172.20.10.2 - 14

| Nube/AWS | | | |
| --- | --- | --- | --- |
| Host | IP LAN | IP VPN | Public |
| ubuntu | 172.31.26.136/20 | 10.0.0.1/24 | 34.224.69.63 |
|  |  |  |  |
|  |  |  |  |
| VIRTUAL |  | 10.0.0.9/24 |  |

| PC1/rodri | | | |
| --- | --- | --- | --- |
| Host | IP LAN | IP VPN |  |
| kuproxy1 | 172.20.10.? | 10.0.0.10/24 |  |
| kumaster1 | 172.20.10.? | 10.0.0.20/24 |  |
| kuworker1 | 172.20.10.? | 10.0.0.30/24 |  |
| VIRTUAL |  | 10.0.0.9/24 |  |

| PC2/carlos | | | |
| --- | --- | --- | --- |
| Host | IP LAN | IP VPN |  |
| kuproxy2 | 172.20.10.? | 10.0.0.11/24 |  |
| kumaster2 | 172.20.10.? | 10.0.0.21/24 |  |
| kuworker2 | 172.20.10.? | 10.0.0.31/24 |  |
| VIRTUAL |  | 10.0.0.9/24 |  |

| PC3/casa | | | |
| --- | --- | --- | --- |
| Host | IP LAN | IP VPN |  |
| kuproxy3 | 172.20.10.? | 10.0.0.13/24 |  |
| kumaster3 | 172.20.10.? | 10.0.0.22/24 |  |
| kuworker3 | 172.20.10.? | 10.0.0.32/24 |  |
| VIRTUAL |  | 10.0.0.9/24 |  |

| PC4/cuadrado | | | |
| --- | --- | --- | --- |
| Host | IP LAN | IP VPN |  |
| kuproxy3 | 172.20.10.? | 10.0.0.15/24 |  |
| kumaster3 | 172.20.10.? | 10.0.0.25/24 |  |
| kuworker3 | 172.20.10.? | 10.0.0.35/24 |  |
| VIRTUAL |  | 10.0.0.9/24 |  |

| PC5/Pruebas | | | |
| --- | --- | --- | --- |
| Host | IP LAN | IP VPN |  |
| ubuntu/Desktop | 172.20.10.? | 10.0.0.14/24 |  |

**Comandos útiles**

Aplicar yaml

kubectl apply -f “nombre.yaml”

Creación

kubectl create namespace “nombre namespace”

Información

kubectl get pods -o wide

kubectl get namespace -o wide

kubectl get nodes

Borrado

kubectl delete namespace “nombre namespace”

kubectl delete pods “nombre pods”

kubectl delete pod “nombre pod” -n “nombre namespace”

**Meter los equipos en la VPN**

**IMPORTANTE SIEMPRE EN ROOT**

Sudo apt-get update

apt install wireguard

(como sudo el de abajo)

wg genkey | tee /etc/wireguard/client\_private.key | wg pubkey > /etc/wireguard/client\_public.key

Sx4LA9NC2DfGYpR0c8tfuBhPuMe9hJcuTj9iaIuzcAU=

sudo nano /etc/wireguard/wg0.conf

[Interface]

Address = 10.0.0.2/24

PrivateKey = <CLAVE\_PRIVADA\_DEL\_CLIENTE>

[Peer]

PublicKey = <CLAVE\_PUBLICA\_DEL\_SERVIDOR>

Endpoint = <IP\_PUBLICA\_DEL\_SERVIDOR>:51820

AllowedIPs = 0.0.0.0/0

PersistentKeepalive = 25

sudo systemctl enable wg-quick@wg0

sudo systemctl start wg-quick@wg0

**Creacion maquina haproxy keepalived**

**Haproxy Keepalived**

Configuración de HAproxy

sudo apt install keepalived haproxy psmisc -y

vi /etc/haproxy/haproxy.cfg

frontend kubernetes-frontend

bind \*:6443

mode tcp

option tcplog

default\_backend kubernetes-backend

backend kubernetes-backend

mode tcp

option tcplog

option tcp-check

balance roundrobin

default-server inter 10s downinter 5s rise 2 fall 3 slowstart 60s maxconn 250 maxqueue 256 weight 100

