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

1. Check the status of clusters

```
chenghy1017@cloudshell:~/volume (my-project-342502)$ gcloud container clusters list
WARNING: The following zones did not respond: us-east7-c, us-east7, us-east7-a, us-ea
st7-b. List results may be incomplete.
NAME: kubia
LOCATION: us-west1
MASTER_VERSION: 1.21.6-gke.1503
MASTER_IP: 34.83.213.183
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.21.6-gke.1503
NUM_NODES: 3
STATUS: RUNNING
```

2. create a dick size of 10G

```
chenghy1017@cloudshell:~/volume (my-project-342502)$ gcloud compute disks create --si
ze=10GiB --zone=us-west1-a mongodb
WARNING: You have selected a disk size of under [200GB]. This may result in poor I/O
performance. For more information, see: https://developers.google.com/compute/docs/di
sks#performance.
Created [https://www.googleapis.com/compute/v1/projects/my-project-342502/zones/us-we
st1-a/disks/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-deploy
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
~
```

```
chenghy1017@cloudshell:~/signature (my-project-342502)$ kubectl apply -f mongodb-depl
oy.yaml
deployment.apps/mongodb-deploy created
```

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

```
chenghy1017@cloudshell:~/signature (my-project-342502)$ kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
mongodb-deploy-57dc68b4bd-fcpxb   1/1     Running   0          97s
```

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

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

```
chenghy1017@cloudshell:~/signature (my-project-342502)$ kubectl apply -f mongodb-serv
ice.yaml
service/mongodb-service created
```

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

```
chenghy1017@cloudshell:~/signature (my-project-342502)$ kubectl get svc
NAME              TYPE           CLUSTER-IP     EXTERNAL-IP      PORT(S)           AGE
kubernetes        ClusterIP      10.36.0.1      <none>           443/TCP           2d21
h
mongodb-service   LoadBalancer   10.36.10.29    34.83.212.124    27017:30806/TCP   5m25
s
```

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

```
chenghy1017@cloudshell:~/signature (my-project-342502)$ kubectl exec -it mongodb-depl
oy-57dc68b4bd-fcpxb -- bash
root@mongodb-deploy-57dc68b4bd-fcpxb:/#
```

Try

mongo External-IP

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

```
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:
        2022-03-17T21:32:53.333+00:00: Using the XFS filesystem is strongly recommend
ed with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-f
ilesystem
        2022-03-17T21:32:55.104+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, e
tc).

        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 p
roduct
        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.disableFr
eeMonitoring()
---
>
```

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

```
---
> exit
bye
root@mongodb-deploy-57dc68b4bd-fcpxb:/# exit
exit
chenghy1017@cloudshell:~/signature (my-project-342502)$
```

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

```
chenghy1017@cloudshell:~/signature (my-project-342502)$ node
Welcome to Node.js v12.14.1.
Type ".help" for more information.
>
```

Enter the following line by line

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://34.83.212.124/mydb"
MongoClient.connect(url,{ useNewUrlParser: true, useUnifiedTopology: true }, function(err,
client){
    if (err)
        throw err;
    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);
    });
});
```

Get Result

```
... });
undefined
> {
  _id: new ObjectId("6233b011595ae2e9affa5bc2"),
  student_id: 11111,
  student_name: 'Bruce Lee',
  grade: 84
}
```

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

1. Create a studentServer

```
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);
```

2. Create Dockerfile

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

3. Build the studentserver docker image

```
chenghy1017@cloudshell:~/signature/studentserver (my-project-342502)$ sudo docker bui
ld -t haruorange/studentserver .
Sending build context to Docker daemon  4.608kB
Step 1/4 : FROM node:7
 ---> d9aed20b68a4
Step 2/4 : ADD studentServer.js /studentServer.js
 ---> b057801be8e5
Step 3/4 : ENTRYPOINT ["node", "studentServer.js"]
 ---> Running in 2e735bde3115
Removing intermediate container 2e735bde3115
 ---> 43e79e1b7d23
Step 4/4 : RUN npm install mongodb
 ---> Running in 17f74bbb78af
```

```
Successfully built e9d5b9eca223
Successfully tagged haruorange/studentserver:latest
chenghy1017@cloudshell:~/signature/studentserver (my-project-342502)$
```

4. Push the docker image

```
chenghy1017@cloudshell:~/signature/studentserver (my-project-342502)$ sudo docker pus
h haruorange/studentserver
Using default tag: latest
The push refers to repository [docker.io/haruorange/studentserver]
b9bc6d0467a6: Pushed
e58ac591b2b6: 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:feb10f59006ad639b5c49465537a3cb288c71a222832b1e34ab818e95ba30e
30 size: 2424
chenghy1017@cloudshell:~/signature/studentserver (my-project-342502)$
```

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 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)
```

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

```
chenghy1017@cloudshell:~/signature/bookshelf (my-project-342502)$ sudo docker build -
t haruorange/bookshelf .
Sending build context to Docker daemon   5.12kB
Step 1/8 : FROM python:alpine3.7
alpine3.7: Pulling from library/python
48ecbb6b270e: Pull complete
692f29ee68fa: Pull complete
6439819450d1: Pull complete
3c7be240f7bf: Pull complete
ca4b349df8ed: Pull complete
```

4. Push the docker image to your dockerhub

```
chenghy1017@cloudshell:~/signature/bookshelf (my-project-342502)$ docker push haruora
nge/bookshelf
Using default tag: latest
The push refers to repository [docker.io/haruorange/bookshelf]
```

