Université de Picardie Jules Verne

Informatique – Master CCM

INSSET – Saint-Quentin

Conteneurs Applicatifs et Micro-Services M2

C. Drocourt

cyril.drocourt@u-picardie.fr

Cours 5.1 : Kubernetes - Installation

V2023.02

Table des matières

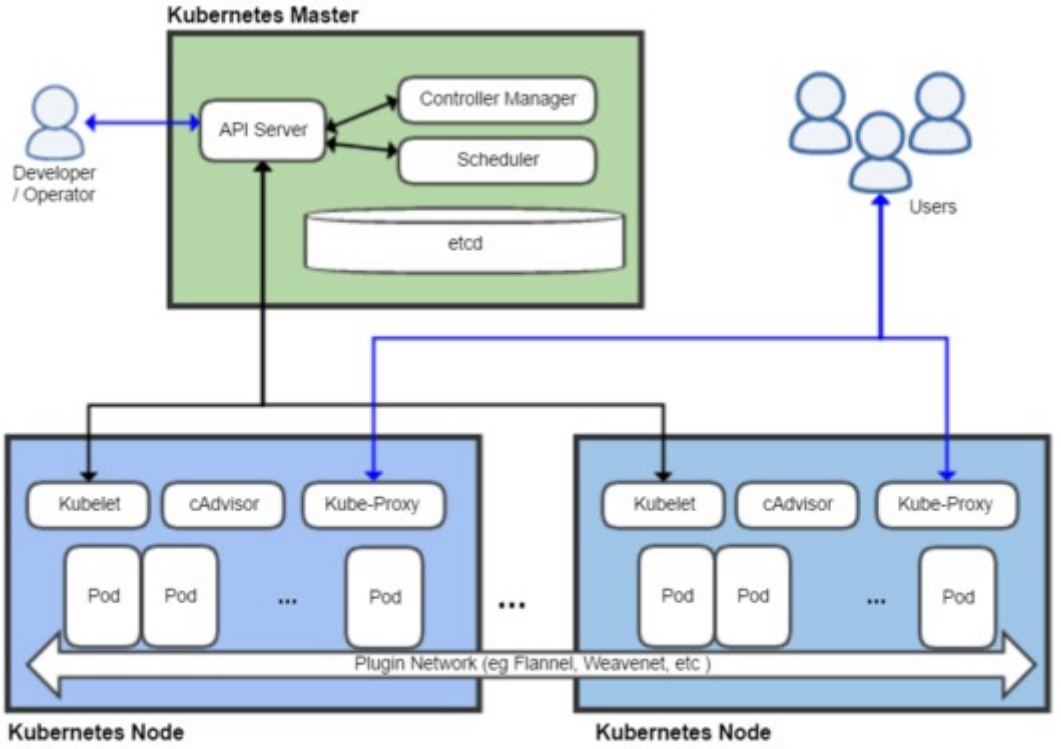
1 - Introduction

Kubernetes (ou K8s), est un orchestrateur de conteneur au même titre que Swarm mais conçu par Google et **beaucoup plus complexe**.

Il permet de réaliser les opérations de déploiement, de partage, de montée en charge, … Pour cela il comprend habituellement un nœud maître et plusieurs nœuds de travail exécutant des Pods.

Kubernetes est architecturé autour d'un grand nombre composants qui sont les Pods, les Labels, et les Services. Il **n'est pas destiné exclusivement à Docker**, il fonctionne aussi avec rkt, runC, containerd, ...

Kubernetes utilise d'autres technologies en complément comme « etcd » pour stocker les données, « flannel » pour le réseau, …


Kubernetes Master
Developer / Operator
API Server
Controller Manager
Scheduler
etcd
Users
Kubelet
cAdvisor
Kube-Proxy
Pod
Pod
...
Pod
...
Kubelet
cAdvisor
Kube-Proxy
Pod
Pod
...
Pod
Plugin Network (eg Flannel, Weavenet, etc)
Kubernetes Node
Kubernetes Node

2 - Installation Nœud maitre

Il faut d'abord désactiver le swap :

```
root@ub20:~# swapoff -a
root@ub20:~# cat /etc/fstab
UUID=25b1f4e7-2f6b-4cea-912d-1ca4 /  ext4  errors=remount-ro 0 1
# swap was on /dev/sda2 during installation
#UUID=5cd8ac14-f01e-4f24-9a0c-d703 none  swap  sw   0  0
```

