

Vagrantfile, scripts et manifestes Kubeadm

I. Étapes pour configurer et utiliser Vagrant

Installer Vagrant

Téléchargez et installez Vagrant (Windows 64 bits) depuis le site officiel : <https://www.vagrantup.com/>.

Créez un dossier sur votre machine pour contenir vos fichiers Vagrant.

Initialiser Vagrant dans le dossier :

Ouvrez une invite de commande ou un terminal, accédez au dossier que vous venez de créer, et exécutez la commande suivante : **vagrant init**

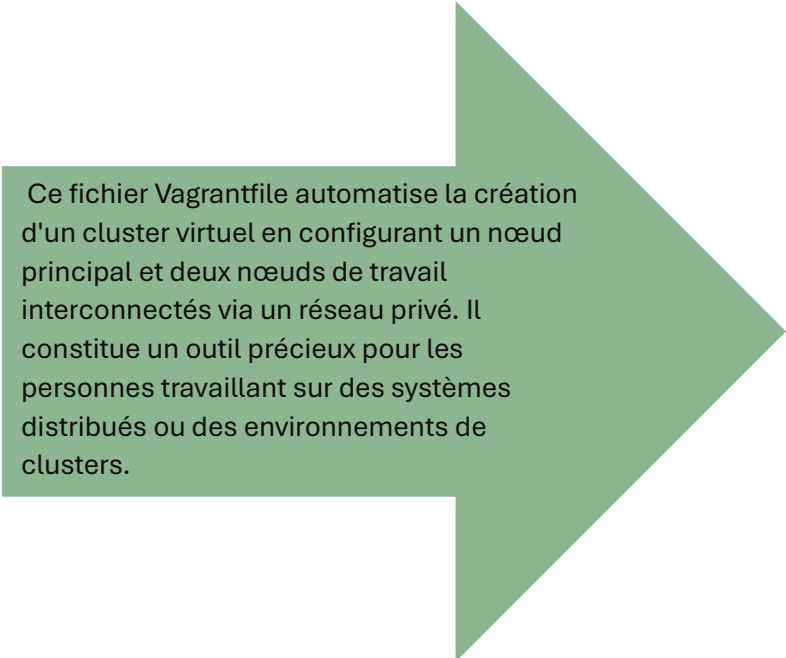
Ouvrir Visual Studio Code :

Une fois le fichier Vagrantfile généré, ouvrez Visual Studio Code à l'intérieur du dossier en utilisant la commande : **code .**

Remplacez le contenu existant du fichier par le nouveau code approprié.

Lancer Vagrant :

Après avoir sauvegardé les modifications, exécutez la commande suivante pour créer et configurer vos machines virtuelles : **vagrant up**



Ce fichier Vagrantfile automatise la création d'un cluster virtuel en configurant un nœud principal et deux nœuds de travail interconnectés via un réseau privé. Il constitue un outil précieux pour les personnes travaillant sur des systèmes distribués ou des environnements de clusters.

```
Vagrant.configure("2") do |config|

  config.vm.provision "shell", inline: <<-SHELL

    apt-get update -y

    echo "10.0.0.10  master-node" >> /etc/hosts

    echo "10.0.0.11  worker-node01" >> /etc/hosts

    echo "10.0.0.12  worker-node02" >> /etc/hosts

  SHELL

  config.vm.define "master" do |master|

    master.vm.box = "bento/ubuntu-22.04"

    master.vm.hostname = "master-node"

    master.vm.network "private_network", ip: "10.0.0.10"

    master.vm.provider "virtualbox" do |vb|

      vb.memory = 4048

      vb.cpus = 2

    end

  end

  (1..2).each do |i|

    config.vm.define "node0#{i}" do |node|

      node.vm.box = "bento/ubuntu-22.04"

      node.vm.hostname = "worker-node0#{i}"

      node.vm.network "private_network", ip: "10.0.0.1#{i}"

      node.vm.provider "virtualbox" do |vb|

        vb.memory = 2048

        vb.cpus = 1

      end

    end

  end

end
```

II. Configuration du cluster Kubernetes à l'aide de Kubeadm

1. Étape 1 : Activer le trafic ponté iptables sur tous les nœuds

Exécutez les commandes suivantes sur tous les nœuds pour que les tables IP voient le trafic ponté. Ici, nous ajustons certains paramètres du noyau et les définissons à l'aide de sysctl

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

sudo modprobe overlay
sudo modprobe br_netfilter

# sysctl params required by setup, params persist across reboots
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

# Apply sysctl params without reboot
sudo sysctl --system
```

2. Étape 2 : Désactiver le swap sur tous les nœuds

Pour que kubeadm fonctionne correctement, vous devez désactiver le swap sur tous les nœuds à l'aide de la commande suivante.

