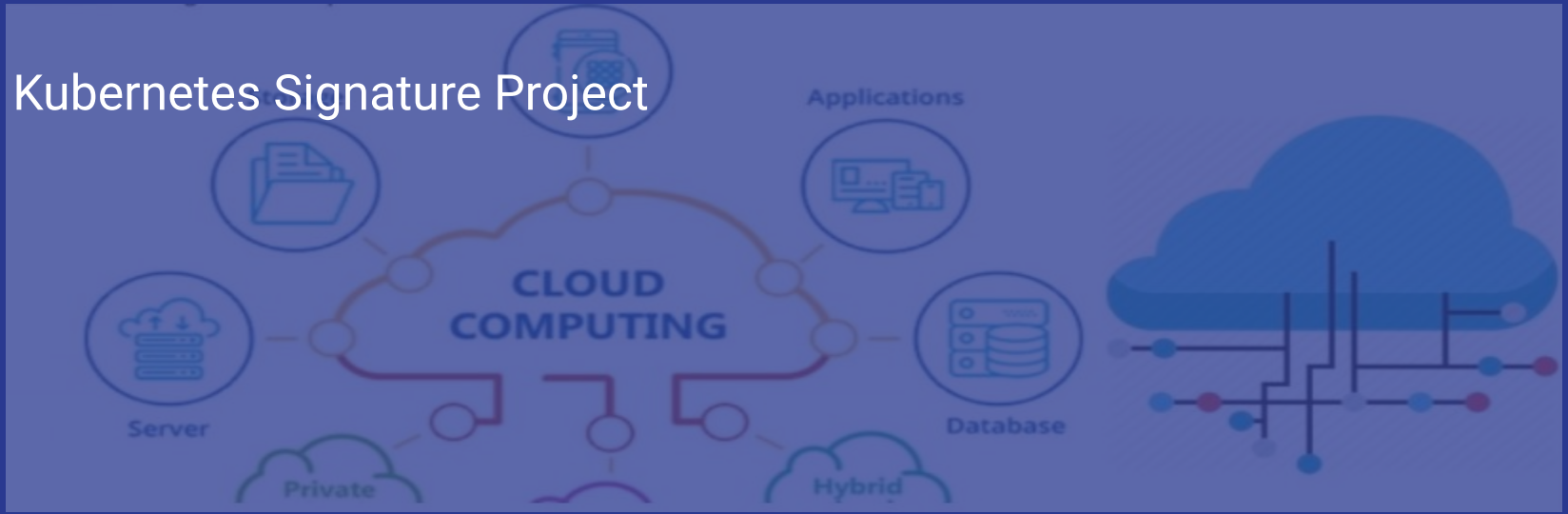
MongoDB + Python Flask Web Framework + REST API +GKE

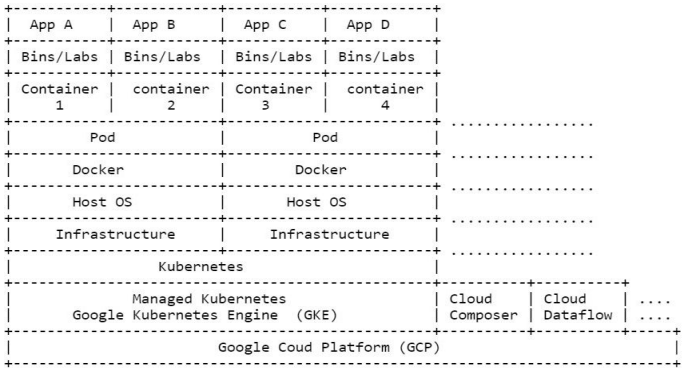
Kubernetes Signature Project



Shoumya Singh

Table of Content

Introduction



kubernetes

- Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.
- Kubernetes provides you with a framework to run distributed systems resiliently. It takes care of scaling and failover for your application, provides deployment patterns, and more.
- **In this project we use the concepts of Ingress ,ConfigMaps and Service ,on kubernetes to host two different applications but will it hosted under the same domain but in different path.**
- **Both the applications will be using MongoDB hosted on Google kubernetes Engine via Google Cloud Computing to stare the data.**

