Step 1

1. **Create a cluster on GKE**

gcloud container clusters create kubia --num-nodes=1 --machine-type=e2-micro --region=us-west1

```
kubeconfig entry generated for kubia.
NAME    LOCATION  MASTER_VERSION    MASTER_IP     MACHINE_TYPE  NODE_VERSION     NUM_NODES  STATUS
kubia   us-west1  1.18.16-gke.502   34.82.48.182  e2-micro      1.18.16-gke.502  3          RUNNING
he19547@cloudshell:~ (cs571-demo-project-302123)$
```

2. **Create a Persistent Volume**

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

```
he19547@cloudshell:~ (cs571-demo-project-302123)$ 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, see: https://developers.g
oogle.com/compute/docs/disks#performance.
ERROR: (gcloud.compute.disks.create) Could not fetch resource:
 - The resource 'projects/cs571-demo-project-302123/zones/us-west1-a/disks/mongodb' already exists

he19547@cloudshell:~ (cs571-demo-project-302123)$
```

3. **Create a mongodb deployment with yaml filec**

vi mongodb-deployment.yaml

Kubectl apply -f mongodb-deployment.yaml

```
he19547@cloudshell:~ (cs571-demo-project-302123)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
he19547@cloudshell:~ (cs571-demo-project-302123)$ cat 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:
        # by default, the image is pulled from docker hub
        - image: mongo
          name: mongo
          ports:
            - containerPort: 27017
          volumeMounts:
            - name: mongodb-data
              mountPath: /data/db
      volumes:
        - name: mongodb-data
          gcePersistentDisk:
            pdName: mongodb
            fsType: ext4
```

apiVersion: apps/v1

kind: Deployment

metadata:

```
    name: mongodb-deployment
spec:
    selector:
        matchLabels:
            app: mongodb
    strategy:
        type: Recreate
    template:
        metadata:
            labels:
                app: mongodb
        spec:
            containers:
                # by default, the image is pulled from docker hub
                - image: mongo
                    name: mongo
                    ports:
                        - containerPort: 27017
                    volumeMounts:
                        - name: mongodb-data
                            mountPath: /data/db
            volumes:
                - name: mongodb-data
                    gcePersistentDisk:
                        pdName: mongodb
                        fsType: ext4
```

4. Check the pods and start running

kubectl get pods

```
he19547@cloudshell:~ (cs571-demo-project-302123)$ kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
mongodb-deployment-554cbb9965-txbdn     1/1     Running   0          2m44s
```

5. Create mongodb-service.yaml

vi mongodb-service.yaml

kubectl apply -f mongodb-service.yaml

```
he19547@cloudshell:~ (cs571-demo-project-302123)$ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
he19547@cloudshell:~ (cs571-demo-project-302123)$ cat mongodb-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  type: LoadBalancer
  ports:
      # service port in cluster
    - port: 27017
      # port to contact inside container
      targetPort: 27017
  selector:
    app: mongodb
```

apiVersion: v1

kind: Service

metadata:

  name: mongodb-service

spec:

  type: LoadBalancer

  ports:

      # service port in cluster

    - port: 27017

      # port to contact inside container

      targetPort: 27017

  selector:

    app: mongodb

6. Check the service（the external-ip is needed, mine is 34.82.14.45）

Kubectl get svc

```
he19547@cloudshell:~ (cs571-demo-project-302123)$ kubectl get svc
NAME              TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)           AGE
kubernetes        ClusterIP      10.3.240.1     <none>         443/TCP           27m
mongodb-service   LoadBalancer   10.3.245.96    34.82.14.45    27017:31580/TCP   111s
he19547@cloudshell:~ (cs571-demo-project-302123)$
```

7. Check mongoDB

kubectl exec -it mongodb-deployment-replace-with-your-pod-name -- bash

```
he19547@cloudshell:~ (cs571-demo-project-302123)$ kubectl exec -it mongodb-deployment-554cbb9965-txbdn -- bash
root@mongodb-deployment-554cbb9965-txbdn:/# 34.82.14.45
```

mongo External-ip

And how to exit

```
root@mongodb-deployment-554cbb9965-txbdn:/# mongo 34.82.14.45
MongoDB shell version v4.4.5
connecting to: mongodb://34.82.14.45:27017/test?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("c0f8bf95-39d4-4d45-979a-0b943a0b2f48") }
MongoDB server version: 4.4.5
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        https://docs.mongodb.com/
Questions? Try the MongoDB Developer Community Forums
        https://community.mongodb.com
---
The server generated these startup warnings when booting:
        2021-04-11T22:12:56.249+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http
godb.org/core/prodnotes-filesystem
        2021-04-11T22:12:57.639+00:00: Access control is not enabled for the database. Read and write access to data and configurati
icted
---
---
        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-554cbb9965-txbdn:/# exit
exit
```

