Table des matières

Installer et Manipuler des Conteneurs dans un Cluster

3 VM Debian identique
Hyperviseur PROXMOX
VLAN Dédié
Spec: 2 GO Ram / 2 CPU / 20 Go

Pre-requis

Préparation VM issue de template:

Contrôle des mac address:

```
ip link | grep link/ether
```

Contrôle des uuid system:

```
cat /sys/class/dmi/id/product_uuid
```

Ajouter un HostName a chaque serveur:
(ex: debian-K8s1 & debian-K8s2 pour les Worker & debian-K8s.Master pour le Master)

```
nano /etc/hostname
```

Ajouter une configuration en ip Fixe: (Pour chaque VM)
Ex: 6 & 7 pour les Worker et 8 pour le master.

```
nano /etc/network/interface
```

```
# The primary network interface
auto ens18
iface ens18 inet static
address 10.200.20.6/28
gateway 10.200.20.1
```

A faire pour chaque VM **Ajouter les entrées dans le Fichier /etc/hosts: (sur toutes le VM)**

```
10.200.20.6 debian-K8s1.unifi.vm
10.200.20.6 debian-K8s1
```

```
10.200.20.7 debian-K8s2.unifi.vm
10.200.20.7 debian-K8s2
10.200.20.8 debian-K8s0.Master.unifi.vm
10.200.20.8 debian-K8s0.Master
```

————————————–
Ajouter dans /etc/resolve.conf
A faire si pas de DNS dans l'infra.
nameserver 127.0.0.1
————————————–

Config par defaut resolv.conf:

```
domain unifi.vm
search unifi.vm.
nameserver 208.67.220.123
nameserver 208.67.222.123
```

Charger les modules requis et appliquer les paramètres au kernel.

```
cat << EOF | sudo tee /etc/modules-load.d/containerd.conf
overlay
br_netfilter
EOF
```

Permettre à iptable de voir le traffic PONTÉ

```
sudo modprobe br_netfilter
sudo modprobe overlay
```

Contrôler:

```
lsmod | grep br_netfilter
```

```
br_netfilter           32768  0
bridge                253952  1 br_netfilter
```

Pour que les iptables de votre nœud Linux voient correctement le trafic ponté, vous devez vous assurer que net.bridge.bridge-nf-call-iptables est défini sur 1 dans votre configuration sysctl:

```
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
```

Recharger avec les nouveaux paramètres

```
sudo sysctl --system
```

- Désactiver le Swap dans /etc/fstab avec un #

- Passer aussi cette commande

```
sudo -i
swapoff -a
exit
```

Toutes ces actions sont à faire sur tout vos serveur Master & Worker

Docker

Mettre à jours ces paquets:

```
sudo apt-get update
```

Pré-requis

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

Créer le répertoire keyrings dans /etc/apt/

```
sudo mkdir -p /etc/apt/keyrings
```

Télécharger les clé gpg

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

Ajouter les sources

echo \

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

Installer Docker Engine:

```
sudo apt-get update
```

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

```
plugin -y
```

Conf Containerd

Éffacer la conf par défaut et regénérer en une

```
rm -rf /etc/containerd/config.toml
$ containerd config default | sudo tee /etc/containerd/config.toml
```

Chercher dans la section [plugins.“io.containerd.grpc.v1.cri”.containerd.runtimes.runc.options] passer la valeur de SystemdCgroup à true dans le nouveau fichier **config.toml** généré ou passer directement cette commande qui fera exactement la même chose:

```
$ sed -i 's/SystemdCgroup = false/SystemdCgroup = true/'
/etc/containerd/config.toml
```

Ajouter votre utilisateur dans le groupe docker

```
sudo usermod -aG docker <username>
```

Test si l'installation de Docker est correcte:

```
sudo docker run hello-world
```

Après cela redémarrer le systeme

```
systemctl reboot
```

Toutes ces actions sont à faire sur tout vos serveur Master & Worker

K8s

Mettre à jour vos paquets et installer les pré-requis

Pré-requis

```
sudo apt-get update && sudo apt-get install -y apt-transport-https curl
```

