Installation et configuration de Kubernetes

1.1.1 Introduction

Kubernetes est une plateforme open source développée par Google pour la gestion des applications conteneurisées. Il vous permet de gérer, de mettre à l'échelle et de déployer automatiquement vos applications conteneurisées dans l'environnement en cluster.

Avec Kubernetes, vous pouvez :

• Orchestrer des conteneurs sur plusieurs hôtes,

• Mettre à l'échelle les applications conteneurisées avec toutes les ressources à la volée

• Et disposer d'un environnement de gestion de conteneur centralisé.

A titre d'exemple voici comment installer et configurer Kubernetes sur Ubuntu 21.04.

Nous utiliserons 3 serveurs Ubuntu 18.04LTS avec les privilèges « root »:

• « master » avec l'IP 192.168.8.100 (Maître Kubernetes)

• « worker1 » avec l'IP 192.168.8.101 (noeuds de travail Kubernetes)

• « worker2 » avec l'IP 192.168.8.102 (noeuds de travail Kubernetes)

ATTENTION

Si vous souhaitez dupliquer une VM pour éviter des installations multiples de Ubuntu 18.04 server il vous faudra changer les noms d'hôte des serveurs pour que les « Workers » puis être visible dans le « master »

Pour cela éditer le fichier « /etc/cloud/cloud.cfg

```
master:~#nano /etc/cloud/cloud.cfg
```

Puis modifier la ligne « preserve_hostname: false » en la remplaceant par « preserve_hostname: true

```
# This will cause the set+update hostname module to not operate (if true)
preserve_hostname: true
```

Renommer le nom d'hôte puis redémarrer le serveur :

```
root@master:~# echo master > /etc/hostname
root@master:~# reboot
```

Voici les étapes à réaliser :

- Installation de Kubeadm
- Configuration des hôtes
- Installation de Docker
- Désactivation de la SWAP
- Installation des paquets Kubeadm
- Initialisation du cluster Kubernetes
- Ajout de noeuds de travail au cluster Kubernetes
- Réalisation de tests

1.1.2 Préparation des hôtes

Dans cette première étape, nous allons préparer ces 3 serveurs pour l'installation de Kubernetes. Exécutez donc toutes les commandes sur les noeuds maître et de travail.

Nous allons préparer tous les serveurs pour l'installation de Kubernetes en modifiant la configuration existante sur les serveurs et en installant également certains packages, y compris docker et kubernetes.

Editez le fichier hosts sur tous les serveurs à l'aide de l'éditeur nano :

```
sudo nano /etc/hosts
```

Collez la configuration des hôtes ci-dessous :

```
192.168.8.102 master
192.168.8.100 worker1
192.168.8.101 worker2
```

Sauvegarder et quitter.

Maintenant testez le ping sur tous les serveurs nom d'hôte :

```
ping -c 3 master
ping -c 3 worker1
ping -c 3 worker2
```

Assurez-vous que toutes les adresses IP sont résolues en tant que nom d'hôte.

```
root@master:~# ping -c 3 master
PING localhost.localdomain (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=1 ttl=64 time=0.067
ms
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=2 ttl=64 time=0.038
ms
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=3 ttl=64 time=0.045
ms
```

Les commandes qui vont suivre sont à exécuter sur tous les nœuds

```
sudo mkdir /etc/docker

cat <<EOF | sudo tee /etc/docker/daemon.json

{

  "exec-opts": ["native.cgroupdriver=systemd"],

  "log-driver": "json-file",

  "log-opts": {

    "max-size": "100m"

  },

  "storage-driver": "overlay2"

}

EOF
```

Désactiver la Swap

Afin de configurer les serveurs Linux Kubernetes, nous devons désactiver le SWAP.

Vérifiez la liste de swap et désactivez-la.

```
sudo swapon -s
sudo swapoff -a
```

Pour désactiver le SWAP de manière permanente, nous devons éditer le fichier '/etc/fstab'.

```
sudo nano /etc/fstab
```

Faites un commentaire sur le type de partition SWAP.

```
#/dev/mapper/hakase--labs--vg-swap_1 none swap sw 0 0
```

Enregistrez et quittez, puis redémarrez le système.

```
sudo reboot
```

1.1.3 Installation des Kubeadm

Dans ce tutoriel, nous allons utiliser les packages Kubeadm pour configurer le cluster Kubernetes. Nous installerons ceux-ci à partir du référentiel officiel Kubernetes.

