

INSTALLATION ET MANIPULATION D'UN CLUSTER KUBERNETES AVEC MINIKUBE

Introduction

Pour les utilisateurs qui souhaitent essayer Kubernetes au quotidien, il existe une multitude de solutions dont la **solution Minikube**.

Minikube est un outil facilitant l'exécution locale de Kubernetes. Il exécute un cluster Kubernetes à nœud unique dans une machine virtuelle (VM), dès lors votre VM devient à la fois un nœud de type Master et Worker.

Cette partie du cours, vous explique **comment installer et manipuler l'outil Minikube**.

Prérequis

Hyperviseur

On va d'abord commencer par vérifier si la virtualisation est prise en charge sous notre machine Linux :

```
egrep --color 'vmx|svm' /proc/cpuinfo
```

Vérifiez ensuite que la sortie est non vide. Si c'est bien le cas, alors vous pouvez passer à l'étape suivante.

La prochaine étape consiste à installer un hyperviseur, dans mon cas je vais utiliser **VirtualBox**. Pour ceux qui souhaitent **utiliser leur machine physique en tant que nœud**, je vous montrerai plus tard comment s'y prendre lorsque nous aborderons la

création de notre cluster Kubernetes.

kubectl

Il est recommandé au préalable d'**installer kubect**l, afin d'interagir avec notre cluster Kubernetes.

Pour ce faire, premièrement commençons par installer le binaire kubectl :

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/$(uname -m)/bin/linux/amd64/kubectl)
chmod +x kubectl
```

Après cela, déplaçons le binaire kubectl dans le dossier d'exécution des utilisateurs :

```
sudo mv ./kubectl /usr/local/bin/kubectl
```

Enfin, testons notre installation en vérifiant la version de kubectl :

```
kubectl version
```

Résultat

```
Client Version: version.Info{Major:"1", Minor:"15", GitVersion:"v1.15.1", GitCommit:"d4617b297c00c24bb10d6b1888b5063e879f3684",
```

Installation de Minikube

Une fois les prérequis satisfaits, on peut à ce moment-là, passer à l'étape d'installation de Minikube. À cet effet, nous allons commencer par **télécharger le binaire Minikube**, comme suit :

```
curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64 && chmod +x minikube
```

Déplaçons ensuite l'exécutable Minikube dans le dossier binaire des utilisateurs :

```
sudo mv minikube /usr/local/bin
```

Analysons ensuite le bon déroulement de notre installation en révélant la version de notre Minikube :

```
minikube version
```

Résultat

```
minikube version: v1.2.0
```

Création du cluster Kubernetes avec Minikube


À présent, il est temps de démarrer Minikube afin de **créer notre premier cluster Kubernetes**. La commande que nous allons exécuter va créer et configurer une machine virtuelle qui exécute un cluster Kubernetes à un seul nœud, elle configurera également notre installation de kubectl de manière à communiquer avec notre cluster.

```
minikube start
```

Résultat

```
? minikube v1.2.0 on linux (amd64)
? Downloading Minikube ISO ...
  129.33 MB / 129.33 MB [=====] 100.00% 0s
? Creating virtualbox VM (CPUs=2, Memory=2048MB, Disk=20000MB) ...
? Configuring environment for Kubernetes v1.15.0 on Docker 18.09.6
? Downloading kubeadm v1.15.0
? Downloading kubelet v1.15.0
? Pulling images ...
? Launching Kubernetes ...
? Verifying: apiserver proxy etcd scheduler controller dns
? Done! kubectl is now configured to use "minikube"
```

Si on retourne sur notre hyperviseur, on peut percevoir notre nouvelle VM fraîchement créée :



minikube
2.6 En fonction

Général

Nom : minikube
Système d'exploitation : Linux 2.6 / 3.x / 4.x (64-bit)
Emplacement du fichier de paramètres : /home/hatim/.minikube/machines/minikube/minikube

Système

Mémoire vive : 2048 Mo
Processeurs : 2
Ordre d'amorçage : Optique, Optique, Disque dur
Accélération : VT-x/AMD-V , Pagination imbriquée, PAE/NX , Paravirtualisation KVM

Affichage

Mémoire vidéo : 8 Mo
Contrôleur graphique : VBoxVGA
Serveur de bureau à distance : Désactivé
Enregistrement : Désactivé

Stockage

Contrôleur : SATA
Port SATA 0 : [Lecteur optique] boot2docker.iso (129,33 Mio)
Port SATA 1 : disk.vmdk (Normal, 19,53 Gio)

Son

Pilote hôte : PulseAudio
Contrôleur : ICH AC97

Réseau

Interface 1: Réseau para-virtuel (NAT)
Interface 2: Réseau para-virtuel (Réseau privé hôte, 'vboxnet0')

USB

Désactivé

Dossiers partagés

Dossiers partagés : 1

Description

Aucune

Si vous souhaitez utiliser votre machine physique en tant que nœud, alors utilisez l'option `--vm-driver` de la commande `minikube start` avec la valeur `none`. Cette option exécutera les composants Kubernetes sur votre machine hôte et non sur une machine virtuelle **à condition de posséder le moteur Docker sur votre machine hôte Linux**. Ce qui nous donne la commande suivante :