server kumaster1 172.20.10.20:6443 check

server kumaster2 172.20.10.21:6443 check

server kumaster3 172.20.10.22:6443 check

#server kuproxy0 192.168.1.10:6443 check

#server kuproxy2 192.168.1.12:6443 check

peers mypeers

peer lb0 172.20.10.10:1024

peer lb2 172.20.10.12:1024

)

systemctl enable haproxy

systemctl restart haproxy

systemctl enable haproxy

Configuración de Keepalived

/etc/keepalived/keepalived.conf

global\_defs {

notification\_email {

}

router\_id LVS\_DEVEL

vrrp\_skip\_check\_adv\_addr

vrrp\_garp\_interval 0

vrrp\_gna\_interval 0

}

vrrp\_script chk\_haproxy {

script "killall -0 haproxy"

interval 2

weight 2

}

vrrp\_instance haproxy-vip {

state BACKUP

priority 100

interface ens33

virtual\_router\_id 60

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

unicast\_src\_ip 172.20.10.10

unicast\_peer {

172.20.10.11

172.20.10.12

}

virtual\_ipaddress {

172.20.10.9/28

}

systemctl enable keepalived

—------------------------------------------------------------

**Creación de la máquina Master**

Cambiar el nombre del host

/etc/hosts

/etc/hostname

sudo apt update

sudo apt upgrade

sudo swapoff -a

sudo sed -i '/ swap / s/^\(.\*\)$/#\1/g' /etc/fstab

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

sudo apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates

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](http://containerd.io/)

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

sudo systemctl restart containerd

sudo systemctl enable containerd

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 update

sudo apt install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

**Creación del cluster en 1 master**

kubeadm init –control-plane-endpoint IPMASTER:6443 –upload-certs

**IMPORTANTE**Al realizar el init nos dará unas claves que tenemos que guardar para que el resto de nodos se puedan unir al CLUSTER

mkdir -p $HOME/.kube

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

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

**Unión al Cluster MASTER**

kubeadm join 192.168.1.9:6443 --token yiu5lw.vlf1royrxc255ehi \

--discovery-token-ca-cert-hash sha256:022c50bdd812e69468cb6505f5637fd8f3afceffdece62f9e692f4dba0e2e424 \

--control-plane --certificate-key 54412941d589bb2c19f89901943ca2e34123d1d5a063264d4cc976f50c1ea6cb

kubectl apply -f <https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml>

kubectl get pods -n kube-system

kubectl get nodes

**Union al cluster Worker**

kubeadm join 192.168.1.9:6443 --token yiu5lw.vlf1royrxc255ehi \

--discovery-token-ca-cert-hash sha256:022c50bdd812e69468cb6505f5637fd8f3afceffdece62f9e692f4dba0e2e424

**FINAL TODOS LOS NODOS**  
kubectl apply -f <https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml>

**Creación de servicios para los worker**

Aplicar yaml

kubectl apply -f “nombre.yaml”

Creacion de maquinas / Servicios  
apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

spec:

replicas: 3 # Número de réplicas (pods) que deseas ejecutar

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:latest # Imagen de Docker de NGINX

ports:

- containerPort: 80 # Puerto en el contenedor

**Creación de Servicio red**

apiVersion: v1

kind: Service

metadata:

name: nginx-service

spec:

selector:

app: nginx

ports:

