# GUÍA DE DESPLIEGUE DEL ESCENARIO

## Instalación VMs

Software de virtualización: Oracle VM VirtualBox 7.0.14 r161095 (Qt5.15.2)

Sistema operativo (imagen ISO): Ubuntu Server 22.04.4 LTS (web oficial)

Almacenamiento nodo master: 42,57 GB

Almacenamiento nodo worker: 54GB

Memoria nodo master: 3,4GB

Memoria nodo worker: >10GB

CPUs virtuales master: 2

CPUs virtuales worker: 8

Adaptador 1: e

Conectado a: Adaptador puente

Modo promiscuo: Permitir todo

Habilitar virtualización anidada (en carpeta de instalacion de Vbox para windows)

.\VBoxManage modifyvm <<nombre vm>> --nested-hw-virt on

Montar carpeta compartida

Sudo mount –t vboxsf <<nombre carpeta compartida creada>> <<ruta de montaje>>

### Instalación del SO

Base de instalación: Estándar

Configuración IP inicial: DHCP

Configuraciones adicionales: Instalado OpenSSH server package

## Instalación nodo k8master

#### Instalación paquetes básicos:

sudo apt update

sudo apt install -y nano net-tools git make gcc apt-transport-https ca-certificates curl

#### Nombres de dominio (/etc/hosts)

192.168.1.100 k8master

192.168.1.111 k8worker01

192.168.1.222 k8worker02

#### Configuración manual IP y ruta por defecto

sudo nano /etc/netplan/00-installer-config-yaml

network:

ethernets:

enp0s3:

addresses: [192.168.1.100/24]

nameservers:

addresses: [8.8.8.8]

routes:

* to: default

via: 192.168.1.1

version: 2

sudo netplan apply

#### Configuración básica

sudo su

sudo ufw disable

echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables

echo '1' > /proc/sys/net/ipv4/ip-forward

swapoff –a (SESIÓN)

Sudo sed –i ‘/ swap / s/^\(.\*\)$/#\1/g’ /etc/fstab

Sudo tee /etc/modules-load.d/containerd.conf <<EOF

Overlay

Br\_netfilter

EOF

modprobe br\_netfilter

modprobe overlay

Modprobe kvm

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

#### Instalación Kubectl, Kubelet, Kubeadm (root)

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

#### Instalación CRI cri-dockerd

apt-get update

apt-get install -y kubelet kubeadm kubectl

apt-mark hold kubelet kubeadm kubectl

install -o root -g root -m 0755 cri-dockerd /usr/local/bin/cri-dockerd

install packaging/systemd/\* /etc/systemd/system

sed -i -e 's,/usr/bin/cri-dockerd,/usr/local/bin/cri-dockerd,' /etc/systemd/system/cri-docker.service

systemctl daemon-reload

systemctl enable --now cri-docker.socket

#### Docker (se puede emplear la clave y repositorio anteriores))

# Add Docker's official GPG key:

sudo apt-get update

sudo apt-get install 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

# Add the repository to Apt sources:

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 docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

sudo groupadd docker

sudo usermod -aG docker $USER

## Instalación nodo k8worker01

#### Instalación paquetes básicos

sudo apt update

sudo apt install -y nano net-tools git make gcc apt-transport-https ca-certificates curl

#### Nombres de dominio (/etc/hosts)

192.168.1.100 k8master

192.168.1.111 k8worker01

192.168.1.222 k8worker02

#### Configuración manual IP y vía a internet por defecto

/etc/netplan/00-installer-config-yaml

network:

ethernets:

enp0s3:

addresses: [192.168.1.111/24]

nameservers:

addresses: [8.8.8.8]

routes:

* to: default

via: 192.168.1.1

version: 2

sudo netplan apply

#### Configuración básica:

sudo su

sudo ufw disable

modprobe br\_netfilter

echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables

echo '1' > /proc/sys/net/ipv4/ip-forward

swapoff –a (SESIÓN)

nano /etc/fstab

#### Instalación Kubectl, Kubelet, Kubeadm (root/v1.29)

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

apt-get update

apt-get install -y kubelet kubeadm kubectl

apt-mark hold kubelet kubeadm kubectl

#### Instalación CRI cri-dockerd

install -o root -g root -m 0755 cri-dockerd /usr/local/bin/cri-dockerd

install packaging/systemd/\* /etc/systemd/system

sed -i -e 's,/usr/bin/cri-dockerd,/usr/local/bin/cri-dockerd,' /etc/systemd/system/cri-docker.service

systemctl daemon-reload

systemctl enable --now cri-docker.socket

#### Docker (se puede emplear la clave y repositorio anteriores))

(Add Docker's official GPG key:)

sudo apt-get update

sudo apt-get install 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

( Add the repository to Apt sources)

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 docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

sudo groupadd docker

sudo usermod -aG docker $USER

#### Agregación de nodos al cluster

kubeadm join 192.168.1.100:6443 --token <<TOKEN>> --discovery-token-ca-cert-hash sha256:<<HASH>> --cri-socket unix:///var/run/cri-dockerd.sock

sudo systemctl restart containerd

#### Crear carpeta para el volumen persistente

mkdir /home/k8worker01/kubedata

## Instalación nodo k8worker02

#### Instalación paquetes básicos

sudo apt update

sudo apt install -y nano net-tools git make gcc apt-transport-https ca-certificates curl