```
minikube start --vm-driver=none
```

Vous pouvez aussi personnaliser votre nœud en utilisant certaines options de la commande `minikube start`. Avant de personnaliser notre machine, voici déjà la configuration par défaut de certaines ressources d'un nœud Minikube :

```
DefaultMemory    = 2048 MB      # Quantité de la RAM
DefaultCPUS       = 2           # Nombre de processeurs
DefaultDiskSize   = 20 GB       # Taille du disque virtuel
DefaultVMDriver   = virtualbox  # L'hyperviseur utilisé
```

Vous pouvez **retrouver la configuration complète par défaut de Minikube** dans le fichier suivant :

```
cat ~/.minikube/machines/minikube/config.json
```

Résultat

```
{
  "ConfigVersion": 3,
  "Driver": {
    "IPAddress": "192.168.99.105",
    "MachineName": "minikube",
    "SSHUser": "docker",
    "SSHPort": 38707,
    ...
    "CPU": 2,
    "Memory": 2048,
    "DiskSize": 20000,
    "NatNicType": "virtio",
    ...
  },
  "Name": "minikube"
}
```

Désormais, personnalisons notre VM ! Nous allons créer un cluster Kubernetes contenant un seul nœud avec deux fois plus puissance que la configuration par défaut. Ainsi nous exécuterons la commande ci-dessous :

```
minikube start --memory=4096 --cpus=4 disk-size=40g
```

Résultat

```
? minikube v1.2.0 on linux (amd64)
? Creating virtualbox VM (CPUs=4, Memory=4096MB, Disk=40000MB) ...
? Configuring environment for Kubernetes v1.15.0 on Docker 18.09.6
? Downloading kubeadm v1.15.0
? Downloading kubelet v1.15.0
? Pulling images ...
? Launching Kubernetes ...
? Verifying: apiserver proxy etcd scheduler controller dns
? Done! kubectl is now configured to use "minikube"
```

Pour **obtenir les noms des champs configurables de notre nœud Minikube**, nous exécuterons alors la commande suivante :

```
minikube config -h
```

Résultat

```
* vm-driver
* feature-gates
* v
* cpus
* disk-size
* host-only-cidr
* memory
* log_dir
...
* embed-certs
```

Manipulation du cluster Kubernetes avec Minikube

Commençons par **vérifier l'état de notre cluster Kubernetes**, comme suit :

```
minikube status
```

Résultat

```
host: Running
kubelet: Running
apiserver: Running
kubectl: Correctly Configured: pointing to minikube-vm at 192.168.99.105
```

On va utiliser l'outil kubectl afin de **récupérer la liste des nœuds de notre cluster**

Kubernetes :

```
kubectl get nodes
```

Sans aucune surprise, on ne récupère qu'un seul nœud de type Master :

NAME	STATUS	ROLES	AGE	VERSION
minikube	Ready	master	15m	v1.15.0

Nous pouvons aussi **récupérer des informations sur notre cluster Kubernetes** grâce à la commande ci-dessous :

```
kubectl cluster-info
```

Résultat

```
Kubernetes master is running at https://192.168.99.105:8443
KubeDNS is running at https://192.168.99.105:8443/api/v1/namespaces/kube-system/serv
```

Si jamais, vous rencontrez quelques soucis avec votre cluster Kubernetes, n'hésitez alors pas à **fouiller dans les logs de Minikube** afin de connaître la source du problème :

