Docker - Installation docker + swarm

Difficulté : Confirmé

Notions : docker, container, portainer, docker swarm.

I. Introduction

> Introduction à la containerisation

Cette procédure à pour but de décrire la mise en place d'une infrastructure docker + portainer en standalone et en swarm.

> plus d'informations sur : https://docs.docker.com/

> **Prérequis** :

- (ici) Ubuntu server 22.04
- 4vcpu
- 4Gb de RAM
- 20Gb disque OS
- disque supplémentaire pour docker (taille dépendante du contenu prévu, prévoir minimum 10Gb)

II. Installation de docker

> **Attention** : Avant toute installation, il est conseillé de mettre à jour son système.

> **Information** : Un assistant d'installation contenant les instructions sur mesures est disponible ici : get-docker

2.1 Installer les prérequis

Tout d'abord, il faut installer les prérequis.

```
sudo apt install ca-certificates curl gnupg lsb-release
```

```
root@template:/home/sc4d-# apt install ca-certificates curl gnupg lsb-release
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
lsb-release est déjà la version la plus récente (11.1.0ubuntu4).
lsb-release passé en « installé manuellement ».
ca-certificates est déjà la version la plus récente (20211016ubuntu0.22.04.1).
ca-certificates passé en « installé manuellement ».
curl est déjà la version la plus récente (7.81.0-1ubuntu1.7).
curl passé en « installé manuellement ».
gnupg est déjà la version la plus récente (2.2.27-3ubuntu2.1).
gnupg passé en « installé manuellement ».
Les paquets suivants ont été installés automatiquement et ne sont plus nécessaires :
  libflashrom1 libftdi1-2
Veuillez utiliser « apt autoremove » pour les supprimer.
0 mis à jour, 0 nouvellement installés, 0 à enlever et 1 non mis à jour.
root@template:/home/sc4d-# _
```

2.2 Installer docker

Ajouter la clé des dépôts.

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

```
root@template:/home/sc4d-# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dear
mor -o /usr/share/keyrings/docker.gpg
root@template:/home/sc4d-#
```

Vérifier la présence de la clé avec :

```
ls /etc/apt/keyrings/docker.gpg
```

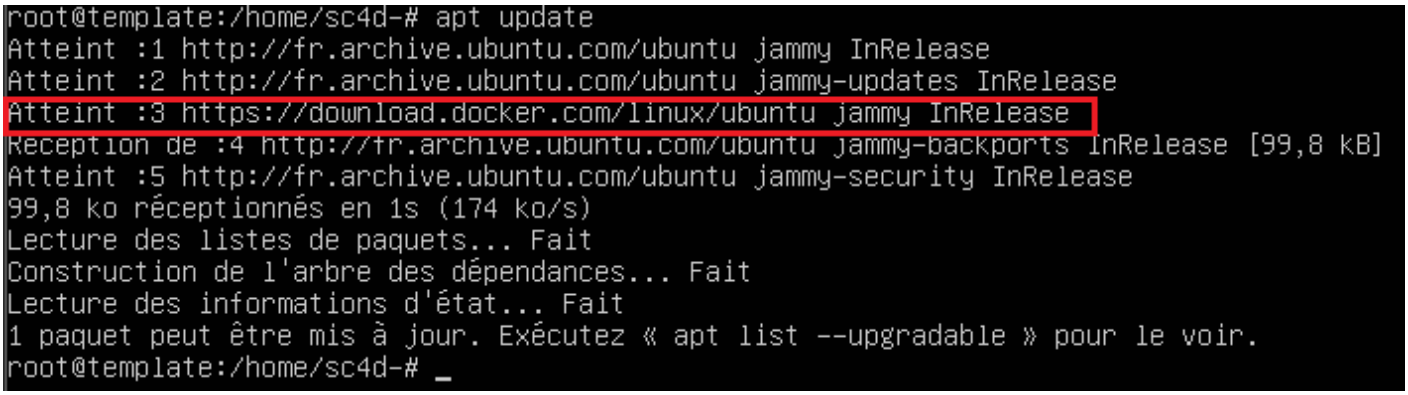
```
root@template:/home/sc4d-# ls /etc/apt/keyrings/docker.gpg
/etc/apt/keyrings/docker.gpg
```

Ajouter le dépôt :

