**Comment installer le cluster Kubernetes sur Debian 12 | 11**

**Conditions préalables**

* Debian 12/11 minimalement installée
* 2 processeurs/processeurs virtuels
* 2 Go de RAM
* 20 Go d'espace disque libre
* Utilisateur Sudo avec droits d'administrateur
* Connectivité Internet stable
* Assurez-vous que chaque nœud peut communiquer avec les autres via une connexion réseau fiable.

Pour la démonstration, j'utilise trois systèmes Debian 12.

Nœud maître (k8s-master) – 192.168.1.23

Nœud de travail 1 (k8s-worker01) – 192.168.1.24

Nœud de travail 2 (k8s-worker02) – 192.168.1.25

1) Définir le nom d'hôte et mettre à jour le fichier d'hôtes

Connectez-vous à chaque nœud (nœuds maître et woker) et définissez leur nom d'hôte à l'aide de la commande hostnamectl.

$ sudo hostnamectl set-hostname "k8s-master.linuxtechi.local" // Exécuté sur le nœud maître

$ sudo hostnamectl set-hostname "k8s-worker01.linuxtechi.local" // Exécuté sur le 1er nœud de travail

$ sudo hostnamectl set-hostname "k8s-worker02.linuxtechi.local" // Exécuté sur le 2ème nœud de travail

Ajoutez également les entrées suivantes dans le fichier /etc/hosts sur tous les nœuds,

192.168.1.23 k8s-master.linuxtechi.local k8s-master

192.168.1.24 k8s-worker01.linuxtechi.local k8s-worker01

192.168.1.25 k8s-worker02.linuxtechi.local k8s-worker02

2) Désactiver l'échange sur tous les nœuds

Pour que kubelet fonctionne correctement, il est recommandé de désactiver le swap. Exécutez les commandes suivantes sur les nœuds maître et travailleur pour désactiver le swap.

$ sudo swapoff -a

$ sudo sed -i '/ swap / s/^\(.\*\)$/#\1/g' /etc/fstab

3) Ajouter des règles de pare-feu pour le cluster Kubernetes

Si le pare-feu du système d'exploitation est activé sur vos systèmes Debian, autorisez respectivement les ports suivants sur les nœuds maître et travailleur.

Sur le nœud maître, exécutez

$ sudo ufw autorise 6443/tcp

$ sudo ufw autorise 2379/tcp

$ sudo ufw autorise 2380/tcp

$ sudo ufw autorise 10250/tcp

$ sudo ufw autorise 10251/tcp

$ sudo ufw autorise 10252/tcp

$ sudo ufw autorise 10255/tcp

$ sudo ufw recharger

Sur les nœuds de travail,

$ sudo ufw autorise 10250/tcp

$ sudo ufw autorise 30000:32767/tcp

$ sudo ufw recharger

Remarque : Si le pare-feu est désactivé sur vos systèmes Debian 12/11, vous pouvez ignorer cette étape.

4) Installez le temps d'exécution de Containerd sur tous les nœuds

Containerd est le temps d'exécution de conteneur standard de l'industrie et pris en charge par Kubernetes. Alors, installez containersd sur tous les nœuds maîtres et travailleurs.

Avant d'installer containersd, définissez les paramètres de noyau suivants sur tous les nœuds.

$ chat <<EOF | sudo tee /etc/modules-load.d/containerd.conf

recouvrir

br\_netfilter

EOF

$ superposition sudo modprobe

$ sudo modprobe br\_netfilter

$ chat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-k8s.conf

net.bridge.bridge-nf-call-iptables = 1

net.ipv4.ip\_forward = 1

net.bridge.bridge-nf-call-ip6tables = 1

EOF

Pour apporter les modifications ci-dessus à l'effet, exécutez

$ sudo sysctl --système

Maintenant, installez conatinerd en exécutant [la commande apt](https://www.linuxtechi.com/apt-command-ubuntu-debian-linux/) suivante sur tous les nœuds.

$ sudo apt mise à jour

$ sudo apt -y installer conteneur

Ensuite, configurez containers pour qu'il fonctionne avec Kubernetes, exécutez sous la commande sur tous les nœuds

$ configuration par défaut de conteneurd | sudo tee /etc/containerd/config.toml >/dev/null 2>&1

Définissez cgroupdriver sur systemd sur tous les nœuds,

Modifiez le fichier ' /etc/containerd/config.toml ' et recherchez la section ' [plugins.”io.containerd.grpc.v1.cri”.containerd.runtimes.runc.options] ' et modifiez ' SystemdCgroup = false' à 'SystemdCgroup = vrai '

$ sudo vi /etc/containerd/config.toml

Enregistrez et quittez le fichier.