```
minikube logs
```

Pour information, cette commande vient avec trois options qui peuvent vous être bien utiles :

- **-f** ou **--follow** : suivre en permanence les logs de Minikube (correspond à un **tail -f** sous Linux)
- **-n** ou **--length** : nombre de lignes à afficher (50 par défaut)

- `--problems` : afficher uniquement les logs qui pointent vers des problèmes connus

Vous n'aurez nullement besoin de vous connecter à la VM Minikube, mais si l'envie vous en dit, alors voici la commande destinée à **se connecter directement en ssh à votre nœud Minikube** :

```
minikube ssh
```

Résultat

De la même façon, vous pouvez aussi vérifier si votre minikube est à jour de la façon suivante :

```
minikube update-check
```

Résultat

```
CurrentVersion: v1.2.0
LatestVersion: v1.2.0
```

Information

Si votre version n'est pas à jour, il suffit alors de reproduire la procédure de téléchargement de Minikube, vue précédemment.

Dans le chapitre précédent, je vous avais annoncé, qu'il existait deux façons pour communiquer avec un cluster Kubernetes, soit comme vu antérieurement, en utilisant le binaire kubectl, soit depuis une interface web de management nommé Dashboard. Ça tombe bien car minikube a pensé à nous, dans l'intention de nous **faciliter le lancement d'un Dashboard** :

```
minikube dashboard
```

Résultat

```
? Enabling dashboard ...
? Verifying dashboard health ...
? Launching proxy ...
? Verifying proxy health ...
? Opening http://127.0.0.1:34751/api/v1/namespaces/kube-system/services/http:kubernetes-dashboard/proxy
Fontconfig warning: "/usr/share/fontconfig/conf.avail/05-reset-dirs-sample.conf", line 48: cannot open font
[9491:9511:0731/234116.883991:ERROR:browser_process_sub_thread.cc(203)] Waited 5 ms for shutdown
Ouverture dans une session de navigateur existante.
```

Vous n'avez plus besoin de votre cluster Kubernetes, et vous souhaitez vous en débarrasser facilement ? Pas de soucis, challenge accepted ! Vous pouvez facilement sans aucune résistance **supprimer votre cluster Minikube**, à l'aide de la commande suivante :

```
minikube delete
```

Résultat

```
? Deleting "minikube" from virtualbox ...
? The "minikube" cluster has been deleted.
```

Conclusion

Il faut bien l'avouer, Minikube reste un outil qui facilite grandement le provisionnement et la gestion des clusters Kubernetes à un seul nœud. C'est l'outil

idéal pour tester et découvrir l'orchestrateur Kubernetes, de ce fait il est optimisé pour des flux de travail en mode test/développement.

Pour les personnes qui visitent mon site pour la première fois (déjà je vous souhaite la bienvenue), je tenais à vous dire que j'ai l'habitude à chaque fin de chapitre de rajouter un **aide-mémoire**, et en voici un pour minikube :

```
# Modifier la config minikube
minikube config
    minikube config set memory 1024 : Exemple pour reconfigurer la mémoire vive

# Accéder à l'interface web d'administration Kubernetes
minikube dashboard

# Récupérer l'adresse IP du cluster en cours d'exécution
minikube ip

# Obtenir les logs de l'instance en cours d'exécution pour du débogage
minikube logs
    -f ou --follow : Suivre en permanence les logs de Minikube
    -n ou --length : Nombre de lignes à afficher (50 par défaut)
    --problems : Afficher uniquement les logs qui pointent vers des problèmes connus

# Se connecter en ssh sur le nœud Minikube
minikube ssh

# Démarrer un cluster Kubernetes local
minikube start
    --cpu <int> : Nombre de processeurs alloués au minikube VM (2 par défaut)
    --disk-size <string> : Taille de disque allouée à la VM minikube (format <number>
    --memory <int> : Quantité de RAM allouée à la VM mini-cube en Mo (par défaut 2048
    --vm-driver <string> : Hyperviseur à utiliser (par défaut VirtualBox)

# Supprimer un cluster Kubernetes local
minikube delete

# Obtenir le statut d'un cluster Kubernetes local
minikube status

# Arrêter un cluster Kubernetes en cours d'exécution
minikube stop

# Afficher le numéro de la version actuelle et la plus récente de Minikube
minikube update-check

# Afficher la version de Minikube
minikube version
```