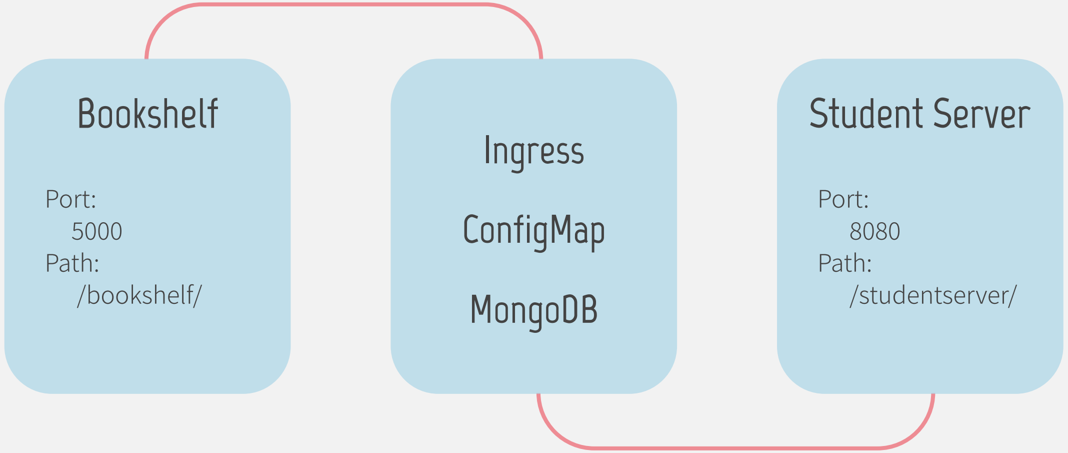
Cloud Computing
Kubernetes

MongoDB + Python Flask Web Framework + REST API + GKE

Avery Niou
2022

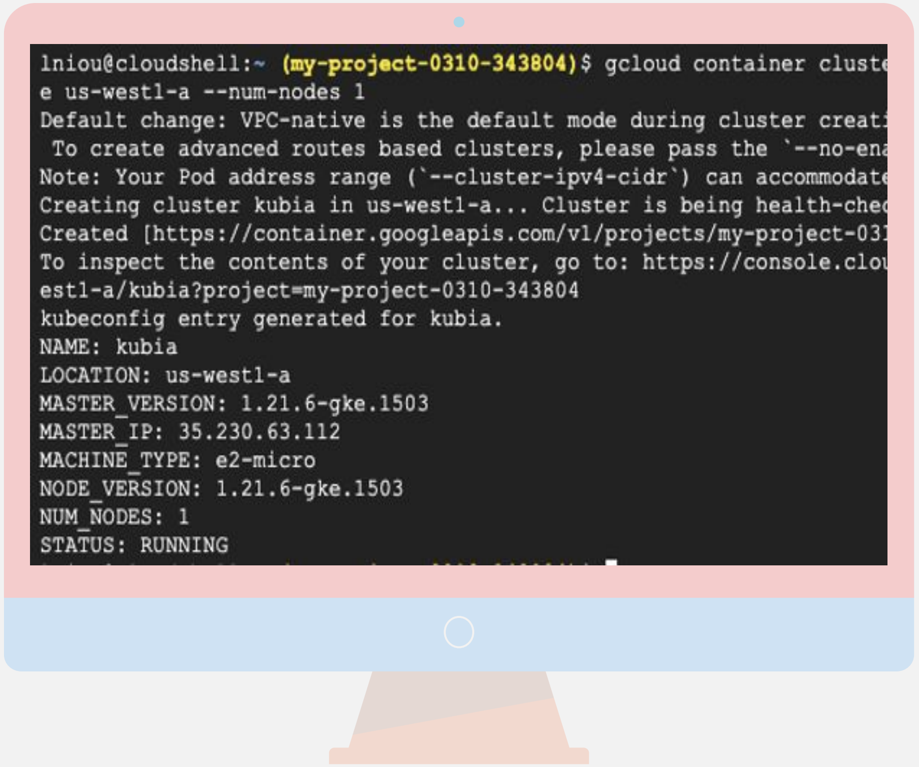
Project Overview

1. Using GCP to have Persistent Volume for MongoDB.
2. Create cluster on GKE.
3. Create MongoDB deployment pod and service get external IP.
4. Create simple node.js server and push on docker.
5. Create a python Flask bookshelf REST API and deploy on GKE.
6. Create ConfigMap for both applications to store MongoDB URL and MongoDB name.
7. Expose 2 application using ingress, we can put them on the same Domain but different path.