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key
add -
cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main

```

```
EOF
```

Installer Kublet Kubeadm et Kubectl

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

Passer vos paquets en OLD pour eviter de les mettre à jours par errreur

```
sudo apt-mark hold kubelet kubeadm kubectl
```

Resultat:

```
root@debian-K8s0:/home/sylvain# sudo apt-mark hold kubelet kubeadm kubectl
kubelet passé en figé (« hold »).
kubeadm passé en figé (« hold »).
kubectl passé en figé (« hold »).
```

Notez: Configurer le driver de cgroup utilisé par la kubelet sur un “nœud master”
Lorsque vous utilisez Docker, kubeadm détecte automatiquement le pilote (driver) de cgroup pour kubelet et le configure dans le fichier /var/lib/kubelet/config.yaml lors de son éxecution. si vous utilisez un autre CRI, vous devez passer votre valeur cgroupDriver avec kubeadm init, comme ceci :
##
apiVersion: kubelet.config.k8s.io/v1beta1
kind: KubeletConfiguration
cgroupDriver: <value>
##

Relancer les nouveaux services installé:

```
sudo systemctl daemon-reload && sudo systemctl restart kubelet
```

Toutes ces actions sont à faire sur tout vos serveur Master & Worker

Init du Master

sur le Master Seulement:

Initialiser le cluster Kubernetes.

```
kubeadm init --apiserver-advertise-address=10.200.20.6 --pod-network-
```

```
cidr=172.20.0.0/16
```

```
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

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

kubeadm join 10.200.20.9:6443 --token 3kiw1t.t5pf607g66oowm0x \
    --discovery-token-ca-cert-hash
sha256:ac817f253c44949f4bede5c8a64775b7ba5c62663c7580dd42b01fdddffed056
```

Charger les image par défaut

```
kubeadm config images pull
```

Ajout Pod Réseau (CNI)

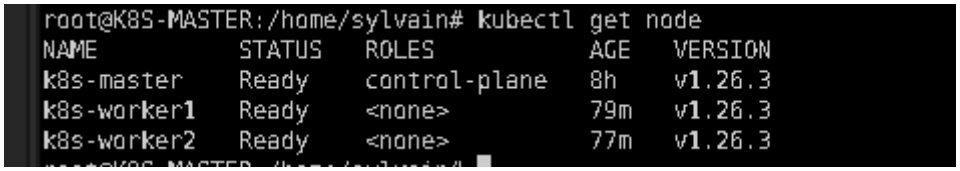
kubectl apply -f https://docs.projectcalico.org/v3.16/manifests/calico.yaml

Dépannage

Master Port Closed

```
sudo -i
swapoff -a
exit
strace -eopenat kubectl version
Puis ressayer la commande Join.
```

- CNI configuration: /etc/cni/net.d
- kubectl get nodes (Affiche tout les nodes et leur status



- kubeadm reset (sert a effacer un essais init/join infructueux)
- kubectl cluster-info (info cluster)

```
root@K8S-MASTER:/home/sylvain# kubectl cluster-info
Kubernetes control plane is running at https://10.200.20.6:6443
CoreDNS is running at https://10.200.20.6:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

