
Configurer un nœud Worker Kubernetes avec containerd sur Debian 12

Ajoute un nœud worker à un cluster Kubernetes existant avec containerd comme runtime sur Debian 12.

[Systèmes & Réseaux](#) [DevOps](#) 15 min de lecture [Niveau Intermédiaire](#)

Document généré le 25/05/2026 à 20h09 · nouv.fr/wiki/worker-kubernetes-debian12-containerd

Sommaire

12 section(s) · 15 min de lecture

1. Mise à jour du système et installation des outils de base

2. Installation de containerd

3. Désactivation du swap

4. Configuration des modules du noyau et paramètres réseau

5. Ajouter le dépôt Kubernetes et la clé GPG

6. Installer kubelet, kubeadm et kubectl

7. Rejoindre le cluster Kubernetes

↳ Générer la commande de jointure (depuis le nœud Master)

↳ Rejoindre le cluster (sur le nœud Worker)

8. Vérification du cluster (depuis le Master)

9. Dépannage — Flannel / CNI

Récapitulatif — Ordre d'installation

1. Mise à jour du système et installation des outils de base

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y curl gnupg2 apt-transport-https ca-certificates software-properties-
common
```

📄 Copier

2. Installation de containerd

```
sudo apt install -y containerd
sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml
sudo sed -i 's/SystemdCgroup = false/SystemdCgroup = true/' /etc/containerd/config.toml
sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd
```

📄 Copier

L'option `SystemdCgroup = true` est obligatoire pour que containerd s'intègre correctement avec kubelet via systemd.

3. Désactivation du swap

Le swap doit être désactivé — Kubernetes ne fonctionne pas correctement avec le swap actif.

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

📄 Copier

*La commande `sed` commente la ligne swap dans `/etc/fstab` pour que la désactivation soit **persistante après redémarrage**.*

4. Configuration des modules du noyau et paramètres réseau

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF

sudo modprobe overlay
sudo modprobe br_netfilter
```

📄 Copier

```
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward = 1
EOF

sudo sysctl --system
```

📄 Copier

Paramètre	Rôle
overlay	Système de fichiers en couches pour les conteneurs
br_netfilter	Permet au bridge réseau de passer par iptables
ip_forward	Active le routage IP entre les interfaces

5. Ajouter le dépôt Kubernetes et la clé GPG

```
sudo mkdir -p /etc/apt/keyrings

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.32/deb/Release.key |
  sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.32/deb/ /" |
sudo tee /etc/apt/sources.list.d/kubernetes.list
```

📄 Copier

6. Installer kubelet, kubeadm et kubectl

```
sudo apt update
sudo apt install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

📄 Copier

apt-mark hold empêche une mise à jour automatique des composants Kubernetes, ce qui pourrait casser le cluster.

7. Rejoindre le cluster Kubernetes

Générer la commande de jointure (depuis le nœud Master)

Sur le **nœud Master**, exécuter la commande suivante pour générer la commande de jointure :

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

📋 Copier

Cela retourne une commande du type :

```
kubeadm join 192.168.1.100:6443 --token abcdef.0123456789abcdef  
--discovery-token-ca-cert-hash sha256:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

📋 Copier

Rejoindre le cluster (sur le nœud Worker)

Sur **chaque nœud Worker**, exécuter la commande fournie par le master :

```
sudo kubeadm join 192.168.1.100:6443 --token abcdef.0123456789abcdef  
--discovery-token-ca-cert-hash sha256:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

📋 Copier

Remplacer l'IP, le token et le hash par les valeurs réelles retournées par le master.

8. Vérification du cluster (depuis le Master)

```
kubectl get nodes
```

📋 Copier

Résultat attendu :

NAME	STATUS	ROLES	AGE	VERSION
master	Ready	control-plane	5m	v1.32.0
worker-1	Ready	<none>	2m	v1.32.0
worker-2	Ready	<none>	2m	v1.32.0

✂ Copier

Si un worker reste en NotReady, vérifier que containerd tourne bien (`systemctl status containerd`) et que le swap est bien désactivé (`free -h`).

9. Dépannage — Flannel / CNI

Si le nœud worker reste en `NotReady` après le join ou que les pods Flannel sont en `CrashLoopBackOff`, c'est souvent lié à un chemin CNI manquant. Créer le lien symbolique suivant **sur le worker** :

```
sudo ln -s /opt/cni/bin /usr/lib/cni
```

✂ Copier

Puis redémarrer les pods Flannel depuis le **master** :

```
kubectl rollout restart daemonset/kube-flannel-ds -n kube-flannel
kubectl get pods -n kube-flannel
```

✂ Copier

Récapitulatif — Ordre d'installation

Étape	Action	Où
1	Mettre à jour le système et installer les dépendances	Worker
2	Installer et configurer containerd (<code>SystemdCgroup = true</code>)	Worker
3	Désactiver le swap (temporaire + persistant via fstab)	Worker
4	Charger les modules noyau et paramètres réseau	Worker
5	Ajouter le dépôt Kubernetes v1.32	Worker
6	Installer kubelet, kubeadm, kubectl (avec hold)	Worker
7	Générer la commande de jointure	Master
8	Exécuter <code>kubeadm join</code>	Worker
9	Vérifier <code>kubectl get nodes → STATUS Ready</code>	Master
10	Si Flannel KO : <code>sudo ln -s /opt/cni/bin /usr/lib/cni</code>	Worker