8. Insert records into mongoDB and

npm install mongodb

Npm link mongodb

Vi mongodb1.js(create as you want.js and remember to change the external-ip the second line 34.82.14.45)

Node mongodb1.js

```
he19547@cloudshell:~ (cs571-demo-project-302123)$ cat mongodb1.js
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://34.82.14.45/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);
     });
});
he19547@cloudshell:~ (cs571-demo-project-302123)$ npm install mongodb

added 17 packages, changed 1 package, and audited 19 packages in 1s

1 package is looking for funding
  run `npm fund` for details

found 0 vulnerabilities
he19547@cloudshell:~ (cs571-demo-project-302123)$ npm link mongodb

removed 17 packages, changed 1 package, and audited 3 packages in 945ms

found 0 vulnerabilities
he19547@cloudshell:~ (cs571-demo-project-302123)$ node mongodb1.js
null
3
```

var MongoClient = require('mongodb').MongoClient;

var url = "mongodb://34.82.14.45/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);
    });
});
```

Step 2

1. Create studentServer.js

```
he19547@cloudshell:~ (cs571-demo-project-302123)$ cat studentServer.js
var http = require('http');
var url = require('url');
var mongodb = require('mongodb');
const {
  MONGO_URL,
  MONGO_DATABASE
}  = process.env;
// - Expect the request to contain a query
//   string with a key 'student_id' and a student ID as
//   the value. For example
//     /api/score?student_id=1111
// - The JSON response should contain only 'student_id', 'student_name'
//   and 'student_score' properties. For example:
//   {
//       "student_id": 1111,
//       "student_name": Bruce Lee,
//       "student_score": 84
//   }
//

var MongoClient = mongodb.MongoClient;
var uri = `mongodb://${MONGO_URL}/${MONGO_DATABASE}`;
// Connect to the db
console.log(uri);

var server = http.createServer(function (req, res) {
  var result;
  // req.url = /api/score?student_id=11111
  var parsedUrl = url.parse(req.url, true);

  var student_id = parseInt(parsedUrl.query.student_id);
  // match req.url with the string /api/score
  if (/^\/api\/score/.test(req.url)) {
    // e.g., of student_id 1111

    MongoClient.connect(uri,{ useNewUrlParser: true, useUnifiedTopology:
true }, function(err, client){
        if (err)
          throw err;
        var db = client.db("studentdb");
        db.collection("students").findOne({"student_id":student_id},
(err, student) => {
          if(err)
               throw new Error(err.message, null);
```

```
               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);
```

var http = require('http');

var url = require('url');

```
var mongodb = require('mongodb');
const {
   MONGO_URL,
   MONGO_DATABASE
}   = process.env;
// - Expect the request to contain a query
//      string with a key 'student_id' and a student ID as
//      the value. For example
//          /api/score?student_id=1111
// - The JSON response should contain only 'student_id', 'student_name'
//      and 'student_score' properties. For example:
//      {
//              "student_id": 1111,
//              "student_name": Bruce Lee,
//              "student_score": 84
//      }
//

var MongoClient = mongodb.MongoClient;
var uri = `mongodb://${MONGO_URL}/${MONGO_DATABASE}`;
// Connect to the db
console.log(uri);

var server = http.createServer(function (req, res) {
   var result;
   // req.url = /api/score?student_id=11111
   var parsedUrl = url.parse(req.url, true);
```

```
    var student_id = parseInt(parsedUrl.query.student_id);
    // match req.url with the string /api/score
    if (/^\/api\/score/.test(req.url)) {
        // e.g., of student_id 1111

        MongoClient.connect(uri,{ useNewUrlParser: true, useUnifiedTopology:
true }, function(err, client){
                if (err)
                    throw err;
                var db = client.db("studentdb");
                db.collection("students").findOne({"student_id":student_id},
(err, student) => {
                    if(err)
                            throw new Error(err.message, null);

                    if (student) {
                            res.writeHead(200, { 'Content-Type': 'application/json'
})
                            res.end(JSON.stringify(student)+ '\n')
                    }else {
                            res.writeHead(404);
                            res.end("Student Not Found \n");
                    }
                });
    } else {
    res.writeHead(404);
    res.end("Wrong url, please try again\n");
    }
```

});

server.listen(8080);

2. Create Dockerfile

```
he19547@cloudshell:~ (cs571-demo-project-302123)$ cat Dockerfile
FROM node:7
ADD studentServer.js /studentServer.js
ENTRYPOINT ["node","studentServer.js"]
RUN npm install mongodb
he19547@cloudshell:~ (cs571-demo-project-302123)$
```

3. Build the studentserver docker image

docker build -t yourdockerhubID/studentserver .

