# Orchestration with Kubernetes

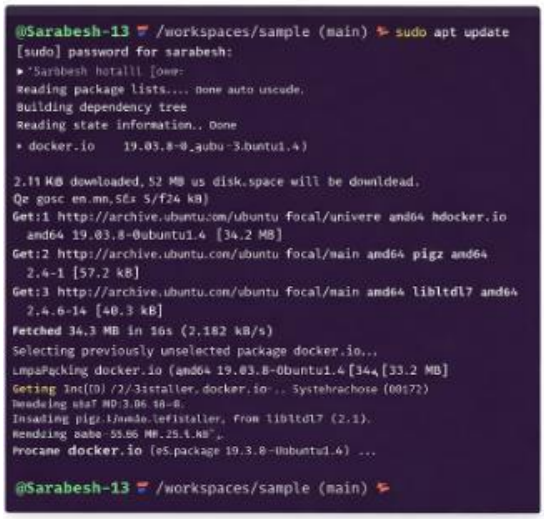
## 1. Objective

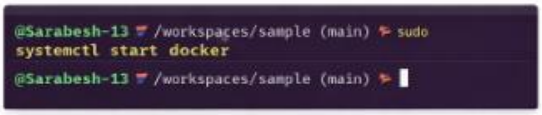
To deploy a UDP-based client–server application using Docker and Kubernetes on a local Minikube cluster and verify communication through a NodePort service.

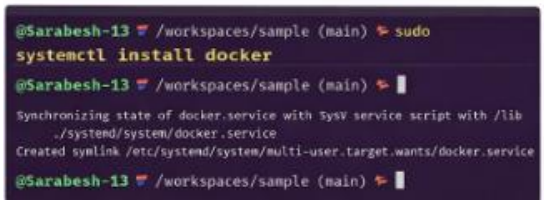
## 2. Install Docker

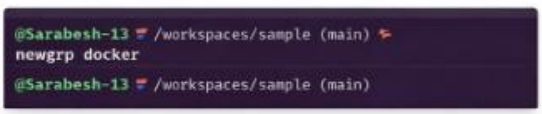
### Commands:

sudo apt update
sudo apt install -y docker.io
sudo systemctl start docker
sudo systemctl enable docker
sudo usermod -aG docker $USER
newgrp docker




## 3. Install kubectl

### Commands:

kubectl version –client

```
@Sarabesh-13 ➜/workspaces/sample (main) $ kubectl version --client
Client Version: v1.31.0
Kustomize Version: v5.4.2
```

# 4. Install Minikube

## Commands:

minikube version

```
@Sarabesh-13 ➜/workspaces/sample (main) $ minikube version
minikube version: v1.38.0
commit: de81223c61ab1bd97dcfcfa6d9d5c59e5da4a0cf
```

# 5. Start Minikube Cluster

## Command:

minikube start --driver=docker

```
@Sarabesh-13 ➜/workspaces/sample (main) $ minikube start --driver=docker
😄  minikube v1.38.0 on Ubuntu 24.04 (docker/amd64)
✨  Using the docker driver based on user configuration
❗  Starting v1.39.0, minikube will default to "containerd" container runtime. See #21973 for more info.
📌  Using Docker driver with root privileges
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.49 ...
💾  Downloading Kubernetes v1.35.0 preload ...
    > gcr.io/k8s-minikube/kicbase...:  514.16 MiB / 514.16 MiB  100.00% 72.56 M
    > preloaded-images-k8s-v18-v1...:  271.45 MiB / 271.45 MiB  100.00% 14.04 M
🔥  Creating docker container (CPUs=2, Memory=3072MB) ...
🐳  Preparing Kubernetes v1.35.0 on Docker 29.2.0 ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: storage-provisioner, default-storageclass

❗  /usr/local/bin/kubectl is version 1.31.0, which may have incompatibilities with Kubernetes 1.35.0.
    ▪ Want kubectl v1.35.0? Try 'minikube kubectl -- get pods -A'
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

# 6. Verify Kubernetes Cluster

## Command:

kubectl get nodes

```
@Sarabesh-13 ➜/workspaces/sample (main) $ kubectl get nodes
NAME       STATUS   ROLES           AGE    VERSION
minikube   Ready    control-plane   9m1s   v1.35.0
```

# 7. Configure Docker to Use Minikube Environment

## Command:

eval $(minikube docker-env)

```
@Sarabesh-13 ➜/workspaces/sample (main) $ eval $(minikube docker-env)
```

# 8. Build UDP Server Docker Image

## Command:

docker build -t udp-server:1.0 .

