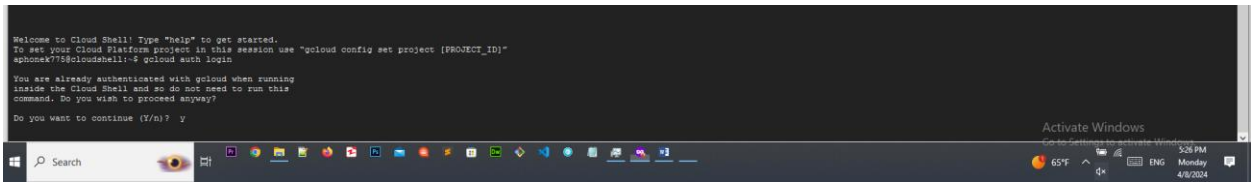
Aung Phone Kyaw

19930

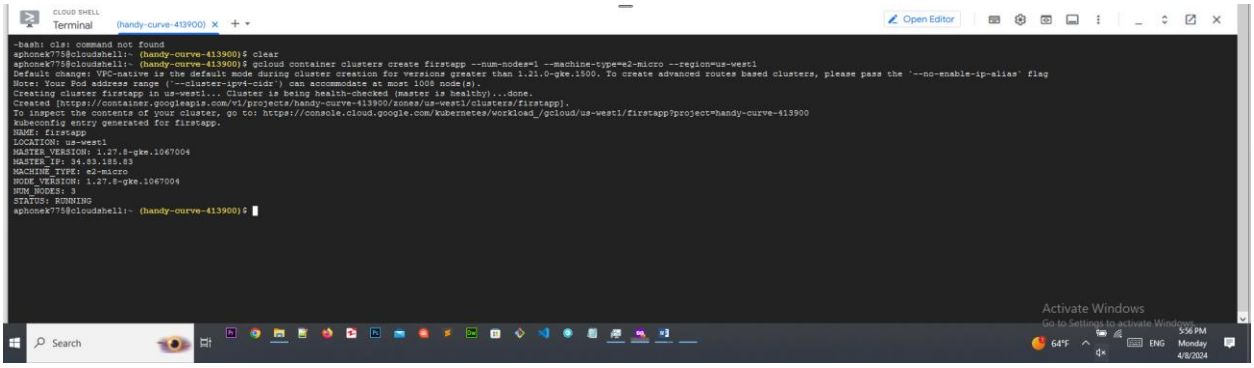
CS571

Signature project

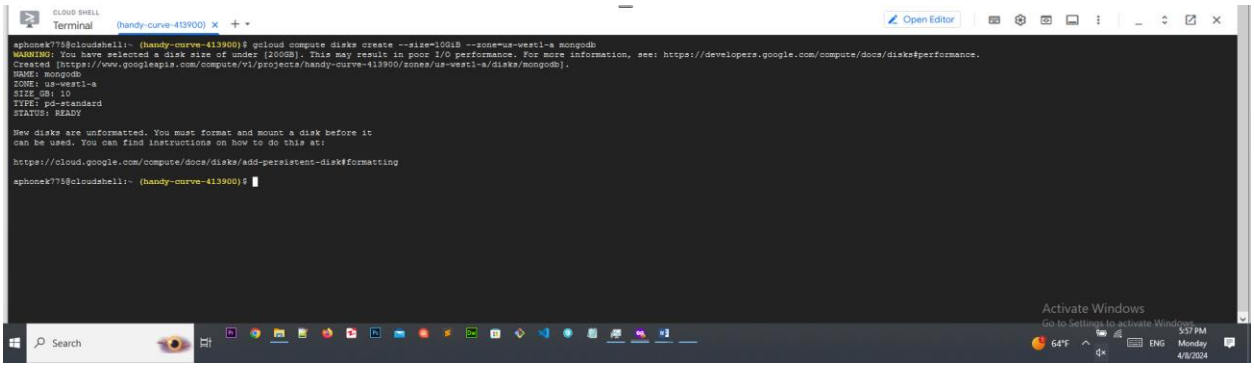
<u>Create MongoDB using Persistent Volume on GKE, and insert records into it</u>



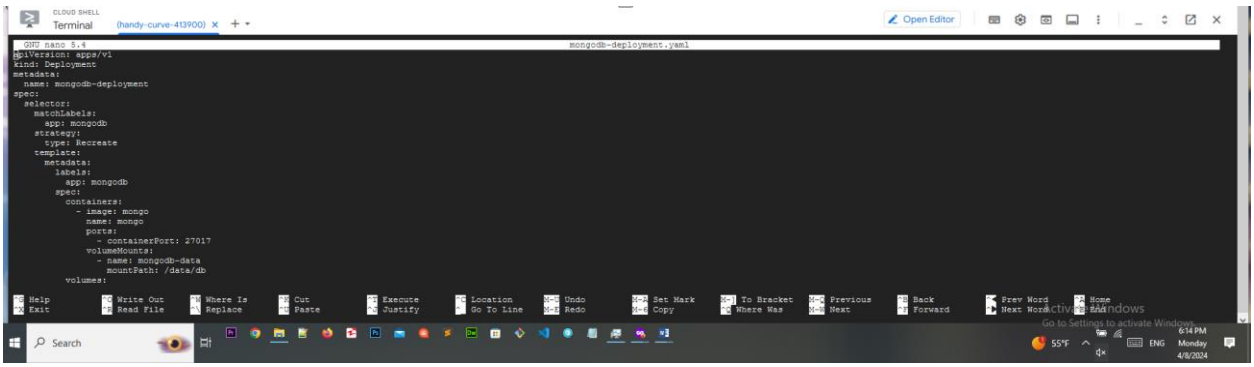
Login to gcp account using this command “gcloud auth login”



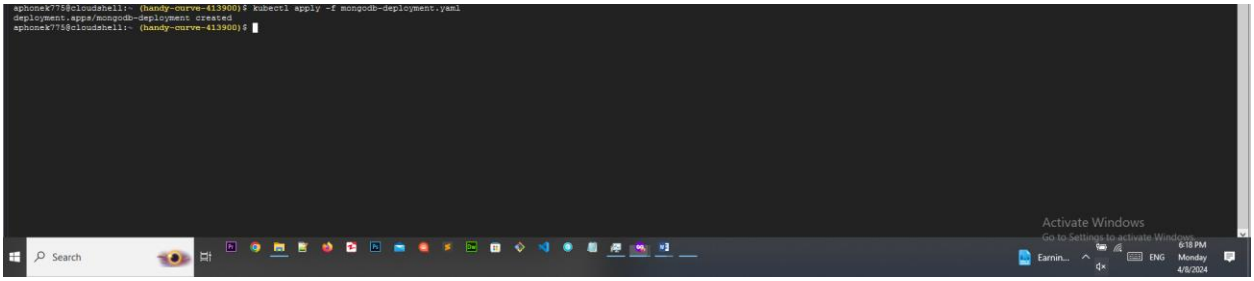
Create a cluster using this command “gcloud container clusters create firstapp --num-nodes=1 --machine-type=e2-micro --region=us-west1”