Installez « apt-transport-https ».

```
sudo apt install -y apt-transport-https
```

Ajoutez la clé APT de Kubernetes

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

Et ajoutez le référentiel Kubernetes en créant un nouveau fichier kubernetes.list dans le répertoire « /etc/apt/sources.list.d ».

```
 cd /etc/apt/ && nano sources.list.d/kubernetes.list
```

Coller le référentiel Kubernetes ci-dessous.

```
deb http://apt.kubernetes.io/ kubernetes-xenial main
```

Maintenant, mettez à jour le référentiel et installez les paquets kubeadm en utilisant les commandes « apt » ci-dessous.

```
sudo apt update && apt install -y kubeadm kubelet kubectl
```

Attendez l'installation des packages kubeadm.

```
 root@master:~# sudo apt install -y kubeadm kubelet kubectl
 Reading package lists... Done
Building dependency tree Reading state information... Done
 kubeadm is already the newest version (1.16.3-00).
 kubectl is already the newest version (1.16.3-00).
 kubelet is already the newest version (1.16.3-00).
 0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
```

1.1.4 Initialisation du cluster

Dans cette étape, nous allons initialiser Kubernetes sur le noeud 'master'.

ATTENTION

```
sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
sudo kubeadm reset
```

Exécutez toutes les commandes de cette étape uniquement sur le serveur 'master'.

Initialisez le cluster Kubernetes à l'aide de la commande kubeadm ci-dessous.

```
sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --apiserver-advertise-
address=192.168.8.100 --kubernetes-version "1.22.4"
```

« --apiserver-advertise-address = » détermine l'adresse IP sur laquelle Kubernetes doit annoncer son serveur d'API.

« --pod-network-cidr = » spécifie la plage d'adresses IP du réseau de pod. Nous utilisons le réseau virtuel 'flannel'. Si vous souhaitez utiliser un autre réseau de pods tel que weave-net ou Calico, modifiez l'adresse IP de la plage.

RAPPEL

En cas de problème vous avez la possibilité de recommencer cette étape en réalisant au préalable un

```
Kubeadm reset
```

Puis effacer le repertoire $HOME/.kube

```
mkdir -p $HOME/.kube
```

ATTENTION : le Token ne sera plus retrouvable

Copiez donc la commande « kubeadm join » complète (avec le token) dans votre éditeur de texte.

La commande sera utilisée pour enregistrer de nouveaux noeuds de travail dans le cluster Kubernetes.

Maintenant, pour utiliser Kubernetes, nous devons exécuter certaines commandes, comme indiqué dans le résultat.

Créez un nouveau répertoire de configuration ".kube" et copiez le répertoire de configuration "admin.conf" à partir du répertoire "/etc/kubernetes".

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

Ensuite, déployez le réseau de flanelles sur le cluster Kubernetes à l'aide de la commande « kubectl ».

```
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-
flannel.yml
```

```
root@master:~# kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-
flannel.yml
…
```

ATTENTION

Attendez une minute, avant de passer à la suite.

Vérifiez le noeud Kubernetes et les pods à l’aide des commandes ci-dessous.

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

Le noeud 'master' s'exécute en tant que cluster 'maître' avec le statut 'prêt'

```
root@master:~# kubectl get nodes
```

Tous les pods 'kube-system' nécessaires au cluster sont opérationnels.

```
root@master:~# kubectl get nodes
NAME     STATUS ROLES  AGE VERSION
worker1 Ready master 72m v1.16.3
```

Tous les pods 'kube-system' nécessaires au cluster sont opérationnels.

```
root@master:~# kubectl get pods --all-namespaces
```

L'initialisation et la configuration du maître de cluster Kubernetes sont terminées

Ajout des nœuds de travail au cluster

Dans cette étape, nous allons ajouter les deux travailleurs (ou worker) « worker1 » et « worker2 » au noeud Kubernetes.

Connectez-vous au serveur 'worker1' et exécutez la commande kubeadm join obtenue lors de l'initialisation du cluster.