```
sudo swapoff -a
(crontab -l 2>/dev/null; echo "@reboot /sbin/swapoff -a") | crontab - || true
```

L'fstab entrée garantira que l'échange est désactivé lors des redémarrages du système.

```
* Applying /etc/sysctl.conf ...
vagrant@worker-node02:~$ sudo vim /etc/fstab
```

```
# /boot was on /dev/sda2 during curtin installation
/dev/disk/by-uuid/5e5081dd-7961-4507-a56b-471be7d54906 /boot ext4 defaults 0 1
#/swap.img none swap sw 0 0
#VAGRANT-BEGIN
# The contents below are automatically generated by Vagrant. Do not modify.
vagrant /vagrant vboxsf uid=1000,gid=1000,_netdev 0 0
#VAGRANT-END
~
```

3. Étape 3 : installer CRI-O Runtime sur tous les nœuds

Nous utiliserons CRI-O au lieu de [Docker](#) pour cette configuration car [Kubernetes a déconseillé le moteur Docker](#)

Exécutez les commandes suivantes **sur tous les nœuds** pour installer les dépendances requises et la dernière version de CRIO

```
sudo apt-get update -y
sudo apt-get install -y software-properties-common gpg curl apt-transport-https
ca-certificates
```

```
curl -fsSL https://pkgs.k8s.io/addons/cri-o/prerelease/main/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/cri-
o-apt-keyring.gpg
```

```
echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addons/cri-o/prerelease/main/deb//" |
sudo tee /etc/apt/sources.list.d/cri-o.list
```

```
sudo mkdir -p /etc/apt/keyrings
sudo chmod 755 /etc/apt/keyrings
```

```
echo "KUBELET_EXTRA_ARGS=--node-ip=$local_ip" | sudo tee /etc/default/kubelet
```

```
sudo apt-get update -y
sudo apt-get install -y cri-o

sudo systemctl daemon-reload
sudo systemctl enable crio --now
sudo systemctl start crio.service
```

Installez crictl.

crictl, un utilitaire CLI pour interagir avec les conteneurs créés par l'environnement d'exécution du conteneur.

```
VERSION="v1.30.0"
wget https://github.com/kubernetes-sigs/cri-
tools/releases/download/$VERSION/crictl-$VERSION-linux-amd64.tar.gz
sudo tar zxvf crictl-$VERSION-linux-amd64.tar.gz -C /usr/local/bin
rm -f crictl-$VERSION-linux-amd64.tar.gz
```

4. Étape 4 : installer Kubeadm, Kubelet et Kubectl sur tous les nœuds

Téléchargez la clé GPG pour le référentiel Kubernetes APT **sur tous les nœuds**.

```
KUBERNETES_VERSION=1.30
```

```
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://pkgs.k8s.io/core:/stable:/v${KUBERNETES_VERSION}/deb/Release.key
| sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v${KUBERNETES_VERSION}/deb/ /" | sudo tee
/etc/apt/sources.list.d/kubernetes.list
```

Mettre à jour le dépôt apt

```
sudo apt-get update -y
```

Vous pouvez utiliser les commandes suivantes pour rechercher les dernières versions. Installez la première version 1.30 afin de pouvoir vous entraîner à la mise à niveau du cluster.

