LIGNIERE
Timothée
BTS2 SIO
Date de création
09/11/23
Date de maj
23/11/23

Docker

Sommaire

1 - Problématique

Il peut être fastidieux de toujours reconfigurer une application qu'on le connaît déjà du bout des doigts et c'est en plus une perte de temps dans certains cas.

Cependant, il est possible de rendre le déploiement de services plus rapide à l'aide de Docker.

Docker est un logiciel permettant de lancer des applications rapidement dans des conteneurs logiciels ; il faut imaginer un conteneur comme une machine virtuelle, en plus primitif, car, elle ne contient que ce qui lui est nécessaire. Ce qui est un avantage, puisque, cela signifie que l'application peut être lancée alors qu'elle n'est pas censée fonctionner dans son état initial sur un OS en particulier par exemple.

2 - Notice d'installation

L'installation de Docker est différente selon chaque OS (parfois sur des points insignifiants, mais suffisant pour que cela ne fonctionne pas forcément), je me concentrerais ici sur l'installation sur Debian 11 et 12 en 64bits.

En cas de problème durant l'installation, se référé à https://docs.docker.com/engine/install/

Pour commencer, vérifier que tous les paquets qui pourraient créer des conflits sont désinstallés :

```
for pkg in docker.io docker-doc docker-compose podman-docker containerd runc; do sudo apt-get remove $pkg; done
```

Puis rajouter le dépositoire de Docker à celui mémorisé de votre machine :

```
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg


echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

Après cela, installer les paquets de Docker :

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

Enfin, vérifier l'installation à l'aide de la commande suivante :

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

Cette dernière commande télécharge et fait fonctionner une image docker test. Quand il se met à fonctionner, il renvoit un texte de confirmation puis s'éteint.

Si vous souhaitez utiliser la commande « docker » sans avoir les droits de root, alors faites la commande suivante pour, vous rajoutez à la liste des utilisateurs ayant accès à Docker :

```
sudo usermod -a -G docker <Votre nom d'utilisateur>
```

Pensez à recharger votre session, dans le cas contraire, le changement ne s'appliquera pas forcément et vous ne pourriez donc pas accéder à la commande sans la faire fonctionner en root.

3 – Commandes docker essentielles

Nous allons désormais nous intéresser aux commandes les plus basiques et certaines de leurs options.

3.1 – Docker run

Docker run permet de lancer les conteneurs et de les télécharger avant de les lancer s'ils ne sont pas encore présents sur la machine, elle s'écrit de la manière suivante :

```
docker run <Options> Image <Command> <Arg...>
```

« Image » peut soit correspondre au nom ou à l'ID du conteneur selon si le but est de créer une nouvelle machine ou d'en démarrer une déjà créer auparavant.

Il existe plusieurs options avec la commande run afin d'interagir avec le container. Il est tout à fait possible d'utiliser plusieurs options sur une seule ligne de commande.

3.1.1 – Donner un nom à un container

Pour obliger un nom particulier à un conteneur qui n'a pas encore été créer utiliser l'option **-name** :

```
docker run –name <Nom donner au conteneur> <Image Docker>
```

3.1.2 – Faire fonctionner un container en arrière-plan

Pour faire fonctionner un conteneur en arrière-plan, rajoutez l'option **-d** dans la commande :

```
docker run –d <Image Docker>
```

3.1.3 – Relier deux containers entre eux

Dans certains cas, un container a besoin d'un autre service qui est, lui aussi, hébergé par Docker, dans ce genre de cas, on peut relier par l'option **-link**

```
docker run -link=<Nom du containeur relier> <Image Docker>
```

3.1.4 – Réacheminement de port (Port Forwarding)

Dans le cas où il est nécessaire qu'un service ait accès à un port du container ou si on cherche à accéder à l'interface web d'un service, il faut forcément faire du Port forwarding sur l'un des ports de notre machine hôte de Docker avec l'option **-p** :

```
docker run -p <Port de la machine hôte>:<Port du container> <Image Docker>
```

3.1.5 – Relier un dossier avec le container

Si vous souhaitez accéder aux fichiers de configuration de votre service de manière permanente sur votre machine, vous pouvez le faire à l'aide de l'option **-v** :

```
docker run -v /<Chemin absolu vers un dossier de l'hôte>:/<Chemin absolu vers un dossier du container> <Image Docker>
```

3.1.6 – Attribuer un périphérique de stockage

Dans le cas où votre machine a besoin d'un grand espace de stockage, vous pouvez directement lui attribuer un disque présent sur votre machine hôte, pour cela, vous aurez besoin de l'emplacement de votre disque sur l'hôte et de l'option **--device** :