```
@Sarabesh-13 ➜/workspaces/sample (main) $ docker build -t udp-server:1.0 .
[+] Building 3.3s (11/11) FINISHED                                                                  docker:default
 => [internal] load build definition from Dockerfile                                                         0.0s
 => => transferring dockerfile: 277B                                                                         0.0s
 => [internal] load metadata for docker.io/library/ubuntu:22.04                                              1.7s
 => [auth] library/ubuntu:pull token for registry-1.docker.io                                                0.0s
 => [internal] load .dockerignore                                                                            0.0s
 => => transferring context: 2B                                                                              0.0s
 => [1/5] FROM docker.io/library/ubuntu:22.04@sha256:c7eb020043d8fc2ae0793fb35a37bff1cf33f156d4d4b1          0.0s
 => [internal] load build context                                                                            0.0s
 => => transferring context: 2.08kB                                                                          0.0s
 => CACHED [2/5] RUN apt update && apt install -y g++     && rm -rf /var/lib/apt/lists/*                     0.0s
 => CACHED [3/5] WORKDIR /app                                                                                0.0s
 => [4/5] COPY udpserver.cpp .                                                                               0.0s
 => [5/5] RUN g++ udpserver.cpp -o udp_server                                                                0.9s
 => exporting to image                                                                                       0.5s
 => => exporting layers                                                                                      0.5s
 => => writing image sha256:7e5066699756905e80b0826201e3fc647d6f33d8866a2adf0564cd8b05164efb                 0.0s
 => => naming to docker.io/library/udp-server:1.0                                                            0.0s
```

# 9. Deploy UDP Server Using Kubernetes

## Commands:

kubectl apply -f deployment.yaml
kubectl apply -f service.yaml

```
@Sarabesh-13 ➜/workspaces/sample (main) $ kubectl apply -f deployment.yaml
deployment.apps/udp-server created
```

```
@Sarabesh-13 ➜/workspaces/sample (main) $ kubectl apply -f service.yaml
service/udp-server-service created
```

# 10. Verify Pod and Service Status

## Commands:

kubectl get pods
kubectl get svc

```
@Sarabesh-13 ➜/workspaces/sample (main) $ kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
udp-server-7cc7bbf8fb-pvk88   1/1     Running   0          23m
```

```
@Sarabesh-13 ➜/workspaces/sample (main) $ kubectl get svc
NAME                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes           ClusterIP   10.96.0.1       <none>        443/TCP          100m
udp-server-service   NodePort    10.103.41.73    <none>        4003:30003/UDP   76m
```

# 11. Verify Service Endpoints

## Command:

kubectl get endpoints udp-server-service

```
@Sarabesh-13 ➜/workspaces/sample (main) $ kubectl get endpoints udp-server-service
Warning: v1 Endpoints is deprecated in v1.33+; use discovery.k8s.io/v1 EndpointSlice
NAME                 ENDPOINTS          AGE
udp-server-service   10.244.0.5:4003    77m
```

# 12. Check Server Logs

## Command:

kubectl logs <udp-server-pod-name>

```
@Sarabesh-13 ➜/workspaces/sample (main) $ kubectl logs -f udp-server-7cc7bbf8fb-pvk88
UDP Server running on port 4003
Client: hi
Client: welcome all
Client: hey
^C
```

# 13. Test UDP Server Using Client

## Commands:

minikube ip
./udp_client <minikube-ip> <nodeport>

```
@Sarabesh-13 ➜/workspaces/sample (main) $ minikube ip
192.168.49.2
```

```
@Sarabesh-13 ➜/workspaces/sample (main) $ ./udp_client $(minikube ip) 30003
Client: hi
Server: ACK
Client: welcome all
Server: ACK
Client: ^C
```

# 14. Optional Debugging (If `ss` Command Does Not Work)

If the `ss` command is not available inside the container:

### Commands:

```
kubectl exec -it <pod-name> -- apt update
kubectl exec -it <pod-name> -- apt install -y iproute2 net-tools
```

Then retry:

```
kubectl exec -it <pod-name> -- ss -anu
```

# 15. Result

The UDP server was successfully deployed on a Minikube Kubernetes cluster and verified using a UDP client via NodePort service.