```
apt-cache madison kubeadm | tac
```

Spécifiez la version comme indiqué ci-dessous. Ici, j'utilise **1.30.0-1.1**

```
sudo apt-get install -y kubelet=1.30.0-1.1 kubectl=1.30.0-1.1 kubeadm=1.30.0-1.1
```

```
sudo apt-mark hold kubelet kubeadm kubectl
```

Nous disposons désormais de tous les utilitaires et outils nécessaires pour configurer les composants Kubernetes à l'aide de kubeadm.

Ajoutez l'IP du nœud à KUBELET_EXTRA_ARGS.

! Remarque : Avant d'exécuter le script, vérifiez quelle interface réseau (par exemple, eth0, ens33, ou autre) est associée à l'adresse IP de votre machine. Utiliser la commande `ip addr`

```
sudo apt-get install -y jq
local_ip="$(ip --json addr show eth0 | jq -r '.[0].addr_info[] | select(.family
== "inet") | .local')"
cat > /etc/default/kubelet << EOF
KUBELET_EXTRA_ARGS=--node-ip=$local_ip
EOF
```

5. Étape 5 : Initialiser Kubeadm sur le nœud maître pour configurer le plan de contrôle

! Remarque : Cette configuration est spécifique au Nœud maître car c'est lui qui nécessite la définition explicite de l'IP pour gérer les nœuds de travail dans un cluster Kubernetes

Si vous utilisez une adresse IP privée pour le nœud maître :

Définissez les variables d'environnement suivantes. Remplacez-les 10.0.0.10 par l'adresse IP de votre nœud maître.

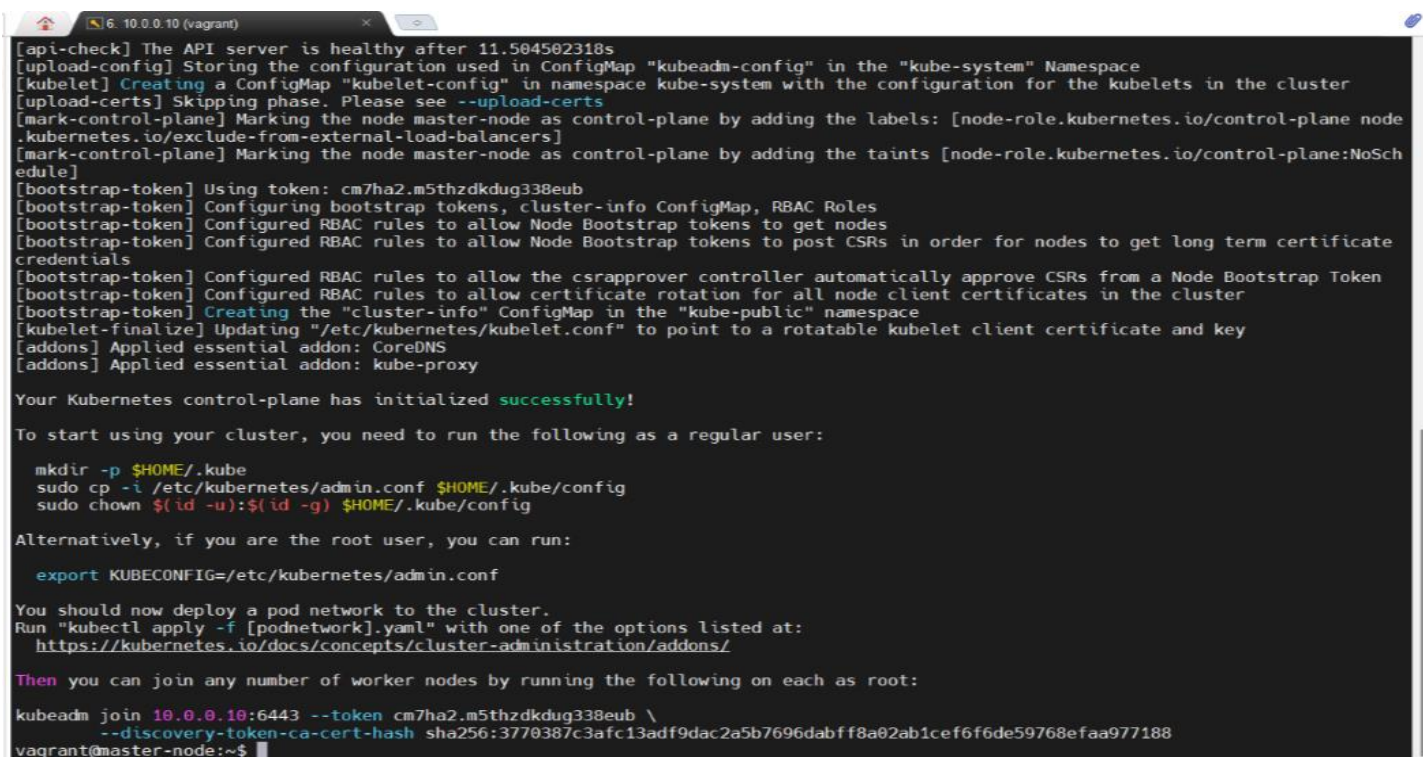
```
IPADDR="10.0.0.10"
NODENAME=$(hostname -s)
POD_CIDR="192.168.0.0/16"
```

Pour une **configuration basée sur une adresse IP privée**, utilisez la commande init suivante.