Table of Content

Project Description

This kubernetes project uses the following techniques to run on Google Kubernetes Engine.

kubernetes

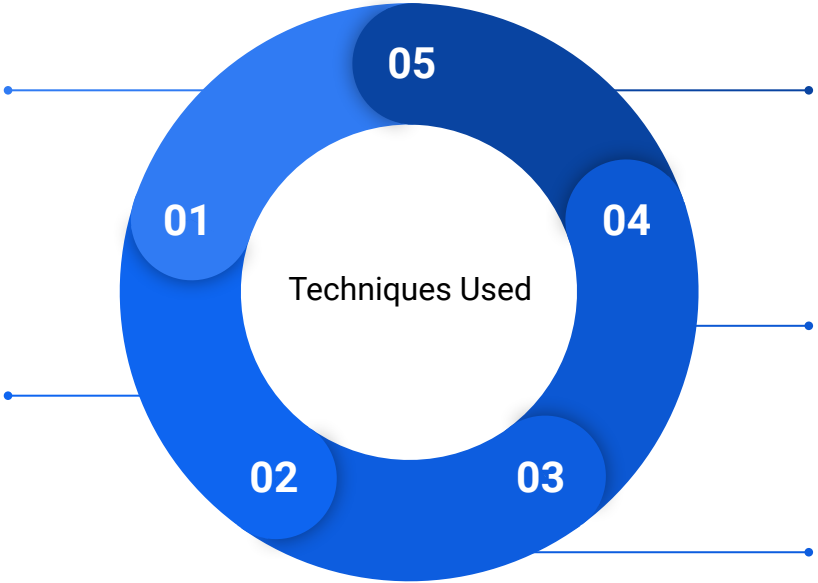
Pods - to run 2 applications

Pod 1 student records:Node.js webserver

Pod 2 Bookstore: MongoDB+Python Flask Web Framework+REST API

Service

for outside access of the application



ConfigMaps

to store MongoDB service address, in case MongoDB is down and restarts with a different service address, and with ConfigMaps, we don't need to build the docker image again with the new address.

Ingress

to expose both applications under same domain but different path.

Persistence Volumes

to store the data with MongoDB

Desired Results

1. **Access a student's Score**.
   - curl cs571.project.com/studentserver/api/score?student_id=11111

2. **List all the books**
   - curl cs571.project.com/bookshelf/books

3. **Add a book**
   - curl -X POST -d "{\"book_name\": \"cloud computing\",\"book_author\": \"unkown\", \"isbn\": \"123456\" }" http://cs571.project.com/bookshelf/book

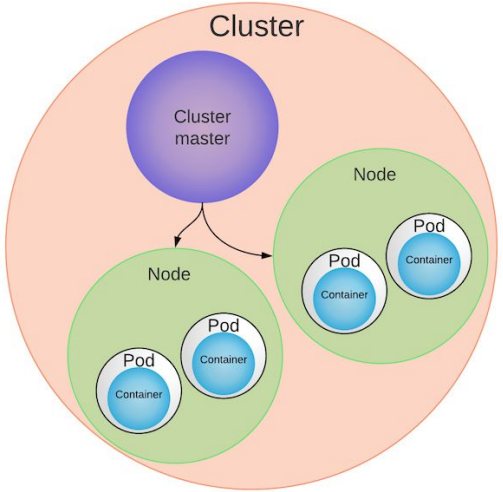
4. **Update a book**
   - curl -X PUT -d "{\"book_name\": \"123\",\"book_author\": \"test\", \"isbn\": \"123updated\" }" http://cs571.project.com/bookshelf/book/id

