

Documentation Technique

Kubernetes

Réalisé par:

Mohamed Dhia Souissi

Sommaire:

I. Guide d'installation et configuration d'un cluster avec kubernetes

I- Guide d'installation et configuration d'un cluster avec kubernetes

*installer docker :

1. se connecter à votre machine en mode root.

2. Mise à jour des packages et s'assurer que les APT fonctionnent avec HTTPS et que les certificats ca sont installés.

```
# sudo apt-get update
# sudo apt-get install apt-transport-https ca-certificates
```

3. Ajoutez une nouvelle clé GPG

```
# sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys
58118E89F3A912897C070ADB76221572C52609D
```

4. Ouvrez le fichier docker.list dont le chemin est : `/etc/apt/sources.list.d/docker.list` Si le fichier n'existe pas, créez-le.

5. Supprimez le contenu de ce fichier.

6. Ajoutez une entrée pour votre système d'exploitation Ubuntu.

*Ubuntu 12.04 lts:

```
deb https://apt.dockerproject.org/repo ubuntu-precise main
```

*Ubuntu 14.04 lts:

```
deb https://apt.dockerproject.org/repo ubuntu-trusty main
```

* Ubuntu 15.10 Wily

```
deb https://apt.dockerproject.org/repo ubuntu-wily main
```

*Ubuntu 16.04 lts:

```
deb https://apt.dockerproject.org/repo ubuntu-xenial main
```

7. Enregistrez et fermez le fichier `/etc/apt/sources.list.d/docker.list`

8. Mettre à jour les packages apt.

```
#sudo apt-get update
```

9. Purger l'ancienne pension si elle existe.

```
#sudo apt-get purge lxc-docker
```

10. Vérifiez qu'on va télécharger du bon répertoire.

```
#apt-cache policy docker-engine
```

11. Pour Ubuntu Trusty, Wily et Xenial, il est recommandé d'installer le paquet de noyau `linux-image-extra`. Ce dernier vous permet d'utiliser le pilote aufs de stockage.

```
#sudo apt-get update
```

```
#sudo apt-get install linux-image-extra-$(uname -r)
```

12. Installez Docker.

```
#sudo apt-get update
```

```
#sudo apt-get install docker-engine
```

13. Lancez le service Docker.

```
#sudo service docker start
```

14. Vérifiez que Docker est bien installé.

```
#sudo docker run hello-world
```

*installer kubernetes : (cette étape est nécessaire pour tous les machines qui appartiennent au cluster qu'on va le configurer)

-Clonez le projet kubernetes-config trouvé sur github dhiaSouissi/kubernetes-config par la commande suivante :

```
#git clone https://github.com/dhiaSouissi/kubernetes-config
```

-Allez dans le répertoire kubernetes-config et exécutez la commande suivante :

```
#chmod +x *
```

-Lancez le script build.sh par :

```
#./build.sh
```

-Après l'exécution de ce script, un répertoire **binaries** sera créé dans votre répertoire. Allez dans ce répertoire et tapez ces commandes :

```
#sudo cp kubectl /usr/local/bin/kubectl
```

```
# sudo chmod +x /usr/local/bin/kubectl
```

-Vérifiez que kubernetes est bien installé par la commande suivante :

```
#kubectl --help
```

Dans la machine master :

-Allez sous le répertoire kubernetes-config.

-Tapez la commande :

```
#. /master.sh
```

En cas d'un message d'erreur (!!! [0810 06:18:35] flannel failed to start. Exiting...)

Retapez la commande et répondre par non si il vous demande de nettoyer /var/lib/kubelet et de stopper les anciens conteneurs.

-Vérifiez par la commande :

```
#kubectl get nodes
```

Dans la machine slave :

-Tapez les commandes suivantes :

```
#export MASTER_IP= adresse IP de master
```

```
# ./worker.sh
```

```
#kubectl config set-cluster moncluster - --server=http://adresseIP_Master :8080
```

```
#kubectl config set-context moncluster - --cluster=moncluster
```

```
#kubectl config use-context moncluster
```

-Pour vérifier, tapez la commande :

```
#kubectl get nodes
```

-Pour voir le Dashboard kubernetes, allez vers ce lien :

<http://adresseIP-master:8080/ui>

-Exemple d'utilisation :

```
# kubectl run nginx --image=nginx --port=90
```

```
#kubectl get pods
```

Le Dashboard est accessible via **http://adresseMaster:8080/ui**

kubernetes All user namespaces ▼

Workloads + DEPLOY APP ↑ UPLOAD YAML

Replication controllers

Name	Labels	Pods	Age	Images
✓ nginx	run: nginx	1 / 1	12 minutes	nginx

Pods

Name	Status	Restarts	Age	Cluster IP	CPU (cores)	Memory (bytes)
✓ nginx-e1pei	Running	0	12 minutes	10.1.9.4

kubernetes All user namespaces ▼

Pods > nginx-e1pei EDIT DELETE

OVERVIEW

Details	Network
Name: nginx-e1pei	Node: 192.168.42.159
Namespace: default	IP: 10.1.9.4
Start time: 11/8/16 09:20	
Labels: run: nginx	
Status: Running	
Images: nginx	

kubernetes

Workloads > Nodes

Name	Labels
192.168.42.159	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux kubernetes.io/hostname: 192.168.42.159
192.168.42.161	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux kubernetes.io/hostname: 192.168.42.161

Inconvénient de cette solution :

On ne peut pas afficher les conteneurs de Docker créés par des commandes propre à docker dans ce Dashboard. Ceci s'explique par :

- Des raisons de concurrence entre Google propriétaire de kubernetes et Docker.Inc propriétaire de Docker. En effet, Google avec Kubernetes veut dominer le monde de virtualisation par la création d'un Dashboard qui facilite la création des conteneurs virtuels par des interfaces graphiques, sachant que kubernetes utilise indirectement docker.

➔ Conclusion :

Kubernetes ne permet pas d'afficher dans son propre Dashboard les conteneurs créés par les commandes de docker.