Create a cluster on GKE

```
$ gcloud container clusters create kubia
  --machine-type=e2-micro
  --zone us-west1-a
  --num-nodes 1
```

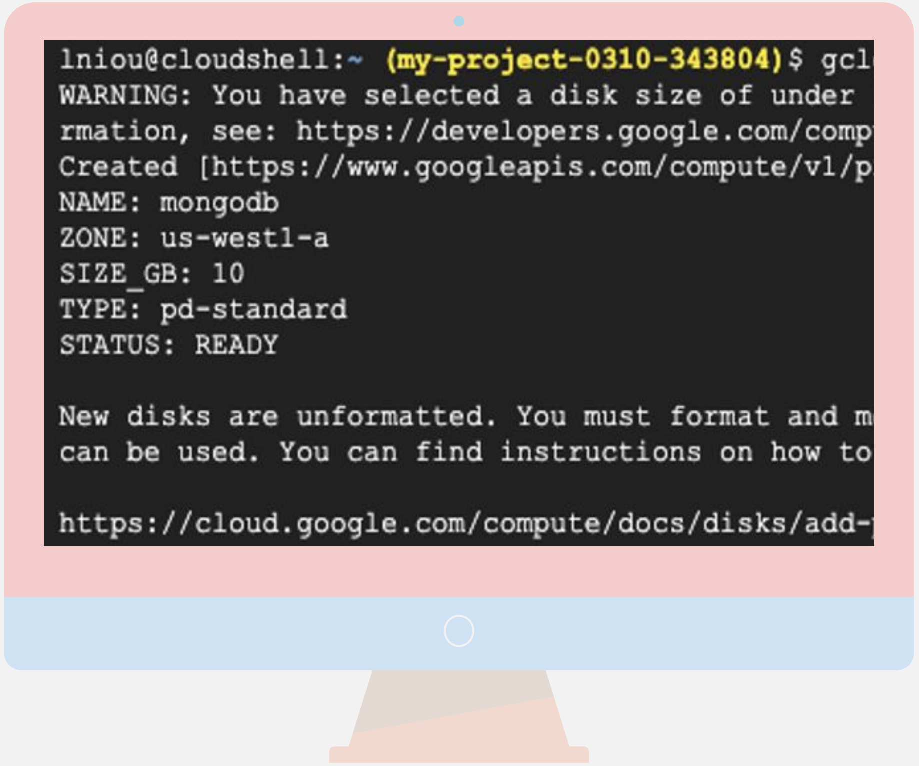


```
lniou@cloudshell:~ (my-project-0310-343804)$ gcloud container cluste
e us-west1-a --num-nodes 1
Default change: VPC-native is the default mode during cluster creat
 To create advanced routes based clusters, please pass the `--no-ena
Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate
Creating cluster kubia in us-west1-a... Cluster is being health-chec
Created [https://container.googleapis.com/v1/projects/my-project-031
To inspect the contents of your cluster, go to: https://console.clo
est1-a/kubia?project=my-project-0310-343804
kubeconfig entry generated for kubia.
NAME: kubia
LOCATION: us-west1-a
MASTER_VERSION: 1.21.6-gke.1503
MASTER_IP: 35.230.63.112
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.21.6-gke.1503
NUM_NODES: 1
STATUS: RUNNING
```

02

Create Persistent Volume on GCP

```
$ gcloud compute disks create
  --size=10GiB
  --zone=us-west1-a
  mongodb
```

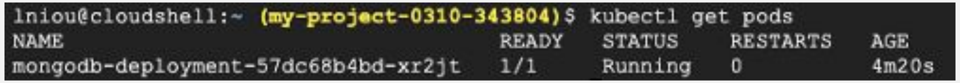


```
lniou@cloudshell:~ (my-project-0310-343804)$ gcl
WARNING: You have selected a disk size of under
rmation, see: https://developers.google.com/comp
Created [https://www.googleapis.com/compute/v1/p
NAME: mongodb
ZONE: us-west1-a
SIZE_GB: 10
TYPE: pd-standard
STATUS: READY

New disks are unformatted. You must format and m
can be used. You can find instructions on how to

https://cloud.google.com/compute/docs/disks/add-
```

Create a mongodb deployment.

```
$ kubectl apply -f mongodb_deployment.yaml
$ kubectl get pods
```

