Turning on the minikube: (to run Kubernetes on my own lab)

```
minikube v1.31.2 on Ubuntu 22.04 (vbox/amd64)
Using the docker driver based on existing profile
Starting control plane node minikube in cluster minikube
Pulling base image ...
Restarting existing docker container for "minikube" ...
Preparing Kubernetes v1.27.4 on Docker 24.0.4 ...
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
```

Menu of DevOPs tool:

```
DevOps Tool
1. Build Docker Image
2. Deploy to Kubernetes
3. Monitor Kubernetes Resources
4. Start Flask App
5. Exit
Enter your choice:
```

Menu: option1 screenshot:

```
#0 building with "default" instance using docker driver

#1 [internal] load .dockerignore
#1 transferring context:
#1 transferring context: 2B 0.1s done
#1 DONE 0.2s

#2 [internal] load build definition from Dockerfile
#2 transferring dockerfile: 106B 0.1s done
#2 DONE 0.2s

#3 [internal] load metadata for docker.io/library/python:3.8-slim
#3 DONE 1.8s

#4 [1/3] FROM docker.io/library/python:3.8-slim@sha256:3324295e27596c2a4b3e53615af5ca3b280674208a24906db76812628c2ce4c8
#4 DONE 0.0s

#5 [internal] load build context
#5 transferring context: 4.88kB 0.8s done
#5 DONE 1.0s

#6 [2/3] WORKDIR /app
#6 CACHED

#7 [3/3] COPY . .
#7 DONE 1.2s

#8 exporting to image
#8 exporting layers
#8 exporting layers 0.2s done
#8 writing image sha256:91e9a463b3ed17027ee74bce4afce99cf8cda3d9354e6c140d2c10c6e50236e7 done
#8 naming to docker.io/library/mypythonapp
#8 naming to docker.io/library/mypythonapp done
#8 DONE 0.2s
```

Menu: option 2 screenshot:

```
DevOps Tool
1. Build Docker Image
2. Deploy to Kubernetes
3. Monitor Kubernetes Resources
4. Start Flask App
5. Exit
Enter your choice: 2
deployment.apps/python-app unchanged
deployment.apps/mysql configured
service/mysql-service unchanged
```

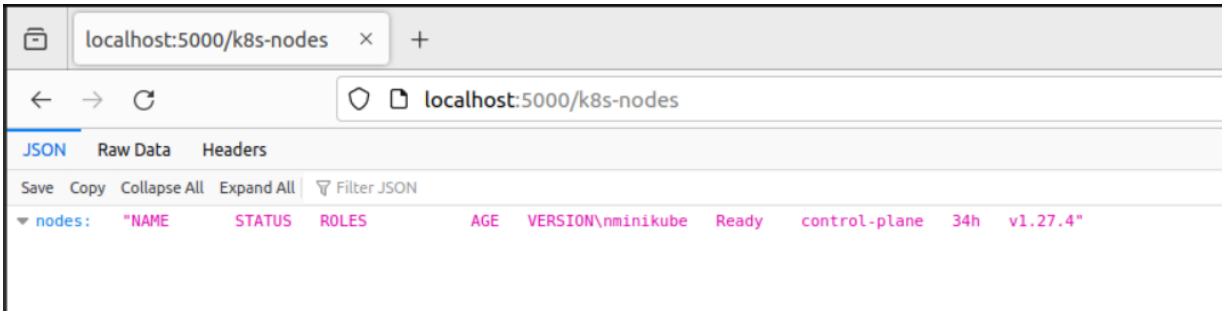
Menu: option 3 screenshot:

```
DevOps Tool
1. Build Docker Image
2. Deploy to Kubernetes
3. Monitor Kubernetes Resources
4. Start Flask App
5. Exit
Enter your choice: 3
NAME                                  READY   STATUS             RESTARTS   AGE
flaskapp-76787c997c-rdv5q             0/1     ImagePullBackOff   0          34h
flaskapp-86cf945cdc-cffdd             0/1     ImagePullBackOff   0          24h
flaskapp-86cf945cdc-mnfvv             0/1     ImagePullBackOff   0          34h
mysql-58757fbd97-95q5g                1/1     Running            0          2m46s
python-app-5cb4894949-nhdj8           0/1     ImagePullBackOff   0          19h
python-script-deployment-659f5c4b7-k9tsl   0/1     ImagePullBackOff   0          21h
```

Menu: option 4 screenshot: (note this is juts a test server for demonstration purposes)

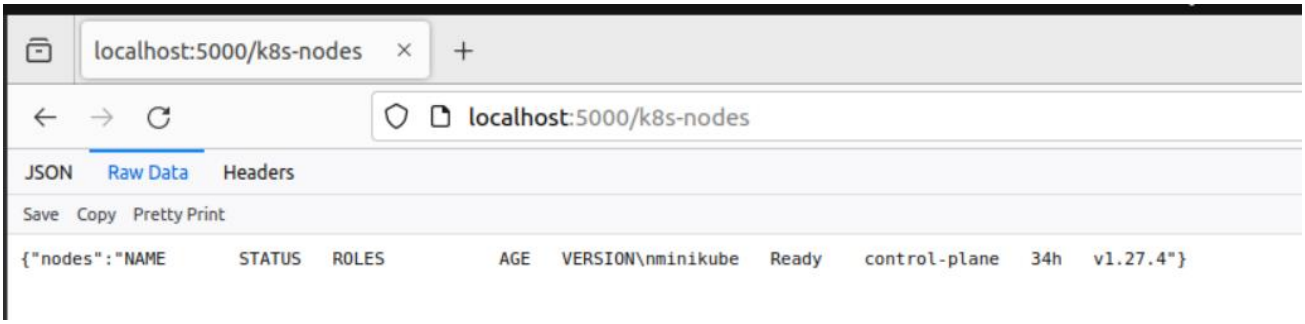
```
DevOps Tool
1. Build Docker Image
2. Deploy to Kubernetes
3. Monitor Kubernetes Resources
4. Start Flask App
5. Exit
Enter your choice: 4
 * Serving Flask app 'Menu'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://10.0.2.15:5000
Press CTRL+C to quit
```

Access of data (nodes) via http while flask is open; screenshots: 1 of 3 (Note* this is just a test server)
JSON:



Access of data (nodes) via http while flask is open; screenshots: 2 of 3 (Note* this is just a test server)

Raw Data:



Access of data (nodes) via http while flask is open; screenshots: 2 of 3 (Note* this is just a test server)
Headers: