

LAB 1-2 : Installer Un Cluster Kubernetes Sur Ubuntu

Conditions préalables

- Plusieurs serveurs Linux exécutant Ubuntu (1 nœud principal, plusieurs nœuds de travail)
- Un compte utilisateur sur chaque système avec des privilèges `sudo` ou `root`
- Le gestionnaire de paquets `apt`, inclus par défaut
- Docker-CE installé sur tous les nœuds

1. Étapes d'installation de Kubernetes sur Ubuntu

Pour utiliser Kubernetes, vous devez installer un moteur de conteneurisation. Actuellement, la solution de conteneur la plus populaire est Docker. Docker doit être installé sur Ubuntu, à la fois sur le nœud maître et sur les nœuds de travail.

Étape 1: configurer le référentiel Kubernetes

Les packages Kubernetes ne sont pas disponibles à partir des référentiels officiels d'Ubuntu. Cette étape doit être effectuée sur le **nœud maître** et sur **chaque nœud de travail** (workers) que vous prévoyez d'utiliser pour la configuration de votre conteneur.

La commande suivante permet d'installer `curl` sur Ubuntu.

```
# sudo apt-get install curl
```

Entrez la commande suivante pour récupérer la clé de signature des paquets.

```
# curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add
```

Entrez la commande suivante pour ajouter les référentiels Kubernetes sous Ubuntu .

```
# sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"
```

Étape 2: Installez kubelet , beadm et kubectl

Ces 3 packages de base sont nécessaires pour pouvoir utiliser Kubernetes. Installez le (s) package (s) suivant (s) sur chaque nœud:

Installez les outils Kubernetes avec la commande.

```
# sudo apt-get install -y kubelet=1.21.3-00 kubeadm=1.21.3-00 kubectl=1.21.3-00
# systemctl enable kubelet
# systemctl start kubelet
```

Vérifier l'installation :

```
# kubeadm version
```

Avant de déployer un cluster, assurez-vous de définir les noms d'hôte, de configurer le pare-feu et les paramètres du noyau.

Étape 3: Définir le nom d'hôte sur les nœuds

Pour donner un nom d'hôte unique à chacun de vos nœuds, utilisez cette commande:

Sur le **nœud Master**

```
# hostnamectl set-hostname Master-Node
```

Sur les nœuds workers

```
# hostnamectl set-hostname Worker-Node-1
```

Créez une entrée d'hôte ou un enregistrement DNS pour résoudre le nom d'hôte de tous les nœuds:

```
# nano /etc/hosts
```

Avec l'entrée:

```
IP-Serveur Master-Node
IP-Worker1 Worker-Node-1
...
```

Étape 4: Configurer le pare-feu

Les nœuds, les conteneurs et les pods doivent pouvoir communiquer à travers le cluster pour exécuter leurs fonctions. Le Firewall est activé par défaut dans Ubuntu sur le front-end.

Ajoutez les ports suivants en entrant les commandes répertoriées.

Sur le **nœud maître** ainsi que les workers, entrez :

```
# sudo ufw allow 6443/tcp
# sudo ufw allow 2379:2380/tcp
# sudo ufw allow 10248/tcp
# sudo ufw allow 10250/tcp
# sudo ufw allow 10251/tcp
# sudo ufw allow 10252/tcp
# sudo ufw allow 10255/tcp
# sudo ufw allow 6783/tcp
# sudo ufw allow 6783/udp
# sudo ufw allow 6784/udp
```

Étape 5: Mettre à jour les paramètres Iptables

Définissez `net.bridge.bridge-nf-call-iptables` sur '1' dans votre fichier de configuration `sysctl`.

Cela garantit que les paquets sont correctement traités par les tables IP pendant le filtrage et la redirection de port.

```
# cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
# sysctl --system
```

Étape 6: Désactivez SWAP

Enfin, nous devons désactiver SWAP pour permettre au kubelet de fonctionner correctement:

```
# sed -i '/swap/d' /etc/fstab
# swapoff -a
```

2. Déployer un cluster Kubernetes

Étape 1: créer un cluster avec kubeadm

Sur le nœud Master uniquement :

Initialisez un cluster en exécutant la commande suivante:

```
# kubeadm init --pod-network-cidr=10.244.0.0/16
```

