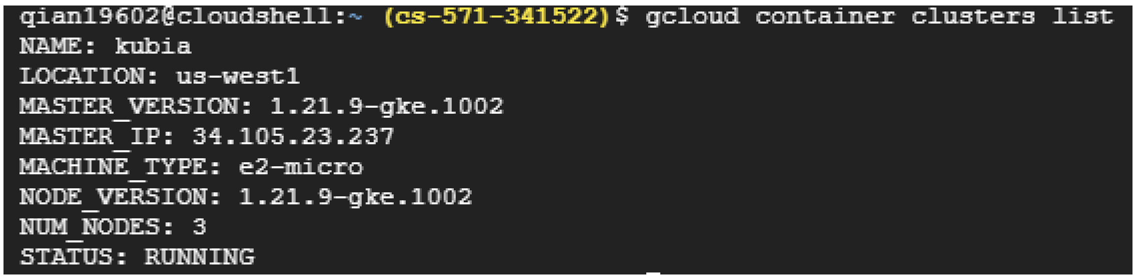
CS571 Signature Project

Name:Jin Qian

ID: 19602

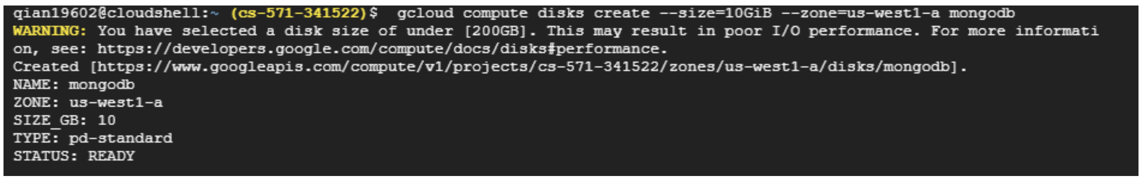
Step1 Create MongoDB using Persistent Volume on GKE, and insert records into it 1. Create a cluster as usual on GKE

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



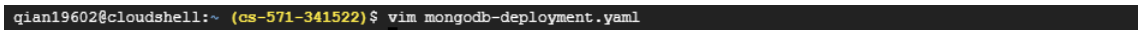
```
qian19602@cloudshell:~ (cs-571-341522)$ gcloud container clusters list
NAME: kubia
LOCATION: us-west1
MASTER_VERSION: 1.21.9-gke.1002
MASTER_IP: 34.105.23.237
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.21.9-gke.1002
NUM_NODES: 3
STATUS: RUNNING
```

2. create a Persistent Volume first
 gcloud compute disks create --size=10GiB --zone=us-west1-a mongodb



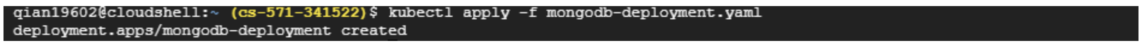
```
qian19602@cloudshell:~ (cs-571-341522)$  gcloud compute disks create --size=10GiB --zone=us-west1-a mongodb
WARNING: You have selected a disk size of under [200GB]. This may result in poor I/O performance. For more informati
on, see: https://developers.google.com/compute/docs/disks#performance.
Created [https://www.googleapis.com/compute/v1/projects/cs-571-341522/zones/us-west1-a/disks/mongodb].
NAME: mongodb
ZONE: us-west1-a
SIZE_GB: 10
TYPE: pd-standard
STATUS: READY
```

3. Now create a mongodb deployment with this yaml filec



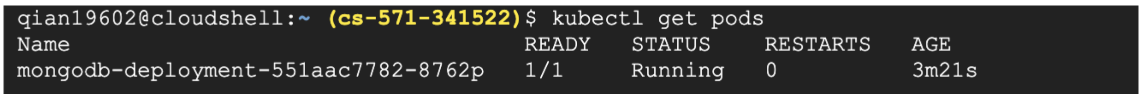
```
qian19602@cloudshell:~ (cs-571-341522)$ vim mongodb-deployment.yaml
```

kubectl apply -f mongodb-deployment.yaml



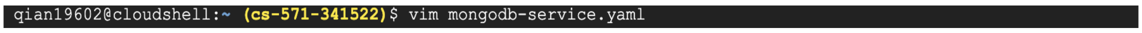
```
qian19602@cloudshell:~ (cs-571-341522)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
```

4. Check if the deployment pod has been successfully created and started running
kubectl get pods



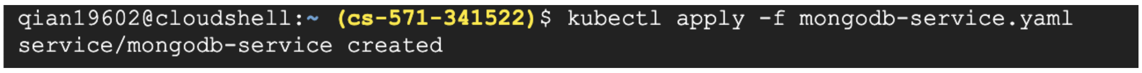
```
qian19602@cloudshell:~ (cs-571-341522)$ kubectl get pods
Name                                READY   STATUS    RESTARTS   AGE
mongodb-deployment-551aac7782-8762p  1/1     Running   0          3m21s
```