```
sudo kubeadm init --apiserver-advertise-address=$IPADDR --apiserver-cert-extras=$IPADDR --pod-network-cidr=$POD_CIDR --node-name $NODENAME --ignore-preflight-errors Swap
```

--ignore-preflight-errors Swap n'est en fait pas nécessaire car nous avons initialement désactivé l'échange.

Lors d'une initialisation réussie de kubeadm, vous devriez obtenir une sortie avec l'emplacement du fichier kubeconfig et la commande join avec le jeton comme indiqué ci-dessous. Copiez-le et enregistrez-le dans le fichier. Nous en aurons besoin pour joindre le nœud worker au master .



```
[api-check] The API server is healthy after 11.504502318s
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config" in namespace kube-system with the configuration for the kubelets in the cluster
[upload-certs] Skipping phase. Please see --upload-certs
[mark-control-plane] Marking the node master-node as control-plane by adding the labels: [node-role.kubernetes.io/control-plane node.kubernetes.io/exclude-from-external-load-balancers]
[mark-control-plane] Marking the node master-node as control-plane by adding the taints [node-role.kubernetes.io/control-plane:NoSchedule]
[bootstrap-token] Using token: cm7ha2.m5thzdkdug338eub
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to get nodes
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

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 10.0.0.10:6443 --token cm7ha2.m5thzdkdug338eub \
--discovery-token-ca-cert-hash sha256:3770387c3afc13adf9dca2a5b7696dabff8a02ab1cef6f6de59768efaa977188
vagrant@master-node:~$
```

Utilisez les commandes suivantes à partir de la sortie pour créer le kubeconfigmaster afin de pouvoir l'utiliser kubectl pour interagir avec l'API du cluster.

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Maintenant, vérifiez le kubeconfig en exécutant la commande kubectl suivante pour répertorier tous les pods de l' kube-systemspace de noms.

```
kubectl get po -n kube-system
```

```
unknown command "Swap" for "kubeadm init"
To see the stack trace of this error execute with --v=5 or higher
vagrant@master-node:~$ mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
vagrant@master-node:~$ kubectl get po -n kube-system
NAME                                READY   STATUS    RESTARTS   AGE
coredns-55cb58b774-7xt7w            0/1     Pending   0           21m
coredns-55cb58b774-jdn8j            0/1     Pending   0           21m
etcd-master-node                    1/1     Running   1           21m
kube-apiserver-master-node           1/1     Running   1           21m
kube-controller-manager-master-node 1/1     Running   1           21m
kube-proxy-f9f7g                     1/1     Running   1           21m
kube-scheduler-master-node           1/1     Running   1           21m
vagrant@master-node:~$
```

Vous vérifiez tous les états de santé des composants du cluster à l'aide de la commande suivante.

