Lab1 – Déploiement d'un cluster Kubernetes Brahim HAMDI

Introduction

Dans ce Lab, nous allons déployer un cluster avec kubeadm de trois nœuds: un master et deux workers.

Boot et préparation de l'environnement

1. Installer le plugin vagrant vagrant-vbguest :

vagrant plugin install vagrant-vbguest

2. Git clonez le répo suivant de la formation :

git clone https://github.com/brahimhamdi/k8s-lab.git

o Sous le répertoire k8s, démarrer et provisionner les 3 VMs.

cd k8s-lab

vagrant up

Cette dernière commande:

- Démarre les 3 VM : k8s-master, k8s-worker1 et k8s-worker2
- Installe containerd et kubernetes
- Il est ensuite possible de se connecter en SSH sur chaque machine avec les commandes suivantes :

vagrant ssh k8s-master

vagrant ssh k8s-worker1

vagrant ssh k8s-worker2

 Kubernetes ne supporte pas le swap (la mémoire virtuelle) et il est donc nécessaire de le désactiver.

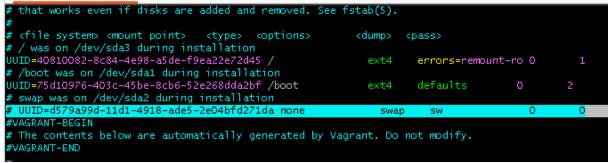
Tapez les 2 commandes suivantes sur chaque VM pour désactiver le swap:

sudo swapoff -a

Et pour le désactiver définitivement pour les prochains redémarrage, éditez le fichier /etc/fstab :

sudo nano /etc/fstab

puis commentez la ligne de montage automatique du swap :



Initialisation du cluster sur master

- **3.** Sur le nœud k8s-master lancer la commande suivante pour initialiser le cluster: sudo kubeadm init --apiserver-advertise-address=192.168.205.100 --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=NumCPU Les options :
 - --api-advertise-address=192.168.205.100 : IP annoncée par le master
 - --pod-network-cidr=10.244.0.0/16 : Sous réseau des pods
 - --ignore-preflight-errors=NumCPU : Ignorer le nombre de cpus (kubernetes exige au moins 2 cpus pour le master)

L'opération prend quelques minutes suivant la qualité de la connexion.

- 4. Toujours sous master, nous allons repasser en mode non root pour la suite.
 - Pour configurer kubectl sur le master, taper les 3 commandes suivantes (en mode non root):

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

- **5.** Pour terminer et rendre définitivement le node Ready, il faut rajouter un plugin réseau. Nous allons utiliser flannel.
 - Télécharger flannel, et ajouter la ligne "- --iface=eth1" au fichier "kubeflannel.yml" (après la ligne "- --kube-subnet-mgr"):

wget https://raw.githubusercontent.com/flannel-io/flannel/master/Documentation/kube-flannel.yml nano kube-flannel.yml

• Puis appliquer le fichier kube-flannel.yml : kubectl apply -f kube-flannel.yml

6. Avant d'ajouter worker1 et worker2, on doit vérifier l'état du cluster et du master ainsi que le système kubernetes avec les commandes suivantes :

```
kubectl cluster-info
kubectl get pods -n kube-system
kubectl get nodes
```

Nous sommes maintenant prêt à rajouter les workers nodes.

Déploiement des worker nodes

7. Pour rajouter un worker node, il suffit d'utiliser la commande *join* (sur le worker node et en mode root) affichée précédemment à la fin de la commande *kubeadm init*.

```
sudo kubeadm join 192.168.205.100:6443 --token ...
```

Si vous avez perdu cette commande, vous pouvez la regénérer plus tard avec la commande *sudo kubeadm token create --print-join-command* depuis le nœud master.

° Sur master surveiller la liste des nodes :

kubectl get nodes -o wide