5. Create a service for the mongoDB, so it can be accessed from outside



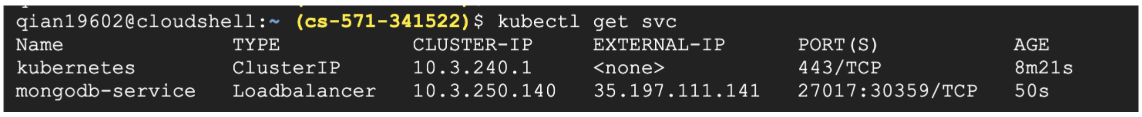
```
qian19602@cloudshell:~ (cs-571-341522)$ vim mongodb-service.yaml
```

kubectl apply -f mongodb-service.yaml



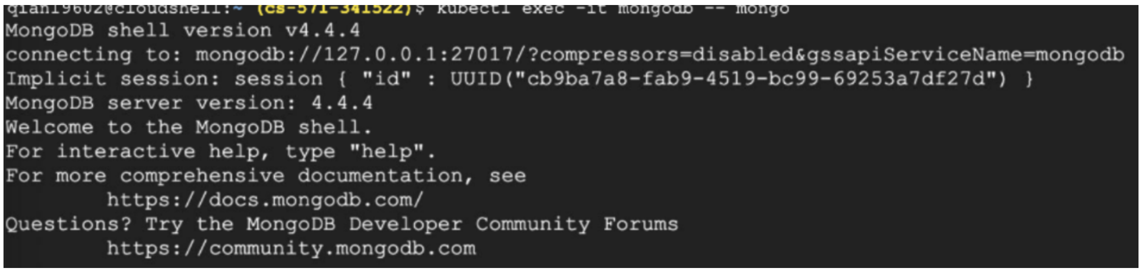
```
qian19602@cloudshell:~ (cs-571-341522)$ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
```

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



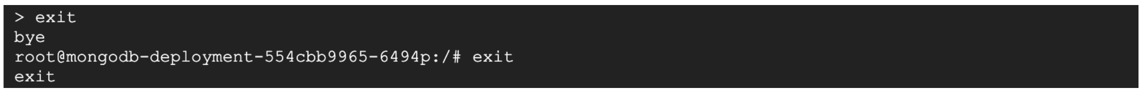
```
qian19602@cloudshell:~ (cs-571-341522)$ kubectl get svc
Name             TYPE           CLUSTER-IP     EXTERNAL-IP      PORT(S)           AGE
kubernetes       ClusterIP      10.3.240.1     <none>           443/TCP           8m21s
mongodb-service  Loadbalancer   10.3.250.140   35.197.111.141   27017:30359/TCP   50s
```

7. Now try and see if mongoDB is functioning for connections using the External-IP kubectl exec -it mongodb-deployment-replace-with-your-pod-name -- bash Now you are inside the mongodb deployment pod



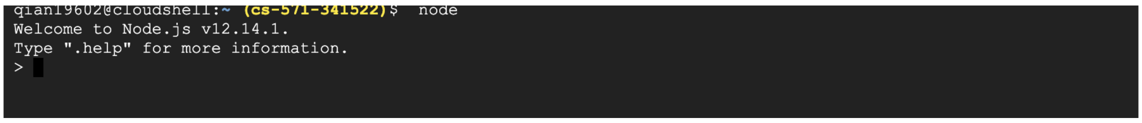
```
qian19602@cloudshell:~ (cs-571-341522)$ kubectl exec -it mongodb -- mongo
MongoDB shell version v4.4.4
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("cb9ba7a8-fab9-4519-bc99-69253a7df27d") }
MongoDB server version: 4.4.4
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
```

8. Type exit to exit mongodb and back to our google console

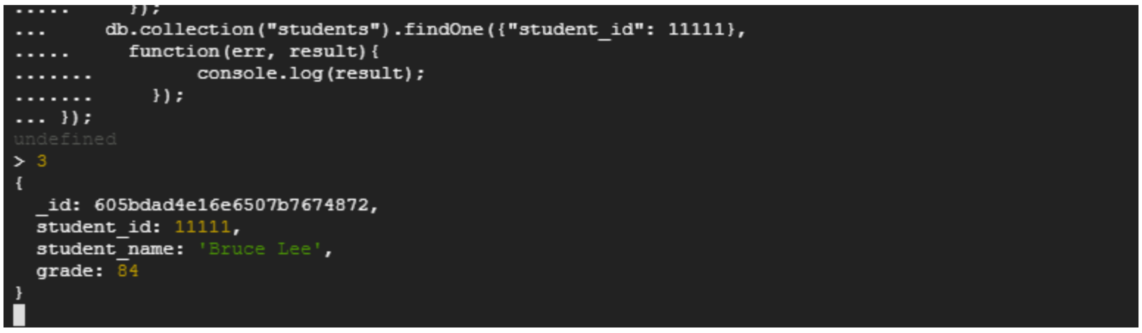


```
> exit
bye
root@mongodb-deployment-554cbb9965-6494p:/# exit
exit
```