```
kubectl get --raw='/readyz?verbose'
```

```
vagrant@master-node:~$ kubectl get --raw='/readyz?verbose'
+ping ok
+log ok
+etcd ok
+etcd-readiness ok
+informer-sync ok
+poststarthook/start-apiserver-admission-initializer ok
+poststarthook/generic-apiserver-start-informers ok
+poststarthook/priority-and-fairness-config-consumer ok
+poststarthook/priority-and-fairness-filter ok
+poststarthook/storage-object-count-tracker-hook ok
+poststarthook/start-apiextensions-informers ok
+poststarthook/start-apiextensions-controllers ok
+poststarthook/crd-informer-synced ok
+poststarthook/start-service-ip-repair-controllers ok
+poststarthook/rbac/bootstrap-roles ok
+poststarthook/scheduling/bootstrap-system-priority-classes ok
+poststarthook/priority-and-fairness-config-producer ok
+poststarthook/start-system-namespaces-controller ok
+poststarthook/bootstrap-controller ok
+poststarthook/start-cluster-authentication-info-controller ok
+poststarthook/start-kube-apiserver-identity-lease-controller ok
+poststarthook/start-kube-apiserver-identity-lease-garbage-collector ok
+poststarthook/start-legacy-token-tracking-controller ok
+poststarthook/aggregator-reload-proxy-client-cert ok
+poststarthook/start-kube-aggregator-informers ok
+poststarthook/apiservice-registration-controller ok
+poststarthook/apiservice-status-available-controller ok
+poststarthook/apiservice-discovery-controller ok
+poststarthook/kube-apiserver-autoregistration ok
+autoregister-completion ok
+poststarthook/apiservice-openapi-controller ok
+poststarthook/apiservice-openapi-v3-controller ok
+shutdown ok
readyz check passed
vagrant@master-node:~$
```

Vous pouvez obtenir les informations du cluster à l'aide de la commande suivante.

```
kubectl cluster-info
```

```
vagrant@master-node:~$ kubectl cluster-info
Kubernetes control plane is running at https://10.0.0.10:6443
CoreDNS is running at https://10.0.0.10:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
vagrant@master-node:~$
```

Par défaut, les applications ne sont pas planifiées sur le nœud maître. Si vous **souhaitez utiliser le nœud maître pour planifier des applications**, altérez le nœud maître.

```
kubectl taint nodes --all node-role.kubernetes.io/control-plane-
```

6. Étape 6 : joindre les nœuds de travail au nœud maître Kubernetes

Nous avons également configuré les utilitaires **cri-o**, **kubelet** et **kubeadm** sur les **nœuds de travail**.

Maintenant, joignons le nœud de travail au nœud maître à l'aide de la commande **Kubeadm join** que vous avez obtenue dans la sortie lors de la configuration du nœud maître.

Si vous avez oublié de copier la commande join, exécutez la commande suivante dans le nœud maître pour recréer le jeton avec la commande join

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

Voici à quoi ressemble la commande. À utiliser sudo si vous exécutez en tant qu'utilisateur normal. Cette commande effectue l' [amorçage TLS](#) pour les nœuds.

```
sudo kubeadm join 10.128.0.37:6443 --token j4eice.33vgvyf5cxw4u8i \
--discovery-token-ca-cert-hash
sha256:37f94469b58bcc8f26a4aa44441fb17196a585b37288f85e22475b00c36f1c61
```

En cas d'exécution réussie, vous verrez le résultat indiquant : « Ce nœud a rejoint le cluster ».

```
sudo rm -rf /var/lib/kubelet/*
vagrant@worker-node01:~$ sudo kubeadm join 10.0.0.10:6443 --token mdwqw8.dnx2ftuu7cr3sei0 --discovery-token-ca-cert-hash sha256:427817994
4860b5d60538bf4f0d7420f34daa7ae02d7bd8bf82
[preflight] Running pre-flight checks
[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-check] Waiting for a healthy kubelet. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 1.026709653s
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

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 control-plane to see this node join the cluster.
vagrant@worker-node01:~$
```

Exécutez maintenant la **commande kubectl** à partir du **nœud maître** pour vérifier si le nœud est ajouté au maître.

```
kubectl get nodes
```

```
vagrant@master-node:~$ kubectl get node
NAME          STATUS  ROLES    AGE  VERSION
master-node   Ready   control-plane  130m  v1.30.0
worker-node01 Ready   <none>    69m  v1.30.0
worker-node02 Ready   <none>    72m  v1.30.0
vagrant@master-node:~$
```

7. Étape 7 : Installer le plug-in Calico Network pour la mise en réseau des pods

Kubeadm ne configure aucun plug-in réseau. Vous devez installer un plug-in réseau de votre choix pour la mise en réseau [des pods Kubernetes](#) et activer la politique réseau. J'utilise le plugin réseau Calico pour cette configuration.

Exécutez les commandes suivantes pour installer l'opérateur [de plug-in réseau Calico](#) sur le cluster.

