Installation d'un cluster Kubernetes (CentOS 7)

- Etape 1 - Préparer les hôtes du cluster
- Etape 2 - Déployer le cluster Kubernetes

Etape 1 - Préparer les hôtes du cluster

*La procédure suivante couvre la préparation des hôtes Kubernetes sur **CentOS 7.6 (1810)** uniquement*

Avant l'installation et le déploiement d'un cluster Kubernetes, quelques étapes sont nécessaires afin d'assurer le fonctionnement idéal du cluster.

L'ensemble de cette procédure est à effectuer sur tous les hôtes du cluster !

Étapes préliminaires

- Activer le module noyau "br_netfilter" sur tous les hôtes du cluster, ce qui est requis par Kubernetes

```
modprobe br_netfilter
```

- La commande modprobe permet le chargement d'un module noyau durant l'exécution, mais cela n'est pas persistant après un redémarrage des serveurs. Créer le fichier `/etc/sysctl.d/k8s.conf` et ajouter les lignes suivantes :

```
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
```

- Appliquer les règles sysctl directement

```
sysctl --system
```

- Changer l'état de SELinux d'enforced à permissive pour éviter les problèmes à l'exécution de Kubernetes. La bonne pratique voudrait une configuration fine de SELinux ;)

```
setenforce Permissive
```

- Éditer le fichier `/etc/selinux/config`

```
SELINUX=permissive
SELINUXTYPE=targeted
```

Installation de Docker-CE 18.09

- Installer tout d'abord les dépendances pour Docker-CE

```
yum install -y yum-utils device-mapper-persistent-data lvm2
```

- Ajouter le repository YUM

```
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

- Installer Docker-CE à la version requise (18.09)

```
yum install -y docker-ce-18.09.8-3.el7.x86_64 docker-ce-cli-18.09.8-3.el7.x86_64 containerd.io
```

- Démarrer le service Docker et activer le démarrage du service au boot

```
systemctl enable --now docker
```

Configuration du Docker Daemon

Kubernetes nécessite une configuration particulière de Docker au niveau de cgroups. Cette configuration n'est pas impérative mais est fortement conseillée. Elle consiste à switcher le driver cgroup ***cgroupfs***, par défaut utilisé par Docker, par le driver cgroup ***systemd***.

Explications : https://kubernetes.io/docs/setup/production-environment/container-runtimes/#cgroup-drivers

- Créer un fichier `/etc/docker/daemon.json` et ajouter le contenu suivant :

```
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2",
  "storage-opts": [
    "overlay2.override_kernel_check=true"
  ]
```

```
}
```

- Il convient finalement de redémarrer le daemon Docker pour prendre en compte les changements

```
systemctl restart docker
```

- Vérifier que le driver cgroup **systemd** est bien chargé

```
docker info | grep Cgroup
Cgroup Driver: systemd
```

Configuration du pare-feu

Kubernetes requiert l'ouverture d'un certain nombre de ports sur les pare-feu de vos machines.

- Sur le Kubernetes Master

Port(s)	Description
6443	API Kubernetes
2379-2380	API Serveur etcd
10250	API Kubelet
10251	Kube-Scheduler
10252	Kube-Controller Manager
10255	Read-Only Kubelet API
179	Port BGP (pour Calico)

```
firewall-cmd --permanent --add-port={6443,2379-2380,10250,10251,10252,10255,179}/tcp
```

- Sur les minions (workers)

Port(s)	Description
10250	API Kubelet
10255	Read-Only Kubelet API
30000-32767	Ports Services NodePorts

179	Port BGP (pour Calico)

```
firewall-cmd --permanent --add-port={10250,10255,30000-32767,179}/tcp
```

Configuration NetworkManager pour Calico

Cette section n'est valable que si vous utilisez NetworkManager et Calico comme CNI sur votre cluster.

NetworkManager manipule la table de routage pour les interfaces dans namespace réseau par défaut, où les veth Calico sont ancrées pour les connexions aux conteneurs. Ceci peut interférer sur la capacité de routage de l'agent Calico.

- Créez le fichier de configuration suivant dans `/etc/NetworkManager/conf.d/calico.conf` pour empêcher NetworkManager d'interférer avec les interfaces :

```
[keyfile]
unmanaged-devices=interface-name:cali*;interface-name:tunl*
```

- Redémarrez NetworkManager

```
systemctl restart NetworkManager
```

That's it ! On se retrouve à l'étape 2!

Etape 2 - Déployer le cluster Kubernetes

Une fois l'ensemble de vos hôtes Kubernetes préparés avec Docker installé, l'ensemble des tweaks configurés et vos règles firewall, il est désormais possible de déployer votre cluster.

Déploiement du Kubernetes master

Déploiement des minions (workers)