9. We need to insert some records into the mongoDB for later use node



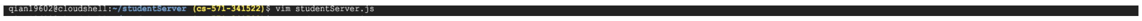
```
qian19602@cloudshell:~ (cs-571-341522)$  node
Welcome to Node.js v12.14.1.
Type ".help" for more information.
>
```



```
.....     });
...     db.collection("students").findOne({"student_id": 11111},
.....     function(err, result){
.......         console.log(result);
.......     });
... });
undefined
> 3
{
  _id: 605bdad4e16e6507b7674872,
  student_id: 11111,
  student_name: 'Bruce Lee',
  grade: 84
}
```

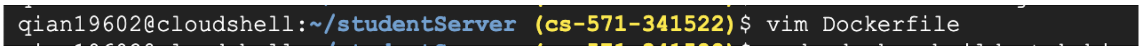
Step2 Modify our studentServer to get records from MongoDB and deploy to GK

1. Create a studentServer



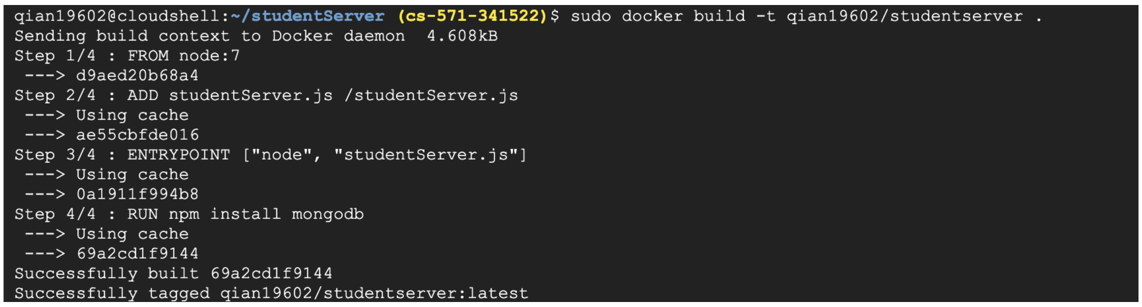
```
qian19602@cloudshell:~/studentServer (cs-571-341522)$ vim studentServer.js
```

2. Create Dockerfile



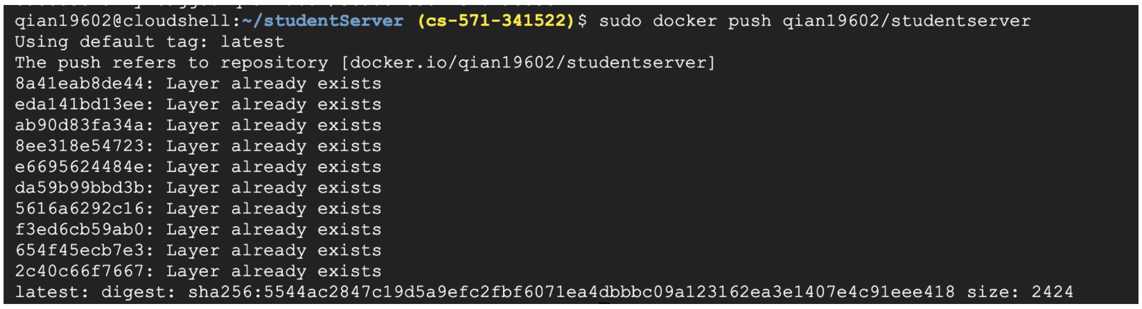
```
qian19602@cloudshell:~/studentServer (cs-571-341522)$ vim Dockerfile
```

3. Build the studentserver docker image docker build -t yourdockerhubID/studentserver . Make sure there is no error



```
qian19602@cloudshell:~/studentServer (cs-571-341522)$ sudo docker build -t qian19602/studentserver .
Sending build context to Docker daemon  4.608kB
Step 1/4 : FROM node:7
 ---> d9aed20b68a4
Step 2/4 : ADD studentServer.js /studentServer.js
 ---> Using cache
 ---> ae55cbfde016
Step 3/4 : ENTRYPOINT ["node", "studentServer.js"]
 ---> Using cache
 ---> 0a1911f994b8
Step 4/4 : RUN npm install mongodb
 ---> Using cache
 ---> 69a2cd1f9144
Successfully built 69a2cd1f9144
Successfully tagged qian19602/studentserver:latest
```