5. **Delete a book**
   - curl -X DELETE cs571.project.com/bookshelf/book/id

Table of Content

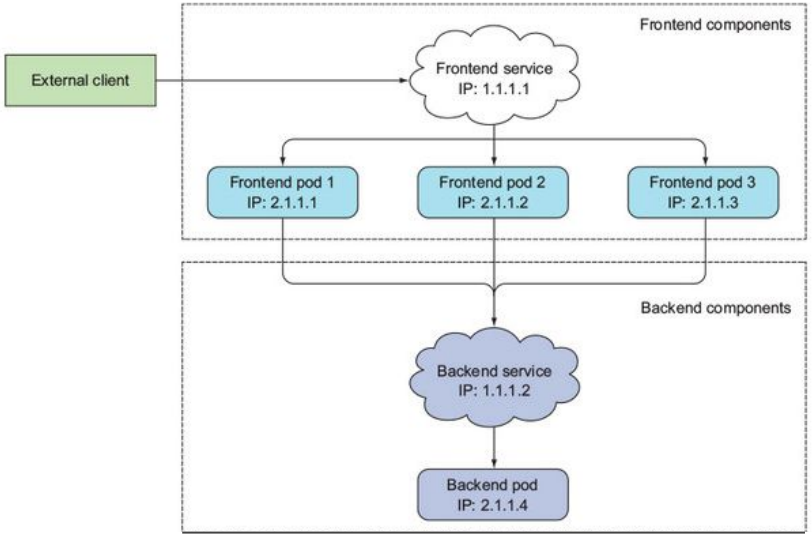
Pods

- A pod is building block of Kubernetes and core unit of management ,its a collection of one or more containers.
- Instead of deploying containers individually, we always deploy and operate on a pod of containers.
- At runtime, pods can be scaled by creating replica sets, which ensure that the deployment always runs the desired number of pods.



- In this project we will have 2 Pods
  - **Pod 1**: student records
    - Node.js web server that allows users to access certain student record from given student_id.
  - **Pod 2**: bookstore
    - MongoDB + Python Flask Web Framework + REST API to allow users to perform standard List, Insert, Update, Delete operation to the data stored inside MongoDB

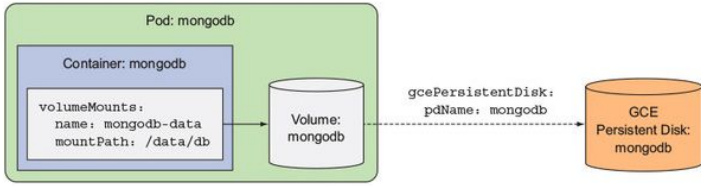
Service



- A Kubernetes Service is a resource you create to make a single, constant point of entry to a group of pods providing the same service.

- **Each service has an IP address and port that never change while the service exists.**
  - Clients can open connections to that IP and port, and those connections are then routed to one of the pods backing that service.
  - This way, clients of a service don't need to know the location of individual pods providing the service, allowing those pods to be moved around the cluster at any time.

