.168.49.2   80      99s
```

14. 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

```
NAME     CLASS   HOSTS                ADDRESS        PORTS   AGE
server   nginx   cs571.project.com    192.168.49.2   80      99s
```

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

```
# /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/host.conf for the resolution order.
#

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

#
# Imaginary network.
#10.0.0.2               myname
#10.0.0.3               myfriend
#
# According to RFC 1918, you can use the following IP networks for private
# nets which will never be connected to the Internet:
#
#       10.0.0.0        -   10.255.255.255
#       172.16.0.0      -   172.31.255.255
#       192.168.0.0     -   192.168.255.255
```

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

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