MongoDB + Python Flask Web Framework + RESTAPI + GKE

By Shruti Kavishwar
San Fransisco Bay University
Guided by: Prof. Henry Chang

Project Overview

- **Objective:** Create and manage a MongoDB database using Google Kubernetes cluster and develop a python Flask web application with REST API, deployed on GKE
- **Components**
  - **MongoDB** setup on **GKE**
  - **REST API** development with **Flask**
  - Application deployment on GKE

Setting Up MongoDB

Steps:

- Create a Google Kubernetes Cluster on Google Cloud Platform
  - $ gcloud container clusters create kubia --num-nodes=1 --machine-type=e2-micro --region=us-west1
- Create a Disk for Mongodb
  - $ gcloud compute disks create --size=10GiB --zone=us-west1-a mongodb

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ gcloud c
ontainer clusters create kubia --num-nodes=1 --machine-type=e2-micro --region
=us-west1
Default change: VPC-native is the default mode during cluster creation for ve
rsions greater than 1.21.0-gke.1500. To create advanced routes based clusters
, please pass the `--no-enable-ip-alias` flag
Note: The Kubelet readonly port (10255) is now deprecated. Please update your
 workloads to use the recommended alternatives. See https://cloud.google.com/
kubernetes-engine/docs/how-to/disable-kubelet-readonly-port for ways to check
 usage and for migration instructions.
Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most
1008 node(s).
Creating cluster kubia in us-west1... Cluster is being health-checked (master
 is healthy)...done.

Created [https://container.googleapis.com/v1/projects/cs571-cloude-computing/
zones/us-west1/clusters/kubia].
To inspect the contents of your cluster, go to: https://console.cloud.google.
com/kubernetes/workload_/gcloud/us-west1/kubia?project=cs571-cloude-computing
kubeconfig entry generated for kubia.
NAME: kubia
LOCATION: us-west1
MASTER_VERSION: 1.29.6-gke.1038001
MASTER_IP: 35.247.66.163
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.29.6-gke.1038001
NUM_NODES: 3
STATUS: RUNNING
```

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ gcloud c
ompute disks create --size=10GiB --zone=us-west1-a mongodb
WARNING: You have selected a disk size of under [200GB]. This may result in p
oor I/O performance. For more information, see: https://developers.google.com
/compute/docs/disks#performance.
Created [https://www.googleapis.com/compute/v1/projects/cs571-cloude-computin
g/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

skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$
```

MongoDB Deployment

Steps:

- Create a Persistent Volume and a Persistent Volume Claim yaml and apply
  - # kubectl apply -f persistent-volume-mongo.yaml
  - # kubectl apply -f persistent-volume-claim-mongo.yaml
- Create a MongoDB deployment YAML (mongodb-deployment.yaml)
- Apply the deployment using
  - # kubectl apply -f mongodb-deployment.yaml
- Create a Service YAML for mongodb to be access from outside.
  - # kubectl apply -f mongodb-service.yaml
- Verify if the service and pod are created.
- Try accessing the MongoDB shell mongosh
  - # kubectl exec -it <mongodb_pod> - - mongosh

YAML file for PV and PVC

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mongodb-pv
spec:
  capacity:
    storage: 10Gi  # Adjust size according to your needs
  accessModes:
    - ReadWriteOnce
  gcePersistentDisk:
    pdName: mongodb  # Ensure this matches your actual GCE PD name
    fsType: ext4
```

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mongodb-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi  # This should match the size defined in the PV
```

YAML files for MongoDB deployment and Service

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

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

Verify pods and service

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ kubectl
get pods
NAME                                READY   STATUS    RESTARTS   AGE
mongodb-deployment-b7579f455-twxrd   1/1     Running   0          95s
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$
```

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ kubectl
get svc
NAME              TYPE           CLUSTER-IP       EXTERNAL-IP    PORT(S)
      AGE
kubernetes        ClusterIP      34.118.224.1     <none>         443/TCP
      22m
mongodb-service   LoadBalancer   34.118.229.107   34.83.243.86   27017:30153/
TCP   93s
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$
```

