Complete Kubernetes Master and Worker setup using kubeadm

**Kubernetes-master-worker-setup-using-kubeadm**

complete master-worker set using kubeadm using script file Step 0: Prerequisites OS: Ubuntu 22.04 on all nodes Users: Root access (or sudo) Network: All nodes can reach each other via private IP Hardware: **Master: 2 CPUs, 2GB RAM minimum Worker: 1 CPU, 2GB RAM minimum Worker: 1 CPU, 2GB RAM minimum**

Disable swap on all nodes (your script already does this) Open necessary ports (optional if using ufw): Master node: 6443 (API server), 2379-2380 (etcd), 10250 (kubelet), 10251 (scheduler), 10252 (controller manager), 30000-32767 (NodePort services) Workers: 10250 (kubelet), 30000-32767 (NodePort)

**vi k8s-cluster-setup.sh**

#!/bin/bash

# ==========================================================

# Automated Kubernetes Cluster Setup using kubeadm

# Works for both Master and Worker nodes

# OS: Ubuntu 22.04 (tested)

# Usage:

# sudo ./k8s-cluster-setup.sh --master <MASTER\_PRIVATE\_IP>

# sudo ./k8s-cluster-setup.sh --worker "<JOIN\_COMMAND>"

# ==========================================================

set -euo pipefail

IFS=$'\n\t'

# ---------- Helpers ----------

log() { echo -e "\e[1;32m$1\e[0m"; }

err() { echo -e "\e[1;31mERROR: $1\e[0m" >&2; }

fatal() { err "$1"; exit 1; }

if [ "$EUID" -ne 0 ]; then

fatal "This script must be run as root (use sudo)."

fi

# ---------- Arguments ----------

MODE="${1:-}"

ARG="$2" || true

if [ -z "$MODE" ]; then

fatal "Usage:

Master: sudo ./k8s-cluster-setup.sh --master <MASTER\_PRIVATE\_IP>

Worker: sudo ./k8s-cluster-setup.sh --worker \"<kubeadm join ...>\""

fi

# ---------- Common setup ----------

log "[1/9] Updating system and installing base packages..."

export DEBIAN\_FRONTEND=noninteractive

apt-get update -y

apt-get upgrade -y

apt-get install -y apt-transport-https ca-certificates curl gnupg lsb-release software-properties-common

log "[2/9] Disabling swap (required by kubelet)..."

swapoff -a

sed -i.bak '/\sswap\s/s/^/#/' /etc/fstab || true

log "[3/9] Loading kernel modules and sysctl settings..."

cat >/etc/modules-load.d/k8s.conf <<'EOF'

overlay

br\_netfilter

EOF

modprobe overlay || true

modprobe br\_netfilter || true

cat >/etc/sysctl.d/99-k8s.conf <<'EOF'

net.bridge.bridge-nf-call-iptables = 1

net.bridge.bridge-nf-call-ip6tables = 1

net.ipv4.ip\_forward = 1

EOF

sysctl --system

# ---------- Install containerd ----------

log "[4/9] Installing and configuring containerd..."

apt-get update -y

# Install containerd package (Ubuntu repo). If you prefer containerd.io from Docker repo, change here.

apt-get install -y containerd

mkdir -p /etc/containerd

containerd config default > /etc/containerd/config.toml || true

# Ensure systemd cgroup is enabled

sed -i 's/SystemdCgroup = false/SystemdCgroup = true/' /etc/containerd/config.toml || true

systemctl daemon-reload

systemctl restart containerd

systemctl enable containerd

# ---------- Install kubeadm, kubelet, kubectl ----------

log "[5/9] Installing kubeadm, kubelet and kubectl..."

mkdir -p /etc/apt/keyrings

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | 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/ /" > /etc/apt/sources.list.d/kubernetes.list

apt-get update -y

apt-get install -y kubelet kubeadm kubectl

apt-mark hold kubelet kubeadm kubectl

# ---------- Disable ufw (optional) ----------

# If UFW is active and not configured, kubeadm networking may be blocked.

if command -v ufw >/dev/null 2>&1; then

if ufw status | grep -q active; then

log "[6/9] Disabling UFW (if you want to keep it enabled, open required ports instead)..."

ufw disable || true

fi

fi

log "[7/9] Base packages installed."

# ---------- Role specific ----------

if [[ "$MODE" == "--master" ]]; then

MASTER\_IP="$ARG"

if [ -z "$MASTER\_IP" ]; then

fatal "Provide the master private IP: sudo ./k8s-cluster-setup.sh --master <MASTER\_PRIVATE\_IP>"

fi

log "🚀 Initializing Kubernetes control-plane (master)..."