4. Push the docker image docker push yourdockerhubID/studentserver

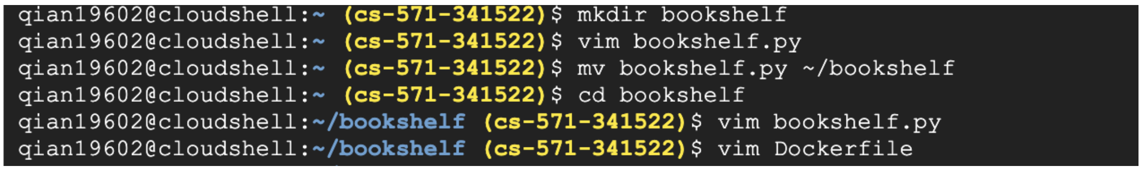


```
qian19602@cloudshell:~/studentServer (cs-571-341522)$ sudo docker push qian19602/studentserver
Using default tag: latest
The push refers to repository [docker.io/qian19602/studentserver]
8a41eab8de44: Layer already exists
eda141bd13ee: Layer already exists
ab90d83fa34a: Layer already exists
8ee318e54723: Layer already exists
e6695624484e: Layer already exists
da59b99bbd3b: Layer already exists
5616a6292c16: Layer already exists
f3ed6cb59ab0: Layer already exists
654f45ecb7e3: Layer already exists
2c40c66f7667: Layer already exists
latest: digest: sha256:5544ac2847c19d5a9efc2fbf6071ea4dbbbc09a123162ea3e1407e4c91eee418 size: 2424
```

Step3 Create a python Flask bookshelf REST API and deploy on GKE

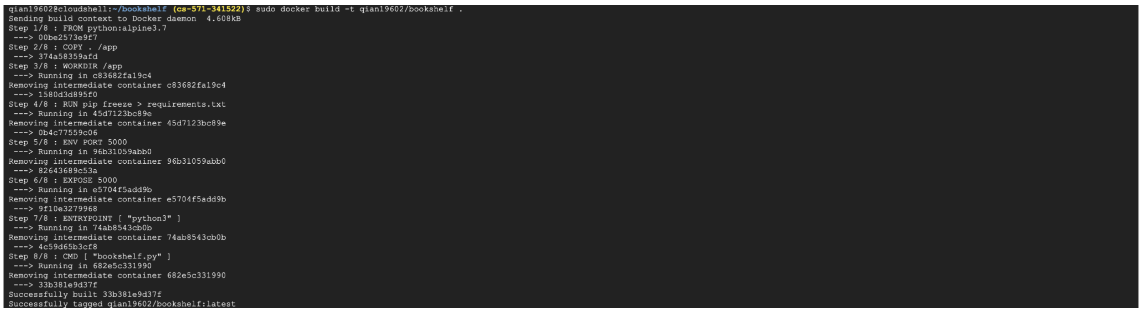
1. Create bookshelf.py

2. Create a Dockerfile



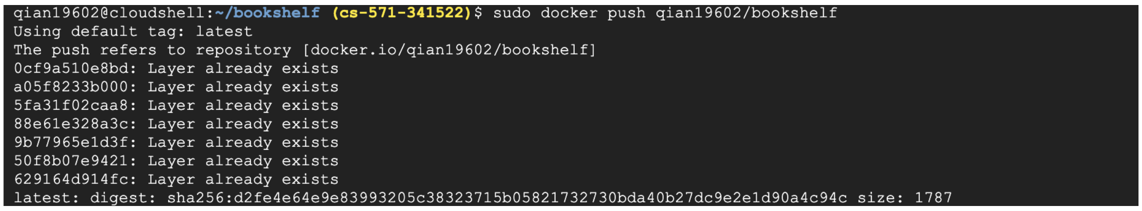
```
qian19602@cloudshell:~ (cs-571-341522)$ mkdir bookshelf
qian19602@cloudshell:~ (cs-571-341522)$ vim bookshelf.py
qian19602@cloudshell:~ (cs-571-341522)$ mv bookshelf.py ~/bookshelf
qian19602@cloudshell:~ (cs-571-341522)$ cd bookshelf
qian19602@cloudshell:~/bookshelf (cs-571-341522)$ vim bookshelf.py
qian19602@cloudshell:~/bookshelf (cs-571-341522)$ vim Dockerfile
```

3. Build the bookshelf app into a docker image docker build -t qian19539/bookshelf .
Make sure this step build successfully



```
qian19602@cloudshell:~/bookshelf (cs-571-341522)$ sudo docker build -t qian19602/bookshelf .
Sending build context to Docker daemon  4.608kB
Step 1/8 : FROM python:alpine3.7
 ---> 00be2573e9f7
Step 2/8 : COPY . /app
 ---> 374a58359afd
Step 3/8 : WORKDIR /app
 ---> Running in c83682fa19c4
Removing intermediate container c83682fa19c4
 ---> 1580d3d895f0
Step 4/8 : RUN pip freeze > requirements.txt
 ---> Running in 45d7123bc89e
Removing intermediate container 45d7123bc89e
 ---> 0b4c77559c06
Step 5/8 : ENV PORT 5000
 ---> Running in 96b31059abb0
Removing intermediate container 96b31059abb0
 ---> 82643689c53a
Step 6/8 : EXPOSE 5000
 ---> Running in e5704f5add9b
Removing intermediate container e5704f5add9b
 ---> 9f10e3279968
Step 7/8 : ENTRYPOINT [ "python3" ]
 ---> Running in 74ab8543cb0b
Removing intermediate container 74ab8543cb0b
 ---> 4c59d65b3cf8
Step 8/8 : CMD [ "bookshelf.py" ]
 ---> Running in 682e5c331990
Removing intermediate container 682e5c331990
 ---> 33b381e9d37f
Successfully built 33b381e9d37f
Successfully tagged qian19602/bookshelf:latest
```