Accessing MongoDB using External IP

Make sure to install below packages to get this working

- $ wget https://downloads.mongodb.com/compass/mongosh-1.10.1-linux-x64.tgz
- $ tar -zxvf mongosh-1.10.1-linux-x64.tgz
- $ sudo cp mongosh-1.10.1-linux-x64/bin/mongosh /usr/local/bin/

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ mongosh
mongodb://34.145.93.62:27017
Current Mongosh Log ID: 66a85f6d450c62913032b1e6
Connecting to:          mongodb://34.145.93.62:27017/?directConnection=true&a
ppName=mongosh+1.10.1
Using MongoDB:          7.0.12
Using Mongosh:          1.10.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/


To help improve our products, anonymous usage data is collected and sent to M
ongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

------
   The server generated these startup warnings when booting
   2024-07-30T03:30:46.775+00:00: Using the XFS filesystem is strongly recomm
ended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/
prodnotes-filesystem
   2024-07-30T03:30:47.799+00:00: Access control is not enabled for the datab
ase. Read and write access to data and configuration is unrestricted
   2024-07-30T03:30:47.799+00:00: vm.max_map_count is too low
------

test>
```

Adding Records to MongoDB

Steps:

- Create a file insert-students.js or simply the mongosh to connect with the database and insert the records
  - $ node insert-students.js

```
const { MongoClient } = require('mongodb');

async function run() {
  const url = "mongodb://35.197.115.51/studentdb";
  const client = new MongoClient(url);

  try {
    // Connect to the MongoDB cluster
    await client.connect();

    // Specify the database and collection
    const db = client.db("studentdb");
    const collection = db.collection("students");

    // Create documents to be inserted
    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 }
    ];
```

Adding Records to MongoDB

```
    // Insert the documents
    const insertResult = await collection.insertMany(docs);
    console.log(`${insertResult.insertedCount} documents were inserted`);

    // Find one document
    const result = await collection.findOne({ student_id: 11111 });
    console.log(result);
  } finally {
    // Close the connection
    await client.close();
  }
}
```

Adding Records to MongoDB

```
> run();
Promise {
  <pending>,
  [Symbol(async_id_symbol)]: 106,
  [Symbol(trigger_async_id_symbol)]: 6
}
> 3 documents were inserted
{
  _id: new ObjectId('66a8676a3819efa7a16ddf66'),
  student_id: 11111,
  student_name: 'Bruce Lee',
  grade: 84
}
```

Deploying Student Server on GKE

Steps:

- Create a student server to access step2.js to access the records from the DB.
- Create a Dockerfile to run the student server in the container
- Build the docker image
  - # docker build -t shrutik2/step2 .
- Tag the docker image
  - # docker tag <image_id> shruti/step2:latest
  - Find the image id using # docker images
- Push the docker image to DockerHub
  - # docker push shrutik2/step2

Creating Student Server script to access Records

```
const http = require('http');
const url = require('url');
const { MongoClient } = require('mongodb');

const { MONGO_URL, MONGO_DATABASE } = process.env;

if (!MONGO_URL || !MONGO_DATABASE) {
  console.error('Environment variables MONGO_URL and MONGO_DATABASE must be set');
  process.exit(1);
}

const uri = `mongodb://${MONGO_URL}/${MONGO_DATABASE}`;
console.log(`Connecting to MongoDB at ${uri}`);

const server = http.createServer(async (req, res) => {
  const parsedUrl = url.parse(req.url, true);
  const student_id = parseInt(parsedUrl.query.student_id);

  if (isNaN(student_id)) {
    res.writeHead(400, { 'Content-Type': 'text/plain' });
    res.end("Invalid student_id, please provide a valid number\n");
    return;
  }

  if (/^\/api\/score/.test(req.url)) {
    try {
      const client = await MongoClient.connect(uri);
      const db = client.db();
      const student = await db.collection("students").findOne({ "student_id": student_id });

      if (student) {
        res.writeHead(200, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify({
          student_id: student.student_id,
          student_name: student.student_name,
          student_score: student.grade,
```

Creating Student Server script to access Records