```
kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
```

Après quelques minutes, si vous vérifiez les pods dans kube-system l'espace de noms, vous verrez des pods calico et des pods CoreDNS en cours d'exécution.

```
kubectl get po -n kube-system
```

```
error: Metrics API not available
vagrant@master-node:~$ kubectl get po -n kube-system
```

NAME	READY	STATUS	RESTARTS	AGE
calico-kube-controllers-5b9b456c66-xlpbv	0/1	ContainerCreating	0	28m
calico-node-5dm2z	1/1	Running	0	28m
calico-node-jvgwj	1/1	Running	0	28m
calico-node-sb4jj	1/1	Running	0	28m
coredns-7db6d8ff4d-kvhnd	1/1	Running	0	144m
coredns-7db6d8ff4d-kzhxh	1/1	Running	0	144m
etcd-master-node	1/1	Running	1	144m
kube-apiserver-master-node	1/1	Running	1	144m
kube-controller-manager-master-node	1/1	Running	3	144m
kube-proxy-q2tsb	1/1	Running	1	83m
kube-proxy-qhmv7	1/1	Running	1	144m
kube-proxy-vzjxv	1/1	Running	1	86m
kube-scheduler-master-node	1/1	Running	3	144m
metrics-server-6455f4d6f7-zgnnn	0/1	Running	0	5m38s

8. Étape 8 : Configurer le serveur de mesures Kubernetes

Kubeadm n'installe pas le composant [serveur de métriques](#) lors de son initialisation. Nous devons l'installer séparément.

Pour vérifier cela, si vous exécutez la commande top, vous verrez l' `Metrics API not available` erreur.

```
root@controlplane:~# kubectl top nodes

error: Metrics API not available
```

Pour installer le serveur de métriques, exécutez le fichier manifeste du serveur de métriques suivant. Il déploie la version du serveur de métriques v0.7.1

```
kubectl apply -f https://raw.githubusercontent.com/techiescamp/kubeadm-scripts/main/manifests/metrics-server.yaml
```

Ce manifeste est extrait du référentiel officiel du [serveur de mesures](#) . J'ai ajouté l' --kubelet-insecure-tls indicateur au conteneur pour le faire fonctionner dans la configuration locale et je l'ai hébergé séparément. Sinon, vous obtiendrez l'erreur suivante.

```
because it doesn't contain any IP SANs" node=""
```

Une fois les objets du serveur de métriques déployés, **il vous faut une minute** pour voir les métriques du nœud et du pod à l'aide de la commande top.

```
kubectl top nodes
```

Vous devriez pouvoir visualiser les métriques des nœuds comme indiqué ci-dessous.

```
vagrant@master-node:~$ kubectl top nodes
NAME           CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
master-node    205m         10%    1060Mi           28%
worker-node01  87m          8%     644Mi            34%
worker-node02  59m          5%     562Mi            30%
vagrant@master-node:~$
```

Vous pouvez également afficher les mesures du processeur et de la mémoire du pod à l'aide de la commande suivante.

```
kubectl top pod -n kube-system
```

```
vagrant@master-node:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-576c6b7b6-qpggl    1/1     Running   0           7m47s
nginx-deployment-576c6b7b6-smxkt    1/1     Running   0           7m47s
vagrant@master-node:~$
```

9. Étape 9 : Déployer un exemple d'application Nginx :

maintenant que nous avons tous les composants pour faire fonctionner le cluster et les applications, déployons un exemple d'application Nginx et voyons si nous pouvons y accéder via un NodePort

Créez un [déploiement](#) Nginx . Exécutez la commande suivante directement sur la ligne de commande. Elle déploie le pod dans l'espace de noms par défaut.

consulter le lien <https://devopscube.com/setup-kubernetes-cluster-kubeadm/> **étape 9**

après : Vérifiez l'état du pod à l'aide de la commande suivante.

```
kubectl get pods
```

Une fois le déploiement terminé, vous devriez pouvoir accéder à la page d'accueil de Nginx sur le NodePort alloué.

⚠ Non sécurisé 10.0.0.11:32000

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Référentiel: <https://devopscube.com/setup-kubernetes-cluster-kubeadm/>