- protocol: TCP

port: 80

targetPort: 80

type: NodePort

**MetalLB**

kubectl edit configmap -n kube-system kube-proxy

apiVersion: kubeproxy.config.k8s.io/v1alpha1

kind: KubeProxyConfiguration

mode: "ipvs"

ipvs:

strictARP: true

kubectl get configmap kube-proxy -n kube-system -o yaml | \

sed -e "s/strictARP: false/strictARP: true/" | \

kubectl diff -f - -n kube-system

kubectl get configmap kube-proxy -n kube-system -o yaml | \

sed -e "s/strictARP: false/strictARP: true/" | \

kubectl apply -f - -n kube-system

kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.14.5/config/manifests/metallb-native.yaml

sudo nano ips.yaml

apiVersion: metallb.io/v1beta1

kind: IPAddressPool

metadata:

name: first-pool

namespace: metallb-system

spec:

addresses:

- 10.0.0.50-10.0.0.60

kubectl apply -f ips.yaml

wget <https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/cloud/deploy.yaml>

kubectl apply -f deploy.yaml

Verificacion de funcionamiento

kubectl get pods -n metallb-system

nano nginx.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

labels:

app: nginx

spec:

replicas: 3

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:latest

ports:

- containerPort: 80

Servicio para el nginx con metallb

apiVersion: v1

kind: Service

metadata:

name: nginx-service

namespace: default

spec:

type: LoadBalancer

selector:

app: nginx

ports:

- protocol: TCP

port: 80

targetPort: 80

kubectl apply -f nginx.yaml

Metallb load balancer

sudo nano metalb-service.yaml

apiVersion: v1

kind: Service

metadata:

name: metallb-service

namespace: default

spec:

selector:

app: metallb-app

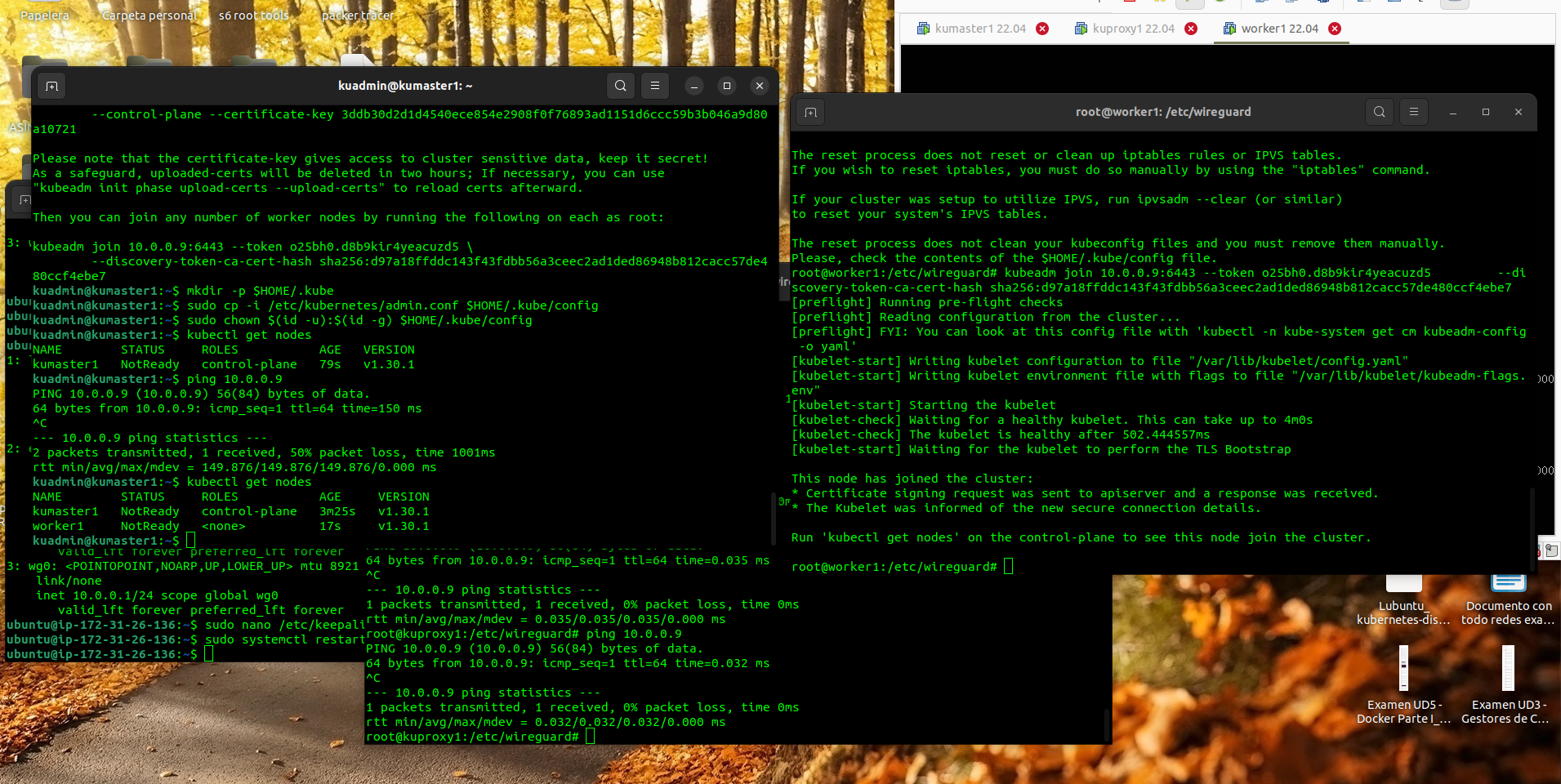
ports:

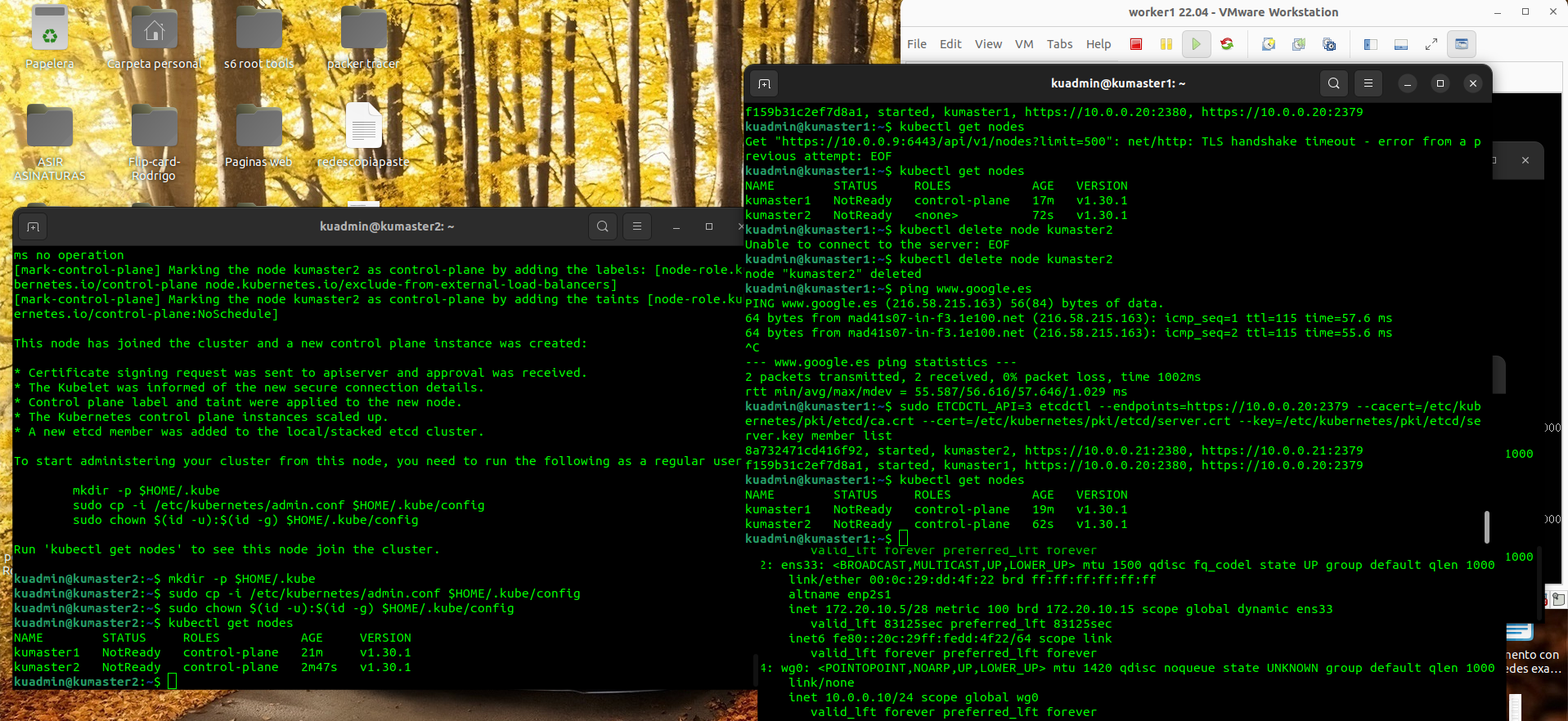
- protocol: TCP

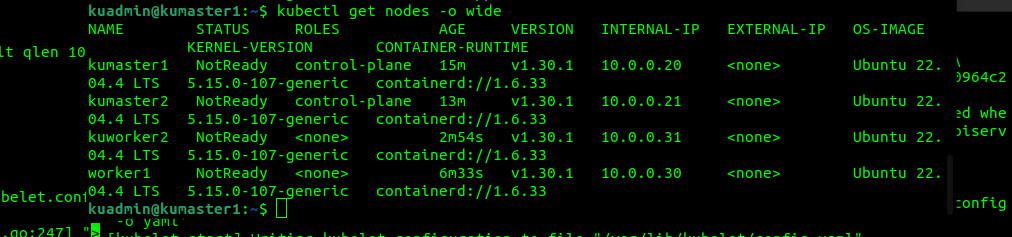
port: 80

targetPort: 8080

type: LoadBalancer







Antes de los join

/etc/default/kubelet

KUBELET\_EXTRA\_ARGS="--node-ip=10.0.0.30"

sudo systemctl daemon-reload

sudo systemctl restart kubelet

**master 1**

sudo kubeadm init --control-plane-endpoint "10.0.0.9:6443" --apiserver-advertise-address "10.0.0.20" --pod-network-cidr "192.168.0.0/16" --upload-certs

**master 2 y 3**

sudo kubeadm join 10.0.0.9:6443 --token h6f881.ut9ou0pxpenvmvm4 \

--discovery-token-ca-cert-hash sha256:167fe96f9062dff6776d177c5c584c698fd36e3513c5ee2084671c8c60a26124 \