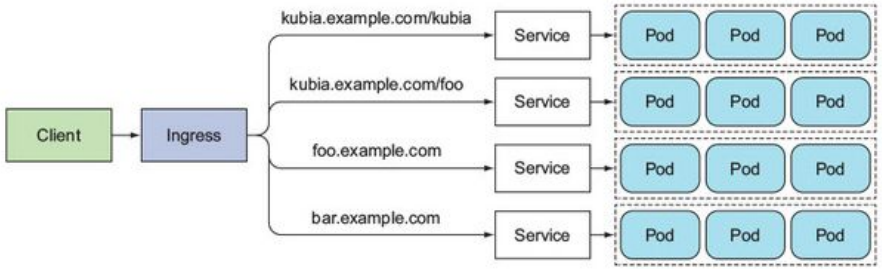
Persistence Volumes



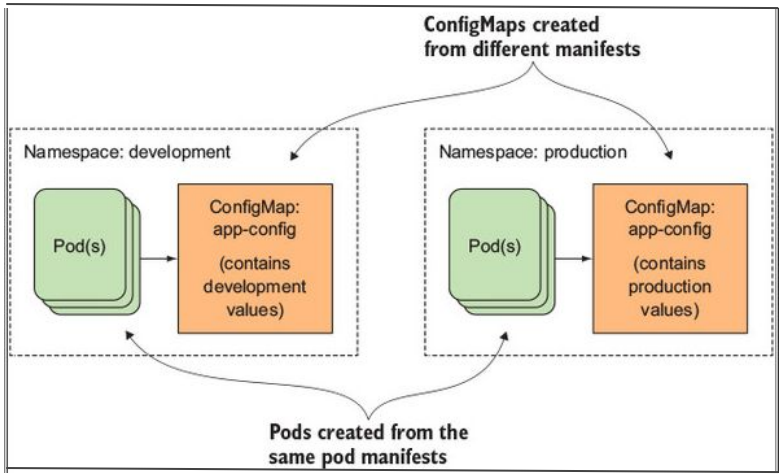
- Kubernetes volumes are a component of a pod and are thus defined in the pod's specification - much like containers.In each container, you can mount the volume in any location of its filesystem. A way to use a pre- or dynamically provisioned persistent storage.
- **If you've been running these examples on Google Kubernetes Engine, which runs your cluster nodes on Google Compute Engine (GCE), you'll use a GCE Persistent Disk as your underlying storage mechanism.**
- **MongoDB is the most popular NoSQL database management system.**
- A part of the NoSQL family of database systems.
  - An open source document-oriented database system
  - MongoDB stores data as JSON-like documents with dynamic schemas.
- Each MongoDB database has many collections.
  - A collection
    - The schema-free equivalent of a table.
  - Each collection has many documents.
  - A document is the basic unit of data
    - Equivalent to a row in a Relational Database Management System (RDMS).
    - It is an ordered set of keys with associated values.

Ingress

- Kubernetes provides isolation between pods and the outside world. If you want to communicate with a service running in a pod, you have to open up a channel for communication. This is referred to as ingress.
- After creating a cluster of nodes, and launching deployments of pods onto the cluster, you still need to add ingress to the cluster to allow external traffic to access your applications.
- There are multiple ways to add ingress to your cluster. The most common ways are by adding either an Ingress controller, or a Load Balancer.
- **In this Project, Ingress is used to expose both applications under same domain but different path.**



