CONFIGMAP: SIGNATURE PROJECT

MONGODB + PYTHON FLASK WEB FRAMEWORK + REST API + GKE

YUBRAJ NIRAULA (20156)

TABLE OF CONTENTS

INTRODUCTION

- PLATFORM USED: GOOGLE CLOUD PLATFORM
- TECHNOLOGIES USED:
  - GOOGLE KUBERNETES ENGINE
  - MONGODB
  - PYTHON FLASK WEB FRAMEWORK
  - REST API
  - NODE.JS
  - DOCKER
- OBJECTIVE: DEPLOY MONGODB ON GKE WITH PERSISTENT STORAGE AND DEPLOY TWO APPLICATIONS (STUDENT SERVER AND BOOKSHELF).

INTRODUCTION

- ARCHITECTURE:
  - GKE CLUSTER SETUP.
  - PERSISTENT VOLUME FOR MONGODB.
  - DEPLOYMENT OF MONGODB, STUDENT SERVER, AND BOOKSHELF APPLICATIONS.
- WHY THIS APPROACH?
  - KUBERNETES FOR SCALABLE AND MANAGED CONTAINER ORCHESTRATION.
  - PERSISTENT STORAGE ENSURES DATA DURABILITY.
  - MICROSERVICES ARCHITECTURE WITH SEPARATE DEPLOYMENTS FOR MONGODB, STUDENT SERVER, AND BOOKSHELF APPLICATIONS.

IMPLEMENTATION

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

```
NAME: kubia
LOCATION: us-west1
MASTER_VERSION: 1.29.6-gke.1038001
MASTER_IP: 35.230.9.113
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.29.6-gke.1038001
NUM_NODES: 3
STATUS: RUNNING
yniraula730@cloudshell:~ (my-project-cs571-423503)$
```

IMPLEMENTATION

2. Create a Persistent Volume first, if you have created a persistent volume for the week10's homework, you can skip this one

1. $ gcloud compute disks create --size=10GiB --zone=us-west1-b mongodb-b

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ gcloud compute disks create --size=10GiB --zone=us-west1-b mongodb
-b
WARNING: You have selected a disk size of under [200GB]. This may result in poor I/O performance. For more information
, see: https://developers.google.com/compute/docs/disks#performance.
Created [https://www.googleapis.com/compute/v1/projects/my-project-cs571-423503/zones/us-west1-b/disks/mongodb-b].
NAME: mongodb-b
ZONE: us-west1-b
SIZE_GB: 10
TYPE: pd-standard
New disks are unformatted. You must format and mount a disk before it
can be used. You can find instructions on how to do this at:

https://cloud.google.com/compute/docs/disks/add-persistent-disk#formatting
```

IMPLEMENTATION

3. Now create a mongodb deployment with this yaml file and apply it.

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

Now apply the yaml file:
$ kubectl apply -f mongodb-deployment.yaml

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ vi mongodb-deployment.yaml
yniraula730@cloudshell:~ (my-project-cs571-423503)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
```

IMPLEMENTATION

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

$ kubectl get pods

Please wait until you see the STATUS is running, then you can move forward.

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ kubectl get pods
NAME                                   READY   STATUS    RESTARTS   AGE
mongodb-deployment-64587b98f4-hwkk6    1/1     Running   0          39s
```

IMPLEMENTATION

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

```
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
```

Then apply the service.

$ kubectl apply -f mongodb-service.yaml

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ 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

Please wait until you see the external-ip is generated for mongodb-service, then you can move forward.

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ kubectl get svc
NAME              TYPE           CLUSTER-IP      EXTERNAL-IP    PORT(S)           AGE
kubernetes        ClusterIP      34.118.224.1    <none>         443/TCP           67m
mongodb-service   LoadBalancer   34.118.232.81   35.227.173.7   27017:31452/TCP   40s
```