```
            student_score: student.grade,
          }));
        } else {
          res.writeHead(404, { 'Content-Type': 'text/plain' });
          res.end("Student Not Found\n");
        }

        client.close();
      } catch (err) {
        console.error('Database error:', err);
        res.writeHead(500, { 'Content-Type': 'text/plain' });
        res.end("Internal Server Error\n");
      }
    } else {
      res.writeHead(404, { 'Content-Type': 'text/plain' });
      res.end("Wrong URL, please try again\n");
    }
});

server.listen(8080, () => {
  console.log('Server listening on port 8080');
});
```

Creating Dockerfile Building the image

```
FROM node:18

# Set the working directory
WORKDIR /app

# Copy package.json and package-lock.json first
COPY package*.json ./

# Install dependencies
RUN npm install

# Copy the rest of your application code
COPY . .

# Expose port
EXPOSE 8080

# Start the application
CMD ["node", "step2.js"]
```

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ docker build -t shrutik2/step2 .
[+] Building 0.4s (10/10) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 224B
 => [internal] load metadata for docker.io/library/node:14
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/5] FROM docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461aa
 => [internal] load build context
 => => transferring context: 30B
 => CACHED [2/5] WORKDIR /app
 => CACHED [3/5] COPY step2.js /app/step2.js
 => CACHED [4/5] RUN npm init -y
 => CACHED [5/5] RUN npm install mongodb
 => exporting to image
 => => exporting layers
 => => writing image sha256:a3868c936db2f8b9e3de19e8125ea0c3186f4ad3530c21917e846eb01e24c12b
 => => naming to docker.io/shrutik2/step2
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ docker push shrutik2/step2
Using default tag: latest
The push refers to repository [docker.io/shrutik2/step2]
1faa8649e5d5: Mounted from shrutik2/571-20022-sfbu
e5b84b4815f9: Mounted from shrutik2/571-20022-sfbu
d28c5bb325ef: Mounted from shrutik2/571-20022-sfbu
50e69ff23347: Mounted from shrutik2/571-20022-sfbu
0d5f5a015e5d: Mounted from shrutik2/571-20022-sfbu
3c777d951de2: Mounted from shrutik2/571-20022-sfbu
f8a91dd5fc84: Mounted from shrutik2/571-20022-sfbu
cb81227abde5: Mounted from shrutik2/571-20022-sfbu
e01a454893a9: Mounted from shrutik2/571-20022-sfbu
c45660adde37: Mounted from shrutik2/571-20022-sfbu
fe0fb3ab4a0f: Mounted from shrutik2/571-20022-sfbu
f1186e5061f2: Mounted from shrutik2/571-20022-sfbu
b2dba7477754: Mounted from shrutik2/571-20022-sfbu
latest: digest: sha256:3cd35a061c7178f077b095208ebb56de9377d9af9c9475393078054bdbbb464d size: 3046
```

Flask Bookshelf REST API

Steps:

- Create bookshelf.py,
- Create requirements.txt
- Create Dockerfile
- Build the Docker image

Flask Bookshelf REST API

```
Flask==2.2.2
Flask-PyMongo==2.3.0
pymongo==4.4.0
```

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

Flask Bookshelf REST API

```
skavishw276@cloudshell:~/signature_project/bookshelf (cs571-cloude-computing)
$ docker build -t shrutik2/bookshelf .
[+] Building 8.7s (9/9) FINISHED                                docker:default
 => [internal] load build definition from Dockerfile                      0.0s
 => => transferring dockerfile: 204B                                      0.0s
 => [internal] load metadata for docker.io/library/python:alpine3.7       0.1s
 => [internal] load .dockerignore                                         0.0s
 => => transferring context: 2B                                           0.0s
 => [internal] load build context                                         0.0s
```

```
skavishw276@cloudshell:~/signature_project/bookshelf (cs571-cloude-computing)
$ docker tag cfa1af15dee5 shrutik2/bookshelf:latest
```

```
skavishw276@cloudshell:~/signature_project/bookshelf (cs571-cloude-computing)
$ docker push shrutik2/bookshelf
Using default tag: latest
The push refers to repository [docker.io/shrutik2/bookshelf]
bc4559d9b192: Pushed
5f70bf18a086: Pushed
79a39072c2f2: 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: sha256:728fce49722d34e2b4dd96c51254e9b668ac93ea5b5994a846e53d
d9a321e13a size: 1993
```

