

# Week 10: Homework: Chapter 7: Configmap: Signature Project: MongoDB + Python Flask Web Framework + REST API + GKE

We are going to deploy two application under one domain into GKE. One built with node.js and another using python. Both application used mongodb server for data source and data operation.

We need to create cluster -

```
$ export PROJECT_ID=signature-project1-19609-v1
$ gcloud config set compute/zone us-west2-a
$ gcloud container clusters create cs571-sgnprj1-cluster
$ gcloud container clusters get-credentials cs571-sgnprj1-cluster --zone us-west2-a
```

Now we need to create storage disk for mongodb.

```
$ gcloud compute disks create --size=10GiB --zone=us-west2-a mongodisk
```

Here is the response of above commands .

```
gcloudshell:~ (signature-project1-19609-v1) $ export PROJECT_ID=signature-project1-19609-v1
gcloudshell:~ (signature-project1-19609-v1) $ gcloud config set compute/zone us-west2-a
Updated property [compute/zone].
gcloudshell:~ (signature-project1-19609-v1) $ gcloud container clusters create cs571-sgnprj1-cluster
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-ip4-cidr ' ) can accommodate at most 1008 node(s).
Creating cluster cs571-sgnprj1-cluster in us-west2-a... Cluster is being health-checked (master is healthy)... done.
Created [https://container.googleapis.com/v1/projects/signature-project1-19609-v1/zones/us-west2-a/clusters/cs571-sgnprj1-cluster].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload/_gcloud/us-west2-a/cs571-sgnprj1-cluster?project=signature-project1-19609-v1
kubeconfig entry generated for cs571-sgnprj1-cluster.
NAME: cs571-sgnprj1-cluster
LOCATION: us-west2-a
MASTER VERSION: 1.21.9-gke.1002
MASTER_IP: 34.94.72.16
MACHINE_TYPE: e2-medium
NODE VERSION: 1.21.9-gke.1002
NUM_NODES: 3
STATUS: RUNNING
gcloudshell:~ (signature-project1-19609-v1) $ gcloud container clusters get-credentials cs571-sgnprj1-cluster --zone us-west2-a
Fetching cluster endpoint and auth data.
kubeconfig entry generated for cs571-sgnprj1-cluster.
gcloudshell:~ (signature-project1-19609-v1) $ mkdir sgnprj1
gcloudshell:~ (signature-project1-19609-v1) $ v1 sgnprj1/mongodb.yaml
gcloudshell:~ (signature-project1-19609-v1) $ gcloud compute disks create --size=10GiB --zone=us-west2-a mongodisk
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/disks#performance.
Created [https://www.googleapis.com/compute/v1/projects/signature-project1-19609-v1/zones/us-west2-a/disks/mongodisk].
NAME: mongodisk
ZONE: us-west2-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
```

Now we will create a load balance service for mongodb deployment and we will deploy mongodb.

Here is the YAML (mongodb-service.yaml) file for mongodb load-balancer-

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

And here is the YAML (mongodb.yaml) file for mongodb deployment -

```
apiVersion: apps/v1
kind: Deployment
```

```

metadata:
  name: mongodb-deployment
spec:
  selector:
    matchLabels:
      app: mongodb
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      volumes:
        - name: mongodb-data
          gcePersistentDisk:
            pdName: mongodisk
            fsType: ext4
      containers:
        - image: mongo
          name: mongodb
          volumeMounts:
            - name: mongodb-data
              mountPath: /data/db
          ports:
            - containerPort: 27017
              protocol: TCP

```

Start minikube and enable ingress, cause we need it later. Then we create mongo db pod

```

$ minikube start
$ minikube addons enable ingress
$ kubectl apply -f sgnprj1/mongodb-service.yaml
$ kubectl create -f sgnprj1/mongodb.yaml

```

Here is the response of above commands -

```

@cloudshell:~ (signature-project1-19609-v1)$ 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% 95.54 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: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
@cloudshell:~ (signature-project1-19609-v1)$ 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
@cloudshell:~ (signature-project1-19609-v1)$ kubectl create -f sgnprj1/mongodb.yaml
pod/mongopod created
@cloudshell:~ (signature-project1-19609-v1)$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
mongopod  0/1     ContainerCreating   0           6s

