***Descomplicando o kubernetes***

**#Docker runtime**

Se o nosso container runtime for o Docker CE, siga esse passaços para fazer a configuração.

**# Adicione ao repositório para fazer a instalação dos pacotes.**

sudo apt update

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

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 containerd.io docker-ce docker-ce-cli

**#Instalando via Script.**

curl -fsSL https://get.docker.com | bash

sudo mkdir -p /etc/systemd/system/docker.service.d

**# Criando o arquivo json config file**

sudo tee /etc/docker/daemon.json <<EOF

{

"exec-opts": ["native.cgroupdriver=systemd"],

"log-driver": "json-file",

"log-opts": {

"max-size": "100m"

},

"storage-driver": "overlay2"

}

EOF

**# Start e habilite na inicialização do sistema o Services**

sudo systemctl daemon-reload

sudo systemctl restart docker

sudo systemctl enable docker

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 enable containerd

Sudo systemctl restart containerd

# **Configure a persistência de carregamento dos módulos.**

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

br\_netfilter

ip\_vs

ip\_vs\_rr

ip\_vs\_sh

ip\_vs\_wrr

nf\_conntrack\_ipv4

overlay

EOF

**# Carregue os módulos**

sudo modprobe overlay

sudo modprobe br\_netfilter

**# A instalação requer sysctl parametros**

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

**Instalando o Kubernets: Verifique se os módulos estão carregados.**

vim /etc/modules-load.d/k8s.conf

br\_netfilter

ip\_vs

ip\_vs\_rr

ip\_vs\_sh

ip\_vs\_wrr

nf\_conntrack\_ipv4

overlay

**Vamos habilitar o repasse de pacotes e fazer com que o iptables gerencie os pacotes que estão trafegando pelas brigdes. Para isso vamos utilizar systcl para parametrizar o kernel: Verificando se o arquivo foi criado.**

nano /etc/sysctl.d/k8s.conf

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

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

net.ipv4.ip\_forward = 1

**Para ler as novas configurações:**

sysctl --system

**Atualize o repositorio: Distribuição Ubuntu 22.04**

# apt-get update -y && apt-get upgrade -y

# apt-get update && apt-get install -y apt-transport-https

# curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/kubernetes-xenial.gpg

# sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"

**Debian**

# curl -fsSLo /etc/apt/keyrings/kubernetes-archive-keyring.gpg <https://packages.cloud.google.com/apt/doc/apt-key.gpg>

# echo "deb [signed-by=/etc/apt/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list

# apt-get update

# apt-get install -y kubelet kubeadm kubectl

# swapoff -a

# vim /etc/fstab comente a linha do swap depois mount –a para conferir.

**Apenas no Nó Master executar o pull das imagens do kubernets:**

# kubeadm config images pull

# kubeadm init

# 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://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"

# kubectl get pods -n kube-system

# kubeadm join --token 39c341.a3bc3c4dd49758d5 IP\_DO\_MASTER:6443 --discovery-token-ca-cert-hash sha256:37092

**Para criar um novo token:**

kubadm token create --print-join-command

**Alguns comandos iniciais para brincar com o Kubernets:**

# kubectl get nodes

# kubectl describe node elliot-03

# kubeadm token create --print-join-command

# echo "source <(kubectl completion bash)" >> ~/.bashrc

# kubectl get namespace

# kubectl get pods -n kube-system

# kubectl get pods --all-namespaces

# kubectl run nginx --image nginx

# kubectl get deployments

# kubectl describe deployment nginx

# kubectl get events

# kubectl get deployment nginx -o yaml

# kubectl get deployment nginx -o yaml > meu\_primeiro.yaml

# kubectl delete deployment nginx

# kubectl create -f meu\_primeiro.yaml

# kubectl delete -f meu\_primeiro.yaml

# kubectl get deploy,pod

# kubectl expose deployment/nginx

# kubectl get svc nginx

# kubectl describe pod nginx-6f858d4d45-qxjlh

# kubectl get pods -o wide

# kubectl delete pods nginx-6f858d4d45-qxjlh

**Instalando Kubernetes em alta disponibilidade.**

Tipos de topologias de K8s multi-master: https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/ha-topology/

Instalação kubeadm, kubelet e kubectl: https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

Instalação Kubernetes multi-master: https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/high-availability/

HAproxy: <https://www.haproxy.org/>

Em breve tutoria de instalação.