ConfigMap for applications

Steps:

- Create ConfigMaps for student server application and bookshelf application
- 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

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  MONGO_URL: 35.197.115.51
  MONGO_DATABASE: studentdb
```

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-
apiVersion: v1
kind: ConfigMap
metadata:
  name: bookshelf-config
data:
 # SERVICE_NAME.NAMESPACE.svc.cluster.local:SERVICE_PORT
  MONGO_URL: 35.197.115.51
  MONGO_DATABASE: studentdb
```

Exposing Application via Ingress

Steps:

- Use Nginx Ingress to expose application
- Create Deployment files for both the applications (studentserver-deployment.yaml , bookshelf-deployment.yaml)
- Create service files for both the deployments (studentserver-service.yaml, bookshelf-service.yaml)
- Start Minikube
  - # minikube start
- Enable Ingress
  - # minikube addons enable ingress
- Deploy services and verify if the resources are created correctly
  - # kubectl apply -f studentserver-deployment.yaml
  - # kubectl apply -f bookshelf-deployment.yaml
  - # kubectl apply -f studentserver-service.yaml
  - # kubectl apply -f bookshelf-service.yaml

Exposing Application via Ingress

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ cat stud
entserver-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:
        - name: web
          image: shrutik2/step2
          imagePullPolicy: Always
          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
```

Exposing Application via Ingress

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ cat book
shelf-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:
        - name: bookshelf-deployment
          image: shrutik2/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
```

Exposing Application via Ingress

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ cat stud
entserver-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: web
spec:
  type: LoadBalancer
  ports:
    - port: 8080
      targetPort: 8080
  selector:
    app: web
```

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ cat book
shelf-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: bookshelf-service
spec:
  type: LoadBalancer
  ports:
    - port: 5000
      targetPort: 5000
  selector:
    app: bookshelf-deployment
```

Exposing Application via Ingress

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ minikube
 addons enable ingress
* ingress is an addon maintained by Kubernetes. For any concerns contact mini
kube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernet
es/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
```

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ kubectl
apply -f studentserver-deployment.yaml
deployment.apps/web created
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ kubectl
apply -f studentserver-configmap.yaml
error: error parsing studentserver-configmap.yaml: error converting YAML to J
SON: yaml: line 2: mapping values are not allowed in this context
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ vi stude
ntserver-configmap.yaml
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ kubectl
apply -f studentserver-configmap.yaml
configmap/studentserver-config created
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ kubectl
apply -f studentserver-service.yaml
service/web created
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$
```

Exposing Application via Ingress

```
skavishw276@cloudshell:~/signature_project/bookshelf (cs571-cloude-computing)
$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created
skavishw276@cloudshell:~/signature_project/bookshelf (cs571-cloude-computing)
$ kubectl apply -f bookshelf-configmap.yaml
error: error when retrieving current configuration of:
Resource: "/v1, Resource=configmaps", GroupVersionKind: "/v1, Kind=ConfigMap"
Name: "", Namespace: "default"
from server for: "bookshelf-configmap.yaml": resource name may not be empty
skavishw276@cloudshell:~/signature_project/bookshelf (cs571-cloude-computing)
$ vi bookshelf-configmap.yaml
skavishw276@cloudshell:~/signature_project/bookshelf (cs571-cloude-computing)
$ kubectl apply -f bookshelf-configmap.yaml
error: error validating "bookshelf-configmap.yaml": error validating data: ki
nd not set; if you choose to ignore these errors, turn validation off with --
validate=false
skavishw276@cloudshell:~/signature_project/bookshelf (cs571-cloude-computing)
$ vi bookshelf-configmap.yaml
skavishw276@cloudshell:~/signature_project/bookshelf (cs571-cloude-computing)
$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
skavishw276@cloudshell:~/signature_project/bookshelf (cs571-cloude-computing)
$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
skavishw276@cloudshell:~/signature_project/bookshelf (cs571-cloude-computing)
$
```