```
docker run --device=<Emplacement du disque attribuer> <Image Docker>
```

3.1.7 – Stratégie de redémarrage du container

Quand le container s'arrête, il ne redémarrera pas de lui-même lui avoir défini dans quel cas il est censé redémarrer avec l'option **--restart** :

```
docker run --restart= « <No / on-failure / always / unless-stopped> » <Image Docker>
```

3.1.8 – Supprimer container lors de son arrêt

Pour diverses raisons, vous pourriez vouloir que votre container se supprime de lui-même lors de son arrêt, pour cela, utiliser l'option **--rm** :

```
docker run --rm=<true / false> <Image Docker>
```

3.2 – Docker pull

Docker pull a pour seul intérêt de télécharger les images de containers, elle n'est donc pas aussi complexe que docker run.

```
docker pull <Image Docker>
```

3.2.1 – Télécharger toutes les images dans le dépôt

Si vous souhaitez télécharger toutes les images se trouvant dans le dépôt où vous avez trouvé votre image de container, utiliser l'option **-a** :

```
docker pull -a <Image Docker>
```

3.2.2 – Supprimer le retour d'information

Si vous ne souhaitez pas vérifier que l'image s'installe correctement, utiliser simplement l'option **-q** :

```
docker pull -q <Image Docker>
```

3.3 – Docker ps

Docker ps vous permet de lister les containers que vous avez mis en place et qui sont en cours de fonctionnement, nous allons aussi voir ses options.

```
docker ps
```

3.3.1 – Lister toutes les machines existantes

Le problème de la commande sans option est qu'elle ne montre que les containers en fonctionnement ; ceux ne fonctionnant pas ne sont donc pas visibles, c'est là l'intérêt de l'option **-a** :

```
docker ps -a
```

3.3.2 – Afficher uniquement les ID des containers

Chaque container lors de sa création a un ID qui lui ait attribué, pour les lister sans information en plus, utiliser l'option **-q** :

```
docker ps -q
```

3.3.3 – Afficher taille du container

Les containers peuvent prendre une place importante au fur et à mesure du temps, pour vérifier la taille de vos containers utiliser l'option **-s** :

```
docker ps -s
```

3.3.4 – Afficher les x derniers containers créé

Votre machine hôte peut commencer à avoir un nombre important de containers, pour vous simplifier la tâche lors de vos vérifications de créations de vos derniers containers, vous pouvez choisir de visualiser uniquement les x derniers containers à l'aide de l'option **-n** :

```
docker ps -n<Nombre voulu>
```

Pour uniquement avoir le dernier container, utiliser l'option **-l** :

```
docker ps -l
```

3.4 – Docker rm

Docker rm vous permet de supprimer les containers dont vous voulez vous débarrasser de la manière suivante :

```
docker rm <ID ou nom du container>
```

3.4.1 – Suppression forcée

Sous certaines conditions, Docker peut refuser la suppression d'un container, pour contrer cela, vous pouvez utiliser l'option **-f** :

```
docker rm -f <ID ou nom du container>
```

3.4.2 – Suppression de lien

Si vous avez précédemment utilisé l'option link lors de votre docker run et que vous souhaitez supprimer le lien entre deux machines, utiliser l'option **-l** :

```
docker rm -l <ID ou nom du container>
```

3.4.3 – Suppression de tous les containers

Il peut être nécessaire de faire table rase sur un projet impliquant plusieurs containers, pour cela, il faudrait supprimer tous les dockers hors cela peut ête fastidieux si vous devez copier un par un l'ID ou le nom de vos containers.

Pour cela, il faut utiliser une combinaison de commande en utilisant docker ps :

```
docker rm $(docker ps -aq)
```

3.5 – Docker images

Lorsque notre machine hôte de docker commence à contenir plusieurs projets, on ne sait plus vraiment quelles images sont installées dessus, certaines pourrait même être obsolètes, pour les lister, on utilise donc docker images :

```
docker images
```

3.5.1 – Voir toutes les images sans exception

Normalement, docker images cache les images intermédiaires, pour les voir il faut utiliser l'option **-a** :

```
docker images -a
```

3.5.2 – Uniquement voir les ID des images

Si vous souhaitez uniquement voir les ID des images, utiliser l'option **-q** :