```
Removing intermediate container 2ec47f3072c8
 ---> 3918f800a5fe
Successfully built 3918f800a5fe
Successfully tagged hhhh9999/studentserver:latest
```

4. Push the docker image

```
he19547@cloudshell:~ (cs571-demo-project-302123)$ docker push hhhh9999/studentserver
Using default tag: latest
The push refers to repository [docker.io/hhhh9999/studentserver]
08aa9b3eb1bf: Pushed
a6c3c2c888b3: 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: sha256:54a603e8743739e713d8dca6b88b955166f27f066e89a64c21cd34d8a87c0b89 size: 2424
```

Step 3

1. Create bookshelf.py

```
he19547@cloudshell:~ (cs571-demo-project-302123)$ cat 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"]
```

```
        "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!"
    )
```

```
        message="All Books deleted!"
    )

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

2. **Create Dockerfile**

```
he19547@cloudshell:~ (cs571-demo-project-302123)$ cat Dockerfile
FROM python:alpine3.7
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
ENV PORT 5000
EXPOSE 5000
ENTRYPOINT [ "python3" ]
CMD [ "bookshelf.py" ]
```

3. **Build the bookshelf app into a docker image**

Docker build -t hhhh9999/bookshelf .

```
he19547@cloudshell:~/kubernetes_project/bookshelf (cs571-demo-project-302123)$ docker build -t hhhh9999/bookshelf .
Sending build context to Docker daemon  22.53kB
Step 1/8 : FROM python:alpine3.7
 ---> 00be2573e9f7
Step 2/8 : COPY . /app
 ---> a689723666e6
Step 3/8 : WORKDIR /app
 ---> Running in da38cf321981
Removing intermediate container da38cf321981
 ---> 1cfaa63b0cc8
Step 4/8 : RUN pip install -r requirements.txt
 ---> Running in 7c93a839403e
Collecting Flask (from -r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/f2/28/2a03252dfb9ebf377f40fba6a7841b47083260bf8bd8e737b0c6952df83f/Flas
k-1.1.2-py2.py3-none-any.whl (94kB)
Collecting Flask-PyMongo (from -r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/67/b8/0322016b9ce09a64fba9018211e7c35fd51380527ffd9ea248744f389239/Flas
k_PyMongo-2.3.0-py2.py3-none-any.whl
Collecting click>=5.1 (from Flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/d2/3d/fa76db83bf75c4f8d338c2fd15c8d33fdd7ad23a9b5e57eb6c5de26b430e/clic
k-7.1.2-py2.py3-none-any.whl (82kB)
Collecting Werkzeug>=0.15 (from Flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/cc/94/5f7079a0e00bd6863ef8f1da638721e9da21e5bacee597595b318f71d62e/Werk
zeug-1.0.1-py2.py3-none-any.whl (298kB)
```

4. Push the docker image to the dockerhub

Docker push hhhh9999/bookshelf

```
Successfully tagged hhhh9999/bookshelf:latest
he19547@cloudshell:~/kubernetes_project/bookshelf (cs571-demo-project-302123)$ docker push hhhh9999/bookshelf
Using default tag: latest
The push refers to repository [docker.io/hhhh9999/bookshelf]
5e86b7507b64: Pushed
911a51651be2: 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: sha256:4a388cf00f30af1f863f174b9d9dc86f70bf188f29f01b9b6c4a2c8f27efb83f size: 1787
he19547@cloudshell:~/kubernetes_project/bookshelf (cs571-demo-project-302123)$ ^C
```

Step 4

1. Create a file named studentserver-configmap.yaml

2. Create a file named bookshelf-configmap.yaml (Remember to change the external ip)

```
he19547@cloudshell:~ (cs571-demo-project-302123)$ cat vi studentserver-configmap.yaml
cat: vi: No such file or directory
apiVersion: v1
kind: ConfigMap
metadata:
  name: studentserver-config
data:
  MONGO_URL: 34.82.14.45
  MONGO_DATABASE: mydb
he19547@cloudshell:~ (cs571-demo-project-302123)$ cat bookshelf-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: bookshelf-config
data:
  # SERVICE_NAME.NAMESPACE.svc.cluster.local:SERVICE_PORT
  MONGO_URL: 34.82.14.45
  MONGO_DATABASE: mydb
he19547@cloudshell:~ (cs571-demo-project-302123)$
```

