

# 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 repo suivant de la formation :

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

- 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 :

```
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sda3 during installation
UUID=40810082-8c84-4e98-a5de-f9ea22e72d45 / ext4 errors=remount-ro 0 1
# /boot was on /dev/sda1 during installation
UUID=75d10976-403c-45be-8cb6-52e268dda2bf /boot ext4 defaults 0 2
# swap was on /dev/sda2 during installation
# UUID=d579a99d-11d1-4918-ade5-2e04bfd271da none swap sw 0 0
#VAGRANT-BEGIN
# The contents below are automatically generated by Vagrant. Do not modify.
#VAGRANT-END
```

## 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 "kube-flannel.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 régénérer 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*