Images et système de fichiers

Images Docker

Une image Docker est un **fichier immuable** qui contient tout le nécessaire pour exécuter une application dans un conteneur.
Elle contient le code source, les bibliothèques, les dépendances, les variables d'environnement, les configurations, etc.

L'image Docker est une photographie figée d’un environnement logiciel, garantissant que l'application s’exécutera systématiquement de la même manière, indépendamment de son environnement.

Une image Docker est construite à partir d'un fichier de configuration appelé **Dockerfile.** Ce fichier décrit étape par étape comment assembler l'image.

> Nous reviendrons plus tard sur le Dockerfile.

Une fois construite, une image peut être stockée localement ou poussée vers un registre Docker (comme Docker Hub ou un registre privé), où elle peut être versionnée et partagée.

Pour identifier une image Docker, on utilise la syntaxe suivante :

```
propriétaire/nom:tag
```

- **propriétaire :** correspond à l'utilisateur ou à l'organisation qui possède l'image sur le registre.
- **nom :** le nom de l’image, comme `nginx`, `ubuntu`, etc.
- **tag :** une étiquette qui permet de versionner l'image. Par défaut, si aucun tag n'est spécifié, Docker utilise `latest`.
  Les tags peuvent représenter des versions (1.0, v2.3.4) ou des environnements (dev, prod).

>

> Certaines images "officielles" ne possèdent pas de propriétaire, donc il n'est pas nécessaire de le préciser.

Quelques exemples :

```
nginx:latest # image officielle de Nginx, version la plus récente.
thibaud/appli:1.2.0 # image personnalisée appartenant à l'utilisateur thibaud, version 1.2.0.
```

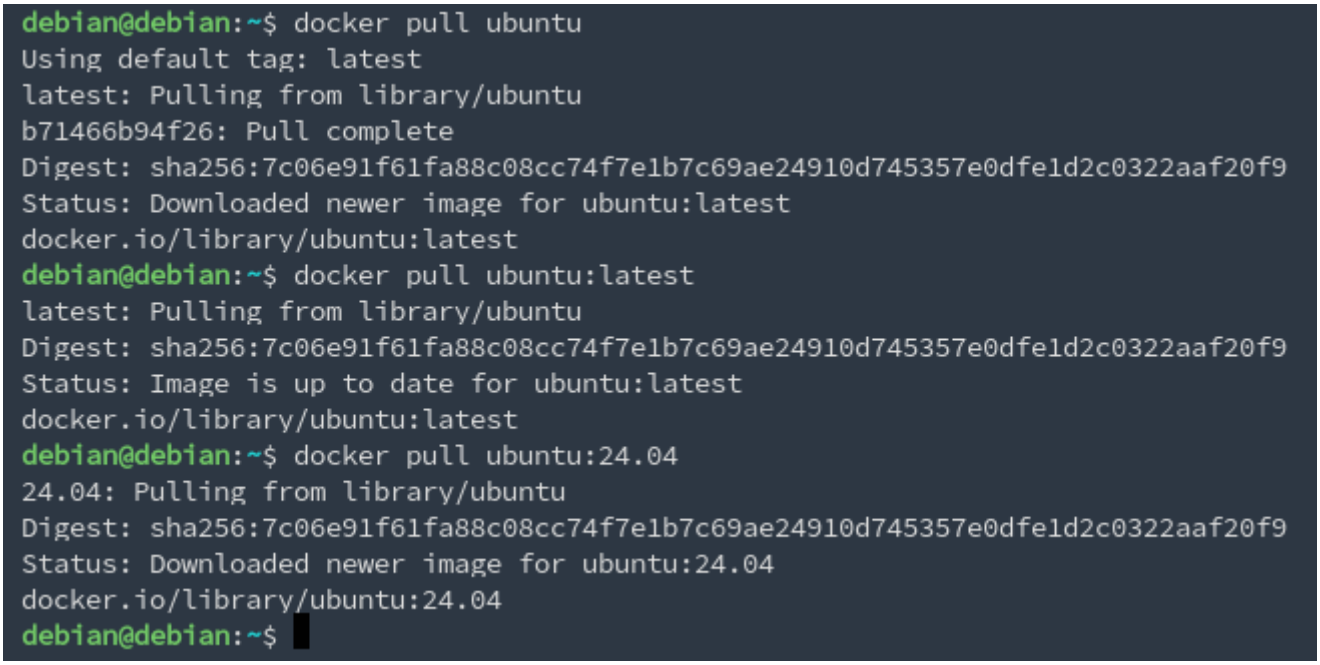
Registry Docker Hub

Comme expliqué ci-dessus, les images Docker doivent être stockées sur un **registre**.

Un registre peut être **public** ou **privé**, par exemple au sein d'une entreprise afin d'y stocker une application propriétaire.

Le registre public et par défaut est le Docker Hub. Docker Hub

La commande `docker pull` permet de télécharger une image depuis un registre, par défaut le registre public **Docker Hub**.



Actuellement, l'image `latest` d'Ubuntu correspond à la version 24.04 sur le Docker Hub. Ces trois identifiants retournent la même image, qui n'est téléchargée qu'une seule fois : `ubuntu`, `ubuntu:latest`, `ubuntu:24.04`.