```

And here is the mongodb service with IP -

```
@cloudshell:~ (signature-project1-19609-v1)$ kubectl get service
NAME                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
kubernetes           ClusterIP     10.40.0.1     <none>         443/TCP          121m
mongodb-service      LoadBalancer 10.40.12.35   34.94.54.39    27017:32449/TCP  6m14s
@cloudshell:~ (signature-project1-19609-v1)$
```

Now we need to check if mongodb can access by this service IP.

```
$ kubectl exec -it mongodb-deployment-68c6f84d89-hcsf4 -- bash
# mongo 34.94.54.39
```

Here is the response -

```
@cloudshell:~ (signature-project1-19609-v1)$ kubectl exec -it mongodb-deployment-68c6f84d89-hcsf4 -- bash
root@mongodb-deployment-68c6f84d89-hcsf4:/# mongo 34.94.54.39
MongoDB shell version v5.0.6
connecting to: mongodb://34.94.54.39:27017/test?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("ca6d9678-e87b-4409-9bad-e4828bcafc8e") }
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/
=====
The server generated these startup warnings when booting:
  2022-03-30T03:32:52.510+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
  2022-03-30T03:32:53.200+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()
-----
> exit
bye
root@mongodb-deployment-68c6f84d89-hcsf4:/# exit
exit
@cloudshell:~ (signature-project1-19609-v1)$
```

Now we need to create our application. We have two applications. They are -

1. Student server
2. Bookshelf

Here is the code for student server -

```
>> 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);

```

>> Dockerfile

```

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

```

>> studentserver-configmap.yaml

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  MONGO_URL: 34.82.46.209
  MONGO_DATABASE: mydb

```

>> studentserver-service.yaml

```

apiVersion: v1
kind: Service
metadata:
  name: studentserver-service
  annotations:
    cloud.google.com/neg: '{"ingress": true}'
spec:
  type: LoadBalancer
  ports:
    # service port in cluster
    - port: 8080
    # port to contact inside container
    targetPort: 8080
  selector:
    app: studentserver-deployment

```

```
>> studentserver-deployment.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: us-west2-docker.pkg.dev/signature-project1-19609-v1/cs571-
sgnprj1-repo/shkr19609-sgnprj1-student-server
            imagePullPolicy: Always
            name: studentserver-deployment
            command: [ "/bin/bash", "-ce", "tail -f /dev/null" ]
            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
```

## Here is the code for bookshelf application -

```
>> 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)

```

>>> Dockerfile

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

>> requirements.txt

```
Flask==2.1.0
Flask-PyMongo==2.3.0
```

>> bookshelf-configmap.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: bookshelf-config
data:
  # SERVICE_NAME.NAMESPACE.svc.cluster.local:SERVICE_PORT
  MONGO_URL: 34.82.46.209
  MONGO_DATABASE: mydb
```

>> bookshelf-service.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
```

>> 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: us-west2-docker.pkg.dev/signature-project1-19609-v1/cs571-
sgnprj1-repo/shkr19609-sgnprj1-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

```

## ***GCP commands to get output -***

Before proceed, we need to add some data into mongodb student db manually. Here is the code -

>> student-data-manual-input.js

```

var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://34.94.54.39/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);
  });
});

```

Now we need to execute with node. Before execute we need to install mongodb into GCP folder. Here is the commands -



```

$ cd sgnprj1
$ vi student-data-manual-input.js
$ npm install mongodb
$ node student-data-manual-input.js

```

Also we need our common ingress service. Here is the ingress YAML (sgnprg1-ingress.yaml) file -

```

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: sgnprg1-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
  rules:
    - host: cs571.signature.project1.local
      http:
        paths:
          - path: /studentserver(/|$)(.*)
            pathType: Prefix
            backend:
              service:
                name: studentserver-service
                port:
                  number: 8080
          - path: /bookshelf(/|$)(.*)
            pathType: Prefix
            backend:
              service:
                name: bookshelf-service
                port:
                  number: 5000