ConfigMaps

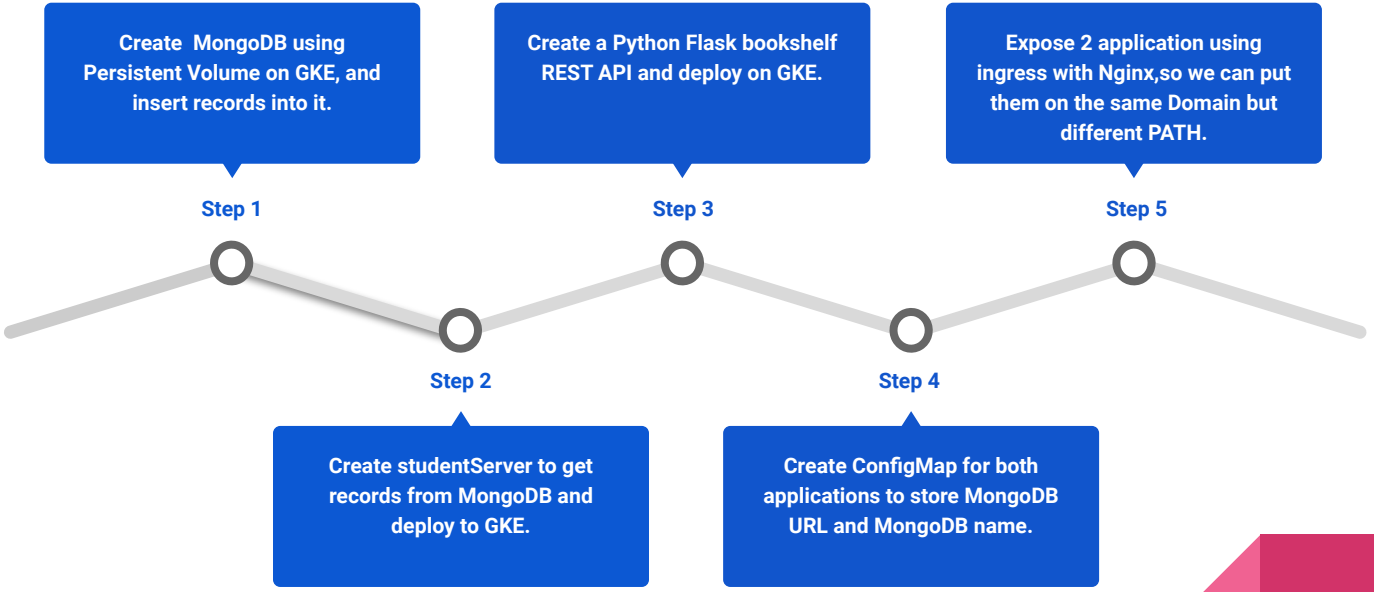


- **In this Project ,ConfigMaps is used to store MongoDB service address, in case MongoDB is down and restarts with a different service address, and with ConfigMaps, we don't need to build the docker image again with the new address.**

- Kubernetes allows separating configuration options into a separate object called a ConfigMap, which is a map containing key/value pairs with the values ranging from short literals to full config files.

Table of Content

Implementation



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

kubernetes

Step 1

- **Create a Kubernetes cluster on GKE.**
  - gcloud container clusters create kubia –num-nodes=1 - -machine-type=e2-micro –region=us-west1

```
NAME   LOCATION  MASTER_VERSION   MASTER_IP     MACHINE_TYPE  NODE_VERSION      NUM_NODES  STATUS
kubia  us-west1  1.18.16-gke.302  34.83.66.49   e2-micro      1.18.16-gke.302   3          RUNNING
singh19566@cloudshell:~ (cs571-demo-project-302019)$
```

- **Create a MongoDB Persistent Volume of 10 GiB according to your zone.**
  - gcloud compute disks create --size=10GiB --zone=us-west1-a mongodb

```
singh19566@cloudshell:~ (cs571-demo-project-302019)$ gcloud compute disks create --size=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,
Created [https://www.googleapis.com/compute/v1/projects/cs571-demo-project-302019/zones/us-west1-a/disks/mongodb].
NAME     ZONE        SIZE_GB  TYPE         STATUS
mongodb  us-west1-a  10       pd-standard  READY
```

- **Create a mongodb deployment with this yaml file.**
  - kubectl apply -f mongodb-deployment.yaml

```
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
```

- **Check if the deployment pod has been successfully created and started running**
  - Kubectl get pods

```
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ kubectl get pods
NAME                                 READY   STATUS    RESTARTS   AGE
mongodb-deployment-554cbb9965-gfgbt   1/1     Running   0          88s
singh19566@cloudshell:~/project (cs571-demo-project-302019)$
```

- **Create a service for the mongodb, so it can be accessed from outside**
  - kubectl apply -f mongodb-service.yaml

```
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
```

- **Wait couple of minutes, and then check if the service is up .**
  - kubectl get svc

```
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ kubectl get svc
NAME              TYPE           CLUSTER-IP     EXTERNAL-IP      PORT(S)           AGE
kubernetes        ClusterIP      10.3.240.1     <none>           443/TCP           39m
mongodb-service   LoadBalancer   10.3.245.195   35.227.171.243   27017:31765/TCP   76s
singh19566@cloudshell:~/project (cs571-demo-project-302019)$
```

- **Then insert some records into the mongodb via starting Node.js for later use**

```
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ node
Welcome to Node.js v12.14.1.
Type ".help" for more information.
> var MongoClient = require('mongodb').MongoClient;
undefined
> var url = "mongodb://35.227.171.243/mydb"
undefined
> // Connect to the db
undefined
> 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);
......... });
..... });
undefined
> 3
{
  _id: 606e9ab6f599f6061465e6c2,
  student_id: 11111,
  student_name: 'Bruce Lee',
  grade: 84
}
```

Step 2

Create studentServer to get records from MongoDB and deploy to GKE.