Wait until mongodb deployment pod is running



```
lniou@cloudshell:~ (my-project-0310-343804)$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
mongodb-deployment-57dc68b4bd-xr2jt 1/1     Running   0          4m20s
```

mongodb_deployment.yaml

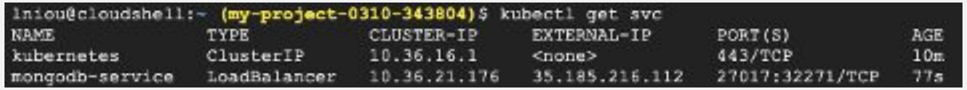
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
spec:
  selector:
       matchLabels:
       app: mongodb
  strategy:
       type: Recreate
  template:
       metadata:
       labels:
       app: mongodb
       spec:
       containers:
       - image: mongo
       name: mongo
       ports:
              - containerPort: 27017
       volumeMounts:
              - name: mongodb-data
              mountPath: /data/db
       volumes:
       - name: mongodb-data
       gcePersistentDisk:
              pdName: mongodb
                     fsType: ext4
```

04

Create service for MongoDB

```
$ kubectl apply -f mongodb-service.yaml
$ kubectl get svc
```

To get External-IP



```
lniou@cloudshell:~ (my-project-0310-343804)$ kubectl get svc
NAME              TYPE           CLUSTER-IP     EXTERNAL-IP      PORT(S)           AGE
kubernetes        ClusterIP      10.36.16.1     <none>           443/TCP           10m
mongodb-service   LoadBalancer   10.36.21.176   35.185.216.112   27017:32271/TCP   77s
```

mongodb-service.yaml

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

05

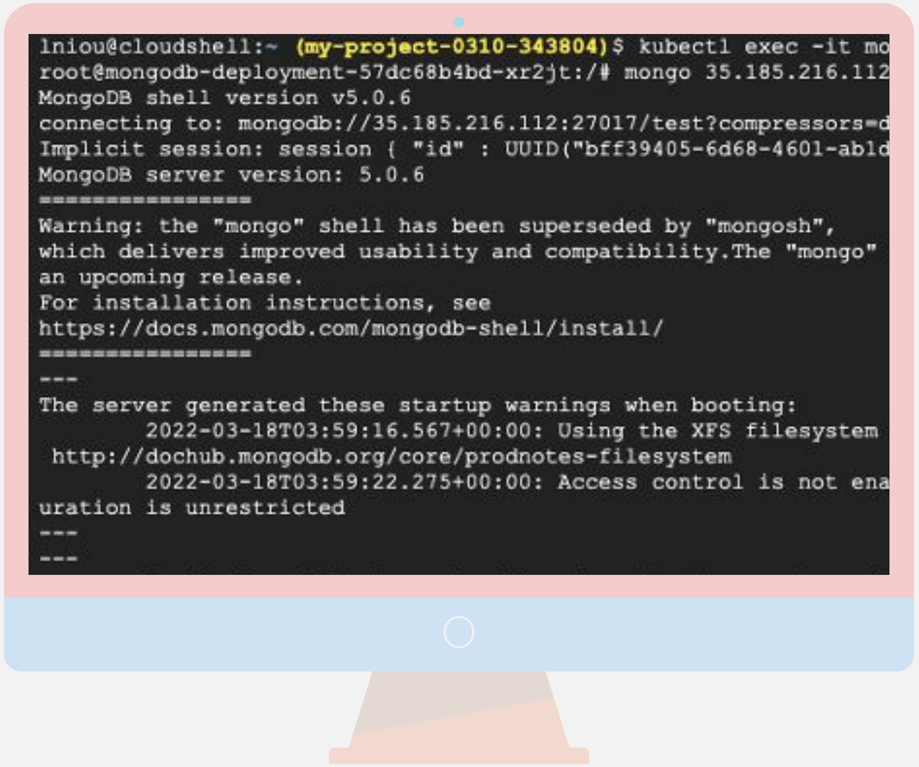
Try MongoDB connections

```
$ kubectl exec
  -it mongodb-deployment-57dc68b4bd-xr2jt
  -- bash
```

Pod name

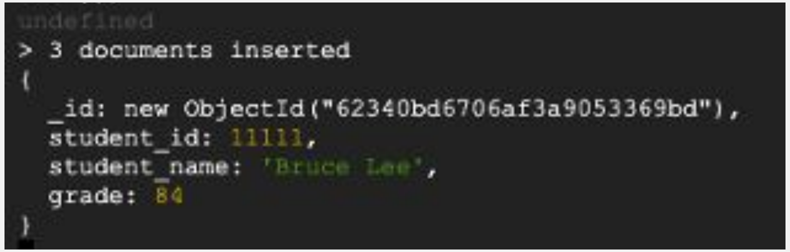
```
# mongo 35.185.216.112
```

External-IP