4. Push the docker image to your dockerhub docker push yourdockerhubID/bookshelf

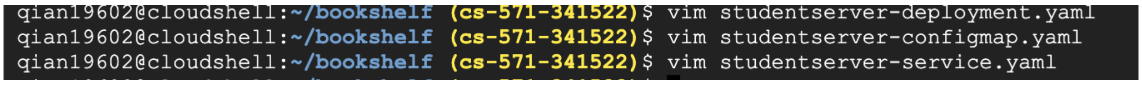


```
qian19602@cloudshell:~/bookshelf (cs-571-341522)$ sudo docker push qian19602/bookshelf
Using default tag: latest
The push refers to repository [docker.io/qian19602/bookshelf]
0cf9a510e8bd: Layer already exists
a05f8233b000: Layer already exists
5fa31f02caa8: Layer already exists
88e61e328a3c: Layer already exists
9b77965e1d3f: Layer already exists
50f8b07e9421: Layer already exists
629164d914fc: Layer already exists
latest: digest: sha256:d2fe4e64e9e83993205c38323715b05821732730bda40b27dc9e2e1d90a4c94c size: 1787
```

Step4 Create ConfigMap for both applications to store MongoDB URL and MongoDB name
1. Create a file named studentserver-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
name: studentserver-config
data:
MONGO_URL: Change-this-to-your-mongoDB-EXTERNAL-IP
MONGO_DATABASE: mydb
2. Create a file named bookshelf-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
name: bookshelf-config
data:
# SERVICE_NAME.NAMESPACE.svc.cluster.local:SERVICE_PORT
MONGO_URL: Change-this-to-your-mongoDB-EXTERNAL-IP
MONGO_DATABASE: mydb

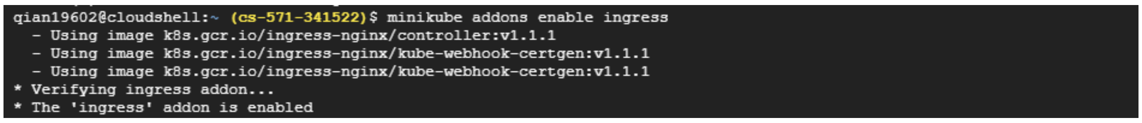
Step5 Expose 2 application using ingress with Nginx, so we can put them on the same Domain but different PATH

1. Create studentserver-deployment.yaml
2. Create bookshelf-deployment.yaml
3. Create sutdentserver-service.yaml
4. Create bookshelf-service.yaml



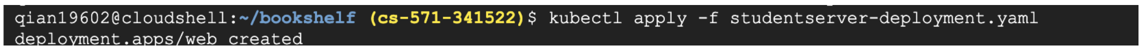
```
qian19602@cloudshell:~/bookshelf (cs-571-341522)$ vim studentserver-deployment.yaml
qian19602@cloudshell:~/bookshelf (cs-571-341522)$ vim studentserver-configmap.yaml
qian19602@cloudshell:~/bookshelf (cs-571-341522)$ vim studentserver-service.yaml
```

5. Start minikube minikube start
6. Start Ingress minikube addons enable ingress



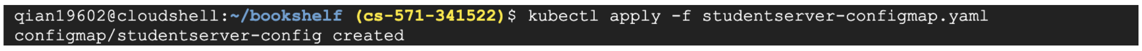
```
qian19602@cloudshell:~ (cs-571-341522)$ minikube addons enable ingress
  - Using image k8s.gcr.io/ingress-nginx/controller:v1.1.1
  - Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
  - Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
* Verifying ingress addon...
* The 'ingress' addon is enabled
```

7. Create studentserver related pods and start service using the above yaml file kubectl apply -f studentserver-deployment.yaml



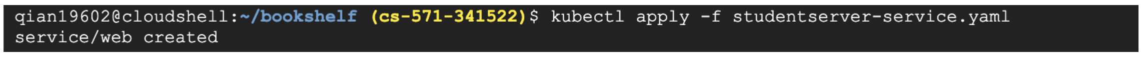
```
qian19602@cloudshell:~/bookshelf (cs-571-341522)$ kubectl apply -f studentserver-deployment.yaml
deployment.apps/web created
```

kubectl apply -f studentserver-configmap.yaml



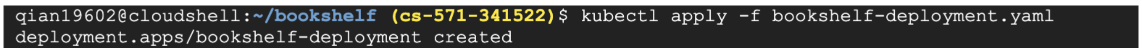
```
qian19602@cloudshell:~/bookshelf (cs-571-341522)$ kubectl apply -f studentserver-configmap.yaml
configmap/studentserver-config created
```

kubectl apply -f studentserver-service.yaml



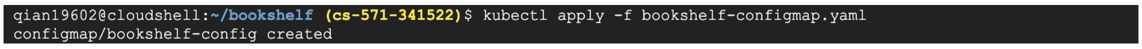
```
qian19602@cloudshell:~/bookshelf (cs-571-341522)$ kubectl apply -f studentserver-service.yaml
service/web created
```