Exposing Application via Ingress

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ kubectl
get pods
NAME                                    READY   STATUS    RESTARTS   AGE
bookshelf-deployment-6755657589-72k85   1/1     Running   0          19m
web-c6bdf97c9-m47pk                     1/1     Running   0          6s
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$
```

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ kubectl
get deployments
NAME                   READY   UP-TO-DATE   AVAILABLE   AGE
bookshelf-deployment   1/1     1            1           21m
web                    1/1     1            1           2m29s
```

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ kubectl
get configmaps
NAME                 DATA   AGE
bookshelf-config     2      19m
kube-root-ca.crt     1      32m
studentserver-config 2      23m
```

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ kubectl
get svc
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)
     AGE
bookshelf-service   LoadBalancer   10.106.157.61   <pending>     5000:32572/T
CP   20m
kubernetes          ClusterIP      10.96.0.1       <none>        443/TCP
     33m
web                 LoadBalancer   10.105.199.15   <pending>     8080:30570/T
CP   23m
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$
```

Exposing Application via Ingress

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ cat stud
entserver-mongo-ingress.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: ImplementationSpecific
            backend:
              service:
                name: web
                port:
                  number: 8080
          - path: /bookshelf(/|$)(.*)
            pathType: ImplementationSpecific
            backend:
              service:
                name: bookshelf-service
                port:
                  number: 5000
```

Exposing Application via Ingress

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ kubectl
apply -f studentserver-mongo-ingress.yaml
ingress.networking.k8s.io/server configured
```

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ kubectl
get ingress
NAME     CLASS   HOSTS               ADDRESS        PORTS   AGE
server   nginx   cs571.project.com   192.168.49.2   80      3m37s
```

Accessing the application

- Add the host you created using ingress in /etc/hosts file

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ vi /etc/
hosts
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ sudo vi
/etc/hosts
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ cat /etc
/hosts | grep "192."
#       192.168.0.0     -   192.168.255.255
192.168.49.2 cs571.project.com
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$
```

Accessing the application

- Using curl command to access the API and retrieve a particular students information
  - # curl cs571.project.com/studentserver/api/score?student_id=11111
  - # curl cs571.project.com/studentserver/api/score?student_id=22222
  - # curl cs571.project.com/studentserver/api/score?student_id=33333
- Performing CRUD operations on Bookshelf application
  - Add Book: # curl -X POST -H "Content-Type: application/json" -d '{"book_name": "cloud computing", "book_author": "unknown", "ISBN": "123456"}' http://cs571.project.com/bookshelf/book
  - Update Book: # curl -X PUT -H "Content-Type: application/json" -d '{"book_name": "Updated Book Name", "book_author": "Updated Author", "ISBN": "Updated ISBN"}' http://cs571.project.com/bookshelf/book/66a91c1c4e229f7d052a2c81
  - Delete book: # curl -X DELETE http://cs571.project.com/bookshelf/book/66a91c1c4e229f7d052a2c81

Accessing the Student Server

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ curl cs5
71.project.com/studentserver/api/score?student_id=11111
{"student_id":11111,"student_name":"Bruce Lee","student_score":84}skavishw276
@cloudshell:~/signature_project (cs571-cloude-computing)$
```

```
@cloudshell:~/signature_project (cs571-cloude-computing)$ curl cs571.project.
com/studentserver/api/score?student_id=22222
{"student_id":22222,"student_name":"Jackie Chen","student_score":93}skavishw2
76@cloudshell:~/signature_project (cs571-cloude-computing)$
```

```
76@cloudshell:~/signature_project (cs571-cloude-computing)$ curl cs571.projec
t.com/studentserver/api/score?student_id=33333
{"student_id":33333,"student_name":"Jet Li","student_score":88}skavishw276@cl
oudshell:~/signature_project (cs571-cloude-computing)$
```

Accessing the Bookshelf – ADD Operation

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ export M
ONGO_URL=35.197.115.51
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ export M
ONGO_DATABASE=studentdb
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$
```

