INSTALLATION AND USING KUBEADM

AWS Setup

1. Ensure that all instances are in the same Security Group.
2. Expose port 6443 in the Security Group to allow worker nodes to join the cluster.
3. Expose port 22 in the Security Group to allows SSH access to manage the instance.

Execute on both Master as well as worker nodes

Disable swap required for kubernetes

sudo swapoff -a

Load necessary kernel modules required for networking

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

overlay

br_netfilter

EOF

sudo modprobe overlay

sudo modprobe br_netfilter

Set Sysctl Parameters: Helps with 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

lsmod | grep br_netfilter

lsmod | grep overlay

Install Containerd:

sudo apt-get update

sudo apt-get install -y ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo \"$VERSION_CODENAME\") stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install -y containerd.io

containerd config default | sed -e 's/SystemdCgroup = false/SystemdCgroup = true/' -e 's/sandbox_image = "registry.k8s.io\/pause:3.6"/sandbox_image = "registry.k8s.io\/pause:3.9"/' | sudo tee /etc/containerd/config.toml

sudo systemctl restart containerd

sudo systemctl status containerd

Install Kubernetes components:

sudo apt-get update

sudo apt-get install -y apt-transport-https ca-certificates curl gpg

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

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

EXECUTE FOR ONLY SINGLE INSTANCE

sudo kubeadm init --pod-network-cidr=192.168.0.0/16

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

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

kubectl taint nodes --all node-role.kubernetes.io/control-plane-

kubectl get nodes

kubectl get pods -A

kubectl apply -f .

EXECUTE ONLY ON MASTER NODE

Initialize in master node:

sudo kubeadm init

Initialize the Cluster:

sudo kubeadm init

Set Up Local kubeconfig:

mkdir -p "$HOME"/.kube

sudo cp -i /etc/kubernetes/admin.conf "$HOME"/.kube/config

sudo chown "$(id -u)":"$(id -g)" "$HOME"/.kube/config

Install a Network Plugin (Calico):

kubectl apply -f
https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/manifests/calico.yaml

Generate Join Command:

kubeadm token create --print-join-command

Copy this generated token for next command.

EXECUTE ON ALL WORKER NODES

Perform pre-flight checks:

sudo kubeadm reset pre-flight checks

Paste the join command you got from the master node and append --v=5 at the end:

sudo kubeadm join <private-ip-of-control-plane>:6443 --token <token> --discovery-token-ca-cert-hash sha256:<hash> --cri-socket "unix:///run/containerd/containerd.sock" --v=5

On master node check if the worker node is updated

kubectl get nodes

For running the example you can check the below command on master node:

kubectl run nginx –image=nginx:latest

OR

Instance: Ubuntu

T3 medium

20gb storage

Number of instances 2

Paste the below commands in the instance creation

```
#!/bin/bash

#Master

# Update and upgrade Ubuntu packages
echo "Updating and upgrading Ubuntu packages..."
sudo apt-get update -y
sudo apt-get upgrade -y

# Disable swap
```

```
echo "Disabling swap..."
sudo swapoff -a
sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab

# Add Kernel Parameters
echo "Adding kernel parameters..."
sudo tee /etc/modules-load.d/containerd.conf <<EOF
overlay
br_netfilter
EOF

sudo modprobe overlay
sudo modprobe br_netfilter

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

sudo sysctl --system

# Install Containerd Runtime
echo "Installing Containerd runtime..."
sudo apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates

echo "Adding Docker's GPG key and repository..."
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/docker.gpg
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

```
sudo apt update
sudo apt install -y containerd.io

echo "Configuring Containerd..."
containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1
sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml

echo "Restarting and enabling Containerd..."
sudo systemctl restart containerd
sudo systemctl enable containerd

# Install Docker
echo "Installing Docker..."
sudo apt update
# sudo apt install -y docker-ce docker-ce-cli
sudo apt install -y apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"
sudo apt update
sudo apt install -y docker-ce docker-ce-cli
sudo usermod -aG docker $USER
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-
$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod 777 /var/run/docker.sock


echo "Starting and enabling Docker..."
sudo systemctl start docker
sudo systemctl enable docker

# Add the Kubernetes signing key and repository
```

```
echo "Adding Kubernetes signing key and repository..."

sudo apt-get update -y

sudo apt-get install -y apt-transport-https ca-certificates curl gpg


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 the package list and install kubelet, kubeadm, and kubectl

echo "Updating package list and installing kubelet, kubeadm, and kubectl..."

sudo apt-get update -y

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl


echo "Kubernetes installation script completed successfully!"
```

on master node terminal

```
sudo hostnamectl hostname Master

bash


sudo kubeadm init --pod-network-cidr=192.168.0.0/16


 mkdir -p $HOME/.kube

 sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

 sudo chown $(id -u):$(id -g) $HOME/.kube/config


kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.1/manifests/calico.yaml


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

on worker node

sudo (generated token paste)

now kubeadm installation has been done

kubectl get nodes - fire this command on master node

Both the master node and worker nodes will be appeared