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null
```

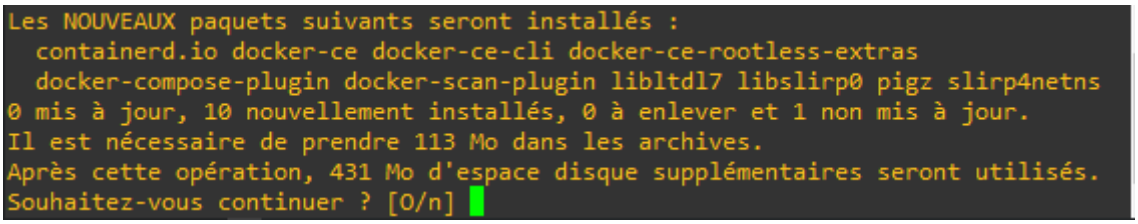
Lancer ensuite une nouvelle fois :

```
apt update
```

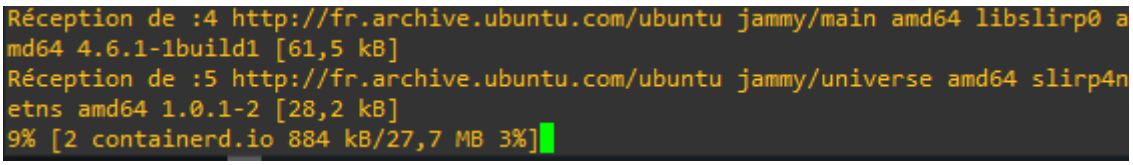


Installer docker et ses composants avec la commande :

```
sudo apt install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```



Valider et attendre la fin de l'installation.



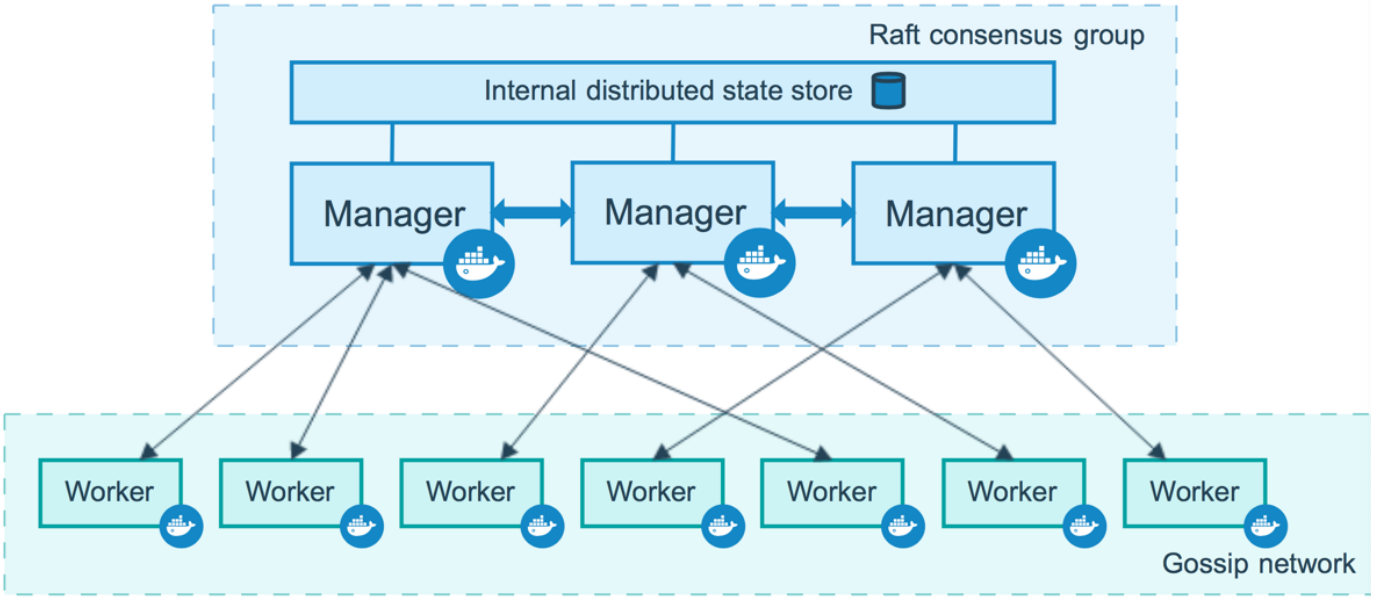
> **Best practice** : Il est conseillé d'utiliser un disque dédié pour docker et de réaliser un point de montage sur "***/var/lib/docker***". Pour cela, suivre la procédure : Ubuntu Server - Ajouter un disque/volume ou Ubuntu Server - Gérer les volumes avec LVM

>

> **Information** : Dans le cadre du déploiement d'un swarm, répéter les opérations ci-dessus sur l'ensemble des nœuds.