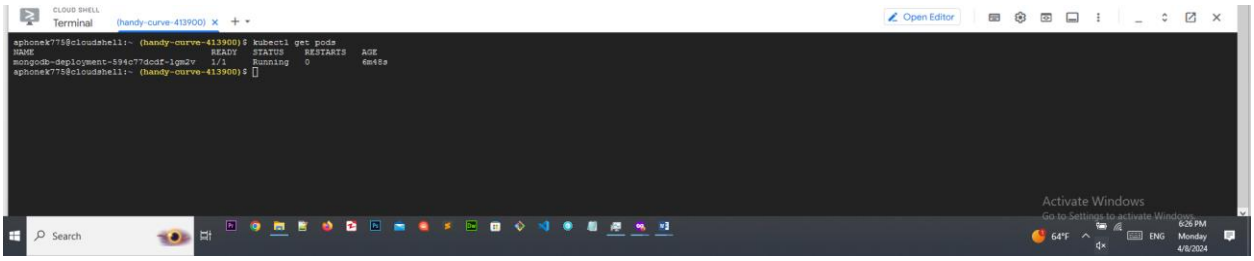
Create a persistent volume using this command “gcloud compute disks create --size=10GiB --zone=us-west1-a mongodb”



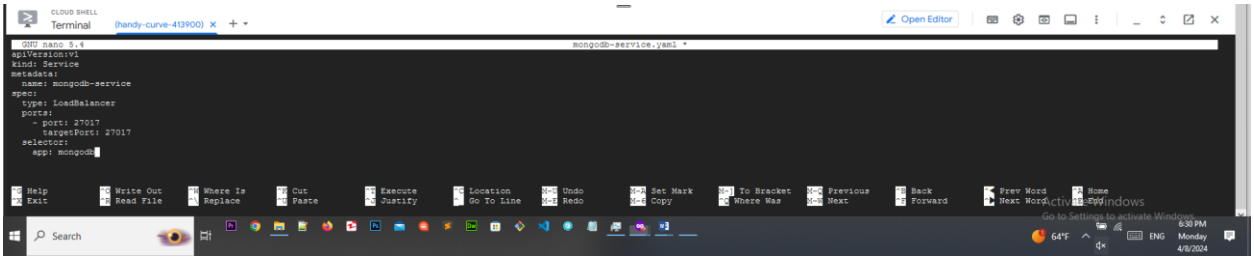
Create a file name called mongodb-deployment.yaml using this command “nano mongodb-deployment.yaml” and enter the configuration



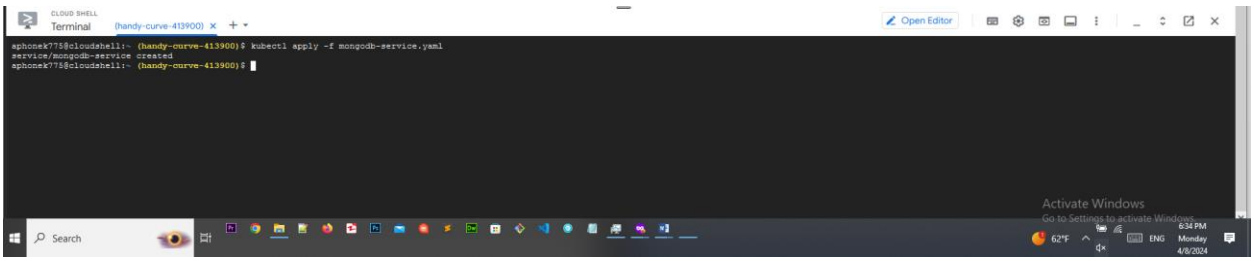
Create mongodb deployment using this command “kubectl apply -f mongodb-deployment.yaml”



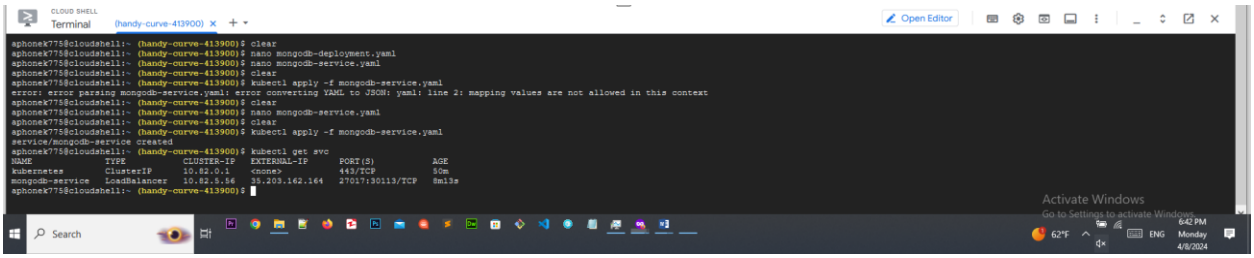
Check if the pod is created by using this command “kubectl get pods”



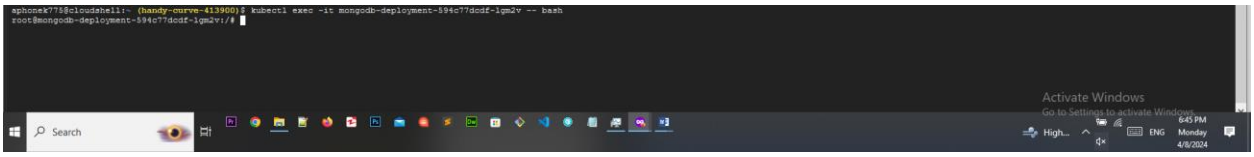
Create a file mongodb-service.yaml for a service to access mongodb from outside by creating service configuration file



Create mongodb service using this command “kubectl apply -f mongodb-service.yaml”

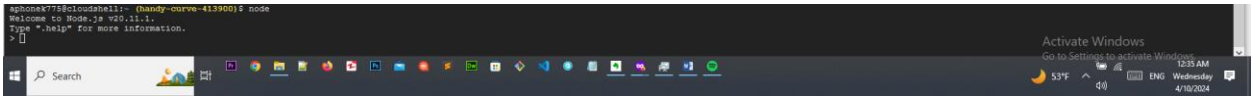


Check if the service is running by using this command “kubectl get svc”