- **Create a studentServer to get records from MongoDB and deploy to GKE,then build the studentserver docker image.**
  - docker build -t yourdockerhubID/studentserver .


```
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ docker build -t shoumya/studentserver .
Sending build context to Docker daemon  3.386MB
Step 1/4 : FROM node:7
7: Pulling from library/node
ad74af05f5a2: Pull complete
2b032b8bbe8b: Pull complete
```

```
Successfully built 1edf2d7df24b
Successfully tagged shoumya/studentserver:latest
singh19566@cloudshell:~/project (cs571-demo-project-302019)$
```

- **Push the docker image**
  - docker push yourdockerhubID/studentserver

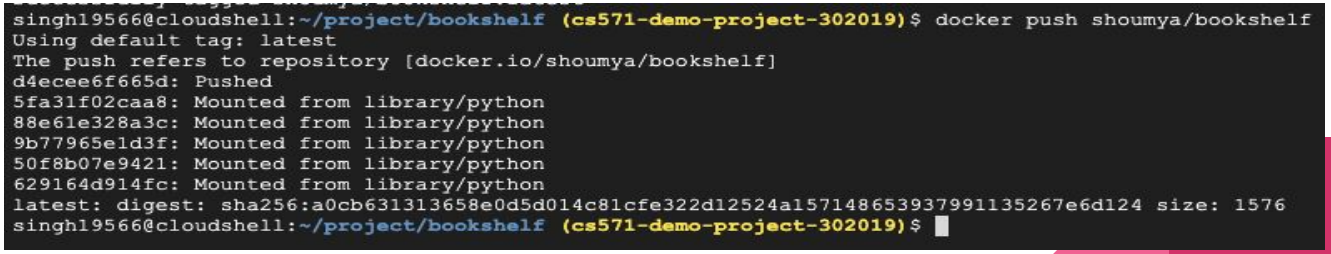
```
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ docker push shoumya/studentserver
Using default tag: latest
The push refers to repository [docker.io/shoumya/studentserver]
52cb1189e4c7: Pushed
f2d344da358b: Pushed
ab90d83fa34a: Mounted from library/node
8ee318e54723: Mounted from library/node
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:b9c50e5d7d3587b96fc270049f1ad16b4df7ad5c7a06d4b0da744efdcacb888c size: 2424
singh19566@cloudshell:~/project (cs571-demo-project-302019)$
```

Create a Python Flask bookshelf REST API and deploy on GKE.

Step 3

- **Build the bookshelf app into a docker image**
  - docker build -t shoumya/bookshelf .

- **Push the docker image to your dockerhub**
  - docker push yourdockerhubID/bookshelf



Step 4

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

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

kubernetes

- **Create a file named studentserver-configmap.yaml**

```
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ vi studentserver-configmap.yaml
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ cat studentserver-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  MONGO_URL: 35.227.171.243
  MONGO_DATABASE: mydb
```

- **Create a file named bookshelf-configmap.yaml**

```
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ vi bookshelf-configmap.yaml
singh19566@cloudshell:~/project (cs571-demo-project-302019)$ cat bookshelf-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: bookshelf-config
data:
  # SERVICE_NAME.NAMESPACE.svc.cluster.local:SERVICE_PORT
  MONGO_URL: 35.227.171.243
  MONGO_DATABASE: mydb
```

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

Step 5

- **Start minikube and addon Ingress**
  - minikube start
  - minikube addons enable ingress

kubernetes

```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ minikube start
* minikube v1.18.1 on Debian 10.9 (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 v1.20.2 preload ...
    > preloaded-images-k8s-v9-v1....: 491.22 MiB / 491.22 MiB  100.00% 169.68 M
* Creating docker container (CPUs=2, Memory=4000MB) ...
* Preparing Kubernetes v1.20.2 on Docker 20.10.3 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v4
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$
```

```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ minikube addons enable ingress
  - Using image us.gcr.io/k8s-artifacts-prod/ingress-nginx/controller:v0.40.2
  - Using image jettech/kube-webhook-certgen:v1.2.2
  - Using image jettech/kube-webhook-certgen:v1.3.0
* Verifying ingress addon...
* The 'ingress' addon is enabled
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$
```