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: 34.83.212.124
  MONGO_DATABASE: mydb
```

2. Create a file named bookshelf-configmap.yaml

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: bookshelf-config
data:
  MONGO_URL: 34.83.212.124
  MONGO_DATABASE: mydb
```

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: haruorange/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
```

2. Create bookshelf-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: bookshelf-deployment
  labels:
    app: bookshelf-deploymenty
spec:
  replicas: 1
  selector:
    matchLabels:
      app: bookshelf-deployment
  template:
    metadata:
      labels:
        app: bookshelf-deployment
    spec:
      containers:
        - image: haruorange/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
```

3. Create sutdentserver-service.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: web
spec:
  type: LoadBalancer
  ports:
  - port: 8080
    targetPort: 8080
  selector:
    app: web
```

4. Create bookshelf-service.yaml

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

5. Start minikube

```
chenghy1017@cloudshell:~/signature (my-project-342502)$ 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% 107.99 M
🔥 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
```

6. Start Ingress

```
chenghy1017@cloudshell:~/signature (my-project-342502)$ minikube addons enable ingres
s
    ▪ Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
    ▪ Using image k8s.gcr.io/ingress-nginx/controller:v1.1.1
    ▪ Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
🔎 Verifying ingress addon...
🌟 The 'ingress' addon is enabled
```

7. Create studentserver related pods and start service using the above yaml file

```
chenghy1017@cloudshell:~/signature (my-project-342502)$ kubectl apply -f studentserve
r-deployment.yaml
deployment.apps/web created
chenghy1017@cloudshell:~/signature (my-project-342502)$ kubectl apply -f studentserve
r-configmap.yaml
configmap/studentserver-config created
chenghy1017@cloudshell:~/signature (my-project-342502)$ kubectl apply -f studentserve
r-service.yaml
service/web created
```

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

```
chenghy1017@cloudshell:~/signature (my-project-342502)$ kubectl apply -f bookshelf-de
ployment.yaml
deployment.apps/bookshelf-deployment created
chenghy1017@cloudshell:~/signature (my-project-342502)$ kubectl apply -f bookshelf-co
nfigmap.yaml
configmap/bookshelf-config created
chenghy1017@cloudshell:~/signature (my-project-342502)$ kubectl apply -f bookshelf-se
rvice.yaml
service/bookshelf-service created
```

9. Check if all the pods are running correctly

```
chenghy1017@cloudshell:~/signature (my-project-342502)$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
bookshelf-deployment-7858cb658-pgz6k  1/1     Running   0          4m47s
web-5df7b4448d-lrkvw                  1/1     Running   0          17s
```

10. 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
```

11. Create the ingress service using the above yaml file

```
chenghy1017@cloudshell:~/signature (my-project-342502)$ kubectl apply -f studentserve
rmongoIngress.yaml
ingress.networking.k8s.io/server created
```

12. Check if ingress is running

```
chenghy1017@cloudshell:~/signature (my-project-342502)$ kubectl get ingress
NAME     CLASS   HOSTS              ADDRESS        PORTS   AGE
server   nginx   cs571.project.com  192.168.49.2   80      24s
```

13. Add Addreee to /etc/hosts

```
# 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-344315516339-default
192.168.49.2    cs571.project.com
~
```

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

```
chenghy1017@cloudshell:~/signature (my-project-342502)$ curl cs571.project.com/studen
tserver/api/score?student_id=11111
{"_id":"6233b011595ae2e9affa5bc2","student_id":11111,"student_name":"Bruce Lee","grad
e":84}
chenghy1017@cloudshell:~/signature (my-project-342502)$ curl cs571.project.com/studen
tserver/api/score?student_id=22222
{"_id":"6233b011595ae2e9affa5bc3","student_id":22222,"student_name":"Jackie Chen","gr
ade":93}
chenghy1017@cloudshell:~/signature (my-project-342502)$ curl cs571.project.com/studen
tserver/api/score?student_id=33333
{"_id":"6233b011595ae2e9affa5bc4","student_id":33333,"student_name":"Jet Li","grade":
88}
```

On another path, you should be able to use the REST API with bookshelf application

Add a book

```
chenghy1017@cloudshell:~/signature (my-project-342502)$ curl -X POST -d "{\"book_name
\": \"cloud computing\",\"book_author\":\"unkown\", \"isbn\": \"123456\" }" http://cs
571.project.com/bookshelf/book
{
  "message": "Task saved successfully!"
}
```

List books

```
chenghy1017@cloudshell:~/signature (my-project-342502)$ curl cs571.project.com/booksh
elf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "6233e233fb285896c716157d"
  },
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123",
    "id": "6233e2d1fb285896c716157e"
  }
]
```

Update a book

```
chenghy1017@cloudshell:~/signature (my-project-342502)$ curl -X PUT -d "{\"book_name\
": \"123\",\"book_author\": \"test\", \"isbn\":\"123updated\" }" http://cs571.project
.com/bookshelf/book/6233e2d1fb285896c716157e
{
  "message": "Task updated successfully!"
}
```

```
chenghy1017@cloudshell:~/signature (my-project-342502)$ curl cs571.project.com/booksh
elf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "6233e233fb285896c716157d"
  },
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123updated",
    "id": "6233e2d1fb285896c716157e"
  }
]
```

Delete a book

```
chenghy1017@cloudshell:~/signature (my-project-342502)$ curl -X DELETE cs571.project.
com/bookshelf/book/6233e2d1fb285896c716157e
{
  "message": "Task deleted successfully!"
}
```