```
lniou@cloudshell:~ (my-project-0310-343804)$ kubectl exec -it mo
root@mongodb-deployment-57dc68b4bd-xr2jt:/# mongo 35.185.216.112
MongoDB shell version v5.0.6
connecting to: mongodb://35.185.216.112:27017/test?compressors=d
Implicit session: session { "id" : UUID("bff39405-6d68-4601-ab1d
MongoDB server version: 5.0.6
================
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility.The "mongo"
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
================
---
The server generated these startup warnings when booting:
        2022-03-18T03:59:16.567+00:00: Using the XFS filesystem
 http://dochub.mongodb.org/core/prodnotes-filesystem
        2022-03-18T03:59:22.275+00:00: Access control is not ena
uration is unrestricted
---
---
```

Insert records in MongoDB for later use



```
undefined
> 3 documents inserted
{
  _id: new ObjectId("62340bd6706af3a9053369bd"),
  student_id: 11111,
  student_name: 'Bruce Lee',
  grade: 84
}
```

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://35.185.216.112/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+" documents inserted");
       //client.close(); if you add this line it won't show findOne
       });
       db.collection("students").findOne({'student_id': 11111},
       function(err, result){
              console.log(result);
       });
});
```

07

Create a 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);
```

08

Build and push studentserver image

Dockerfile

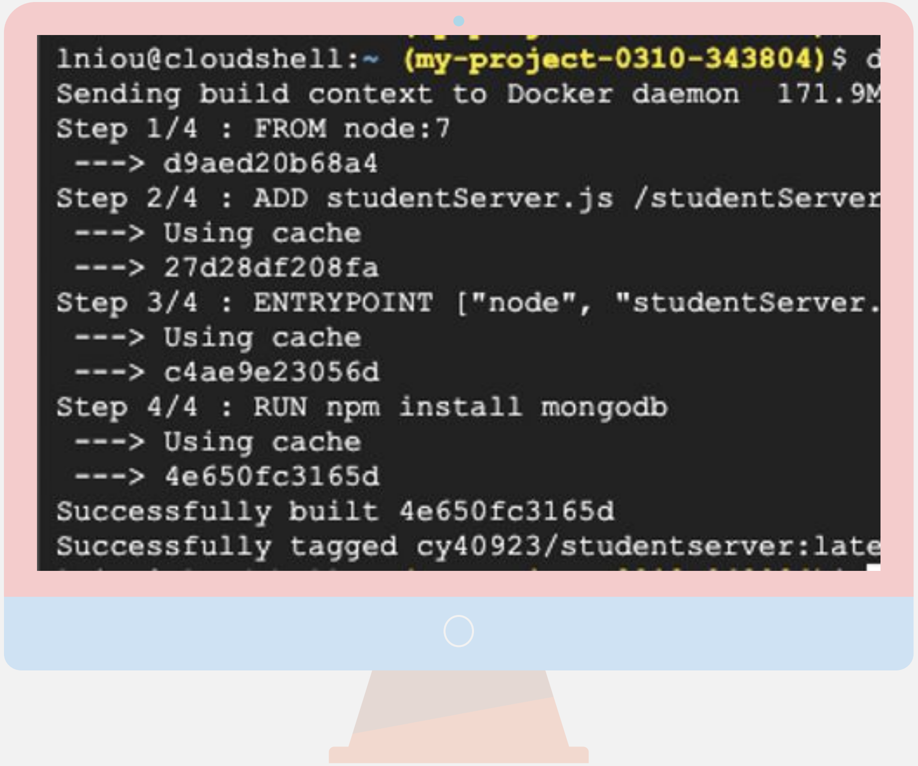
```
FROM node:7
ADD studentServer.js /studentServer.js
ENTRYPOINT ["node", "studentServer.js"]
RUN npm install mongodb
```

Build

```
$ docker build -t ID/repositories .
```

Push

```
$ docker push ID/repositories
```



```
lniou@cloudshell:~ (my-project-0310-343804)$ d
Sending build context to Docker daemon  171.9M
Step 1/4 : FROM node:7
 ---> d9aed20b68a4
Step 2/4 : ADD studentServer.js /studentServer
 ---> Using cache
 ---> 27d28df208fa
Step 3/4 : ENTRYPOINT ["node", "studentServer.
 ---> Using cache
 ---> c4ae9e23056d
Step 4/4 : RUN npm install mongodb
 ---> Using cache
 ---> 4e650fc3165d
Successfully built 4e650fc3165d
Successfully tagged cy40923/studentserver:late
```

09 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)
```