étape optionnelle - création du swarm

Cette étape est optionnelle est n'est nécessaire que dans le cadre de la mise en place d'un cluster. Ou dans le cas de docker d'un ***SWARM***.



2.1.1 Initialisation du swarm

Sur le serveur docker qui sera le master du cluster, Exécuter la commande suivante pour initialiser le swarm.

```
docker swarm init --advertise-addr <adresse ip>
```

```
root@template:/home/sc4d-# docker swarm init --advertise-addr 192.168.1.28
Swarm initialized: current node (zrt0xxnkgd4s2hxk96yo91grz) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-4onnwsb5ya2kl3zwqw0c88qa83u0otnjotrjjtzeh72y27i0rt-cl16owen1
v4hara0f34ejce0 192.168.1.28:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

root@template:/home/sc4d-# _
```

> Plus de paramètres : swarm init

Bien noter lors de l'initialisation du swarm la commande qui permettra au workers de joindre le swarm ainsi que le token.

```
root@template:/home/sc4d-# docker swarm init --advertise-addr 192.168.1.28
Swarm initialized: current node (zrt0xxnkgd4s2hxk96yo91grz) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-4onnwsb5ya2kl3zwqw0c88qa83u0otnjotrjjtzeh72y27i0rt-cl16owen1
v4hara0f34ejce0 192.168.1.28:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

root@template:/home/sc4d-# _
```

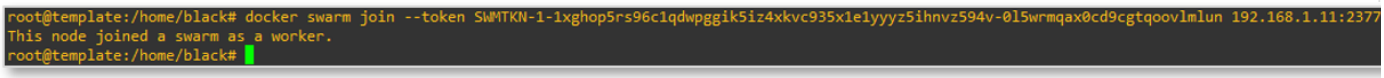
> **Astuce** : Si cette commande avec le token n'est plus affiché. Il est possible de la récupérer en exécutant sur le manager la commande suivante.

```
docker swarm join-token worker
```

2.1.2 Jonction du/des workers

Pour joindre le swarm, entrer la commande suivante sur les workers.

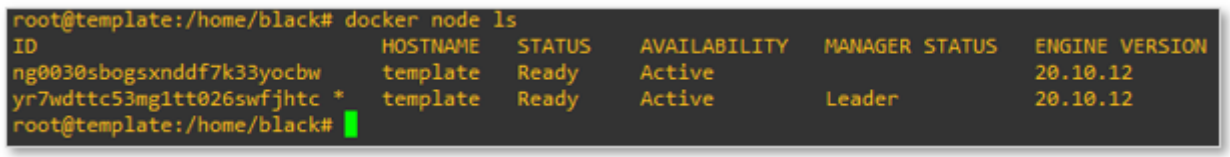
```
docker swarm join --token <token> <ip du master>:<port>
```



2.1.3 Autres opérations

Pour vérifier le status du cluster, utiliser la commande suivante sur un manager.

```
docker node ls
```



Pour quitter le swarm.

```
docker swarm leave
```

Puis sur le dernier nœud du cluster :

```
docker swarm leave --force
```

III. Pricipes de base

> **Information** : La liste des commandes est disponible ici : commandes docker

L'utilisation de docker est assez intuitive une fois que l'on a saisi les bases. Les commandes docker sont assez simples et structurées.

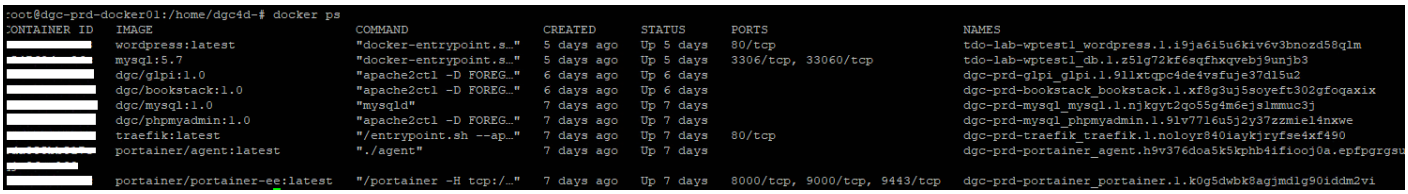
Docker	<objet>	<action>	<nom ou ID>
	image	create	
	network	rm (remove)	
	service	ls (list)	
	stack	prune (delete all unused)	
	volume		

exemples :

```
docker image ls
```

Pour les conteneurs, les commandes sont un peu différentes :

docker ps : liste les conteneurs



docker run <container> : lance le conteneur

docker stop <container> : stoppe le conteneur

3.1 Manipulation des images

> **information** : Les images peuvent être téléchargées depuis dockerhub par exemple : docker HUB

docker image pull <nom:tag> : Pour télécharger une image.

docker image push <nom:tag> : Pour pousser une image sur le repo.

docker image prune : efface les images non utilisées.