Redémarrez et activez le service containersd sur tous les nœuds,

$ sudo systemctl redémarrer le conteneur

$ sudo systemctl activer conteneurd

5) Ajouter le référentiel Kubernetes Apt

Dans Debian 12/11, les packages liés à Kubernetes ne sont pas disponibles dans les référentiels de packages par défaut. Nous devons ajouter un référentiel Kubernetes apt supplémentaire sur tous les nœuds, exécutez

$ sudo apt install gnupg gnupg2 curl software-properties-common -y

$ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/cgoogle.gpg

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

6) Installer les outils Kubernetes

Ensuite, installez les outils Kubernetes, notamment kubeadm, kubelet et kubectl sur tous les nœuds.

$ sudo apt mise à jour

$ sudo apt install kubelet kubeadm kubectl -y

$ sudo apt-mark hold kubelet kubeadm kubectl

7) Installez le cluster Kubernetes avec Kubeadm

Maintenant, nous sommes tous prêts à initialiser le cluster Kubernetes, exécutez la commande suivante uniquement à partir du nœud maître,

$ sudo kubeadm init --control-plane-endpoint=k8s-master

Pour commencer à interagir avec le cluster, exécutez les commandes suivantes sur le nœud maître,

$ mkdir -p $HOME/.kube

$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

$ sudo chown $(id -u):$(id -g) $HOME/.kube/config

Exécutez la commande kubectl suivante pour obtenir des informations sur les nœuds et le cluster,

$ kubectl récupère les nœuds

$ kubectl cluster-info

Sortie des commandes ci-dessus,

Sur vos nœuds de travail, joignez-les au cluster en exécutant la commande affichée lors de l'initialisation du nœud maître. Cela ressemblera à ceci : « Rejoindre Kubeadm »

Remarque : copiez la commande exacte à partir de la sortie de la commande "kubeadm init". Dans mon cas, voici la commande

$ sudo kubeadm join k8s-master:6443 --token 21nm87.x1lgd4jf0lqiiiau \

--discovery-token-ca-cert-hash sha256:28b503f1f2a2592678724c482776f04b445c5f99d76915552f14e68a24b78009

Vérifiez l'état des nœuds en exécutant la commande suivante à partir du nœud maître,

$ kubectl récupère les nœuds

Pour préparer le statut des nœuds, nous devons installer des modules complémentaires de réseau POD comme Calico ou Flannel.

8) Configurer le réseau de pods à l'aide de Calico

Sur le nœud maître, exécutez la commande sous pour installer Calico,

$ kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.1/manifests/calico.yaml

Autorisez les ports Calico dans le pare-feu du système d'exploitation, exécutez sous les commandes ufw sur tous les nœuds,

$ sudo ufw autorise 179/tcp

$ sudo ufw autorise 4789/udp

$ sudo ufw autorise 51820/udp

$ sudo ufw autorise 51821/udp

$ sudo ufw recharger

Vérifiez l'état des pods Calico, exécutez

$ kubectl récupère les pods -n système kube

Très bien, ci-dessus confirme que les nœuds maître et travailleur sont prêts. Maintenant, nous pouvons dire que notre cluster est prêt à gérer la charge de travail.

9) Tester l'installation du cluster Kubernetes

Afin de valider et de tester l'installation du cluster Kubernetes, essayons de déployer une application basée sur nginx via le déploiement. Exécuter sous les commandes,

$ kubectl créer un déploiement nginx-app --image=nginx --replicas 2

$ kubectl expose le déploiement nginx-app --name=nginx-web-svc --type NodePort --port 80 --target-port 80

$ kubectl décrit svc nginx-web-svc

,

Essayez d'accéder à l'application basée sur nginx à l'aide [de la commande curl](https://www.linuxtechi.com/linux-curl-command-examples-part-1/) suivante avec le nodeport 32283.

Remarque : Dans la commande curl, nous pouvons utiliser l'un ou l'autre nom d'hôte du nœud de travail.

$ boucle http://k8s-worker01:32283