Build and push bookshelf image

Dockerfile

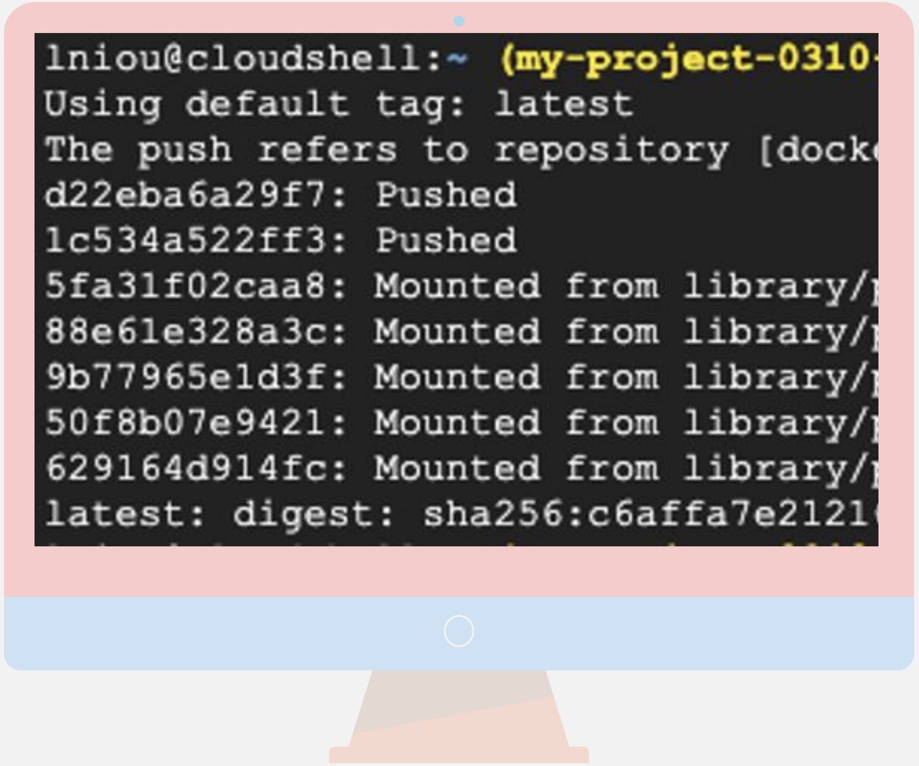
```
FROM python:alpine3.7
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
ENV PORT 5000
EXPOSE 5000
ENTRYPOINT [ "python3" ]
CMD [ "bookshelf.py" ]
```

Build

```
$ docker build -t ID/repositories .
```

Push

```
$ docker push ID/repositories
```



```
lniou@cloudshell:~ (my-project-0310
Using default tag: latest
The push refers to repository [dock
d22eba6a29f7: Pushed
1c534a522ff3: Pushed
5fa31f02caa8: Mounted from library/
88e61e328a3c: Mounted from library/
9b77965e1d3f: Mounted from library/
50f8b07e9421: Mounted from library/
629164d914fc: Mounted from library/
latest: digest: sha256:c6affa7e2121
```

Create ConfigMap

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  MONGO_URL: 35.185.216.112
  MONGO_DATABASE: mydb
```

studentserver-configmap.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  MONGO_URL: 35.185.216.112
  MONGO_DATABASE: mydb
```

bookshelf-configmap.yaml

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

Create deployment.yaml

studentserver-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:
        - image: ID/repositories
        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
```

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: ID/repositories
        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
```

Create service.yaml

studentserver-service.yaml

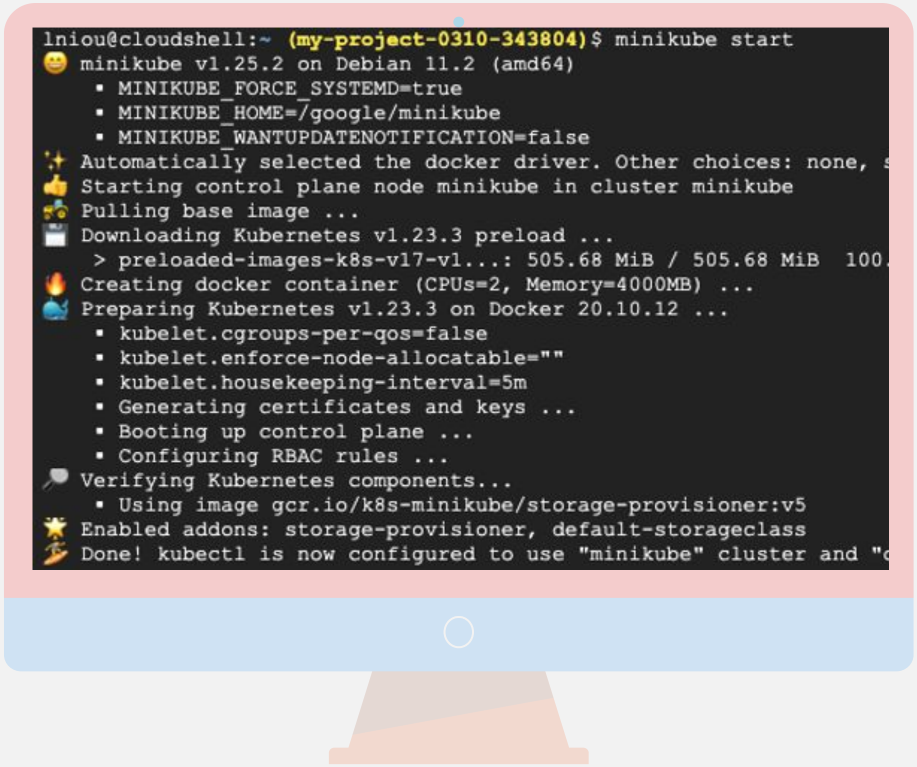
```yaml
apiVersion: v1
kind: Service
metadata:
  name: web
spec:
  type: LoadBalancer
  ports:
     # service port in cluster
     - port: 8080
     # port to contact inside container
     targetPort: 8080
  selector:
     app: web
```