- **Create studentserver related pods and start service using the above yaml file**

```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ kubectl apply -f studentserver-deployment.yaml
deployment.apps/web created
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ kubectl apply -f studentserver-configmap.yaml
configmap/studentserver-config created
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ kubectl apply -f studentserver-service.yaml
service/web created
```

- **Create bookshelf related pods and start service using the above yaml file**

```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
```

- **Check if all the pods are running correctly**

```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
bookshelf-deployment-646c59bd88-nhn2f   1/1     Running   0          8s
web-fcf9f666f-slgtx                   1/1     Running   0          40m
```

- **Create an ingress service yaml file called studentservermongoIngress.yaml**

```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ kubectl apply -f studentservermongoIngress.yaml
ingress.networking.k8s.io/server created
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$
```

- **Check if ingress is running**
  - kubectl get ingress

```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ kubectl get ingress
NAME      CLASS    HOSTS                ADDRESS         PORTS   AGE
server    <none>   cs571.project.com    192.168.49.2    80      2m38s
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$
```

- **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

```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ sudo vi /etc/hosts
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ cat /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-628292330722-default-boost-q5tcj
192.168.49.2 cs571.project.com
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$
```

Table of Content

Test Results

- **If everything goes smoothly, you should be able to access your applications**
  - 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

```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ curl cs571.project.com/studentserver/api/score?student_id=11111
{"_id":"606f3d523e09a2049d7a4199","student_id":11111,"student_name":"Bruce Lee","grade":84}
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ curl cs571.project.com/studentserver/api/score?student_id=22222
{"_id":"606f3d523e09a2049d7a419a","student_id":22222,"student_name":"Jackie Chen","grade":93}
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ curl cs571.project.com/studentserver/api/score?student_id=33333
{"_id":"606f3d523e09a2049d7a419b","student_id":33333,"student_name":"Jet Li","grade":88}
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$
```

- **On another path, we should be able to use the REST API with bookshelf application I.e list all books**
  - curl cs571.project.com/bookshelf/books

```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "606f574b6243ed661467f806"
  }
]
```

Test Results

- **Add a book**
  - curl -X POST -d "{\"book_name\": \"test\",\"book_author\": \"test_1\", \"isbn\": \"99999\" }" http://cs571.project.com/bookshelf/book

```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ curl -X POST -d "{\"book_name\": \"test\",\"book_author\":
\"test_1\", \"isbn\": \"99999\" }" http://cs571.project.com/bookshelf/book
{
  "message": "Task saved successfully!"
}
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "606f574b6243ed661467f806"
  },
  {
    "Book Author": "test_1",
    "Book Name": "test",
    "ISBN": "99999",
    "id": "606f58136243ed661467f807"
  }
]
```

Test Results

- **Update a book**
  - curl -X PUT -d “{\”book_name\”: \”123\”,\”book_author\”: \”test\”, \”isbn\”: \”123updated\” }” http://cs571.project.com/bookshelf/book/id

```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ curl -X PUT -d "{\"book_name\": \"test\",\"book_author\": \"test_1\", \"isbn\":
\"123updated\" }" http://cs571.project.com/bookshelf/book/606f58136243ed661467f807
{
  "message": "Task updated successfully!"
}
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "606f574b6243ed661467f806"
  },
  {
    "Book Author": "test_1",
    "Book Name": "test",
    "ISBN": "123updated",
    "id": "606f58136243ed661467f807"
  }
]
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$
```