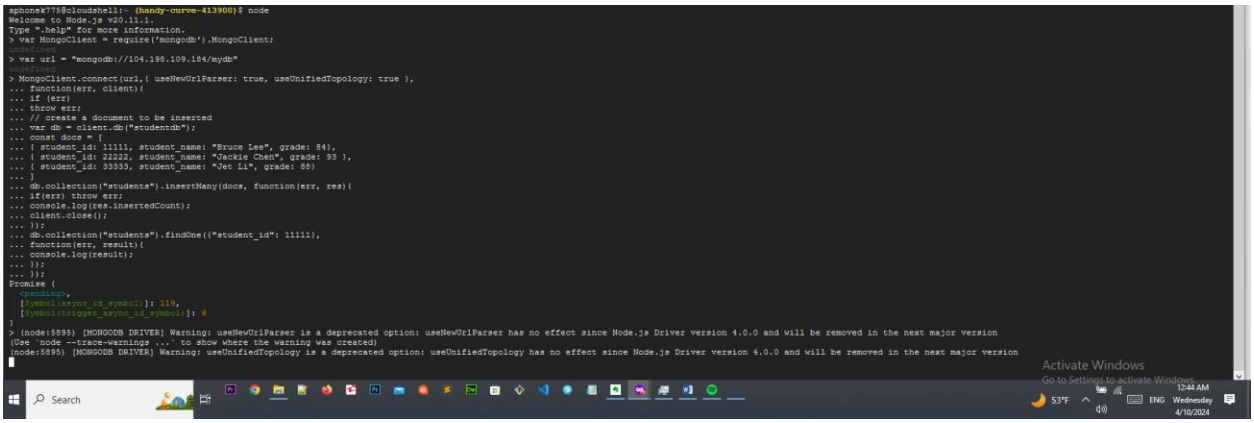


Use created pod using this command “kubectl exec -it mongodb-deployment-pod_name -- bash”