On change le nom du serveur :

```
root@ub20:~# hostname maitre
root@ub20:~# hostname
lemaitre
root@ub20:~# hostname > /etc/hostname
root@ub20:~# hostname -I
10.3.134.130
root@ub20:~# cat /etc/hosts
10.3.134.130 maitre
10.3.134.150 noeud1
10.3.134.170 noeud2
```

On installe ensuite Docker si ce n'est déjà fait :

```
root@maitre:~# apt update
root@maitre:~# apt install apt-transport-https curl
root@maitre:~# apt install docker.io
```

Il faut créer un fichier de configuration spécifique pour changer le « cgroup » de Docker :

```
root@maitre:~# vi /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
root@maitre:~# systemctl restart docker
```

Puis on installe le dépôt Kubernetes à partir du site officiel :

```
root@maitre:~# curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
OK
root@maitre:~# apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"
root@maitre:~# apt update
```

On installe maintenant les paquets nécessaires :

```
root@maitre:~# apt install kubelet=1.25.5-00 kubeadm=1.25.5-00 kubectl=1.25.5-00
```

```
root@maitre:~# kubeadm config images pull
```

On initialise, il faut cependant que le nœud maître possède **2 cpus** :

```
root@maitre:~# kubeadm init --pod-network-cidr=10.244.0.0/16
Your Kubernetes master 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

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/

You can now join any number of machines by running the following on each
node as root:
     kubeadm  join  10.3.134.130:6443  --token  y3hhf7.29iiw6b1vd5g8gde  --
discovery-token-ca-cert-hash
sha256:0bc51e66c08f569deaa0208a2135a9b1ccc55a820a7ee5b207cbdad7792159af
```

Si vous avez oublié le token ou si il a expiré, il est possible d'en générer un autre :

```
root@maitre:~# kubeadm token create --print-join-command
```

La commande « kubeadm » utilise l'interface réseau associée avec la passerelle par défaut pour annoncer l'IP du master. Pour utiliser une autre interface réseau, spécifiez l'option –apiserver-advertise-address=<ip-address>

On réalise les opérations demandées

```
root@maitre:~# exit
inti@maitre:~$ mkdir -p $HOME/.kube
inti@maitre:~$ sudo cp -i /etc/kubernetes/admin.conf
$HOME/.kube/config
inti@maitre:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Alternativement, si vous êtes « root » et que vous voulez travailler en tant que tel, vous pouvez exécuter :

```
root@maitre:~# export KUBECONFIG=/etc/kubernetes/admin.conf
```

On configure la couche réseau en utilisant le plugin « flannel » :

```
root@maitre:~# kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Document
ation/kube-flannel.yml
root@maitre:~# systemctl restart kubelet.service
root@maitre:~# systemctl status kubelet.service
```

On vérifie les nœuds de disponibles :

```
root@maitre:~# kubectl get nodes
NAME       STATUS   ROLES    AGE     VERSION
maitre   Ready    master   8m50s   v1.12.2
```

Par défaut, votre cluster ne déploie pas de pods sur le master pour des raisons de sécurité. Si vous souhaitez pouvoir déployer des pods sur le master, par exemple, pour un cluster Kubernetes mono-machine pour le développement, exécutez:

```
root@maitre:~$ kubectl taint nodes maitre "node-role.kubernetes.io/control-plane:NoSchedule-"
```

Avec un résultat ressemblant à quelque chose comme:

```
node/maitre untainted
```

3 - Nœuds esclaves

Il faut d'abord désactiver le swap :

```
root@ub20:~# swapoff -a
root@ub20:~# cat /etc/fstab
UUID=25b1f4e7-2f6b-4cea-912d-1ca4 /  ext4  errors=remount-ro 0 1
# swap was on /dev/sda2 during installation
#UUID=5cd8ac14-f01e-4f24-9a0c-d703 none  swap  sw   0  0
```

On change également le « hostname » :

```
root@ub20:~# hostname noeud1
root@ub20:~# hostname
noeud1
root@ub20:~# hostname > /etc/hostname
root@ub20:~# hostname -I
10.3.134.150
root@ub20:~# cat /etc/hosts
10.3.134.130 maitre
10.3.134.150 noeud1
10.3.134.170 noeud2
```

→ déconnexion puis reconnexion

On installe ensuite Docker si ce n'est déjà fait :

```
root@noeud1:~# apt update
root@noeud1:~# apt install apt-transport-https curl
root@noeud1:~# apt install docker.io
```

Il faut créer un fichier de configuration spécifique pour changer le « cgroup » de Docker :