8. Create bookshelf related pods and start service using the above yaml file kubectl apply -f bookshelf-deployment.yaml



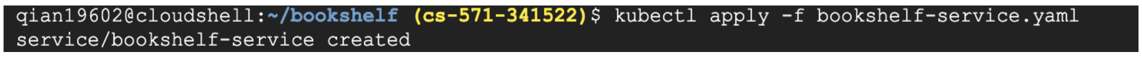
```
qian19602@cloudshell:~/bookshelf (cs-571-341522)$ kubectl apply -f bookshelf-deployment.yaml
deployment.apps/bookshelf-deployment created
```

kubectl apply -f bookshelf-configmap.yaml



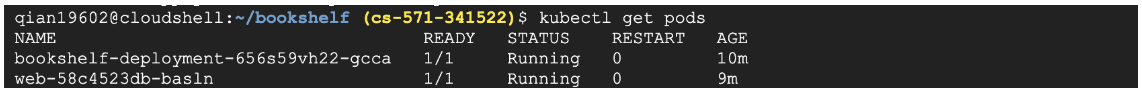
```
qian19602@cloudshell:~/bookshelf (cs-571-341522)$ kubectl apply -f bookshelf-configmap.yaml
configmap/bookshelf-config created
```

kubectl apply -f bookshelf-service.yaml



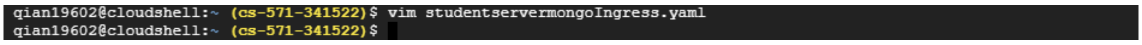
```
qian19602@cloudshell:~/bookshelf (cs-571-341522)$ kubectl apply -f bookshelf-service.yaml
service/bookshelf-service created
```

9. Check if all the pods are running correctly kubectl get pods



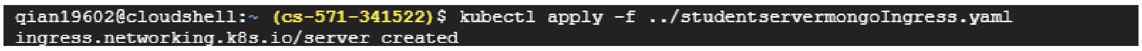
```
qian19602@cloudshell:~/bookshelf (cs-571-341522)$ kubectl get pods
NAME                                  READY   STATUS    RESTART   AGE
bookshelf-deployment-656s59vh22-gcca  1/1     Running   0         10m
web-58c4523db-basln                   1/1     Running   0         9m
```

10. Create an ingress service yaml file called studentservermongoIngress.yaml



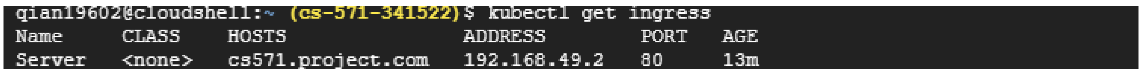
```
qian19602@cloudshell:~ (cs-571-341522)$ vim studentservermongoIngress.yaml
qian19602@cloudshell:~ (cs-571-341522)$
```

11. Create the ingress service using the above yaml file kubectl apply -f ../studentservermongoIngress.yaml



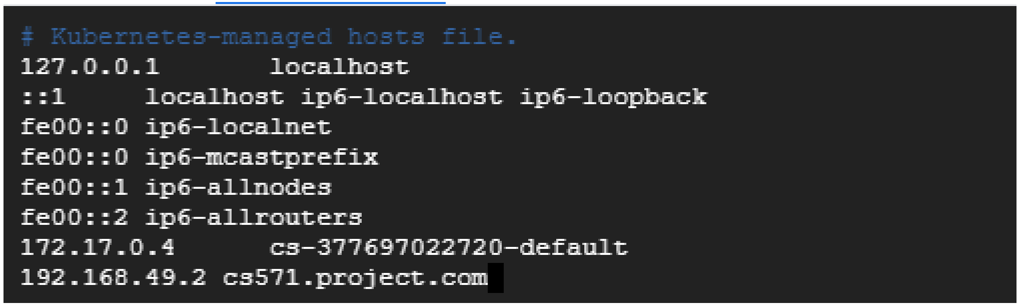
```
qian19602@cloudshell:~ (cs-571-341522)$ kubectl apply -f ../studentservermongoIngress.yaml
ingress.networking.k8s.io/server created
```

12. Check if ingress is running kubectl get ingress Please wait until you see the Address, then move forward



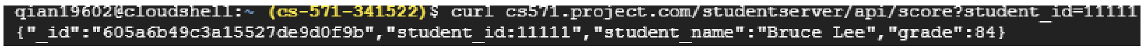
```
qian19602@cloudshell:~ (cs-571-341522)$ kubectl get ingress
Name     CLASS    HOSTS              ADDRESS        PORT   AGE
Server   <none>   cs571.project.com  192.168.49.2   80     13m
```