bookshelf-service.yaml

```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
```

Start minikube and ingress

```
$ minikube start
$ minikube addons enable ingress
```



```
lniou@cloudshell:~ (my-project-0310-343804)$ 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, s
👍  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
🔥  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 "
```

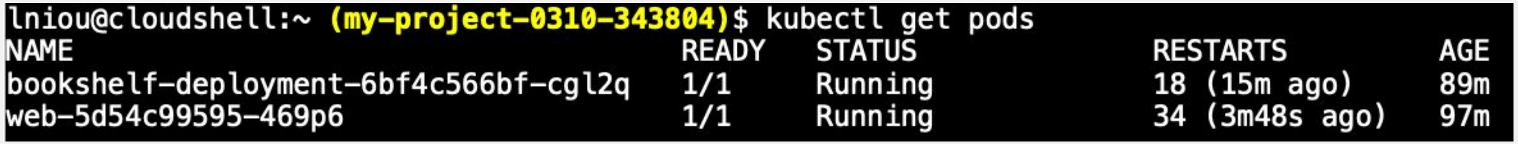
Create pods and services

studentserver

```
$ kubectl apply -f studentserver-deployment.yaml
$ kubectl apply -f studentserver-configmap.yaml
$ kubectl apply -f studentserver-service.yaml
```

bookshelf

```
$ kubectl apply -f bookshelf-deployment.yaml
$ kubectl apply -f bookshelf-configmap.yaml
$ kubectl apply -f bookshelf-service.yaml
```

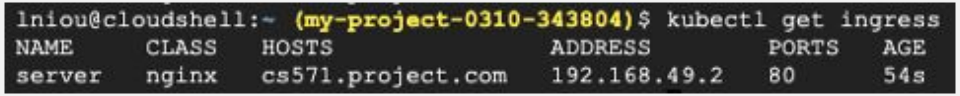


```
lniou@cloudshell:~ (my-project-0310-343804)$ kubectl get pods
NAME                                READY   STATUS    RESTARTS         AGE
bookshelf-deployment-6bf4c566bf-cgl2q   1/1     Running   18 (15m ago)     89m
web-5d54c99595-469p6                1/1     Running   34 (3m48s ago)   97m
```

Create ingress service

```
$ kubectl apply -f ingress.yaml
$ kubectl get ingress
```

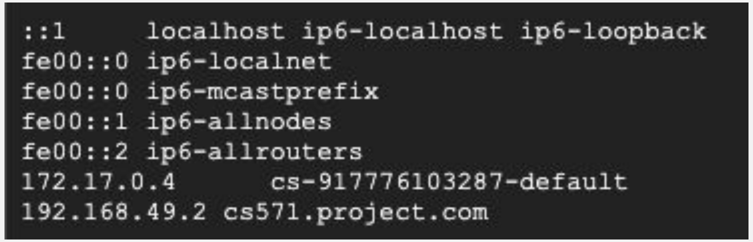
To get Hosts



```
lniou@cloudshell:~ (my-project-0310-343804)$ kubectl get ingress
NAME     CLASS    HOSTS               ADDRESS        PORTS   AGE
server   nginx    cs571.project.com   192.168.49.2   80      54s
```

Add address to /etc hosts

```
$ sudo vi /etc/hosts
```



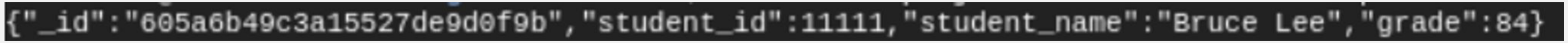
```
::1      localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
fe00::0 ip6-mcastprefix
fe00::1 ip6-allnodes
fe00::2 ip6-allrouters
172.17.0.4       cs-917776103287-default
192.168.49.2 cs571.project.com
```

```
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
```