Le processus peut prendre plusieurs minutes en fonction de la vitesse du réseau. Une fois cette commande terminée, elle affiche un message de jointure kubeadm. Prenez note de l'entrée et utilisez-la pour joindre les nœuds worker au cluster à un stade ultérieur.

```
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

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/

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

kubeadm join 192.168.118.153:6443 --token cridr3.hwkoxrflrggvabyh \
--discovery-token-ca-cert-hash sha256:716f04e56bc5e4b04a617ef121697af755cd3b02f412a213b2a2d80a6f2491
```

```
# export KUBECONFIG=/etc/kubernetes/admin.conf
```

Ajouter la ligne précédente à la fin de fichier .bashrc pour charger la configuration de cluster si on redemarre la machine Master-Node

```
# nano $HOME/.bashrc
→ Ajouter la ligne : export KUBECONFIG=/etc/kubernetes/admin.conf
```

Étape 2 : configurer le réseau de pods (Sur Master-Node)

Un réseau de pods permet aux nœuds du cluster de communiquer. Il existe plusieurs options de mise en réseau Kubernetes disponibles. Utilisez la commande suivante pour installer le module complémentaire réseau Weave Net :

```
# kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl
version | base64 | tr -d '\n')"
```

Si vous décidez d'utiliser Weave Net, modifiez vos règles de pare-feu pour autoriser le trafic pour les ports TCP 6783 et UDP 6783/6784.

Étape 3 : Joindre le nœud de travail au cluster (Sur les Worker-Node)

Comme indiqué à l'étape 1, vous pouvez utiliser la commande `kubeadm join` sur chaque nœud de travail pour le connecter au cluster.

```
kubeadm join ...
```

```
[root@node2 ~]# kubeadm join 192.168.118.153:6443 --token cridr3.hwkoxrflrggvabyh \
> --discovery-token-ca-cert-hash sha256:716f04e56bc5e4b04a617ef121697af755cd3b02f412a213b2a2a2d80a6f2491
[preflight] Running pre-flight checks
[WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The recommended driver is "systemd". Please follow the guide at https://kubernetes.io/docs/setup/cri/
[WARNING SystemVerification]: this Docker version is not on the list of validated versions: 20.10.1. Latest validated version: 19.03
[WARNING Hostname]: hostname "worker-node1" could not be reached
[WARNING Hostname]: hostname "worker-node1": lookup worker-node1 on 192.168.118.2:53: no such host
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiservert and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

En cas de besoin, Régénérer la chaîne de connexion des machines :

```
kubeadm token create --print-join-command
```

Étape 4 : Vérifier l'état du cluster

Vérifiez l'état des nœuds en entrant la commande suivante sur le serveur maître :

```
kubectl get nodes
```

```
[root@master-node ~]# kubectl get nodes
NAME           STATUS    ROLES          AGE    VERSION
master-node    Ready     control-plane,master  120m   v1.20.1
worker-node1   Ready     <none>         70m    v1.20.1
worker-node2   Ready     <none>         58m    v1.20.1
```

Une fois qu'un réseau de pod a été installé, vous pouvez confirmer qu'il fonctionne en vérifiant que le pod CoreDNS est en cours d'exécution en tapant:

```
kubectl get pods --all-namespaces
```

```
[root@master-node ~]# kubectl get pods --all-namespaces
NAMESPACE   NAME                                     READY   STATUS             RESTARTS   AGE
kube-system  coredns-74ff55c5b-268jz                0/1     ContainerCreating   0          121m
kube-system  coredns-74ff55c5b-52p2t                0/1     ContainerCreating   0          121m
kube-system  etcd-master-node                       1/1     Running            0          121m
kube-system  kube-apiserver-master-node             1/1     Running            0          121m
kube-system  kube-controller-manager-master-node    1/1     Running            0          121m
kube-system  kube-flannel-ds-6pp94                  0/1     CrashLoopBackOff    16         60m
kube-system  kube-flannel-ds-mb776                  0/1     CrashLoopBackOff    19         75m
kube-system  kube-flannel-ds-tpd22                  0/1     CrashLoopBackOff    18         72m
kube-system  kube-proxy-fl8s6                       1/1     Running            0          121m
kube-system  kube-proxy-gzb9l                       1/1     Running            0          72m
kube-system  kube-proxy-j74bj                       1/1     Running            0          60m
kube-system  kube-scheduler-master-node             1/1     Running            0          121m
[root@master-node ~]#
```