Step 5

1. Create studentserver-deployment.yaml

```
he19547@cloudshell:~ (cs571-demo-project-302123)$ cat 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: hhhh9999/studentserver
          imagePullPolicy: Always
          name: web
          ports:
            - containerPort: 8080
          env:
            - name: MONGO_URL
              valueFrom:
                configMapKeyRef:
                  name: studentserver-config
                  key: MONGO_URL
            - name: MONGO_DATABASE
              valueFrom:
                configMapKeyRef:
                  name: studentserver-config
                  key: MONGO_DATABASE
```

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: hhhh9999/studentserver
          imagePullPolicy: Always
          name: web
          ports:
            - containerPort: 8080
          env:
            - name: MONGO_URL
              valueFrom:
                configMapKeyRef:
                  name: studentserver-config
                  key: MONGO_URL
            - name: MONGO_DATABASE
              valueFrom:
                configMapKeyRef:
                  name: studentserver-config
                  key: MONGO_DATABASE
```

2. Create bookshelf-deployment.yaml

```
he19547@cloudshell:~ (cs571-demo-project-302123)$ cat 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: hhhh9999/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
```

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: hhhh9999/bookshelf
      imagePullPolicy: Always
      name: bookshelf-deployment
      ports:
        - containerPort: 5000
      env:
        - name: MONGO_URL
          valueFrom:
            configMapKeyRef:
              name: bookshelf-config
              key: MONGO_URL
        - name: MONGO_DATABASE
          valueFrom:
            configMapKeyRef:
              name: bookshelf-config
              key: MONGO_DATABASE
```

3. Create studentserver-service.yaml

```
he19547@cloudshell:~ (cs571-demo-project-302123)$ cat studentserver-sevice.yam
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
```

apiVersion: v1

kind: Service

metadata:

  name: web

spec:

  type: LoadBalancer

  ports:

      # service port in cluster

    - port: 8080

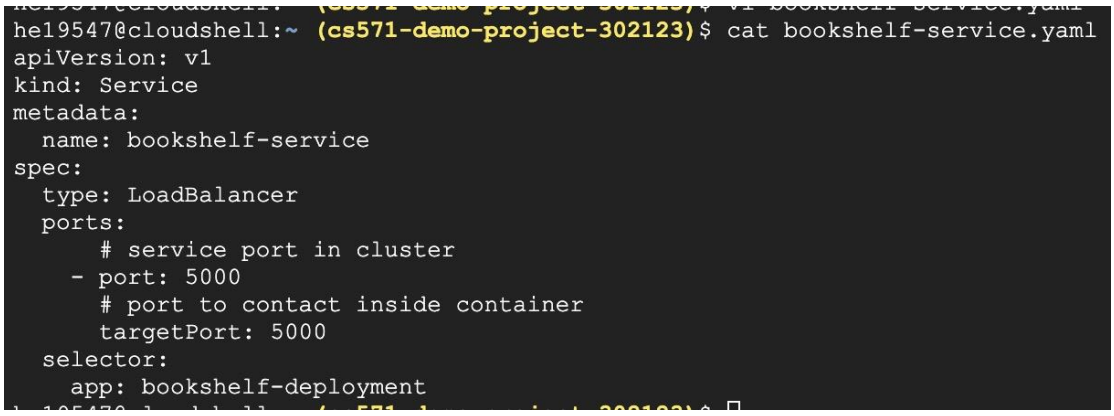
      # port to contact inside container

      targetPort: 8080

  selector:

    app: web

4. Create 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
```

5. Minikube start and start ingress

```
he19547@cloudshell:~/kubernetes_project/studentserver (cs571-demo-project-302123)$ 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: ssh, none
* 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% 142.24 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
he19547@cloudshell:~/kubernetes_project/studentserver (cs571-demo-project-302123)$ 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
```

6. Create studentserver pods and start service