Demo

Student-Server

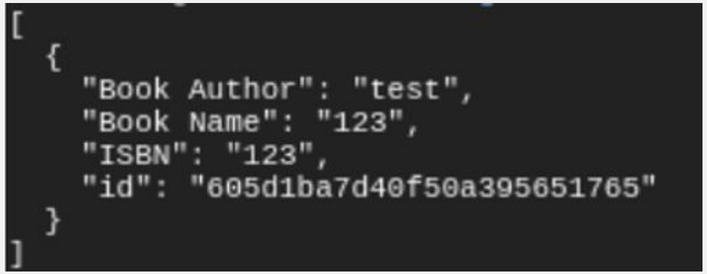
```
$ curl cs571.project.com/studentserver/api/score?student_id=11111
```



```
{"_id":"605a6b49c3a15527de9d0f9b","student_id":11111,"student_name":"Bruce Lee","grade":84}
```

Bookshelf

```
$ curl cs571.project.com/bookshelf/books
```

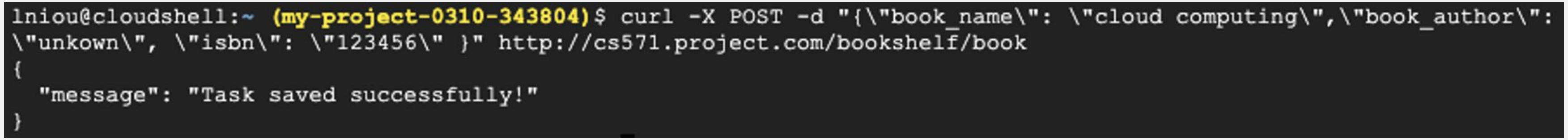


```
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123",
    "id": "605d1ba7d40f50a395651765"
  }
]
```

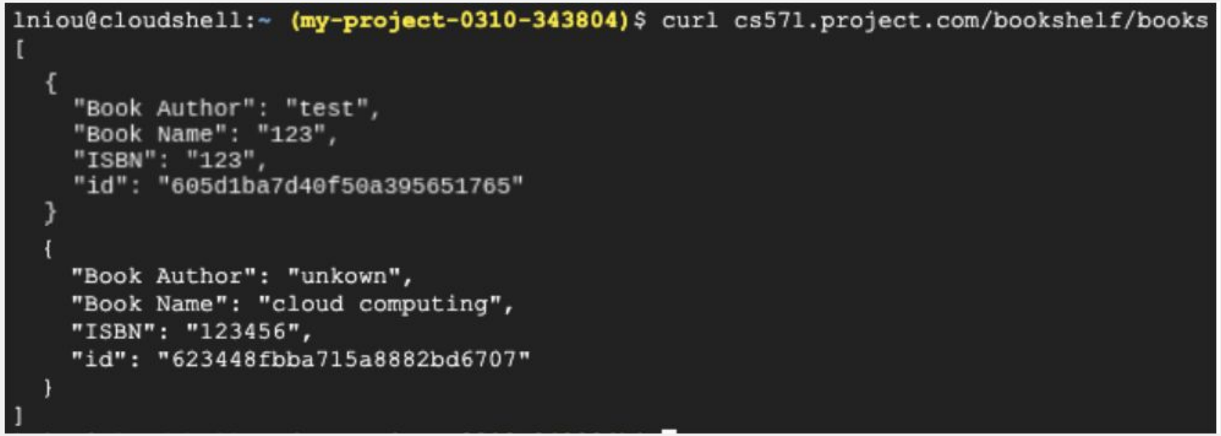
Demo

Bookshelf - Add book

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



```
lniou@cloudshell:~ (my-project-0310-343804)$ curl -X POST -d "{\"book_name\": \"cloud computing\",\"book_author\":
\"unkown\", \"isbn\": \"123456\" }" http://cs571.project.com/bookshelf/book
{
  "message": "Task saved successfully!"
}
```

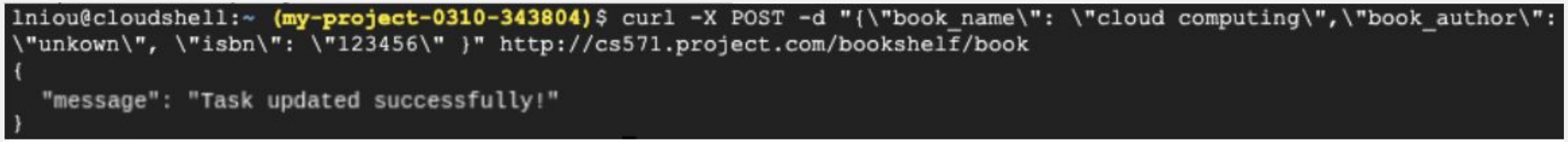


```
lniou@cloudshell:~ (my-project-0310-343804)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123",
    "id": "605d1ba7d40f50a395651765"
  }
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "623448fbba715a8882bd6707"
  }
]
```