3.2 Manipulation des volumes

Les commandes vues plus haut permettent l'essentiel des manipulations.

Il est également bon de savoir les choses suivantes :

- un volume peut être monté sur un ou plusieurs conteneurs simultanément avec des autorisations différentes
- le contenu d'un volume peut être accédé depuis l'hôte docker au même titre qu'un dossier.

3.3 Manipulation des réseaux

Les commandes vues plus haut permettent de manipuler les réseaux.

Lors d'une création de réseau, il est possible d'en fixer les propriétés à l'aide des options suivantes :

```
Options:
      --attachable           Enable manual container attachment
      --aux-address map      Auxiliary IPv4 or IPv6 addresses used by Network driver (default map[])
      --config-from string   The network from which to copy the configuration
      --config-only          Create a configuration only network
  -d, --driver string        Driver to manage the Network (default "bridge")
      --gateway strings      IPv4 or IPv6 Gateway for the master subnet
      --ingress              Create swarm routing-mesh network
      --internal             Restrict external access to the network
      --ip-range strings     Allocate container ip from a sub-range
      --ipam-driver string   IP Address Management Driver (default "default")
      --ipam-opt map         Set IPAM driver specific options (default map[])
      --ipv6                 Enable IPv6 networking
      --label list           Set metadata on a network
  -o, --opt map              Set driver specific options (default map[])
      --scope string         Control the network's scope
      --subnet strings       Subnet in CIDR format that represents a network segment
```

> **Astuce** : chaque réseau reconnaît le nom des conteneurs qui y sont affecté comme nom DNS, il est ainsi possible d'adresser un containeur par son nom.

exemple : le stack 'wordpress' contient deux conteneurs. 'wpweb' et 'wpmysql'.

Si les deux conteneurs partagent un réseau, le conteneur 'wpweb' pourra adresser le conteneur de base de donnée avec le nom 'wpmysql'.

IV. Créer un conteneur ou un service

4.1 sur un hôte standalone

4.1.1 créer via commande :

Dans l'exemple de conteneur suivant, il s'agira d'un service web apache sans base de données avec seulement un volume pour les données et une exposition des ports 80 et 443.

```
docker run -d -p 8001:80 -p 4431:443 --name webserver --restart=always -v
webserver_data:/var/www/html apache:latest
```

Docker run [options] <image>	lance le conteneur avec les options spécifiées basé sur l'image <image>
-d	lance le conteneur en arrière plan avec un ID.
-p 8001:80 -p 4431:443	mappe les ports de l'hôte et les redirige sur les 80 et 443 du conteneur
--name	nom du conteneur
--restart	donne les options de redémarrage du conteneur
-v webserver_data:/var/www/html	mappe le volume webserver_data sur le dossier /var/www/html
apache:latest	image à utiliser

4.1.2 créer une image avec un dockerfile :

pour compiler une image, utiliser la commande :

```
docker image build -f <nomdefichier> -t <nom image> .
```

4.2 sur un swarm

Sur un swarm, la logique change un peu du fait de pouvoir déclarer un service avec des propriété de déploiement au lieu d'un simple conteneur.