IMPLEMENTATION

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> -- mongosh

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ kubectl exec -it mongodb-deployment-64587b98f4-hwkk6 -- mongosh
Current Mongosh Log ID: 66a197b58441151b62149f47
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh
+2.2.10
Using MongoDB:          7.0.12
Using Mongosh:          2.2.10
mongosh 2.2.12 is available for download: https://www.mongodb.com/try/download/shell

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

------
   The server generated these startup warnings when booting
   2024-07-24T21:24:58.100+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine.
 See http://dochub.mongodb.org/core/prodnotes-filesystem
   2024-07-24T21:24:59.429+00:00: Access control is not enabled for the database. Read and write access to data and co
nfiguration is unrestricted
   2024-07-24T21:24:59.429+00:00: vm.max_map_count is too low
------
```

Type exit to exit the current MongoDB shell session.

IMPLEMENTATION

8. Now, you're back in the container's bash shell. From here, you can connect to an external MongoDB instance using:

$ mongosh mongodb://<your external ip>:<port>

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ mongosh mongodb://35.227.173.7:27017
-bash: mongosh: command not found
```

Since mongosh command is not found, we first need to install the MongoDB Shell (mongosh) in your Cloud Shell environment. You can do this with the following commands:

wget https://downloads.mongodb.com/compass/mongosh-1.10.1-linux-x64.tgz

tar -zxvf mongosh-1.10.1-linux-x64.tgz

sudo cp mongosh-1.10.1-linux-x64/bin/mongosh /usr/local/bin/

After installation, you should be able to

connect to your MongoDB

instance using the same command.

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ mongosh mongodb://35.227.173.7:27017
Current Mongosh Log ID: 66a19899b0c73b9a91f531c5
Connecting to:          mongodb://35.227.173.7:27017/?directConnection=true&appName=mongosh+1.10.1
Using MongoDB:          7.0.12
Using Mongosh:          1.10.1

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


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

------
   The server generated these startup warnings when booting
   2024-07-24T21:24:58.100+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engi
 See http://dochub.mongodb.org/core/prodnotes-filesystem
   2024-07-24T21:24:59.429+00:00: Access control is not enabled for the database. Read and write access to data and
nfiguration is unrestricted
   2024-07-24T21:24:59.429+00:00: vm.max_map_count is too low
------

test> exit
```

Here you just accessed your mongoDB using the

External-IP of the pod.

IMPLEMENTATION

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

Enter the following node js code:

```
const { MongoClient } = require('mongodb');
const url = "mongodb://35.227.173.7:27017/mydb"; // Using the EXTERNAL-IP and
port 27017

async function run() {
  try {
    const client = await MongoClient.connect(url);
    const db = client.db("studentdb");

    // Create a document to be inserted
    const docs = [
      { student_id: 11111, student_name: "Bruce Lee", grade: 84 },
      { student_id: 22222, student_name: "Jackie Chan", grade: 93 },
      { student_id: 33333, student_name: "Jet Li", grade: 88 }
    ];

    const insertResult = await db.collection("students").insertMany(docs);
    console.log(`${insertResult.insertedCount} documents were inserted`);

    const student = await db.collection("students").findOne({ "student_id": 11111 });
    console.log(student);

    client.close();
  } catch (err) {
    console.error(err);
  }
}

run();
```

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ node
Welcome to Node.js v20.15.1.
Type ".help" for more information.
>
```

If Everything is correct, you should see the following. 3 means three records was inserted, and we tried search for student_id=11111.

```
... }
undefined
>
> run();
Promise {
  <pending>,
  [Symbol(async_id_symbol)]: 5726,
  [Symbol(trigger_async_id_symbol)]: 6
}
> 3 documents were inserted
{
  _id: new ObjectId('66a19f7216824a6722e06016'),
  student_id: 11111,
  student_name: 'Bruce Lee',
  grade: 84
}
```

IMPLEMENTATION

Step2: Modify our studentServer to get records from MongoDB and deploy to GKE.

1. Create a studentServer.js.

$ vi studentServer.js

2. Create Dockerfile

```
FROM node:14

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY studentServer.js

EXPOSE 8080

ENTRYPOINT ["node", "studentServer.js"]
```

3. Build the studentserver docker image

$ docker build -t yourdockerhubID/studentserver .