Use exit to stop accessing mongo db pod



Use node by typing node to insert data into mongo db

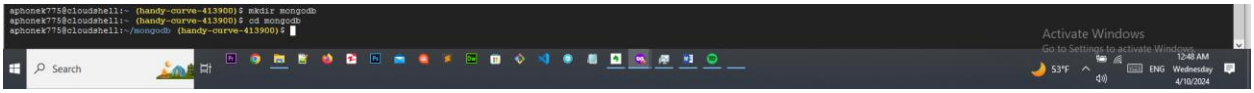


Add data into mogo db using these code

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://EXTERNAL-IP/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);
});
});
```

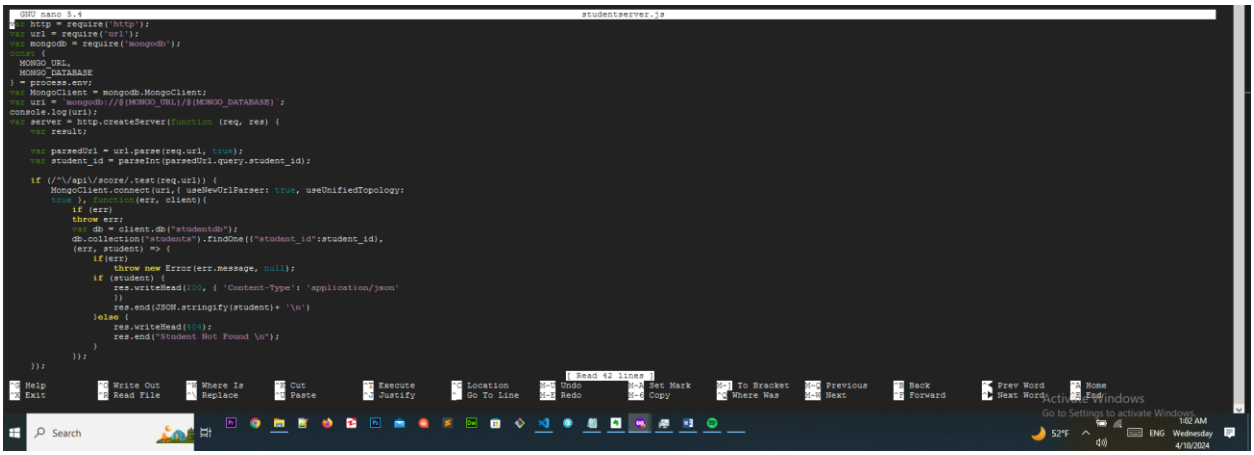
Make sure to enter them line by line.



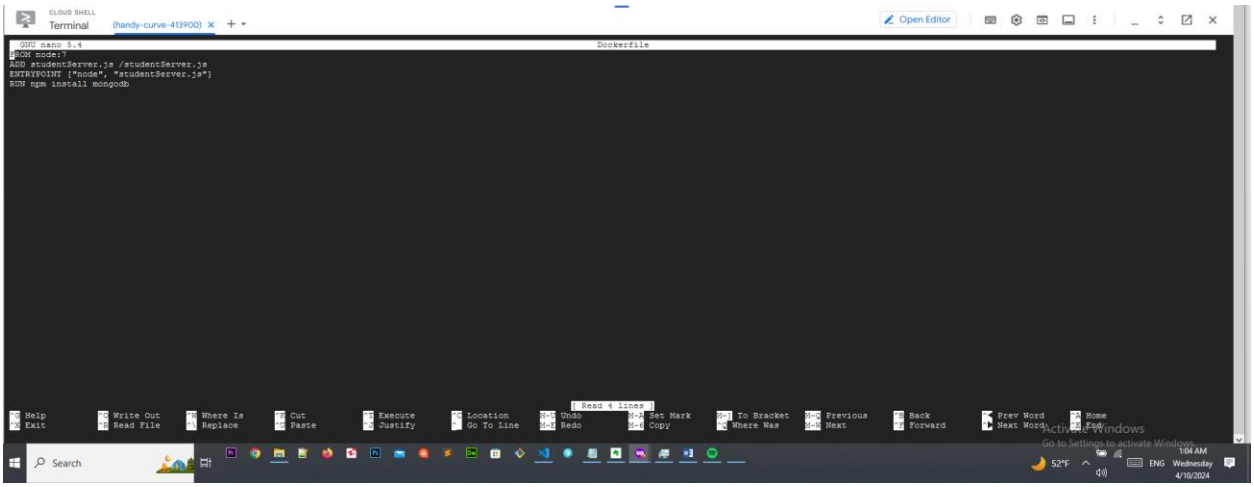
Create a folder called mongodb using this command “mkdir mongodb”

Go to folder path using this command “cd mongodb”

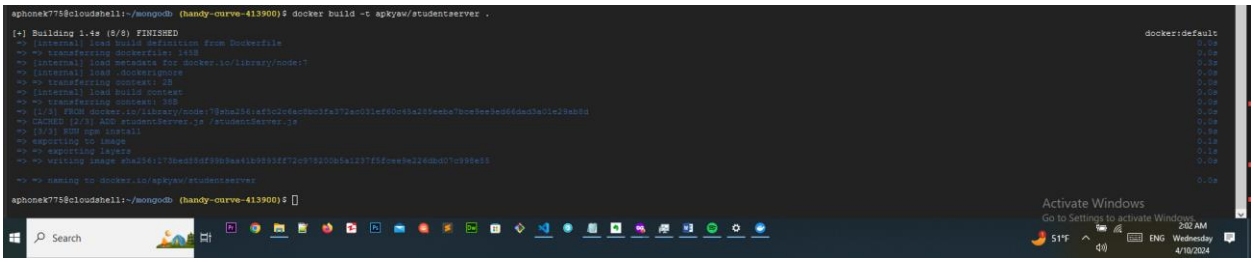
<u>Modify our studentServer to get records from MongoDB and deploy to GKE</u>



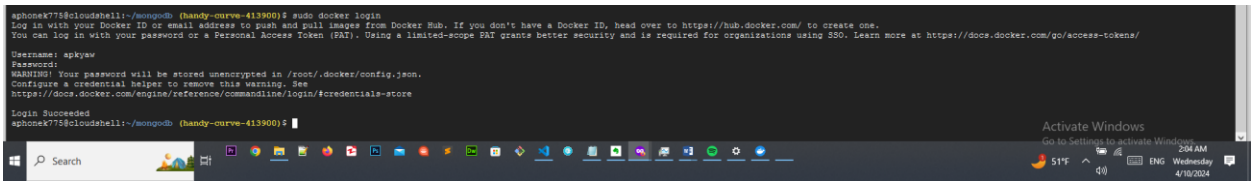
Create a file called studentserver.js and enter these line of codes



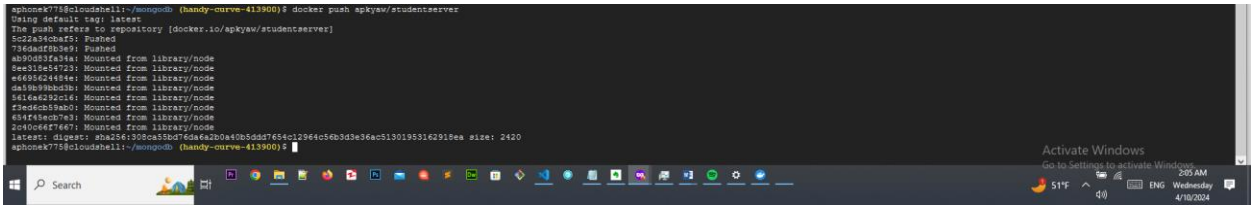
Create Dockerfile and entering these configuration codes



Build the docker file using this command “docker build -t yourdockerrepository/studentserver .”

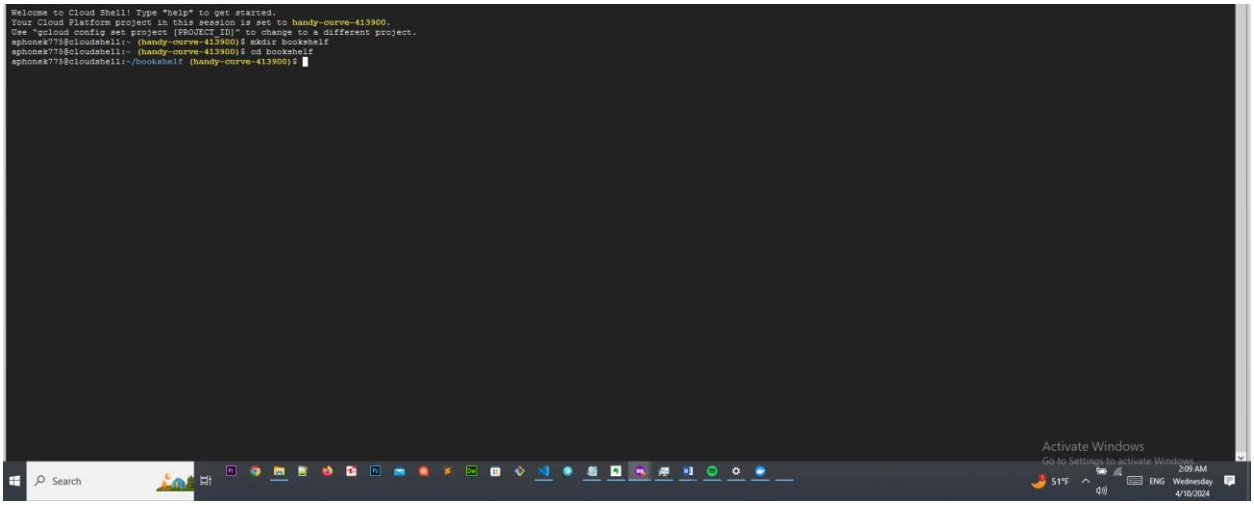


Login to docker account using this command “sudo docker login”

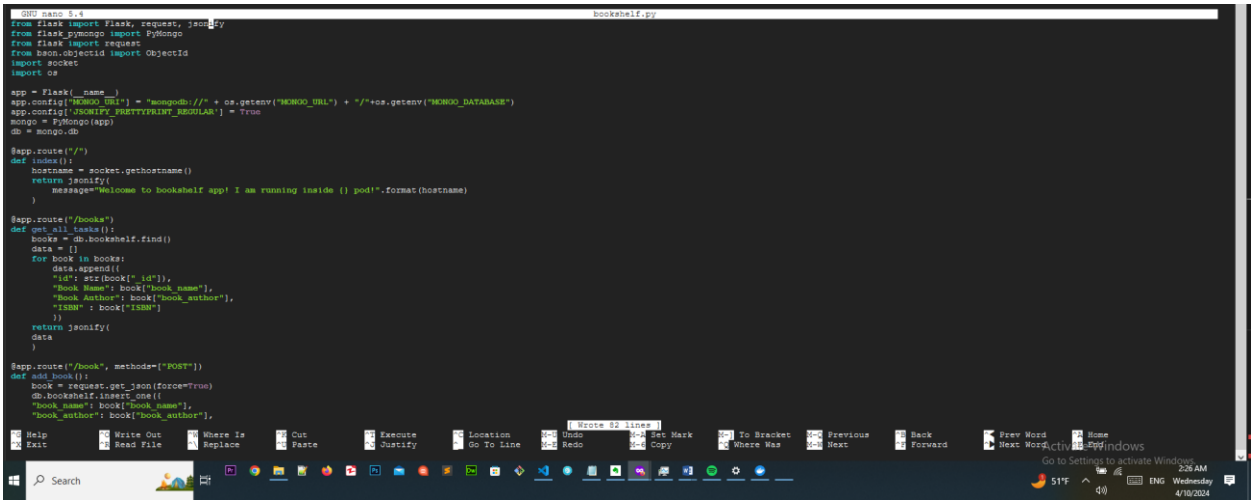


Push the image to docker hub repository using this command “docker push your_repo_name/studentserver”

Create a python Flask bookshelf REST API and deploy on GKE



Create a folder called bookshelf using this command “mkdir bookshelf” and go to bookself folder directory using this “cd bookshelf”

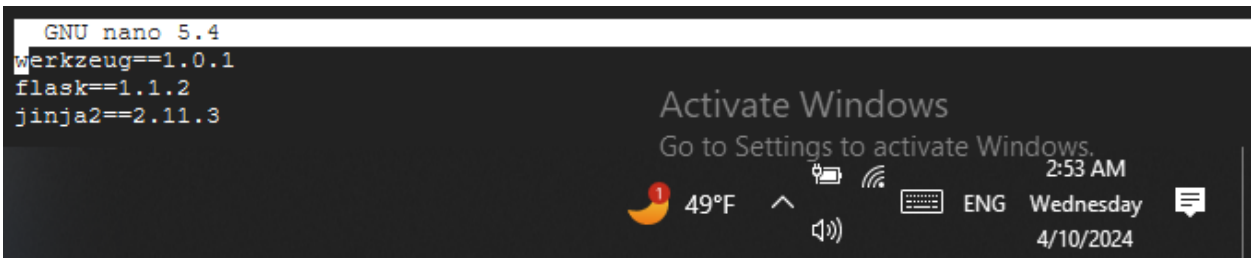


Create a file called bookshelf.py and add these following codes

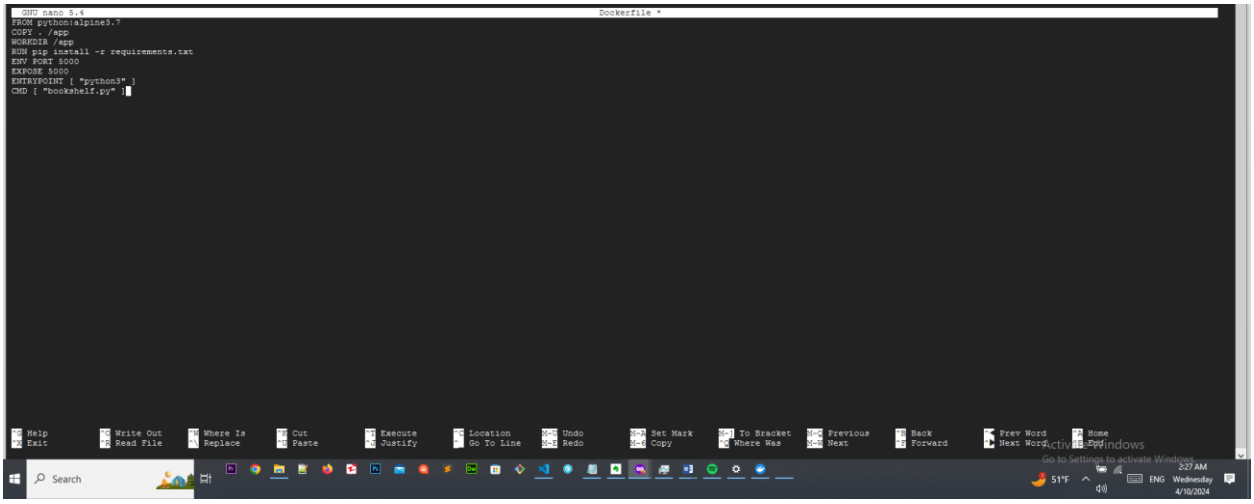
```
from flask import Flask, request, jsonify
from flask_pymongo import PyMongo
from flask import request
from bson.objectid import ObjectId
import socket
import os