#### Nombres de dominio (/etc/hosts)

192.168.1.100 k8master

192.168.1.111 k8worker01

192.168.1.222 k8worker02

#### Configuración manual IP y vía a internet por defecto

/etc/netplan/00-installer-config-yaml

network:

ethernets:

enp0s3:

addresses: [192.168.1.222/24]

nameservers:

addresses: [8.8.8.8]

routes:

* to: default

via: 192.168.1.1

version: 2

sudo netplan apply

#### Configuración básica

sudo su

sudo ufw disable

modprobe br\_netfilter

echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables

echo '1' > /proc/sys/net/ipv4/ip-forward

swapoff -a (SESIÓN)

nano /etc/fstab

#### Instalación Kubectl, Kubelet, Kubeadm (root/v1.29)

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

apt-get update

apt-get install -y kubelet kubeadm kubectl

apt-mark hold kubelet kubeadm kubectl

#### Instalación CRI cri-dockerd

install -o root -g root -m 0755 cri-dockerd /usr/local/bin/cri-dockerd

install packaging/systemd/\* /etc/systemd/system

sed -i -e 's,/usr/bin/cri-dockerd,/usr/local/bin/cri-dockerd,' /etc/systemd/system/cri-docker.service

systemctl daemon-reload

systemctl enable --now cri-docker.socket

#### Docker (se puede emplear la clave y repositorio anteriores))

# Add Docker's official GPG key:

sudo apt-get update

sudo apt-get install 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

# Add the repository to Apt sources:

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 docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

sudo groupadd docker

sudo usermod -aG docker $USER

#### Crear carpeta para el volumen persistente

mkdir /home/k8worker02/kubedata

## Despliegue del cluster con KNE (nodo master)

#### KNE en nodo controller

git clone https://github.com/openconfig/kne.git

cd kne

make install

### Despliegue de cluster

#### Instalación Golang

Version : 1.21.3

curl -O https://dl.google.com/go/go1.21.3.linux-amd64.tar.gz

sudo rm -rf /usr/local/go && sudo tar -C /usr/local -xzf go1.21.3.linux-amd64.tar.gz

rm go1.21.3.linux-amd64.tar.gz

export PATH=$PATH:/usr/local/go/bin

export PATH=$PATH:$(go env GOPATH)/bin

#### Despliegue de cluster

kne deploy kne/deploy/kne/kubeadm.yaml (emplear el del repositorio)

#### Uso de cluster:

mkdir -p $HOME/.kube

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

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

#### Clúster K8

kubectl config view

kubectl get nodes

kubectl get pod –all-namespaces

kubectl get pod –n <<nombre topología>>

kubectl describe nodes

kubectl describe pod –A

kubectl get namespaces

#### Instalación de Flannel (CLI):

kubectl apply -f https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml

#### Instalación de Multus-cni:

git clone https://github.com/k8snetworkplumbingwg/multus-cni.git && cd multus-cni

cat ./deployments/multus-daemonset.yml |kubectl apply -f -

### Agregación de nodos al cluster (desde nodos worker)

kubeadm join 192.168.1.100:6443 --token <<TOKEN>> --discovery-token-ca-cert-hash sha256:<<HASH>> **--cri-socket unix:///var/run/cri-dockerd.sock**

# Creación y uso de topología

### Instalación imagen router arista ceos

docker import cEOS-lab-4.29.2F.tar ceos (tar en el repositorio)

### Crear Topología

Kne create examples/arista/ceos/2ceos/2ceos.yaml (carpeta 2ceos con archivos en el repositorio)

### Acecesos remotos

Router: ssh admin@<<ip del servicio ssh del router>> (usuario/contraseña: admin/admin)

Host (desde el nodo donde está desplegado): docker exec –it <<id del contenedor>> sh

### Configuración de hosts

Ver archivo “configuracion/hostX” y aplicar comandos en cada host

### Comandos testing

traceroute <<IP red interna>>

ping <<IP red interna>> -i <<intervalo>> -s <<tamaño bytes>>

### Comandos router ceos

tcmp dump –i <<nombre interfaz>>

## Despliegue Prometheus (nodo maestro)

kubectl create namespace monitoring

##### Prometheus

git clone <https://github.com/techiescamp/kubernetes-prometheus>

Kubectl create –f <<archivo>>

Desplegar:

* clusterRole.yaml
* config-map.yaml
* prometheus-deployment.yaml
* prometheus-service.yaml

##### Kube-metrics

git clone https://github.com/devopscube/kube-state-metrics-configs.git

kubectl apply -f kube-state-metrics-configs/

##### Node exporter

git clone https://github.com/bibinwilson/kubernetes-node-exporter

kubectl create -f daemonset.yaml

## Despliegue administrador SNMP (nodo maestro)

sudo apt update  
sudo apt install snmp snmp-mibs-downloader

sudo nano /etc/snmp/snmp.conf (aplicar los cambios del archivo del repositorio)

## Uso de scripts

### Instalación de Python3

sudo apt-get install python3

### Instalación de paquetes

pip install kubernetes

pip install prometheus-client

pip install pysnmp

pip install asciichartpy

### Uso del script

python3 <<archivo .py>>