--control-plane --certificate-key 45e51db5c43effe074fa6f3ff96ee17e03dcecf85bda1f577580d49c9ec79b17

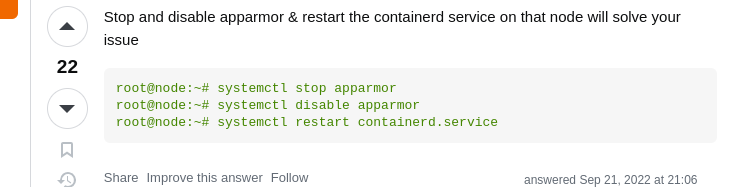
--apiserver-advertise-address "10.0.0.21"

**workers 1 al 3**

sudo kubeadm join 10.0.0.9:6443 --token h6f881.ut9ou0pxpenvmvm4 \

--discovery-token-ca-cert-hash sha256:167fe96f9062dff6776d177c5c584c698fd36e3513c5ee2084671c8c60a26124 --apiserver-advertise-address "10.0.0.31"

sudo kubeadm init --config kubeadm-config.yaml --upload-certs



systemctl stop apparmor

systemctl disable apparmor

systemctl restart containerd.service

<https://stackoverflow.com/questions/49112336/container-runtime-network-not-ready-cni-config-uninitialized>