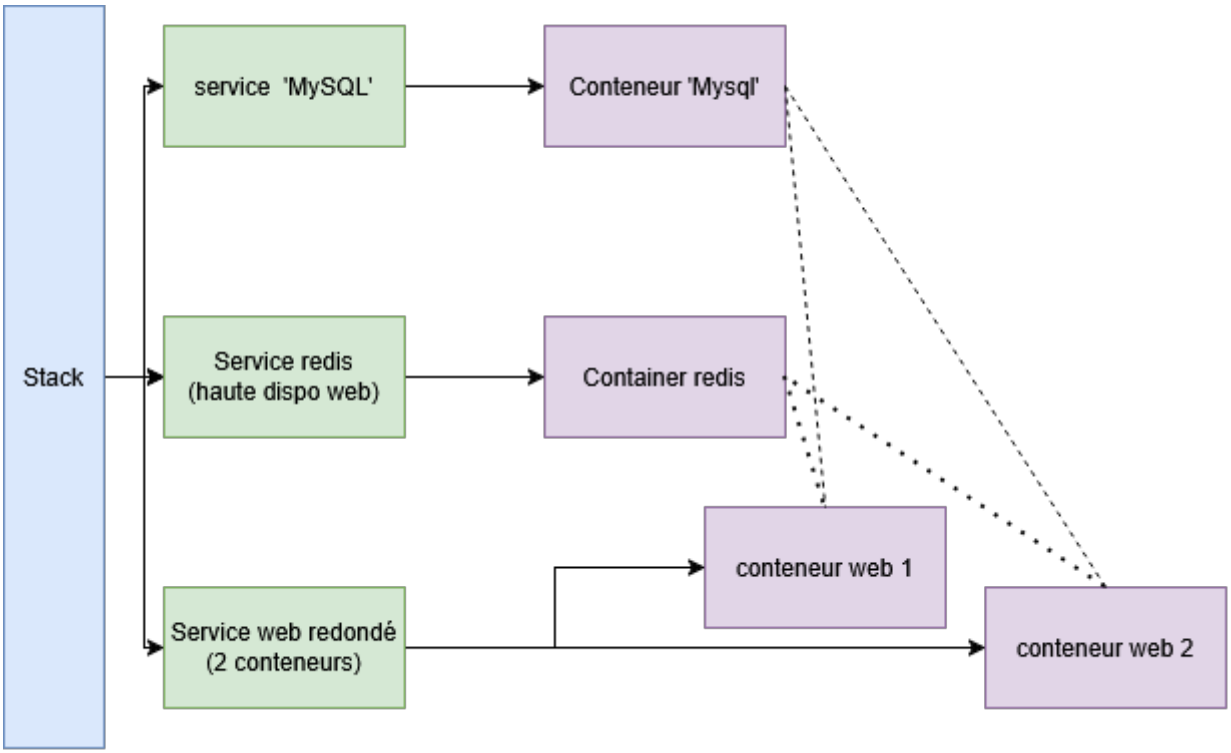
Dans la liste des options supplémentaires, certaines sont notables :

--constraint : ajoute des contraintes de déploiement (que sur un nœud manager, que sur les workers, etc...)

--deployment : ajoute des spécificités au déploiement (labels, nombre de copies, priorités, etc...)

4.2.1 Architecture du service :

La logique change donc par rapport à un déploiement classique, l'on ne parle plus de container directement.



Un Stack deviens un ensemble de services. Qui en fonction de leurs propriétés de déploiement contiennent un ou plusieurs containers.

La définition des propriétés se font donc au niveau du service et non plus au niveau du container.

4.2.2 Le réseau Overlay

Idem pour les réseaux, un nouveau type de réseau fait sont apparition, le type "***overlay***".

Il s'agit d'une encapsulation réseau par dessus la couche 3 pouvant utiliser divers mécanismes afin de créer une topologie virtuelle qui sera partagée entre tous les nœuds du cluster. Ainsi un réseau virtuel ou (VXLAN) sera perçu de la même façon depuis tous les hôtes du cluster et les container dans ce réseaux pourrons communiquer ensemble peut importe le nœud sur lequel ils sont situés.

4.2.2 Les secrets et configurations

Il devient également possible d'utiliser deux mécaniques supplémentaires jusque là absentes :

- les secrets

- les configs

Les secrets :

En effet, il peut être nécessaire de partager une information entre les différents conteneurs d'un même service, et ce indépendamment du nœud sur lequel il se situe. Cependant, cela veut également dire que cette information doit être propagée à l'ensemble des nœuds.

La plupart du temps, il peut s'agir de données sensibles (mot de passe, clé api, etc...).

Pour répondre à cela, il est possible de créer un "***secret***".

```
docker secret create [OPTIONS] SECRET [file|-]
```

Cette information sera alors chiffrée et ne sera accessible en clair qu'a l'intérieur des conteneurs qui la demande sous la forme d'un fichier présent dans '***/run/secrets/<secret_name>***'

Les configurations :

Sur le même principe que les secrets, il s'agit de fichiers déclarés sur le swarm qui sera distribué sur les nœuds.

Cependant celui-ci ne sera pas chiffré et restera modifiable.

```
docker config create [OPTIONS] CONFIG [file|-]
```

Il sera monté sur les conteneurs dans '***/run/configs/<nom_fichier_config>***'.

Cela permet de propager une configuration unique à l'ensemble des conteneurs du service. Par exemple des configurations de Vhost apache.

Revision #4
Created 2026-07-30 07:35:51 UTC by Olivier
Updated 2026-09-16 07:03:08 UTC by Olivier