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"]
```

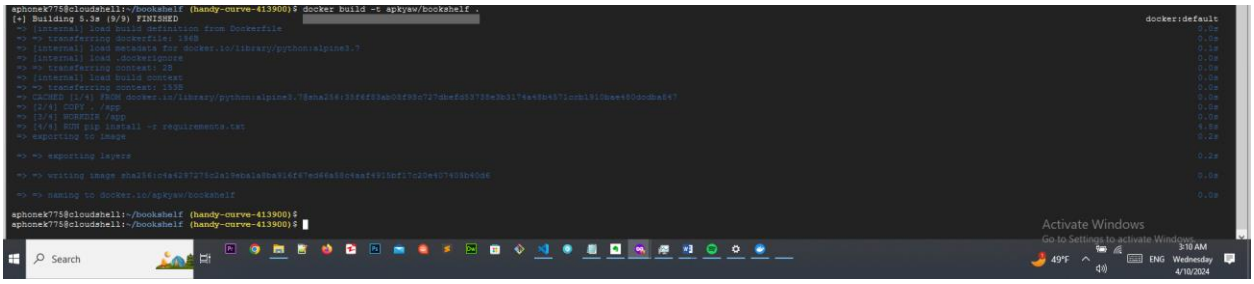
```python
    }})
    if response.matched_count:
        message = "Task updated successfully!"
    else:
        message = "No book found!"
    return jsonify(
    message=message
    )