13. 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 Your /etc/hosts file should look something like this after adding the line, but your address should be different from min

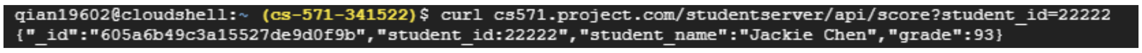


```
# Kubernetes-managed hosts file.
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
fe00::0 ip6-mcastprefix
fe00::1 ip6-allnodes
fe00::2 ip6-allrouters
172.17.0.4      cs-377697022720-default
192.168.49.2 cs571.project.com
```

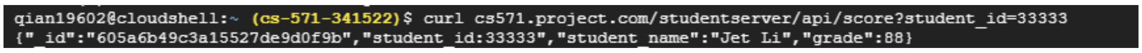
14. If everything goes smoothly, you should be able to access your applications



```
qian19602@cloudshell:~ (cs-571-341522)$ curl cs571.project.com/studentserver/api/score?student_id=11111
{"_id":"605a6b49c3a15527de9d0f9b","student_id:11111","student_name":"Bruce Lee","grade":84}
```

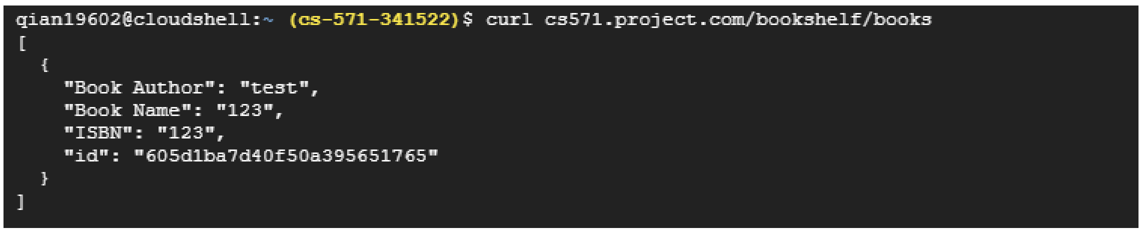


```
qian19602@cloudshell:~ (cs-571-341522)$ curl cs571.project.com/studentserver/api/score?student_id=22222
{"_id":"605a6b49c3a15527de9d0f9b","student_id:22222","student_name":"Jackie Chen","grade":93}
```



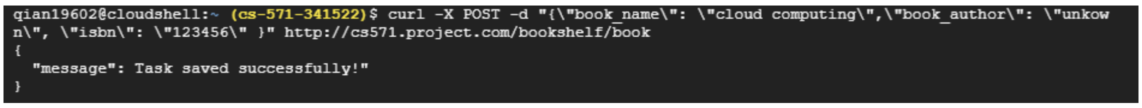
```
qian19602@cloudshell:~ (cs-571-341522)$ curl cs571.project.com/studentserver/api/score?student_id=33333
{"_id":"605a6b49c3a15527de9d0f9b","student_id:33333","student_name":"Jet Li","grade":88}
```

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

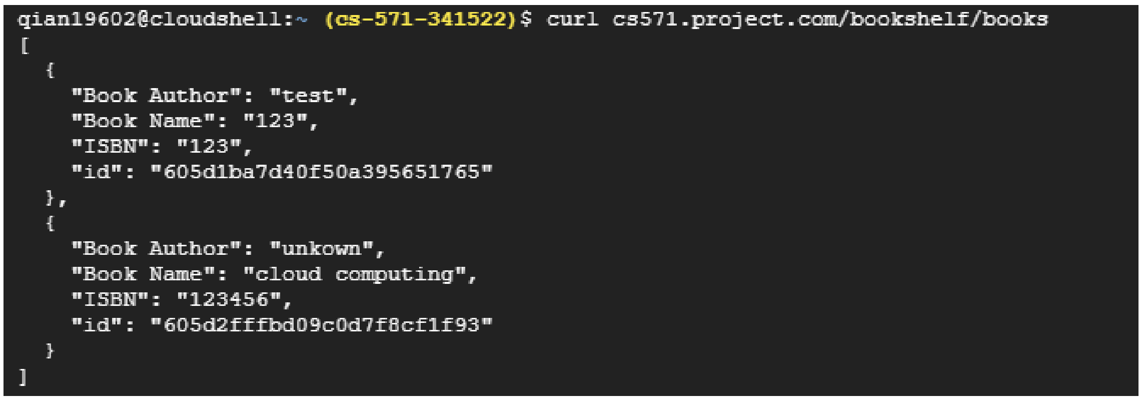


```
qian19602@cloudshell:~ (cs-571-341522)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123",
    "id": "605d1ba7d40f50a395651765"
  }
]
```

