

INGRESS COMMANDS LAB:

- Step 1: helm create fullstack-app [This command for one time only]

This will generate a folder named fullstack-app/ with the following structure:

```
fullstack-app/  
|--- charts/  
|--- templates/  
|--- Chart.yaml  
|--- values.yaml
```

You can then modify values.yaml and templates as per your fullstack app (frontend, backend, MySQL, etc.).

- Step 2: Add and Update Helm Repositories [These commands for first time only]

```
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
```

```
helm repo update
```

```
helm repo list
```

■ Step 3: Install NGINX Ingress Controller

- First time (creates namespace)

```
helm install ingress-nginx ingress-nginx/ingress-nginx --create-namespace --  
namespace ingress-nginx
```

- ▣ Next time (upgrade without creating namespace)

```
helm upgrade ingress-nginx ingress-nginx/ingress-nginx --namespace ingress-nginx
```

- 🔍 Check Ingress Controller Pods

```
kubectl get pods -n ingress-nginx
```

- 🌲 View Ingress Logs

```
kubectl logs -f <ingress-pod-name> -n ingress-nginx
```

```
-----
```

■ Step 4: Install or Upgrade Your Fullstack App

- First time (creates namespace)

```
helm install fullstack-app ./fullstack-app --create-namespace --namespace fullstack-app
```

 Next time (upgrade without recreating namespace)

```
helm upgrade fullstack-app ./fullstack-app --namespace fullstack-app
```

 Check Application Pods

```
kubectl get pods -n fullstack-app
```

 View Application Logs

```
kubectl logs -f <backend-pod-name> -n fullstack-app [this one important]
```

```
kubectl logs -f <frontend-pod-name> -n fullstack-app
```

```
kubectl logs -f <mysql-pod-name> -n fullstack-app
```

 Step 5: List Helm Releases

```
helm list -n fullstack-app
```

■ Step 6: List the services (svc) in the namespace (fullstack-app)

```
kubectl get svc -n fullstack-app
```

■ Step 7: Check Release Status

```
helm status fullstack-app -n fullstack-app
```

■ Step 8: View Manifest

```
helm get manifest fullstack-app -n fullstack-app
```

■ Step 9: Check Release History

```
helm history fullstack-app -n fullstack-app
```

Step 10: Horizontal Pod Autoscaler (HPA)

Check All HPAs

```
kubectl get hpa -n fullstack-app
```

Describe a Specific HPA

```
kubectl describe hpa backend -n fullstack-app
```

```
kubectl describe hpa frontend -n fullstack-app
```

Watch Scaling in Real Time

```
kubectl get hpa -n fullstack-app -w
```

Check Current Pods and Resource Usage

```
kubectl get pods -n fullstack-app
```

Manually Scale (Optional)

```
kubectl scale deployment backend-deployment --replicas=5 -n fullstack-app
```

```
kubectl scale deployment frontend-deployment --replicas=5 -n fullstack-app
```

Monitor Logs During Scaling

```
kubectl logs -f <backend-pod-name> -n fullstack-app
```

■ Step 11: Monitor Pods and Scaling Activity

```
kubectl get pods -n fullstack-app -w
```

■ Step 12: Uninstall the Fullstack App

```
helm uninstall fullstack-app -n fullstack-app
```

■ Step 13: Uninstall NGINX Ingress Controller

```
helm uninstall ingress-nginx -n ingress-nginx
```

Step 14: Delete Namespaces (Cleanup)

```
kubectl delete namespace fullstack-app
```

```
kubectl delete namespace ingress-nginx
```

LAB 15: Ansible

```
wsl --install
```

```
wsl --update
```

```
wsl --list --online
```

```
wsl -d Ubuntu
```

```
sudo apt update && sudo apt upgrade -y
```

```
sudo apt install -y python3 python3-pip python3-venv
```

```
python3 --version
```

```
pip3 --version
```

```
sudo apt install -y software-properties-common git curl apt-transport-https ca-certificates gnupg lsb-release
```

4 Install Ansible

```
sudo add-apt-repository --yes --update ppa:ansible/ansible  
sudo apt install -y ansible  
ansible --version
```

5 Install kubectl (official stable v1.30 repo)

```
sudo mkdir -p /etc/apt/keyrings  
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | \  
sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg  
  
echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] \  
https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /" | \  
sudo tee /etc/apt/sources.list.d/kubernetes.list  
  
sudo apt update  
sudo apt install -y kubectl  
kubectl version --client
```

6 If you already have K3s (Kubernetes lightweight cluster)

```
# Install K3s (lightweight Kubernetes)  
curl -sfL https://get.k3s.io | sh -  
  
# Check K3s service status  
sudo systemctl status k3s
```

```
# Wait until it shows "active (running)"

# Verify if K3s config file exists now
ls /etc/rancher/k3s/k3s.yaml

# Set permissions and copy config to your user
sudo chmod 644 /etc/rancher/k3s/k3s.yaml
mkdir -p ~/.kube
sudo cp /etc/rancher/k3s/k3s.yaml ~/.kube/config
sudo chown $(id -u):$(id -g) ~/.kube/config

# Verify K3s node is registered
kubectl get nodes

mkdir ansible-project
cd ansible-project

ansible-playbook -i inventory.ini playbook-deploy.yml

kubectl get all -n fullstack-app

kubectl get pods -n fullstack-app

kubectl logs -l app=backend -n fullstack-app

ps aux | grep kubectl
```

```
pgrep -a kubectl
```

```
sudo kill -9 port
```

```
kubectl delete namespace fullstack-app
```

LAB 12 kubernetes:

1. Delete the old namespace

```
kubectl delete namespace fullstack-app
```

2. Build and push new Docker images (If code is modified)

```
cd reactfrontend
```

```
docker build -f frontend.Dockerfile -t username/k8-frontend:latest .
```

```
docker push username/k8-frontend:latest
```

```
cd springbootbackend
```

```
docker build -f backend.Dockerfile -t username/k8-backend:latest .
```

```
docker push username/k8-backend:latest
```

3. Reapply deployment configuration

```
kubectl apply -f ./k8/fullstack-deployment.yml
```

4. Check that everything is up and running

```
kubectl get all -n fullstack-app
```