@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)
```



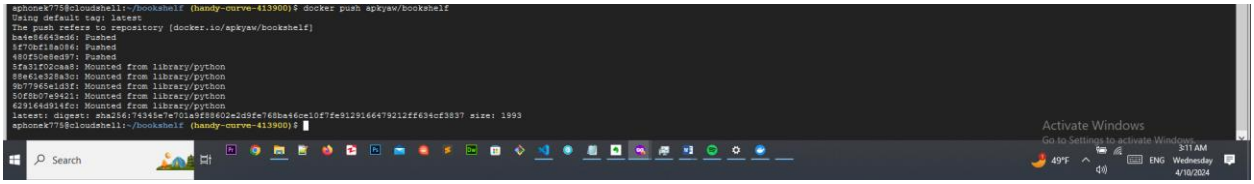
Create requirement.txt file and add these modules



Create a docker file and add these build step codes

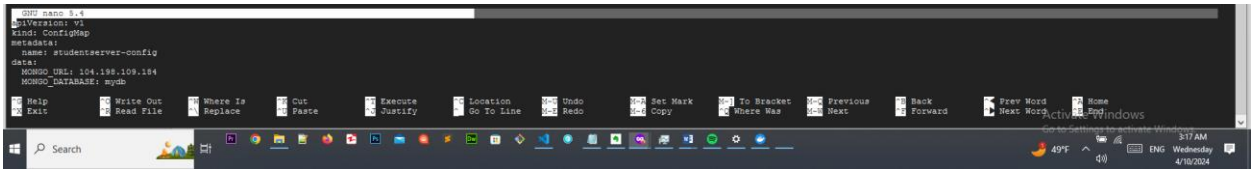


Build the docker image using this command “docker build -t apkyaw/bookshelf .”

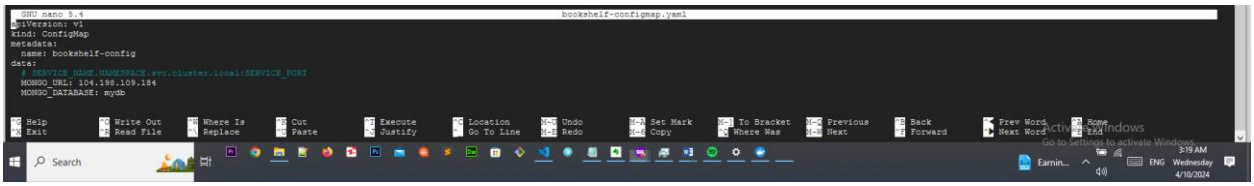


Push docker image to repository using this command “docker push your_repository/bookshelf”

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



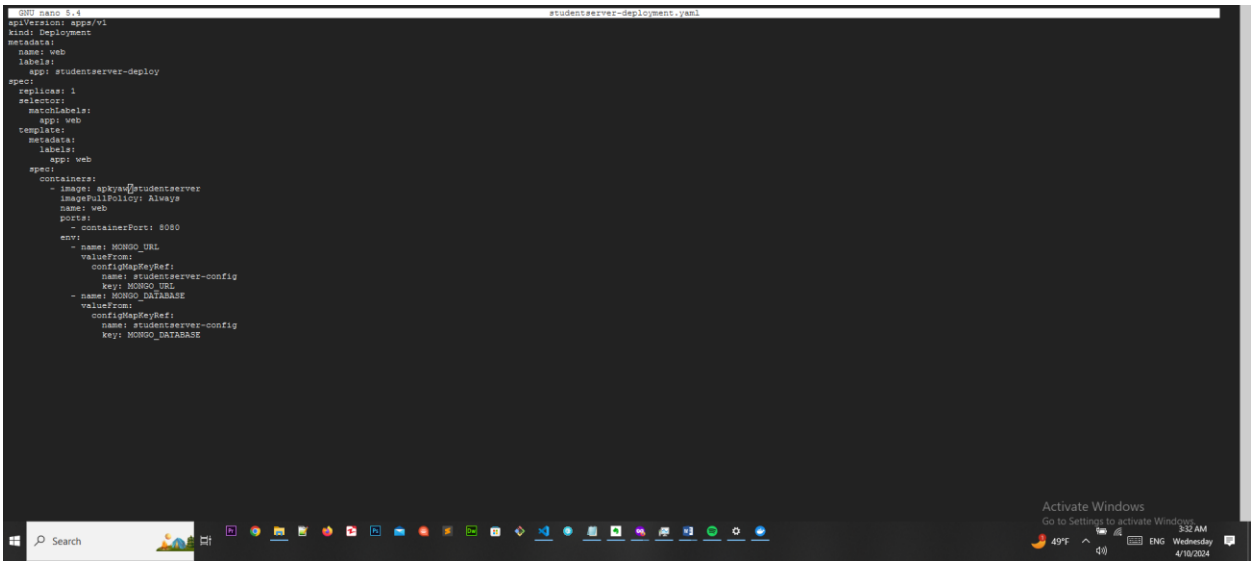
Create file studentserver-configmap.yaml and add these codes



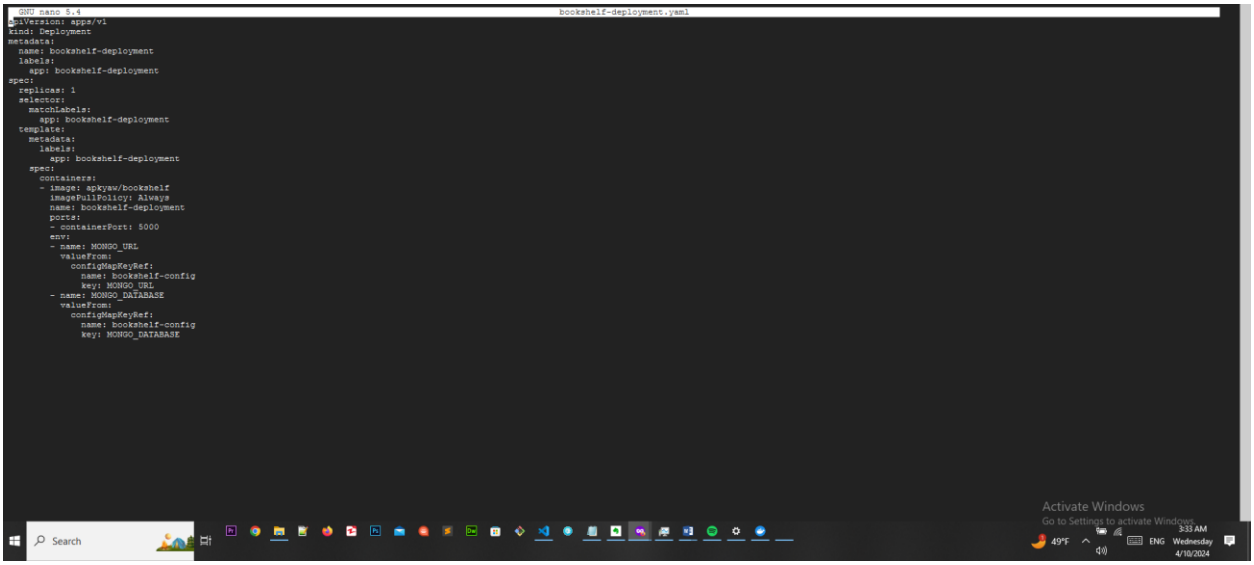
Create file bookshelf-configmap.yaml and add these codes

The reason for these two config files are to avoid rebuilding docker image again if the mongoDB pod restarts with a different External-IP

