k8s complete setup -master & worker node

properly, from scratch, clean DevOps-style 💪
I'll assume you want a **basic Kubernetes cluster using kubead**
m on **Linux (Ubuntu 20.04/22.04)** with:

- 🧠 **Master node** → `10.0.2.14`
- ⚙️ **Worker node** → `10.0.2.190`
- Container runtime: **containerd**
- Network plugin: **Calico** (industry-standard)

I'll give you **end-to-end steps**, copy-paste friendly, with **why** each step exists.

🧱 Architecture Overview

```
Master Node (Control Plane)
IP: 10.0.2.14
 ├── kube-apiserver
 ├── etcd
 ├── controller-manager
 └── scheduler

Worker Node
IP: 10.0.2.190
 └── kubelet + kube-proxy
```

🔹 STEP 0: Prerequisites (BOTH NODES)

✅ OS & User

- Ubuntu 20.04 / 22.04
- Login as **root** or a sudo user

✅ Set hostnames

Master

```
hostnamectl set-hostname k8s-master
```

Worker

```
hostnamectl set-hostname k8s-worker
```

Logout & login again.

✅ Update system

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

✅ Disable swap (VERY IMPORTANT)

```
sudo swapoff -a
sudo sed -i '/ swap / s/^/#/' /etc/fstab
```

Verify:

```
free -h
```

✅ Enable kernel modules

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF

sudo modprobe overlay
sudo modprobe br_netfilter
```

✅ Sysctl settings (networking)

```
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
EOF

sudo sysctl --system
```

🔹 STEP 1: Install Container Runtime (containerd) – BOTH NODES

```
sudo apt install -y containerd
```

Configure containerd

```
sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml
```

Edit config:

```
sudo nano /etc/containerd/config.toml
```

Find:

```
SystemdCgroup = false
```

Change to:

```
SystemdCgroup = true
```

Restart:

```
sudo systemctl restart containerd
sudo systemctl enable containerd
```

Check:

```
containerd --version
```

🔹 STEP 2: Install Kubernetes Components – BOTH NODES

Add Kubernetes repo

```
sudo apt install -y apt-transport-https ca-certificates curl
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/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:/v1.29/deb/ /" | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

Update & install:

```
sudo apt update
sudo apt install -y kubelet kubeadm kubectl
```

Hold versions:

```
sudo apt-mark hold kubelet kubeadm kubectl
```

🔹 STEP 3: Initialize MASTER Node (10.0.2.14)

⚠️ **Run ONLY on master**

```
sudo kubeadm init \
  --apiserver-advertise-address=10.0.2.14 \
  --pod-network-cidr=192.168.0.0/16
```

⏳ Wait 2–3 minutes.

✅ Configure kubectl (MASTER)

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

Test:

```
kubectl get nodes
```

You'll see:

```
k8s-master   NotReady
```

👉 This is normal (network plugin not installed yet)

🔹 STEP 4: Install Network Plugin (Calico) – MASTER

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

Wait:

```
kubectl get pods -n kube-system
```

After 1–2 mins:

```
calico-node   Running
```

Now:

```
kubectl get nodes
```

Master should be:

```
Ready
```

🔹 STEP 5: Join WORKER Node (10.0.2.190)

During `kubeadm init`, you got a join command like this 👇
Run it **on the worker**:

```
sudo kubeadm join 10.0.2.14:6443 \
  --token <token> \
  --discovery-token-ca-cert-hash sha256:<hash>
```

If token expired, regenerate on master:

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

🔹 STEP 6: Verify Cluster (MASTER)

```
kubectl get nodes -o wide
```

Expected:

```
k8s-master   Ready   control-plane   10.0.2.14
k8s-worker   Ready   <none>           10.0.2.190
```

🎉 **Cluster is UP**

🔹 STEP 7: (Optional) Allow Pods on Master (Lab only)

```
kubectl taint nodes k8s-master node-role.kubernetes.io/control-plane-
```

⚠️ Do NOT do this in production.

🔹 STEP 8: Test with Sample App

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

🧠 Common Troubleshooting

Issue	Fix
Node NotReady	Check Calico
kubeadm init fails	Swap not disabled
Pods Pending	Network plugin missing

Issue	Fix
Worker not joining	Firewall / token expired

🏁 Resume-Ready Line

> Built a multi-node Kubernetes cluster using kubeadm with containerd runtime, Calico networking, and manual control-plane & worker node configuration.

If you want next:

- 🔁 HA master setup
- 🔐 TLS / RBAC deep dive
- 🐬 Deploy Dockerized MERN app
- ⚙️ Jenkins → Kubernetes deployment
- 📄 Diagram + YAML examples

Just say **next** 🚀