Rappel : il ne faut pas oublier d'effectuer la configuration présente au début du chapitre sur l'initialisation du cluster présente ici **1.1.4**

```
kubeadm join 192.168.8.102:6443 --token 9y7x5f.nrdvl5eilwsgyg9g \
--discovery-token-ca-cert-hash
sha256:8f3ce1b01c6c394d7297be499da9a52355cc350a3269e90326a7cbc124f3db80
```

Exécutez la même commande « kubeadm join » sur 'worker2' .

ATTENTION

Attendez quelques minutes avant de retourner sur le maitre.

Revenez sur le maître de noeud « **master** » et vérifiez l'état du noeud.

```
kubectl get nodes
```

Vous verrez que les noeuds de travail 'worker1' et 'worker2' font partie du cluster Kubernetes :

RAPPEL

Les noms d'hôte des serveurs doivent être uniques pour que les nodes apparaissent dans le master.

Si ce n'est pas le cas reportez-vous à l'avertissement de la page 50.

Dans ce cas il faut réinitialiser le master comme suit :

```
root@master:~# kubeadm reset
```

Et recommencer à partir de l'étape Initialisation du cluster Kubernetes.

1.1.5 Déploiement d'une application sur le cluster

Dans cette étape, nous allons déployer le serveur Web Nginx dans le cluster. Nous déploierons le serveur Web Nginx à l'aide du modèle YAML, effectuez les commandes suivantes uniquement sur le master.

Créez un nouveau répertoire nommé "nginx" et accédez à ce répertoire.

```
cd /
mkdir -p nginx/
cd nginx/
```

Créez maintenant le fichier Nginx Deployment YAML 'nginx-deployment.yaml' à l'aide de l' éditeur nano

```
sudo nano nginx-deployment.yaml
```

Coller les configurations ci-dessous.

```
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
 name: nginx
spec:
 strategy:
  type: Recreate
 selector:
  matchLabels:
   app: nginx
 replicas: 3 # tells deployment to run 1 pods matching the template
 template: # create pods using pod definition in this template
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx
    ports:
    - containerPort: 80
```

Sauvegarder et quitter. Créez maintenant le déploiement nommé "nginx-deployment" en exécutant la commande kubectl ci-dessous.

```
kubectl create -f nginx-deployment.yaml
```

Après avoir créé un nouveau "nginx-deployment", Vérifiez la liste des déploiements dans le cluster.

```
kubectl get deployments
kubectl describe deployment nginx
```

Le libellé de l'application est "nginx" et a 3 répliques.

Le "nginx-deployment" aura des conteneurs nommés "nginx", basés sur l'image du menu fixe, et exposera le port HTTP 80 par défaut.

Maintenant, vérifiez les pods Kubernetes et vous verrez le pod 'nginx-deployment-xxx' :

```
kubectl get pods
```

Vérifiez les détails du pod :

```
kubectl describe pods nginx-deployment-7f555c9b4b-8lvw6
```

Vous obtiendrez des « pods » de déploiement nginx avec 3 réplicas sur les noeuds de travail.

Ensuite, nous devons créer un nouveau service pour notre déploiement de Nginx.

Créez un nouveau fichier YAML nommé 'nginx-service.yaml'.

```
vim nginx-service.yaml
```

Collez la configuration ci-dessous.

```
apiVersion: v1
kind: Service
metadata:
name: nginx-service
labels:
run: nginx-service
spec:
type: NodePort
ports:
- port: 80
protocol: TCP
selector:
app: nginx
```

Sauvegarder et quitter.

EXPLICATIONS :

Nous créons un nouveau service kubernetes nommé 'nginx-service'.

Le type de service est 'NodePort' avec le port 80 par défaut de TargetPort HTTP.

Le service appartient à l'application nommée 'nginx' en fonction de notre déploiement 'nginx-deployment'.

Créez le service kubernetes à l'aide de la commande kubectl ci-dessous.

```
kubectl create -f nginx-service.yaml
```

Cela donne :

```
root@master:~/nginx# kubectl create -f nginx-service.yaml
service/nginx-service created
```

Maintenant, vérifiez tous les services disponibles sur le cluster et vous obtiendrez le service Nginx dans la liste, puis vérifiez les détails du service.

```
kubectl get service
```

Vous verrez la page par défaut de Nginx.

```
root@worker1:~# curl worker1:30291
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
body {
width: 35em;
margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif;
}
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

L'installation et la configuration de Kubernetes Cluster sur Ubuntu ont été effectuées avec succès.