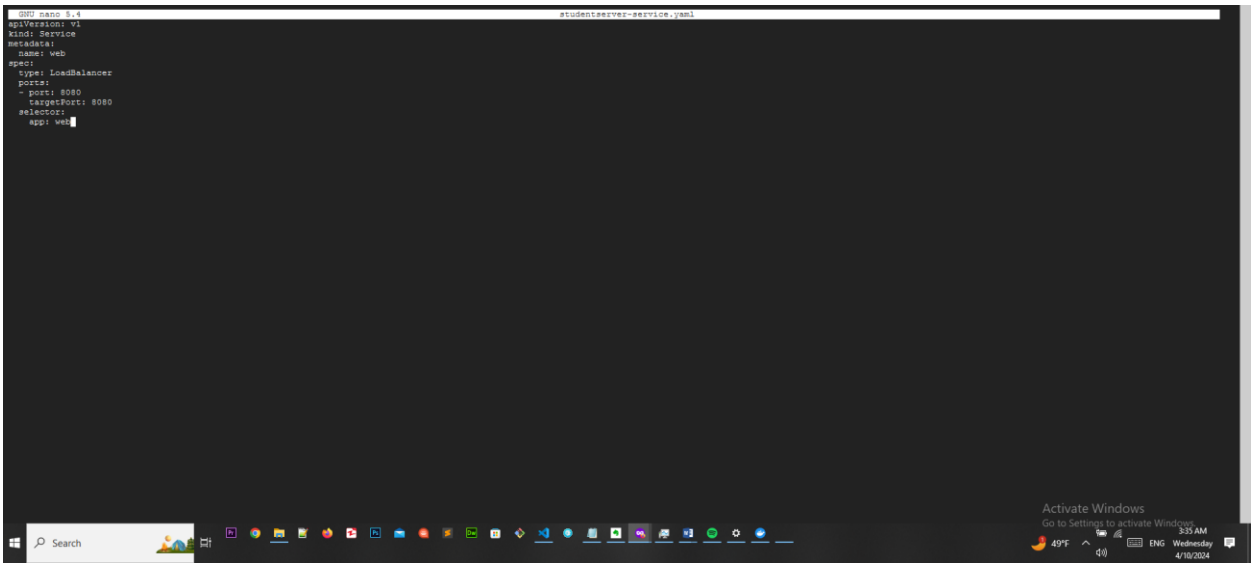
Expose 2 applications using ingress with Nginx, so we can put them on the same Domain but different PATH



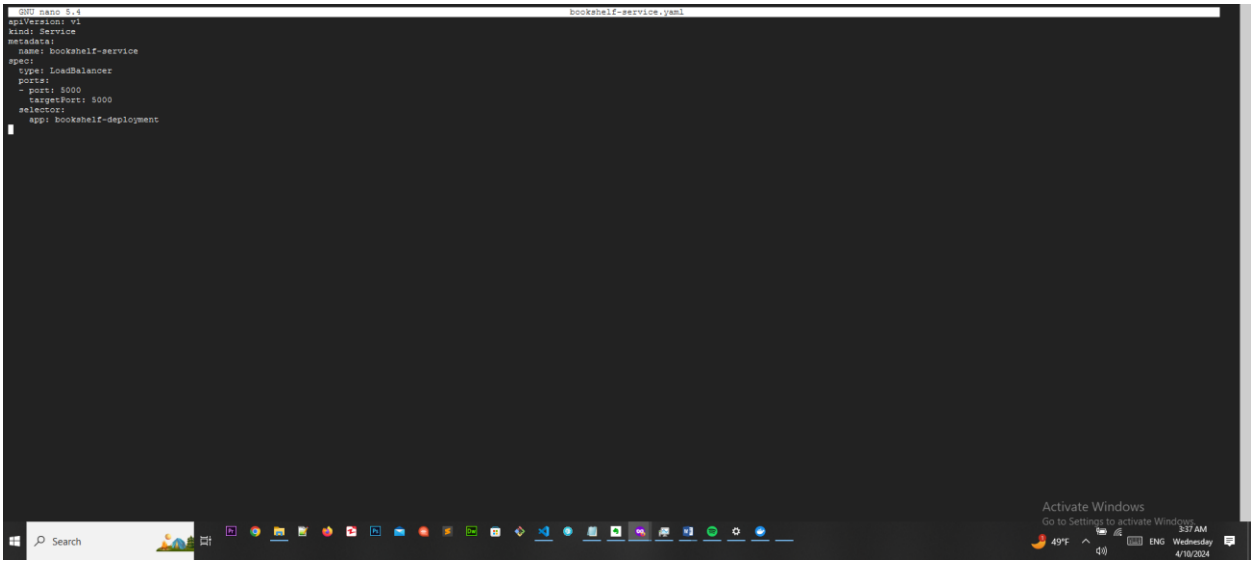
Create a file studentserver-deployment.yaml



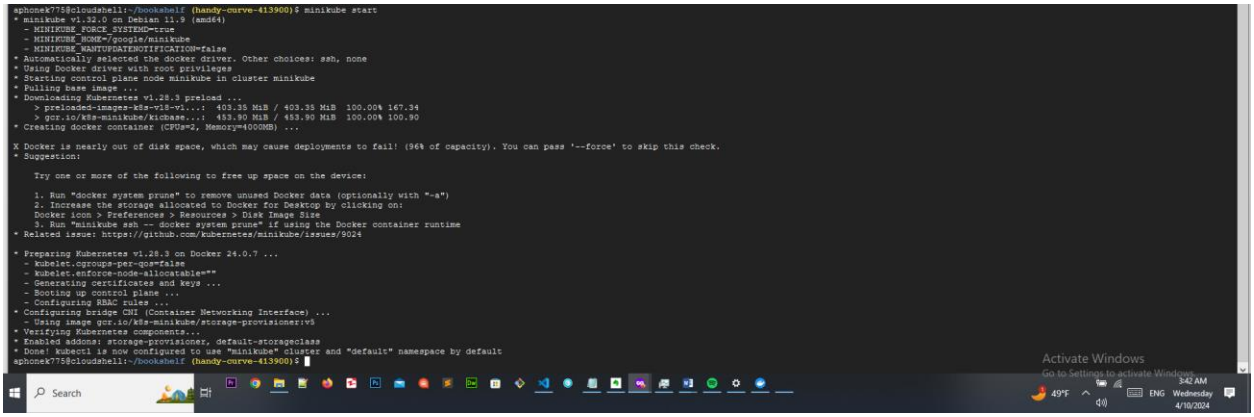
Create a file bookshelf-deployment.yaml



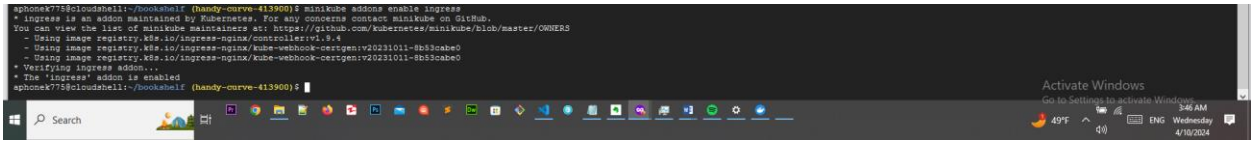
Create a file studentserver-service.yaml



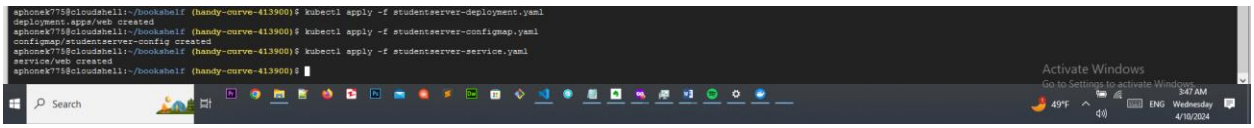
Create a file bookshelf-service.yaml



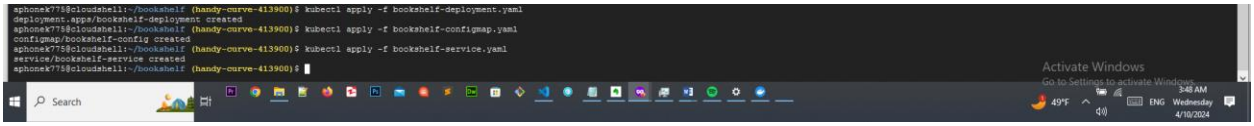
Start minikube



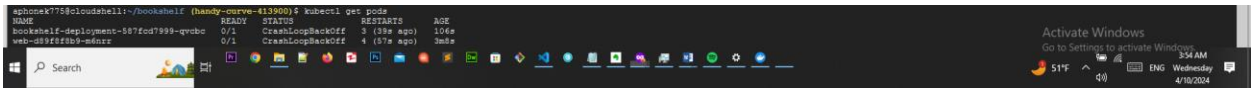
Start ingress



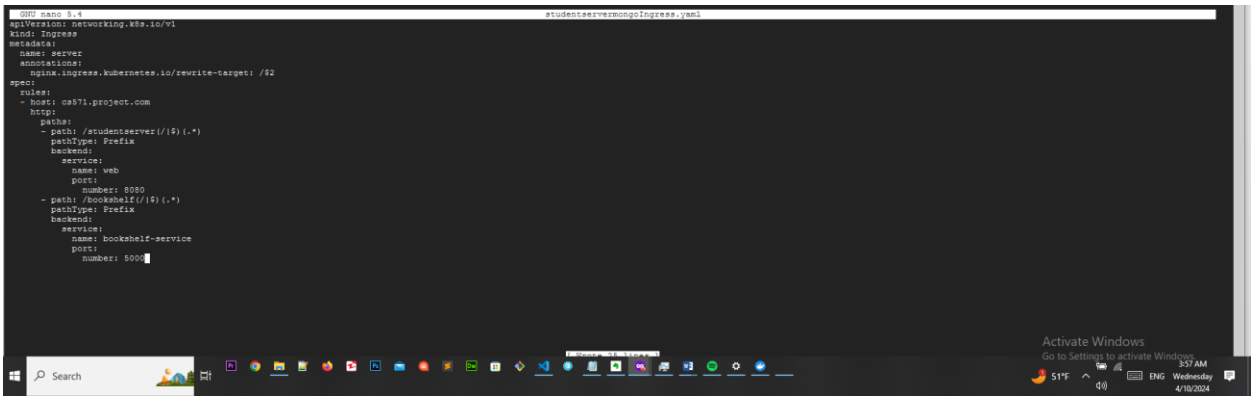
Create student server related pods and start services



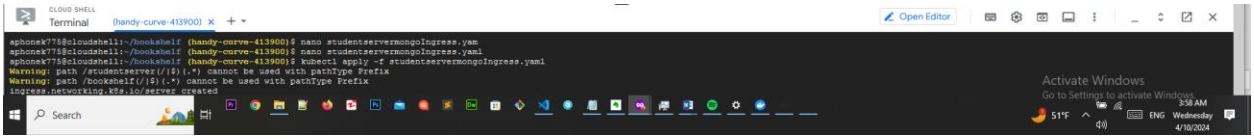
Create book shelf related pods and start services



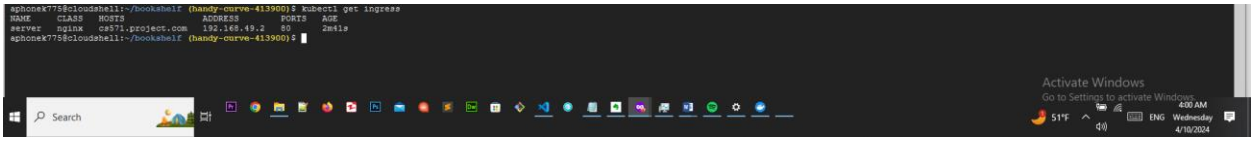