4. Push the docker image

$ docker push yourdockerhubID/studentserver

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ docker push yuvie/studentserver
Using default tag: latest
The push refers to repository [docker.io/yuvie/studentserver]
790d9ca177e4: Pushed
fd82770d705e: Pushed
af03d1317bd1: Pushed
f97a677fa8b4: Pushed
0d5f5a015e5d: Mounted from library/node
3c777d951de2: Mounted from library/node
f8a91dd5fc84: Mounted from library/node
cb81227abde5: Mounted from library/node
e01a454893a9: Mounted from library/node
c45660adde37: Mounted from library/node
fe0fb3ab4a0f: Mounted from library/node
f1186e5061f2: Mounted from library/node
b2dba7477754: Mounted from library/node
latest: digest: sha256:f2eed505be050d9790d6fa66f38d621ad24192b0c86b00154fffff2a35eed76f size: 3047
```

IMPLEMENTATION

Step 3: Create a python Flask bookshelf REST API and deploy on GKE

1. Create a requirements.txt file or edit it if you already have one.

   $ vi requirements.txt

   Flask==2.0.1

   Flask-PyMongo==2.3.0

2. Create bookshelf.py

   $ vi bookshelf.py

3. Create a Dockerfile or edit it if you already have one.

   $ vi Dockerfile

   FROM python:3.7-alpine

   WORKDIR /app

   COPY . /app

   RUN pip install --no-cache-dir -r requirements.txt

   EXPOSE 5000

   ENV PORT=5000

   CMD ["python", "bookshelf.py"]

IMPLEMENTATION

4. Build the bookshelf app into a docker image

$ docker build -t <yourdockerhubID>/bookshelf .

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ docker build -t yuvie/bookshelf .
[+] Building 18.1s (9/9) FINISHED                                                docker:defaul
 => [internal] load build definition from Dockerfile                                      0.0
 => => transferring dockerfile: 496B                                                      0.0
 => [internal] load metadata for docker.io/library/python:3.7-alpine                      0.1
 => [internal] load .dockerignore                                                         0.0
 => => transferring context: 2B                                                           0.0
 => [1/4] FROM docker.io/library/python:3.7-alpine@sha256:f3d31c8677d03f0b3c724446077f229a6ce9d3ac430f5c08cd7df  0.0
 => [internal] load build context                                                         0.2
 => => transferring context: 162.09kB                                                     0.1
 => CACHED [2/4] WORKDIR /app                                                             0.0
 => [3/4] COPY . /app                                                                     2.4
 => [4/4] RUN pip install --no-cache-dir -r requirements.txt                             13.3
 => exporting to image                                                                    2.1
 => => exporting layers                                                                   2.0
 => => writing image sha256:cef8b89a65b1dbddf159c0ee0ba34f210c24a62cbb4039d5697dd7864c80cdb0  0.0
 => => naming to docker.io/yuvie/bookshelf                                                0.0

 1 warning found (use --debug to expand):
 - LegacyKeyValueFormat: "ENV key=value" should be used instead of legacy "ENV key value" format (line 16)
yniraula730@cloudshell:~ (my-project-cs571-423503)$
```

5. Push the docker image to your dockerhub.

$ docker push <yourdockerhubID>/bookshelf

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ docker push yuvie/bookshelf
Using default tag: latest
The push refers to repository [docker.io/yuvie/bookshelf]
cc5b5f7d0b6c: Pushed
e86f15a96ac0: Pushed
015efb81dad4: Pushed
ae2ed3079163: Mounted from library/python
aa3a591fc84e: Mounted from library/python
7f29b11ef9dd: Mounted from library/python
a1c2f058ec5f: Mounted from library/python
cc2447e1835a: Mounted from library/python
latest: digest: sha256:dc07f675082363b82cb7edec873b33607ff73216b2ac1880cce217496
```

IMPLEMENTATION

Step 4: 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: 35.227.173.7
  MONGO_DATABASE: "mydb"
```