# Use pod network cidr compatible with Calico (change if you use different CNI)

kubeadm init --apiserver-advertise-address="$MASTER\_IP" --pod-network-cidr=192.168.0.0/16

log "Configuring kubectl for the ubuntu user..."

# If a non-root user needs kubectl, copy to that user's home. Here we place config in /root/.kube

mkdir -p /root/.kube

cp -i /etc/kubernetes/admin.conf /root/.kube/config

chown root:root /root/.kube/config

export KUBECONFIG=/etc/kubernetes/admin.conf

log "Waiting for kube-apiserver to be ready..."

# Wait until kubectl can access the API

retries=0

until kubectl get componentstatuses >/dev/null 2>&1 || [ $retries -ge 30 ]; do

retries=$((retries+1))

log "Waiting for API server... ($retries/30)"

sleep 5

done

if [ $retries -ge 30 ]; then

err "kubectl couldn't reach API server yet. Check 'kubectl get pods -A' later."

fi

log "Applying Calico CNI manifest..."

# Apply Calico; ensure network is applied after API is available

kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml

log "Generating kubeadm join command (valid 24 hours by default)..."

JOIN\_CMD=$(kubeadm token create --print-join-command)

echo

log "=== COPY THIS JOIN COMMAND FOR WORKERS ==="

echo "$JOIN\_CMD"

echo "========================================="

echo

log "Master setup completed. Use the printed 'kubeadm join ...' command on worker nodes."

elif [[ "$MODE" == "--worker" ]]; then

# Note: pass the whole join command as a single quoted argument

JOIN\_CMD="$ARG"

if [ -z "$JOIN\_CMD" ]; then

fatal "Provide the join command from master: sudo ./k8s-cluster-setup.sh --worker \"kubeadm join ...\""

fi

log "Joining this node to the cluster..."

# run the join command (it may contain backslashes/newlines; evaluate it safely)

eval "$JOIN\_CMD"

log "Worker node joined. Verify on master with 'kubectl get nodes'."

else

fatal "Invalid mode. Use:

Master: sudo ./k8s-cluster-setup.sh --master <MASTER\_PRIVATE\_IP>

Worker: sudo ./k8s-cluster-setup.sh --worker \"<kubeadm join ...>\""

fi

log "All done."

**Step 1: Run your script on the master node** 1.1 Make the script executable: **chmod +x k8s-cluster-setup.sh** 1.2 Run the script with the master IP **sudo ./k8s-cluster-setup.sh --master <MASTER\_PRIVATE\_IP>**

The script will: Install containerd Install kubeadm, kubelet, kubectl Disable swap Load kernel modules Initialize master node Apply Calico CNI Print the kubeadm join ... command

**Step 2: Prepare worker nodes** Copy the k8s-cluster-setup.sh script to each worker nodes both workernode-1 and workernode-2. Run the script with the join command from master:

**Run the below command in Workernode-1** sudo ./k8s-cluster-setup.sh --worker "kubeadm join <MASTER\_IP>:6443 --token --discovery-token-ca-cert-hash sha256:"

example: **cd /home/ubuntu chmod +x k8s-cluster-setup.sh** **sudo ./k8s-cluster-setup.sh --worker "kubeadm join 172.31.38.6:6443 --token 4egq92.mkxl57dxzpgv02xs --discovery-token-ca-cert-hash sha256:ace74c46cfea3df7be4d8e6312d1eb4b3bb21656cbec0df1fc5abbeb615d1895"**

**Run the below command in Workernode-2**

**cd /home/ubuntu chmod +x k8s-cluster-setup.sh sudo ./k8s-cluster-setup.sh --worker "kubeadm join 172.31.38.6:6443 --token 4egq92.mkxl57dxzpgv02xs --discovery-token-ca-cert-hash sha256:ace74c46cfea3df7be4d8e6312d1eb4b3bb21656cbec0df1fc5abbeb615d1895"**

The script will: Install containerd, kubeadm, kubelet Disable swap Join the node to the cluster

**Step 3: Verify cluster status** On the master node: Check node status: **kubectl get nodes**

**You should secc the output**

| **NAME** | **STATUS** | **ROLES** | **AGE** | **VERSION** |
| --- | --- | --- | --- | --- |
| master | Ready | master | 5m | v1.30 |
| worker1 | Ready |  | 2m | v1.30 |
| worker2 | Ready |  | 1m | v1.30 |