Check the pod using kubectl get pods



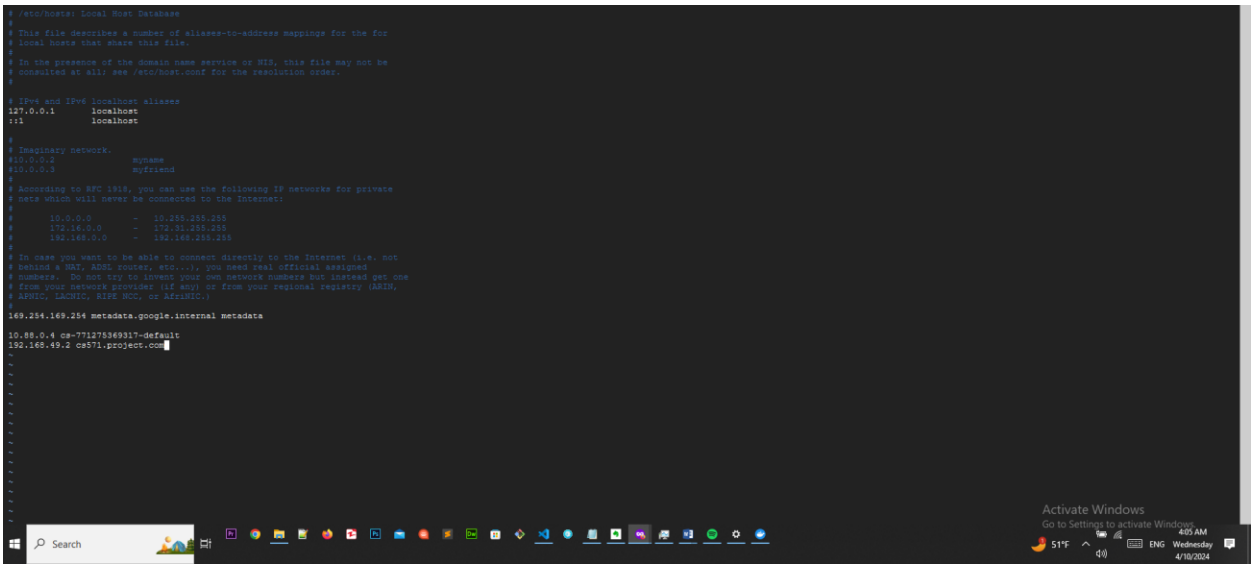
Create file studentservermongoIngress.yaml



Create ingress service using config file



Check created ingress service



Enter sudo vi /etc/hosts to add the address of ingress service

Type your_address cs571.project.com and save the file

Get student info

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

List all the books

curl cs571.project.com/bookshelf/books

Add the books

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

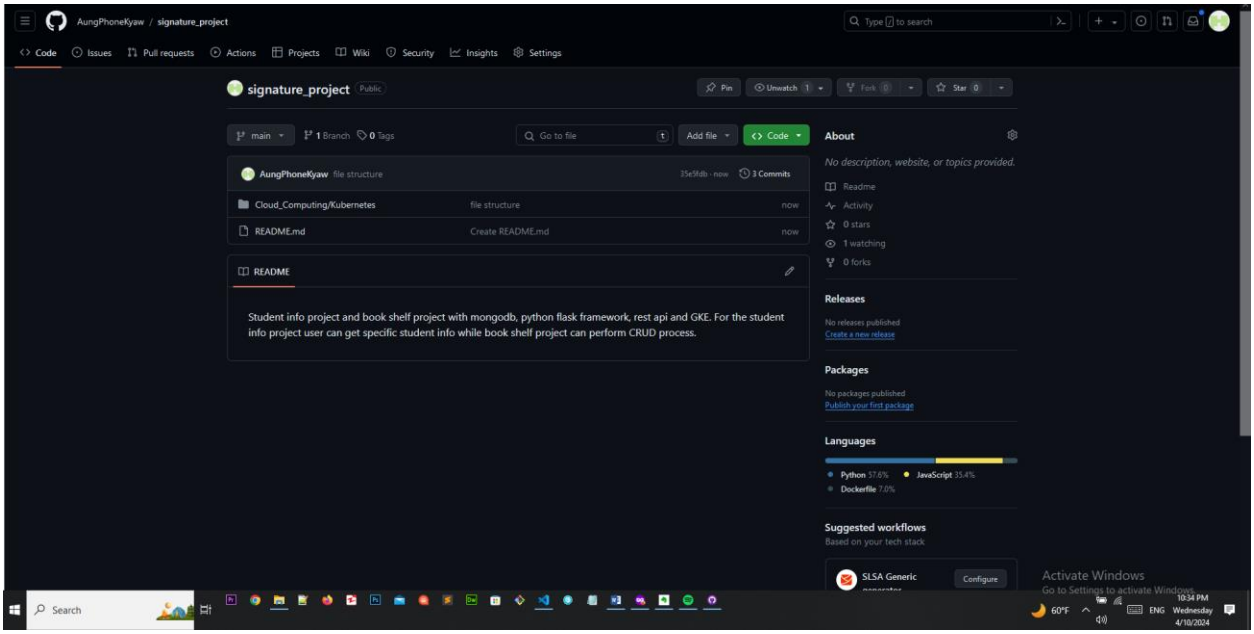
Update a book

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

Delete a book

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

Update your pofolio



Can find the project in this github repository

https://github.com/AungPhoneKyaw/signature_project