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

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: bookshelf-config
data:
  MONGO_URL: 35.227.173.7
  MONGO_DATABASE: "mydb"
```

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

IMPLEMENTATION

Step 5: 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 studentserver-service.yaml.
4. Create bookshelf-service.yaml.
5. Start minikube

   $ minikube start

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ minikube start
* minikube v1.33.1 on Ubuntu 22.04 (amd64)
  - MINIKUBE_FORCE_SYSTEMD=true
  - MINIKUBE_HOME=/google/minikube
  - MINIKUBE_WANTUPDATENOTIFICATION=false
* Automatically selected the docker driver. Other choices: ssh, none
* Using Docker driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.44 ...
* Downloading Kubernetes v1.30.0 preload ...
    > preloaded-images-k8s-v18-v1...:  342.90 MiB / 342.90 MiB  100.00% 212.01
    > gcr.io/k8s-minikube/kicbase...:  481.58 MiB / 481.58 MiB  100.00% 97.46 M
* Creating docker container (CPUs=2, Memory=4000MB) ...
* Preparing Kubernetes v1.30.0 on Docker 26.1.1 ...
  - kubelet.cgroups-per-qos=false
  - kubelet.enforce-node-allocatable=""
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
* 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 "default" namespace by default
```

6. Start Ingress

   $ minikube addons enable ingress

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ minikube addons enable ingress
* ingress is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.1
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.1
  - Using image registry.k8s.io/ingress-nginx/controller:v1.10.1
* Verifying ingress addon...
* The 'ingress' addon is enabled
```

IMPLEMENTATION

7. Create studentserver related pods and start service using the above yaml file.

kubectl apply -f studentserver-deployment.yaml

kubectl apply -f studentserver-configmap.yaml

kubectl apply -f studentserver-service.yaml

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$  kubectl apply -f studentserver-deployment.yaml

kubectl apply -f studentserver-configmap.yaml

kubectl apply -f studentserver-service.yaml
deployment.apps/studentserver-deploy created
configmap/studentserver-config created
service/web created
```

IMPLEMENTATION

8. Create bookshelf related pods and start service using the above yaml file.

kubectl apply -f bookshelf-deployment.yaml

kubectl apply -f bookshelf-configmap.yaml

kubectl apply -f bookshelf-service.yaml

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ kubectl apply -f bookshelf-deployment.yaml

kubectl apply -f bookshelf-configmap.yaml

kubectl apply -f bookshelf-service.yaml
deployment.apps/bookshelf-deployment created
configmap/bookshelf-config created
service/bookshelf-service created
```

IMPLEMENTATION

9. Check if all the pods are running correctly

$ kubectl get pods

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ kubectl get pods
NAME                                    READY   STATUS    RESTARTS      AGE
bookshelf-deployment-68f4d55d7-gmnrn    1/1     Running   0             28m
web-c6bdf97c9-m47pk                     1/1     Running   0             6s
```

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

IMPLEMENTATION

11. Create the ingress service using the above yaml file.

$ kubectl apply -f studentservermongoIngress.yaml

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ vi studentservermongoIngress.yaml
yniraula730@cloudshell:~ (my-project-cs571-423503)$ kubectl apply -f studentservermongoIngress.yaml
Warning: path /studentserver(/|$)(.*) cannot be used with pathType Prefix
Warning: path /bookshelf(/|$)(.*) cannot be used with pathType Prefix
ingress.networking.k8s.io/server created
```

12. Check if the ingress is running

$ kubectl get ingress

Please wait until you see the Address, then move forward.

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ kubectl get ingress
NAME     CLASS   HOSTS                ADDRESS          PORTS   AGE
server   nginx   cs571.project.com    192.168.49.2     80      74s
```

IMPLEMENTATION

13. Add Address 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 mine.

```
# IPv4 and IPv6 localhost aliases
127.0.0.1       localhost
::1             localhost
192.168.49.2 cs571.project.com
#
# Imaginary network.
```