```

Now we are ready to build and deploy our applications. Here is the commands -

```

$ export PRJ_IMG_ROOT=us-west2-docker.pkg.dev
$ gcloud artifacts repositories create cs571-sgnprj1-repo --repository-
format=docker --location=us-west2 --description="Cloud Computing Signature
Project Repository for SFBU, MSCS, Spring 2022, SHKR"
$ gcloud auth configure-docker ${PRJ_IMG_ROOT}
-- build student server and push to repository
$ docker build -t ${PRJ_IMG_ROOT}/${PROJECT_ID}/cs571-sgnprj1-repo/shkr19609-
sgnprj1-student-server ./sgnprj1/student-server/
$ docker push ${PRJ_IMG_ROOT}/${PROJECT_ID}/cs571-sgnprj1-repo/shkr19609-
sgnprj1-student-server
-- build bookshelf and push to repository
$ docker build -t ${PRJ_IMG_ROOT}/${PROJECT_ID}/cs571-sgnprj1-repo/shkr19609-
sgnprj1-bookshelf ./sgnprj1/bookshelf/
$ docker push ${PRJ_IMG_ROOT}/${PROJECT_ID}/cs571-sgnprj1-repo/shkr19609-
sgnprj1-bookshelf

$ kubectl apply -f sgnprj1/student-server/studentserver-configmap.yaml
$ kubectl apply -f sgnprj1/student-server/studentserver-deployment.yaml
$ kubectl apply -f sgnprj1/student-server/studentserver-service.yaml

$ kubectl apply -f sgnprj1/bookshelf/bookshelf-configmap.yaml
$ kubectl apply -f sgnprj1/bookshelf/bookshelf-deployment.yaml

```

```
$ kubectl apply -f sgnprj1/bookshelf/bookshelf-service.yaml
```

```
$ kubectl apply -f sgnprj1/sgnprj1-ingress.yaml
```

After executed all the command above following response we found -

```
@cloudshell:~ (signature-project1-19609-v1)$ kubectl get ingress
NAME          CLASS      HOSTS                                ADDRESS      PORTS      AGE
sgnprj1-ingress  <none>     cs571.signature.project1.local      80          4s

@cloudshell:~ (signature-project1-19609-v1)$ kubectl get service
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
bookshelf-service  LoadBalancer  10.40.12.236   <pending>      5000:31929/TCP    47m
kubernetes       ClusterIP      10.40.0.1      <none>         443/TCP           27h
mongodb-service   LoadBalancer  10.40.12.35    34.94.54.39    27017:32449/TCP   25h
studentserver-service LoadBalancer  10.40.10.21    <pending>      8080:30890/TCP    47m

@cloudshell:~ (signature-project1-19609-v1)$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
bookshelf-deployment-589768865f-mrmlv  1/1     Running   0          47m
mongodb-deployment-68c6f84d89-hcsf4    1/1     Running   0          25h
studentserver-deployment-5c648f8bb5-7cjjq 1/1     Running   0          48m

@cloudshell:~ (signature-project1-19609-v1)$
```

Now we have to wait until our application propagate and GCP assign IP for our applications.

We will able to access this application using following endpoints by curl -

```
>> student server:
```

```
curl http://cs571.signature.project1.local/studentserver/api/score?
student_id=11111
```

```
>> bookshelf: this is a REST server, we need to post data here..
```

```
1. To get the list of books -
```

```
curl http://cs571.signature.project1.local/bookshelf/books
```

```
2. Add a book into server -
```

```
curl -X POST -d '{"book_name": "cloud computing","book_author":
"unkown","isbn": "123456"}'
http://cs571.signature.project1.local/bookshelf/book
```

```
3. Update a book -
```

```
curl -X PUT -d '{"book_name": "123","book_author": "test","isbn":
"123updated"}' http://cs571.signature.project1.local/bookshelf/book/id
```

```
4. Delete a book -
```

```
curl -X DELETE http://cs571.signature.project1.local/bookshelf/book/id
```

As the domain is local, we need to add and entry into “/etc/hosts” file for corresponding IP. As the propagation still in progress, I don’t have IP in my hand. So, I ignore this part.

Github Link: <https://github.com/mkhan-sfbu/cs571-signature-project1>