```
he19547@cloudshell:~/kubernetes_project/studentserver (cs571-demo-project-302123)$ kubectl apply -f studentserver-deployment.
yaml
deployment.apps/web created
he19547@cloudshell:~/kubernetes_project/studentserver (cs571-demo-project-302123)$ kubectl apply -f studentserver-configmap.y
aml
configmap/studentserver-config created
he19547@cloudshell:~/kubernetes_project/studentserver (cs571-demo-project-302123)$ kubectl apply -f studentserver-service.yam
l
service/web created
```

7. Create bookshelf pod and start service

```
he19547@cloudshell:~/kubernetes_project/studentserver (cs571-demo-project-302123)$ cd ../bookshelf/
he19547@cloudshell:~/kubernetes_project/bookshelf (cs571-demo-project-302123)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created
he19547@cloudshell:~/kubernetes_project/bookshelf (cs571-demo-project-302123)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
he19547@cloudshell:~/kubernetes_project/bookshelf (cs571-demo-project-302123)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
```

8. Check pods

```
deployment.apps/web created
he19547@cloudshell:~/kubernetes_project/studentserver (cs571-demo-project-302123)$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
bookshelf-deployment-5b7dfd45c-cpvsw  1/1     Running   0          3m47s
web-787c4cfdb7-855fp                  1/1     Running   0          5s
```

9. Create an ingress service yaml file named studentservermongoIngress.yaml

apiVersion: networking.k8s.io/v1 kind: Ingress

```
metadata:
    name: server
    annotations:
      nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
    rules:
      - host: cs571.project.com
        http:
          paths:
            - path: /studentserver(/|$)(.*)
              pathType: Prefix
              backend:
                service:
                  name: web
                  port:
                  number: 8080
            - path: /bookshelf(/|$)(.*)
              pathType: Prefix
              backend:
                service:
                name: bookshelf-service
                port:
                  number: 5000
```

10. Kubectl get ingress（host address needed)

```
he19547@cloudshell:~/kubernetes_project/studentserver (cs571-demo-project-302123)$ kubectl get ingress
NAME     CLASS    HOSTS              ADDRESS        PORTS   AGE
server   <none>   cs571.project.com  192.168.49.2   80      22s
```

11. Add address

```
he19547@cloudshell:~/kubernetes_project/studentserver (cs571-demo-project-302123)$ vi /etc/hosts
he19547@cloudshell:~/kubernetes_project/studentserver (cs571-demo-project-302123)$ sudo vi /etc/hosts
```

12. Access the applications

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

```
he19547@cloudshell:~/kubernetes_project/studentserver (cs571-demo-project-302123)$ curl cs571.project.com/studentserver/api/s
core?student_id=11111
{"_id":"607391b7bb131403b789592a","student_id":11111,"student_name":"Bruce Lee","grade":84}
```

13. Another path to access the applications

```
he19547@cloudshell:~/kubernetes_project/studentserver (cs571-demo-project-302123)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "6073b45baeefcda681c994e1"
  }
]
```

14. Add a book

```
he19547@cloudshell:~/kubernetes_project/studentserver (cs571-demo-project-302123)$ curl -X POST -d "{\"book_name\": \"kuberne
tes\",\"book_author\":
\"unkown\", \"isbn\": \"123456\" }" http://cs571.project.com/bookshelf/book
{
  "message": "Task saved successfully!"
}
he19547@cloudshell:~/kubernetes_project/studentserver (cs571-demo-project-302123)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123updated",
    "id": "6073b45baeefcda681c994e1"
  },
  {
    "Book Author": "unkown",
    "Book Name": "kubernetes",
    "ISBN": "123456",
    "id": "6073b555aeefcda681c994e2"
  }
]
```

15. **Update a book**

```
he19547@cloudshell:~/kubernetes_project/studentserver (cs571-demo-project-302123)$ curl -X PUT -d "{\"book_name\": \"123\",\"
book_author\": \"test\", \"isbn\": \"123updated\" }" http://cs571.project.com/bookshelf/book/6073b45baeefcda681c994e1
{
  "message": "Task updated successfully!"
}
he19547@cloudshell:~/kubernetes_project/studentserver (cs571-demo-project-302123)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123updated",
    "id": "6073b45baeefcda681c994e1"
  }
]
```

16. Delete a book

```
he19547@cloudshell:~/kubernetes_project/studentserver (cs571-demo-project-302123)$ curl -X DELETE cs571.project.com/bookshelf
/book/6073b45baeefcda681c994e1
{
  "message": "Task deleted successfully!"
}
he19547@cloudshell:~/kubernetes_project/studentserver (cs571-demo-project-302123)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "kubernetes",
    "ISBN": "123456",
    "id": "6073b555aeefcda681c994e2"
  }
]
```