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ curl -X
POST -H "Content-Type: application/json" -d '{"book_name": "cloud computing",
 "book_author": "unknown", "ISBN": "123456"}' http://cs571.project.com/booksh
elf/book
{
  "message": "Book saved successfully!"
}
```

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ curl cs5
71.project.com/bookshelf/books
[
  {
    "Book Author": "unknown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "66a91c1c4e229f7d052a2c81"
  }
]
```

Accessing the Bookshelf – UPDATE Operation

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ curl -X
PUT -H "Content-Type: application/json" \
-d '{"book_name": "Updated Book Name", "book_author": "Updated Author", "ISBN
": "Updated ISBN"}' \
http://cs571.project.com/bookshelf/book/66a91c1c4e229f7d052a2c81
{
  "message": "Book updated successfully!"
}
```

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ curl cs5
71.project.com/bookshelf/books
[
  {
    "Book Author": "Updated Author",
    "Book Name": "Updated Book Name",
    "ISBN": "Updated ISBN",
    "id": "66a91c1c4e229f7d052a2c81"
  }
]
```

Accessing the Bookshelf – DELETE Operation

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ curl -X
DELETE http://cs571.project.com/bookshelf/book/66a91c1c4e229f7d052a2c81
{
  "message": "Book deleted successfully!"
}
```

```
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$ curl cs5
71.project.com/bookshelf/books
[]
skavishw276@cloudshell:~/signature_project (cs571-cloude-computing)$
```

Conclusion

Summary of the Project

Project Overview: This project involved setting up a complete system using MongoDB, the Python Flask web framework, REST API, and deploying everything on Google Kubernetes Engine (GKE). The primary goals were to set up a persistent data storage solution, build a backend server capable of handling RESTful requests, and deploy these services in a scalable and manageable way using GKE.

Key Achievements:

- Successfully set up a MongoDB database with persistent volume storage on GKE.
- Developed a Python Flask application to serve as a REST API, interfacing with the MongoDB database.
- Dockerized the application, ensuring consistent deployment across different environments.
- Utilized ConfigMaps to manage configuration, facilitating easy updates without redeploying Docker images.
- Exposed the applications using Nginx Ingress, allowing for easy access and routing.

Conclusion

Challenges & Resolutions:

- **Handling External-IP for MongoDB:** Initially faced issues with MongoDB connections due to changing External-IP. This was resolved by using ConfigMaps to dynamically manage connection details.
- **Deployment and Configuration Management:** Managing configurations across multiple environments can be challenging. The use of ConfigMaps in Kubernetes simplified this process, allowing for secure and flexible configuration management.
- **Scalability and Load Management:** Ensuring the system could handle varying loads was crucial. Kubernetes' native features like auto-scaling and load balancing were leveraged to manage this aspect efficiently.

Conclusion

Learnings and Best Practices:

- **Importance of Infrastructure as Code (IaC):** Using tools like Kubernetes and Docker highlights the importance of IaC, allowing for consistent and repeatable deployments.
- **Monitoring and Logging:** Setting up proper monitoring and logging is vital for maintaining and troubleshooting the system. Implementing solutions like Prometheus and Grafana for monitoring and ELK stack for logging can provide valuable insights.
- **Security Considerations:** Managing sensitive data and configurations, ensuring secure communication between services, and implementing proper access controls were key security aspects addressed during the project.

Conclusion

Future Enhancements:

- **Enhanced Security Measures:** Implementing more robust security measures, such as network policies, secure API gateways, and encryption of data at rest and in transit.
- **Continuous Integration and Continuous Deployment (CI/CD):** Automating the deployment pipeline using CI/CD tools to streamline updates and feature releases.
- **Scaling and Performance Optimization:** Further optimizing the application's performance and scaling strategies to handle higher loads and improve response times.

GitHub Link:

https://github.com/ShrutiK02/Cloud-Computing/tree/b8c3e19b19f6f8747138e04f41f0a71175ae8f89/Kubernetes/MongoDB%20%2B%20Python%20Flask%20Web%20Framework%20%2B%20REST%20API%20%2B%20GKE