```
root@noeud1:~# vi /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
root@noeud1:~# systemctl restart docker
```

Puis on installe le dépôt Kubernetes à partir du site officiel :

```
root@noeud1:~#  curl  -s  https://packages.cloud.google.com/apt/
doc/apt-key.gpg | apt-key add -
OK
root@noeud1:~# apt-add-repository
        "deb http://apt.kubernetes.io/ kubernetes-xenial main"
root@noeud1:~# apt update
```

On installe les paquets :

```
root@noeud1:~#  apt  install  kubelet=1.25.5-00  kubeadm=1.25.5-00
kubectl=1.25.5-00
```

On peut maintenant joindre le nœud maître :

```
root@noeud1:~#    kubeadm    join    10.3.134.130:6443    --token
y4jsf7.29aj7id3bcj48gde          --discovery-token-ca-cert-hash
sha256:0bc51e66c08f569ad832f23a9135a9b1ccc55a820a7ee5b207cbdad77
92159af
...

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 master to see this node join the
cluster.

root@noeud1:~#
```

On réalise la même opération sur le noeud2 et les tous les autres nœuds de travail.

4 - Commandes de gestion

On peut tout d'abord vérifier la listes de nœuds :

```
root@maitre:~# kubectl get nodes
NAME       STATUS   ROLES    AGE   VERSION
maitre     Ready    master   30m   v1.12.2
noeud1     Ready    <none>   13m   v1.12.2
noeud2     Ready    <none>   23s   v1.12.2
```

Si un problème est détecté, il est possible de consulter les événements :

```
root@maitre:~# kubectl get events
LAST SEEN   TYPE     REASON                    KIND   MESSAGE
...
15m           Normal    NodeReady                 Node    Node
noeud2 status is now: NodeReady
```

Si l'on souhaite obtenir des informations détaillées sur un nœud :

```
root@maitre:~# kubectl describe nodes maitre
Name:               maitre
...
```

5 - Connexion

Il faut d'abord en tant que « root » se connecter sur le docker hub :

```
root@master:~# docker login -u login
Password: XXXXX
```

Ce qui va normalement créer un fichier nommé config.json :

```
root@master:~# cat .docker/config.json
{
        "auths": {
                "https://index.docker.io/v1/": {
                        "auth": "cmDLK45JDKC88341"
                }
        }
}
```

Il n'y a plus qu'à l'utiliser dans Kubernetes :

```
root@master:~# kubectl create secret generic regcred --from-file=.dockerconfigjson=/root/.docker/config.json --type=kubernetes.io/dockerconfigjson
```

ou

```
kubectl create secret docker-registry regcred2 --docker-server=https://index.docker.io/v1/ --docker-username=login --docker-password=mypass --docker-email=moi@u13.org
```

6 - Description de Kubernetes

Kubernetes comporte les objets suivants :

- **Pod** : C'est la plus petite unité manipulable par Kubernetes, elle peut contenir un ou plusieurs conteneurs, mais une unique IP, un seul ensemble de ressources, des options spécifiques, … Tous les conteneurs d'un même Pod s'exécutent sur un même nœud, pour visualiser ceux actifs :

```
root@maitre:~# kubectl get pod
```

- **Service** : Il permet de définir une méthode d'accès distant aux pods,

```
root@maitre:~# kubectl get service
```

- **Volume** : C'est un espace de stockage,
- **Namespace** : Espace virtuel du Cluster permettant d'isoler des composants,

```
root@maitre:~# kubectl get namespace
```

Kubernetes dispose aussi de contrôleurs :

- **ReplicaSet** : Pour gérer et maintenir un ensemble de Pods, pas nécessaire en général car ils le sont par l'intermédiaire des « Deployment »,

```
root@maitre:~# kubectl get rs
```

- **Deployment** : C'est l'instance la plus utilisée, qui permet de déployer des pods sur le cluster,

```
root@maitre:~# kubectl get deployment
```

- **StatefulSet** : Comme un Deployment mais permet de gérer des ressources persistantes,

```
root@maitre:~# kubectl get statefulset
```

- **DaemonSet** : Permet de gérer des instances uniques sur chaque nœud,

```
root@maitre:~# kubectl get daemonset
```

- **Job** : Qui permet de lancer une collection de Pods identiques et de s'assurer qu'un nombre définit se termine,

```
root@maitre:~# kubectl get job
```