- kubectl get pods –all-namespaces

```
root@K8S-MASTER:/home/sylvain# kubectl get pods --all-namespaces --output=wide
NAMESPACE     NAME                                       READY   STATUS    RESTARTS         AGE     IP               NODE          NOMINATED NODE   READINESS GATES
kube-system   calico-kube-controllers-69d88f787c-lkp6p   1/1     Running   0                7h44m   172.20.126.1     k8s-worker2   <none>           <none>
kube-system   calico-node-22n8h                          1/1     Running   1 (7h40m ago)    7h44m   10.200.20.6      k8s-master    <none>           <none>
kube-system   calico-node-cdc41                          1/1     Running   0                16m     10.200.20.8      k8s-worker2   <none>           <none>
kube-system   calico-node-zr28p                          1/1     Running   1 (12m ago)      18m     10.200.20.7      k8s-worker1   <none>           <none>
kube-system   coredns-787d4945fb-rq84w                   1/1     Running   0                7h48m   172.20.235.193   k8s-master    <none>           <none>
kube-system   coredns-787d4945fb-v9hxt                   1/1     Running   0                7h48m   172.20.235.194   k8s-master    <none>           <none>
kube-system   etcd-k8s-master                            1/1     Running   6 (7h40m ago)    7h48m   10.200.20.6      k8s-master    <none>           <none>
kube-system   kube-apiserver-k8s-master                  1/1     Running   38 (7h40m ago)   7h48m   10.200.20.6      k8s-master    <none>           <none>
kube-system   kube-controller-manager-k8s-master         1/1     Running   39 (7h40m ago)   7h48m   10.200.20.6      k8s-master    <none>           <none>
kube-system   kube-proxy-j8dtk                           1/1     Running   0                16m     10.200.20.8      k8s-worker2   <none>           <none>
kube-system   kube-proxy-p9vxt                           1/1     Running   1 (12m ago)      18m     10.200.20.7      k8s-worker1   <none>           <none>
kube-system   kube-proxy-qpvt4                           1/1     Running   1 (7h40m ago)    7h48m   10.200.20.6      k8s-master    <none>           <none>
kube-system   kube-scheduler-k8s-master                  1/1     Running   47 (7h40m ago)   7h48m   10.200.20.6      k8s-master    <none>           <none>
```

- kubectl get pods –output=wide (plus de details)

Si init génère une erreur type Core-DNS ou Kube-Proxy:
Initialiser kubeProxy manuellement:
Faire un :

```
sudo kubeadm init phase addon kube-proxy
```

ensuite faire:

```
kubeadm reset
```

et relancer l'init du CPANE.

Si vous n'avez pas le jeton, vous pouvez l'obtenir en exécutant la commande suivante sur le nœud master:

```
kubeadm token list
```

L'output est similaire à ceci:

```
TOKEN                     TTL  EXPIRES              USAGES
DESCRIPTION            EXTRA GROUPS
8ewj1p.9r9hcjoqgajrj4gi  23h  2018-06-12T02:51:28Z authentication,  The
default bootstrap  system:
                                                   signing           token
generated by     bootstrappers:
                                                                     'kubeadm
init'.        kubeadm:
default-node-token
```

Par défaut, les jetons expirent après 24 heures. Si vous joignez un nœud au cluster après l’expiration

du jeton actuel, vous pouvez créer un nouveau jeton en exécutant la commande suivante sur le nœud maître:

```
kubeadm token create
```

L'output est similaire à ceci:

```
5didvk.d09sbcov8ph2amjw
```

Si vous n'avez pas la valeur –discovery-token-ca-cert-hash, vous pouvez l'obtenir en exécutant la suite de commande suivante sur le nœud master:

```
openssl x509 -pubkey -in /etc/kubernetes/pki/ca.crt | openssl rsa -pubin -
outform der 2>/dev/null | \
   openssl dgst -sha256 -hex | sed 's/^.* //'
```

L'output est similaire à ceci:

```
8cb2de97839780a412b93877f8507ad6c94f73add17d5d7058e91741c9d5ec78
```

(Optionnel) Contrôler votre cluster à partir de machines autres que le master

Afin d'utiliser kubectl sur une autre machine (par exemple, un ordinateur portable) pour communiquer avec votre cluster, vous devez copier le fichier administrateur kubeconfig de votre master sur votre poste de travail comme ceci:

```
scp root@<master ip>:/etc/kubernetes/admin.conf .
kubectl --kubeconfig ./admin.conf get nodes
```

sources :

https://kubernetes.io/fr/docs/setup/production-environment/tools/kubeadm/_print/#check-required-ports

---

Ctrl+Alt+Sup

https://docs.tigera.io/calico/3.25/getting-started/kubernetes/quickstart

---