```
docker images -q
```

3.6 – Docker rmi

À la même manière que la suppression des containers, il est possible de supprimer les images qui ne vous servent plus, elles seront néanmoins réinstallées si vous les demandez avec un docker run ou docker pull :

```
docker rmi <ID ou nom de l'image>
```

3.6.1 – Forcer suppression d'une image

Pour forcer la suppression d'une image en cas de problème, utiliser l'option **-f** :

```
docker rmi -f <ID ou nom de l'image>
```

3.6.2 – Suppression de toutes les images

Si pour quelconque raison, vous souhaitez supprimer toutes vos images, utiliser la commande suivante :

```
docker rmi $(docker images -aq)
```

3.7 – Docker start

Il n’est pas impossible que pour une raison X ou Y, votre machine s’arrête et vous souhaitez la démarrer à nouveau, si c’est le cas utilisé la commande suivante :

```
docker start <Nom du container>
```

3.8 – Docker stop

À l'inverse de docker start, vous souhaitez peut-être arrêter une machine, si telle est le cas, utiliser la commande suivante :

```
docker stop <Nom du container>
```

3.8.1 – Arrêter au bout d'un temps donné

Vous pouvez arrêter un container au bout d'un certain nombre de secondes à l'aide de l'option **-t** :

```
docker stop -t <nombre de secondes> <Nom du container>
```

3.9 – Docker Swarm

Docker swarm est une solution intégrée à Docker qui vous permet de mettre vos containers en redondance, pour cela, vous aurez besoin de plusieurs machines (minimum 2) ayant Docker d'installé.

Swarm fonctionne avec un principe de Master – Slave, le Master s'appelle le Manager et les Slaves s'appellent des Workers.

3.9.1 – Création du swarm

Sur la machine qui aura le rôle de Manager (Master) utilisé la commande suivante pour initier le swarm :

```
docker swarm init --advertise-addr <Adresse de votre machine>
```

Suite à cela, Docker vous fournira une commande à utiliser sur vos Workers (Slaves) pour qu'ils puissent rejoindre le swarm, la commande devrait ressembler à la suivante :

```
docker swarm join –token <Token> <Adresse de votre manager>:<Port>
```

Utiliser donc la commande fournie pour chaque machine que vous voulez relier.

3.9.2 – Déployer sur un swarm

Pour déployer sur un swarm, vous aurez besoin d'utiliser docker-compose avec un fichier docker-compose.yml avec la bonne version et le logiciel que vous désirez ; vous trouverez un grand nombre de références sur Github, je vous invite néanmoins à créer vos propres fichiers de déploiement.

Une fois que vous avez votre fichier, faites :

```
docker stack deploy -c <Fichier de déploiement> <Nom du Stack>
```

3.9.3 – Lister les machines du swarm

Pour lister l'ensemble de vos machines membres de votre swarm pour divers contrôles, faites :

```
docker node ls
```

3.9.4 – Lister les containers sur le swarm

Tout comme vous pourriez avoir besoin de lister vos machines, vous pourriez avoir besoin de lister les containers présent sur le swarm, pour cela faites :

```
docker node ps
```

4 - Annexes

Fiche de recette

Vérification de l'opérationnalité de la solution mise en œuvre : *Docker*

Description du test :

1. *Utiliser docker info*
2. *Utiliser docker run hello-world*

Résultats Attendus :

1. *Information du docker retourner*
2. *« Hello from Docker !»*

Réception Globale : Docker **Auteurs: Timothée LIGNIERE**	**Date: 09/11/23**
Reçu : ☐	
Reçu avec réserve : ☐ ……	
Refusé : ☐ …… Commentaire :	

Recette étape par étape * — ****** *(pour chaque étape, vous devez élaborer dans un fichier distinct un scénario détaillé à faire appliquer au « client » venant valider votre solution)*

Réception Etape 1: *Utiliser la commande docker info, si docker est bien installer, il devrait renvoyer les informations lié au Docker*
Reçu : ☐
Reçu avec réserve : ☐ ……
Refusé : ☐ …… Commentaire :

Réception Etape 2 : *Mettre en place le container hello world avec la commande « docker run –rm=true hello world » afin de vérifier que le moteur de docker fonctionne correctement. Si c'est le cas, la machine devrait renvoyer le message « Hello from Docker ! »*
Reçu : ☐
Reçu avec réserve : ☐ ……
Refusé : ☐ …… Commentaire :