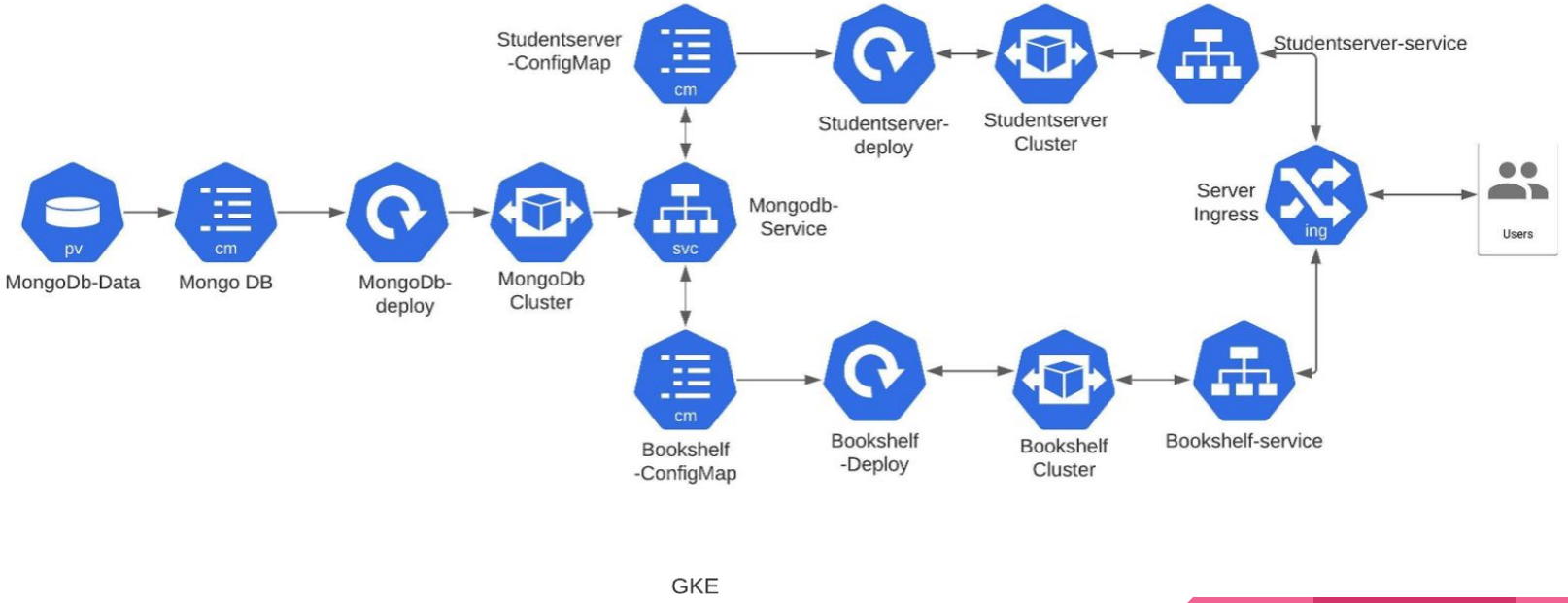
Test Results

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

```
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ curl -X DELETE cs571.project.com/bookshelf/book/606f58136243ed661467f807
{
  "message": "Task deleted successfully!"
}
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "606f574b6243ed661467f806"
  }
]
singh19566@cloudshell:~/project/bookshelf (cs571-demo-project-302019)$
```

Table of Content

Schematic of the Project



Enhancement Ideas

- ➔ To run or process under a real domain name and host both the applications for public test.
- ➔ To run both the applications under HTTPS protocol by adding TLS (Transport Layer Security)to Ingress

Conclusion

- The entire project is developed in Google Cloud Platform which provides different tools from which we are using Google Kubernetes Engine(GKE).
- The Project runs under the test domain **cs571.project.com**
- Its hosting 2 applications as :
  - studentserver
    - node.js server application.
  - bookshelf
    - Python REST API application.

Bibliography/References

- https://npu85.npu.edu/~henry/npu/classes/kubernetes_in_action/first_step/slide/index_slide.html
- https://npu85.npu.edu/~henry/npu/classes/kubernetes_in_action/service/slide/index_slide.html
- https://npu85.npu.edu/~henry/npu/classes/kubernetes_in_action/configmap/slide/index_slide.html
- https://npu85.npu.edu/~henry/npu/classes/kubernetes_in_action/service/slide/Creating_an_Ingress_resource.html
- https://lucid.app/lucidchart/
- https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/
- https://v1-19.docs.kubernetes.io/docs/tasks/access-application-cluster/ingress-minikube/