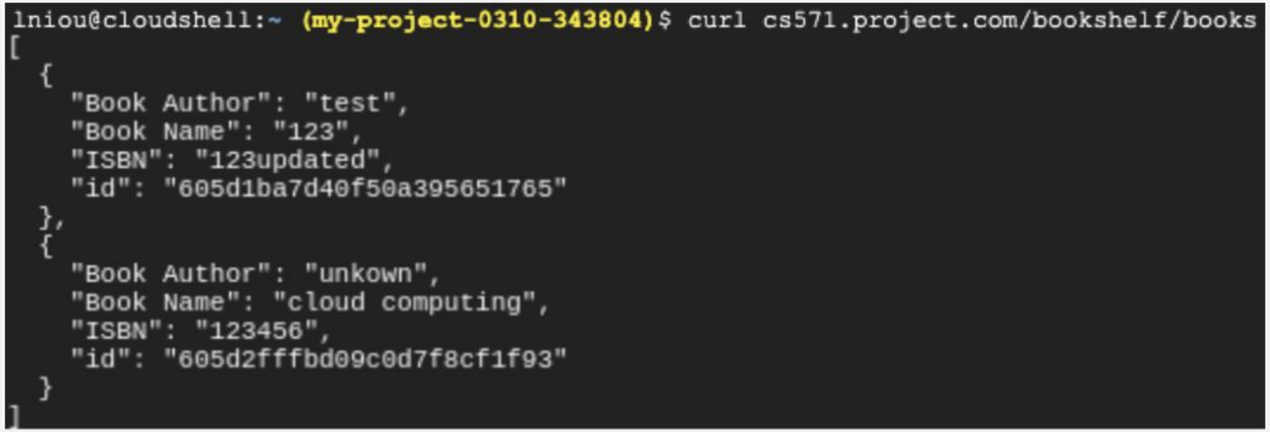
Demo

Bookshelf - Update book

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



```
lniou@cloudshell:~ (my-project-0310-343804)$ curl -X POST -d "{\"book_name\": \"cloud computing\",\"book_author\": \"unkown\", \"isbn\": \"123456\" }" http://cs571.project.com/bookshelf/book
{
  "message": "Task updated successfully!"
}
```

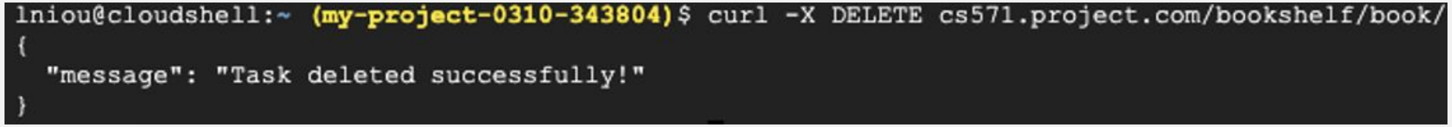


```
lniou@cloudshell:~ (my-project-0310-343804)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123updated",
    "id": "605d1ba7d40f50a395651765"
  },
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "605d2fffbd09c0d7f8cf1f93"
  }
]
```

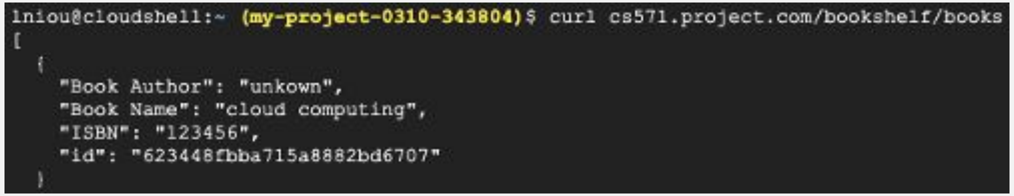
Demo

Bookshelf - Delete book

```
$ curl -X DELETE cs571.project.com/bookshelf/book/ID
```



```
lniou@cloudshell:~ (my-project-0310-343804)$ curl -X DELETE cs571.project.com/bookshelf/book/
{
  "message": "Task deleted successfully!"
}
```



```
lniou@cloudshell:~ (my-project-0310-343804)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "623448fbba715a8882bd6707"
  }
```

THANKS

Do you have any questions?