
🧪 Lab 3.1 – Kubernetes Cluster Bootstrapping with `kubeadm`

> 🧑‍💻 *Each trainee runs this lab on their own **Ubuntu 24.04** VMs (2 per trainee: master and worker).*

🖥️ Lab Setup Options

This lab can be run in two environments:

1. **Instructor-provided VMs in the cloud** – Pre-provisioned Ubuntu 24.04 machines for each trainee.
2. **Local Vagrant environment** – For trainees on physical Ubuntu machines, use Vagrant to spin up 2 interconnected VMs.

Vagrant-based setup instructions will be provided in a separate file.

🎯 Objectives

- Bootstrap a multi-node Kubernetes cluster using `kubeadm`
- Use **containerd** as the container runtime
- Install and configure **Calico** as the CNI plugin
- Validate the cluster and deploy a sample workload

🛠️ Prerequisites

Ensure the following commands are available on both VMs:

```
alias k='kubectl'
source <(kubectl completion bash)
complete -F __start_kubectl k
source ~/.bashrc
```

⚙️ Step 1: Bootstrap the Control Plane (Master Node)

```
# Get the private IP
PRIVATE_IP=$(ip -4 addr show ens4 | grep -oP '(?<=inet\s)\d+(\.\d+){3}' | head -n 1)

# Initialize kubeadm
sudo kubeadm init \
  --apiserver-advertise-address=$PRIVATE_IP \
  --pod-network-cidr=192.168.0.0/16
```

Configure `kubectl` for your user:

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

🌐 Step 2: Install Calico CNI

```
kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.27.0/manifests/calico.yaml
```

🧩 Step 3: Join the Worker Node

On the worker node, use the command provided by `kubeadm init` (with `kubeadm join`). If you lost it, regenerate it from the master:

```
kubeadm token create --print-join-command
```

Then run the output on the worker node.

🔍 Step 4: Verify Your Cluster

```
kubectl get nodes -o wide
kubectl get pods -A
```

You should see Calico pods running in `kube-system` and nodes in Ready state.

🧪 Step 5: Deploy a Test Application

```
kubectl create deployment nginx --image=nginx
kubectl expose deployment nginx --port=80 --type=NodePort
kubectl get svc nginx
```

Try accessing it via the **worker VM's private IP** and the given NodePort.

🧑‍💻 Challenge: Upgrade the cluster

For this challenge, you need to upgrade your cluster to Kubernetes v1.33.2. Note that this is an advanced task and may require additional steps such as unholding packages, updating the APT repository, and applying the upgrade plan.

At the end of this challenge, your cluster should be running Kubernetes v1.33.2 for both the master and worker nodes.

Expected the following output when you check the nodes:

```
$ kubectl get nodes
NAME                                                     STAT
US    ROLES           AGE   VERSION
trainee-demo-master.europe-west1-b.c.ekp-dev.internal   Read
y     control-plane   14m   v1.33.2
trainee-demo-worker.europe-west1-b.c.ekp-dev.internal   Read
y     <none>          12m   v1.33.2
```

> *Nice to have: If you have want you create a backup of your cluster database before the upgrade, you can use tools like `etcdctl`.*

✅ Checklist

- ☐ Cluster bootstrapped with kubeadm
- ☐ Calico CNI deployed and nodes Ready
- ☐ `kubectl` working as non-root user
- ☐ Sample application running
- ☐ Access via NodePort verified