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

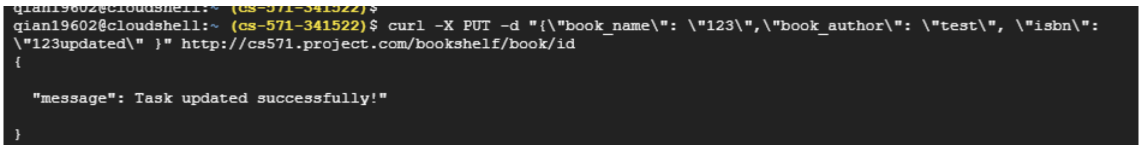


```
qian19602@cloudshell:~ (cs-571-341522)$ curl -X POST -d "{\"book_name\": \"cloud computing\",\"book_author\": \"unkow
n\", \"isbn\": \"123456\" }" http://cs571.project.com/bookshelf/book
{
  "message": Task saved successfully!"
}
```

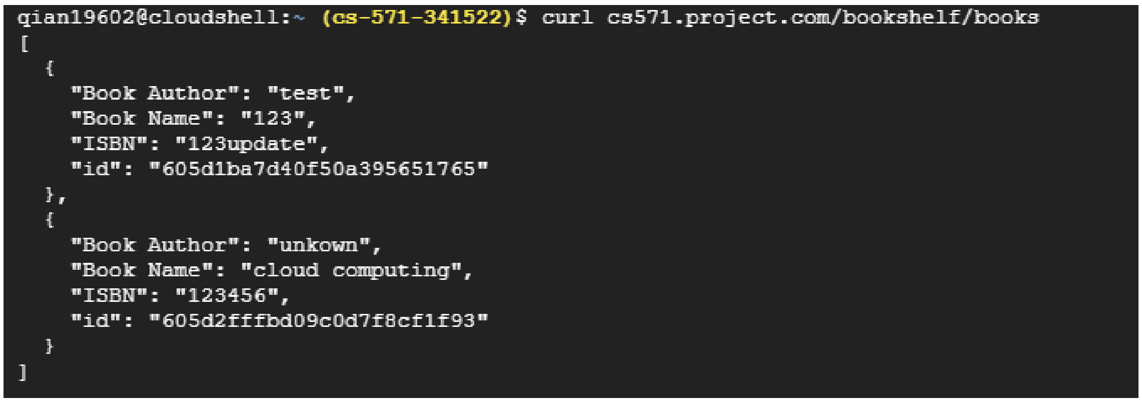


```
qian19602@cloudshell:~ (cs-571-341522)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123",
    "id": "605d1ba7d40f50a395651765"
  },
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "605d2fffbd09c0d7f8cf1f93"
  }
]
```

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

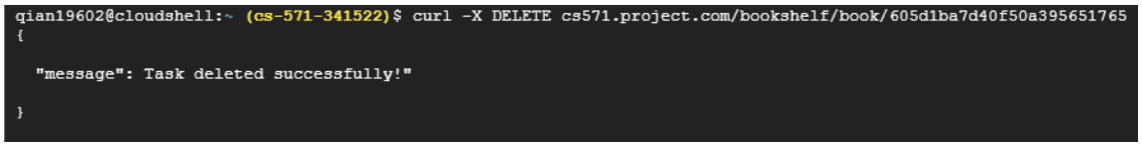


```
qian19602@cloudshell:~ (cs-571-341522)$
qian19602@cloudshell:~ (cs-571-341522)$ curl -X PUT -d "{\"book_name\": \"123\",\"book_author\": \"test\", \"isbn\":
\"123updated\" }" http://cs571.project.com/bookshelf/book/id
{

  "message": Task updated successfully!"

}
```

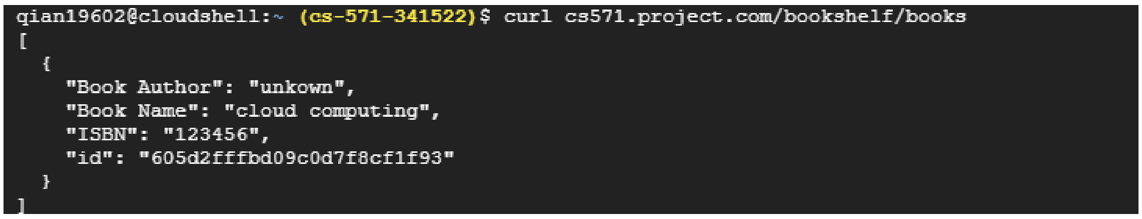


```
qian19602@cloudshell:~ (cs-571-341522)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "test",
    "Book Name": "123",
    "ISBN": "123update",
    "id": "605d1ba7d40f50a395651765"
  },
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "605d2fffbd09c0d7f8cf1f93"
  }
]
```

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



```
qian19602@cloudshell:~ (cs-571-341522)$ curl -X DELETE cs571.project.com/bookshelf/book/605d1ba7d40f50a395651765
{

  "message": Task deleted successfully!"

}
```



```
qian19602@cloudshell:~ (cs-571-341522)$ curl cs571.project.com/bookshelf/books
[
  {
    "Book Author": "unkown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "605d2fffbd09c0d7f8cf1f93"
  }
]
```