TP1 installation de l'environnement de TP

Le but de ce TP est de monter un environnement K8S sur des machines virtuelles afin de réaliser la suite des TPs.

Prérequis

VirtualBox/Vagrant, connexion internet.

Se référer à la documentation d'installation de VirtuaBox et Vagrant.

- https://www.virtualbox.org/wiki/Downloads
- https://www.vagrantup.com/downloads

Tester le lancement d'une machine vagrant avant de commencer.

```
mkdir vagrant-test
cd vagrant-test
vagrant init ubuntu/focal64
vagrant up
vagrant ssh
```

Architecture cible

Nous allons metre en place cette architecture:

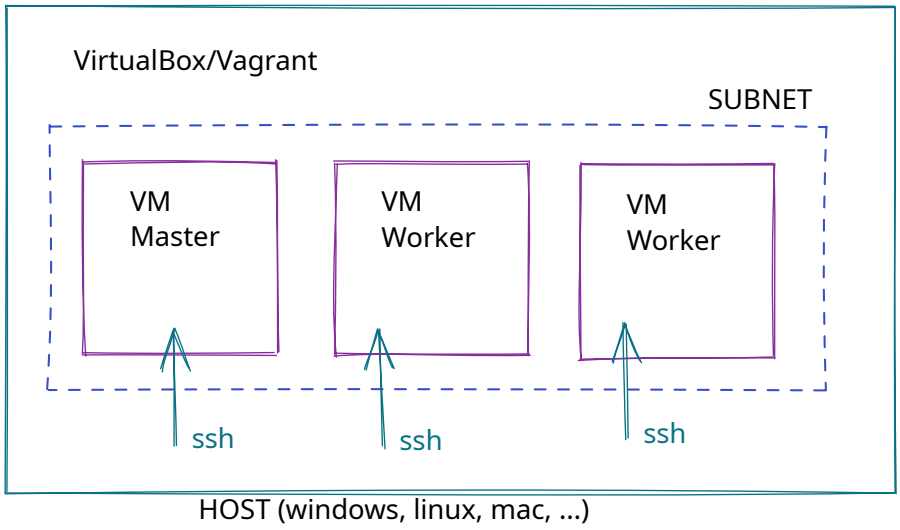

Figure 1: archi.svg

Installation des machines virtuells

- Machine 1 (master): Ubuntu 20.04

- Machines 2 et 3 (workers): Ubuntu 20.04

Créer 3 dossiers

```
mkdir master worker1 worker2
```

Placer le fichier `Vagrantfile` fourni dans chaque dossier, et modifier l'ip pour chaque machine.

- master 192.168.33.10
- worker1 192.168.33.11
- worker2 192.168.33.12

Création du noeud master

Ouvrir un terminal dans le dossier `master`, et lancer la machine virtuelle.

```
cd master
vagrant up
vagrant ssh
```

Sur la machine master, installer Docker et kubernetes

Docker:

```
sudo apt-get install ca-certificates curl gnupg lsb-release

# 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://do
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

sudo apt-get install -y docker-ce docker-ce-cli containerd.io
```

Créer un fichier de configuration docker `/etc/docker/daemon.json`

```
cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
```

```
  },
  "storage-driver": "overlay2"
}
EOF
```

Redémarrer docker

```
sudo systemctl daemon-reload
sudo systemctl restart docker
```

Vérifier que docker se lance bien: `sudo docker run hello-world`

Initialisation de Kubernetes

```
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 sysctl --system

sudo apt-get update
sudo apt-get install -y apt-transport-https curl
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo apt-key add -

cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb  https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
EOF

sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

(attention, changer l'ip si besoin)

```
sudo systemctl restart kubelet
sudo kubeadm init --apiserver-cert-extra-sans my-k8s.local --pod-network-cidr=172.16.0.0/12
```

Si tout se passe bien, le noeud master est correctement initialisé.

Exemple de réponse:

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:
```

```
  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 192.168.33.10:6443 --token fyv2wr.ooobph9jd8x5hoj9 \
    --discovery-token-ca-cert-hash sha256:0ffe4e1b58f9ea80b2f39aa8a5cd3c0655117d09bf1697e376
```

Le retour de la commande nous demande de préparer quelques fichiers de config, et de déployer un POD de getsion du réseau.

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Création du réseau

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

Création d'un worker

Ouvrir un terminal dans le dossier `worker1`, démarrer `vagrant`.

```
cd worker1
vagrant up
vagrant ssh
```

Modifier le hostname de la machine et se reconnecter en ssh.

```
sudo hostname worker1
exit
```

Répéter les étapes d'installation de Docker et Kubernetes.

Cette fois, ne pas lancer la commande kubeadm init. Utiliser la commande `kubeadm join` qui est donnée par le master.

Exemple:

```
sudo kubeadm join 192.168.36.10:6443 --token fyv2wr.ooobph9jd8x5hoj9 --discovery-token-ca-ce
```

Exemple de réponse:

```
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubead
W0124 10:54:12.853193    9820 utils.go:69] The recommended value for "resolvConf" in "Kubele
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubead
```

```
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Depuis le noeud master, vérifier que le worker 1 est bien ajouté.

```
vagrant@ubuntu-focal:~$ kubectl get nodes
NAME           STATUS   ROLES                  AGE     VERSION
ubuntu-focal   Ready    control-plane,master   4m44s   v1.23.2
worker1        Ready    <none>
```

Répéter toutes ces opérations pour le worker2

Vérifier à nouveau que le custler est bien formé depuis le master (kubectl get nodes)

Commandes pour reset après un reboot/restart des machines

Sur le noeud master

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

Sur les workers

```
sudo kubeadm reset
sudo kubeadm join 192.168.56.10:6443 --token xk0ew2.pi59xvo1f7teq2sd --discovery-token-ca-ce
```