À l'inverse, si je lance un `docker run` sur une image qui n'est pas déjà stockée localement, le téléchargement est automatique depuis le Docker Hub :

```
debian@debian:~$ docker run -ti ubuntu:18.04
Unable to find image 'ubuntu:18.04' locally
18.04: Pulling from library/ubuntu
7c457f213c76: Pull complete
Digest: sha256:152dc042452c496007f07ca9127571cb9c29697f42acbfad72324b2bb2e43c98
Status: Downloaded newer image for ubuntu:18.04
root@c59694091d14:/#
root@c59694091d14:/# cat /etc/os-release
NAME="Ubuntu"
VERSION="18.04.6 LTS (Bionic Beaver)"
```

La commande `docker image` permet de gérer les images stockées localement.

```
docker image ls # lister les images
docker image rm ubutu:18.04 # supprimer l'image ubuntu taggée 18.04
```

```
debian@debian:~$ docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED       SIZE
debian        latest    047bd8d81940   3 days ago    120MB
hello-world   latest    1b44b5a3e06a   6 days ago    10.1kB
ubuntu        24.04     e0f16e6366fe   2 weeks ago   78.1MB
ubuntu        latest    e0f16e6366fe   2 weeks ago   78.1MB
ubuntu        18.04     f9a80a55f492   2 years ago   63.2MB
debian@debian:~$ docker image rm ubuntu:18.04
Untagged: ubuntu:18.04
Untagged: ubuntu@sha256:152dc042452c496007f07ca9127571cb9c29697f42acbfad72324b2bb2e43c98
Deleted: sha256:f9a80a55f492e823bf5d51f1bd5f87ea3eed1cb31788686aa99a2fb61a27af6a
Deleted: sha256:548a79621a426b4eb077c926eabac5a8620c454fb230640253e1b44dc7dd7562
debian@debian:~$ docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED       SIZE
debian        latest    047bd8d81940   3 days ago    120MB
hello-world   latest    1b44b5a3e06a   6 days ago    10.1kB
ubuntu        24.04     e0f16e6366fe   2 weeks ago   78.1MB
ubuntu        latest    e0f16e6366fe   2 weeks ago   78.1MB
debian@debian:~$
```

Système de fichiers

L'arborescence des fichiers dans un conteneur Docker est **isolé** du reste de la machine.

Un conteneur Docker contient une arborescence Linux. Exemple ci-dessous :

```
debian@debian:~$ docker run -ti ubuntu bash
root@a9a69b04bd71:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
```

L'arborescence ci-dessus n'est accessible que depuis le conteneur.

- Le conteneur n'a pas accès à l'arborescence de la machine hôte.
- La machine hôte n'a pas accès à l'arborescence du conteneur.

```
debian@debian:~$ echo "Docker c'est pratique" > /home/debian/test.txt
debian@debian:~$ cat /home/debian/test.txt
Docker c'est pratique
debian@debian:~$
debian@debian:~$ docker run -ti debian bash
root@fa14d7193dc3:/# cd /home
root@fa14d7193dc3:/home# ls
root@fa14d7193dc3:/home# mkdir debian
root@fa14d7193dc3:/home# echo "Docker c'est fabuleux" > /home/debian/test.txt
root@fa14d7193dc3:/home# cat /home/debian/test.txt
Docker c'est fabuleux
root@fa14d7193dc3:/home#
root@fa14d7193dc3:/home# exit
exit
debian@debian:~$ cat /home/debian/test.txt
Docker c'est pratique
debian@debian:~$
```

Volatilité

Contrairement à une **image** docker qui est figée, le contenu d'un container est **volatile**.

> Lorsqu'un container est supprimé, tous les fichiers modifiés sont perdus !

Exemple :

```
debian@debian:~$ docker run -ti --name=test debian
root@a4a1252f5a65:/# echo "Coucou Docker !" > /root/coucou.txt
root@a4a1252f5a65:/# cat /root/coucou.txt
Coucou Docker !
root@a4a1252f5a65:/# exit
exit
debian@debian:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS                      PORTS     NAMES
a4a1252f5a65   debian    "bash"    36 seconds ago   Exited (0) 6 seconds ago              test
debian@debian:~$ docker rm test
test
debian@debian:~$ docker run -ti --name=test debian
root@78368e00949e:/# cat /root/coucou.txt
cat: /root/coucou.txt: No such file or directory
root@78368e00949e:/# exit
exit
debian@debian:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS                      PORTS     NAMES
78368e00949e   debian    "bash"    24 seconds ago   Exited (1) 2 seconds ago              test
debian@debian:~$
```

Les deux conteneurs nommés `test` à partir de l'image `debian` ne sont pas les mêmes : ils n'ont pas le même identifiant.

Les données du premier conteneur sont perdues lors de sa suppression.

> Nous verrons plus loin les **volumes Docker** qui permettent de stocker des données persistantes.

Isolation : un exemple avec fastfetch

> L'hôte et le conteneur **partagent** quelques éléments tels que le noyau Linux, le CPU, la RAM disponible et l'espace disque.
> À l'inverse, la distribution Linux, les paquets installés et le système de fichier sont **isolés**.

Revision #37
Created 16 July 2025 08:45:32 by Thibaud FRICHET
Updated 5 September 2025 08:54:09 by Thibaud FRICHET