**Multi-Node Kubernetes Cluster Setup Using Kubeadm**

This readme provides step-by-step instructions for setting up a multi-node Kubernetes cluster using Kubeadm.

Virutal box Guest add on installation

*sudo apt update*

*sudo apt install module-assistant gcc make perl bzip2 tar git linux-headers-generic build-essential dkms*

*sudo apt install linux-headers-$(uname -r)*

*cd ~/Documents/*

**Overview**

This guide provides detailed instructions for setting up a multi-node Kubernetes cluster using Kubeadm. The guide includes instructions for installing and configuring containerd and Kubernetes, disabling swap, initializing the cluster, installing Flannel, and joining nodes to the cluster.

**Prerequisites**

Before starting the installation process, ensure that the following prerequisites are met:

* You have at least two Ubuntu 18.04 or higher servers available for creating the cluster.
* Each server has at least 2GB of RAM and 2 CPU cores.
* The servers have network connectivity to each other.
* You have root access to each server.

**Installation Steps**

The following are the step-by-step instructions for setting up a multi-node Kubernetes cluster using Kubeadm:

Update the system's package list and install necessary dependencies using the following commands:

sudo apt-get update

sudo apt install apt-transport-https curl -y

**Proper /etc/hosts entry for all master and nodes across the cluster**

ip route show

**ip r a default via 192.168.29.1 dev enp0s8 proto dhcp metric** **100**

**Disable swap**

Disable swap using the following command:

sudo swapoff -a

If there are any swap entries in the /etc/fstab file, remove them using a text editor such as nano:

sudo nano /etc/fstab

Enable kernel modules

sudo sysctl -w net.ipv4.ip\_forward=1

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

br\_netfilter

EOF

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf

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

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

EOF

sudo modprobe br\_netfilter

Add some settings to sysctl

sudo sysctl --system

**Install containerd**

To install Containerd, use the following commands:

sudo mkdir -p /etc/apt/keyrings

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

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

sudo apt-get update

**sudo apt-get install containerd.io -y**

**Create containerd configuration**

Next, create the containerd configuration file using the following commands:

sudo mkdir -p /etc/containerd

sudo containerd config default | sudo tee /etc/containerd/config.toml

sudo systemctl restart containerd

**Edit /etc/containerd/config.toml**

Edit the containerd configuration file to set SystemdCgroup to true. Use the following command to open the file:

sudo nano /etc/containerd/config.toml

Set SystemdCgroup to true:

SystemdCgroup = true

Restart containerd:

sudo systemctl restart containerd

**Install Kubernetes**

To install Kubernetes, use the following commands:

sudo apt-get update

*# apt-transport-https may be a dummy package; if so, you can skip that package*

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

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-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

sudo apt-get install -y kubelet=1.27.0-00 kubeadm=1.27.0-00 kubectl=1.27.0-00

crictl config --set runtime-endpoint=unix:///run/containerd/containerd.sock

**To enable internet access inside pod in the cluster**

<https://stackoverflow.com/questions/62664701/resolving-external-domains-from-within-pods-does-not-work/75626926#75626926>

This nameserver was completely different from the result of

resolvectl | grep "Current DNS Server": 10.1.0.2.

By executing **sudo ln -vfns /run/systemd/resolve/resolv.conf /etc/resolv.conf**

**Below after Cluster initialisation**

Add 8.8.8.8 and 8.8.4.4 to your CoreDNS's ConfigMap's forward statement

I changed my forward statement in my

CoreDNS's ConfigMap to this: forward . /etc/resolv.conf 8.8.8.8 8.8.4.4

kubectl edit cm coredns -n kube-system

**Initialize the Cluster (Run only on master)**

Use the following command to initialize the cluster:

kubeadm config images pull

**kubeadm init --apiserver-advertise-address=192.168.29.157 --pod-network-cidr=192.168.0.0/16**

Create a .kube directory in your home directory:

mkdir -p $HOME/.kube

Copy the Kubernetes configuration file to your home directory:

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

Change ownership of the file:

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

**Install Calico (Run only on master)**

Use the following command to install:

curl https://raw.githubusercontent.com/projectcalico/calico/v3.28.0/manifests/calico-typha.yaml -o calico.yaml

As nodes had different network interfaces, calico-node autodetected different networks on nodes. To fix this I added IP\_AUTODETECTION\_METHOD env var to calico-node DaemonSet. In my case I used these env var values:

# Auto-detect the BGP IP address.

- name: IP

value: "autodetect"

- name: IP\_AUTODETECTION\_METHOD

value: cidr=192.168.29.0/24

- name: IP\_AUTODETECTION\_METHOD

value: "interface=eth.\*"

crictl pull offseccw/kalilinux

crictl pull kalilinux/kali-rolling

kubectl create deploy kali --image=offseccw/kalilinux -- sleep 3600000

**Verify Installation**

Verify that all the pods are up and running:

kubectl get pods --all-namespaces

**Join Nodes**

To add nodes to the cluster, run the kubeadm join command with the appropriate arguments on each node. The command will output a token that can be used to join the node to the cluster.

<https://get.helm.sh/helm-v3.15.1-linux-amd64.tar.gz>

<https://github.com/rancher/local-path-provisioner>

<https://github.com/kubernetes-sigs/metrics-server>

wget https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml

spec:

containers:

- args:

- --cert-dir=/tmp

- --secure-port=10250

- --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname

- --kubelet-use-node-status-port

- --metric-resolution=15s

add there two more lines:

- --kubelet-insecure-tls=true

- --kubelet-preferred-address-types=InternalIP