IMPLEMENTATION

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

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

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ curl cs5
71.project.com/studentserver/api/score?student_id=11111
{"student_id":11111,"student_name":"Bruce Lee","student_score":84}
```

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ curl cs5
71.project.com/studentserver/api/score?student_id=22222
"student_id":22222,"student_name":"Jackie Chen","student_score":93}
```

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ curl cs5
71.project.com/studentserver/api/score?student_id=33333
{"student_id":33333,"student_name":"Jet Li","student_score":88}
```

IMPLEMENTATION

15. First you need to export your Mongo URL and Mongo Database and then add a book:

$ Export MONGO_URL=35.227.173.7

$ Export MONGO_DATABASE=studentdb

curl -X POST -H "Content-Type: application/json" -d '{"book_name": "cloud computing", "book_author": "unknown", "ISBN": "123456"}' http://cs571.project.com/bookshelf/book

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ curl -X
POST -H "Content-Type: application/json" -d '{"book_name": "cloud computing",
 "book_author": "unknown", "ISBN": "123456"}' http://cs571.project.com/booksh
elf/book
{
  "message": "Book saved successfully!"
}
```

IMPLEMENTATION

16. Update a book

curl -X PUT -H "Content-Type: application/json" \

-d '{"book_name": "Updated Book Name", "book_author": "Updated Author", "ISBN": "Updated ISBN"}' \

http://cs571.project.com/bookshelf/book/66a91c1c4e229f7d052a2c81

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ curl -X
PUT -H "Content-Type: application/json" \
-d '{"book_name": "Updated Book Name", "book_author": "Updated Author", "ISBN
": "Updated ISBN"}' \
http://cs571.project.com/bookshelf/book/66a91c1c4e229f7d052a2c81
{
  "message": "Book updated successfully!"
}
```

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ curl cs5
71.project.com/bookshelf/books
[
  {
    "Book Author": "unknown",
    "Book Name": "cloud computing",
    "ISBN": "123456",
    "id": "66a91c1c4e229f7d052a2c81"
  }
]
```

IMPLEMENTATION

17. Delete a book

curl -X DELETE http://cs571.project.com/bookshelf/book/66a91c1c4e229f7d052a2c81

```
yniraula730@cloudshell:~ (my-project-cs571-423503)$ curl -X
DELETE http://cs571.project.com/bookshelf/book/66a91c1c4e229f7d052a2c81
{
  "message": "Book deleted successfully!"
}
```

ENHACEMENT IDEAS

- IMPLEMENTING SECURITY BEST PRACTICES (E.G., NETWORK POLICIES, SECRET MANAGEMENT).
- ADDING MONITORING AND LOGGING SOLUTIONS (E.G., PROMETHEUS, ELK STACK).
- AUTOMATING DEPLOYMENT WITH CI/CD PIPELINES.

CONCLUSION

- IN THIS EXERCISE, WE SUCCESSFULLY DEPLOYED MONGODB WITH PERSISTENT STORAGE ON GKE.
- WE DEPLOYED AND TESTED STUDENT SERVER AND BOOKSHELF APPLICATIONS.
- WE IDENTIFIED POTENTIAL AREAS FOR FURTHER IMPROVEMENT AND ENHANCEMENT OF THIS PROJECT.

REFERENCES

- HTTPS://CHATGPT.COM
- MONGODB: THE DEVELOPER DATA PLATFORM | MONGODB
- WHAT IS REST?: REST API TUTORIAL (RESTFULAPI.NET)
- FLASK TUTORIAL - GEEKSFORGEEKS

APPENDIX

- CLOUD-COMPUTING/KUBERNETES/MONGODB + PYTHON FLASK WEB FRAMEWORK + REST API + GKE AT MAIN · YUBRAJNIRAULA/CLOUD-COMPUTING (GITHUB.COM)
- CS571_WEEK11_HW4_Q5_20156_YUBRAJ_NIRAULA - GOOGLE SLIDES

THANK YOU