

# Docker

## Qu'est ce que Docker ?

Meme

Docker est un système de virtualisation légère. En gros, les conteneurs docker sont des machines virtuelles où on ne virtualise pas le matériel (contrairement à une VM VirtualBox ou VMWare), ni le système d'exploitation. On utilise les capacités du noyau linux (comme LXC) pour conteneuriser les processus.

L'idée, c'est de créer des environnements applicatifs portables et prêt à l'emploi. Et ducoup c'est super pratique pour gérer une infrastructure sur un serveur, mais aussi pour lancer rapidement des applications (comme des bases de données notamment) pour développer sur notre poste.

Pour résumer, un conteneur docker c'est une machine virtuelle légère, qui contient les dépendances nécessaire à l'exécution d'un processus.

## Installer Docker

### Sur Linux (par exemple sur le serveur)

```
sudo apt update # Mettre à jours les paquets
sudo apt install apt-transport-https ca-certificates curl software-properties-common # Installer de quoi récupérer
la clé de sécurité de docker
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add - # Récupération de la clé de
sécurité de docker
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable" # Ajout de
dépôt APT de docker
sudo apt update # Mettre à jours les paquets
sudo apt install docker-ce # Installation de docker
sudo systemctl status docker # Pour vérifier que le service tourne bien
```

# Sur windows (par exemple votre poste)

Il faut avoir l'édition "Pro" de Windows ainsi qu'avoir son système à jours. Il faut tout d'abord installer WSL (Windows Subsystem for Linux) :

```
wsl --install
```

## Installation via Winget

```
winget install Docker.DockerDesktop
```

## Installation via Chocolatey

```
choco install docker-desktop
```

## Installation manuelle

Téléchargez et exécutez l'installateur via [ce lien](#).

# Concepts généraux

## Notion d'image

Une image docker, c'est la recette du conteneur, on la construit à partir d'un Dockerfile, qui va décrire ce que contient l'image en terme de dépendances, librairies et exécutable, ainsi que comment le processus doit être exécuté.

## Notion de conteneur

Un conteneur, c'est une instance d'une image en train de tourner.

## Docker Hub

Docker Hub, est un dépôt central qui permet de publier des images docker, pour permettre à d'autres gens de les utiliser facilement.

# Lancer un conteneur à partir d'une image

“ Pour la suite du tutoriel, ajoutez `sudo` à toutes les commandes docker si vous êtes sur Linux. Si vous êtes sur Windows, n'oubliez pas de lancer le docker daemon en lançant l'app "Docker Desktop".

Pour lancer une image docker, il faut utiliser la commande `docker run`. Essayons de lancer un conteneur à partir de l'image MySQL (au hasard), disponible à [ce lien](#).

La commande pour lancer le conteneur est la suivante :

```
docker run -p 3306:3306 --name mon-mysql -e MYSQL_ROOT_PASSWORD=monMdpTresSecret -d mysql:latest
```

Explications :

- `-p` : permet de spécifier les ouvertures de port du conteneur sous la forme `externe:interne` (externe = port de la machine hôte, interne = port du conteneur). Ici on forward le port `3306`, qui est utilisé par MySQL
- `--name mon-mysql` : donner le nom "mon-mysql" au conteneur
- `-e` : permet de passer une variable d'environnement au conteneur. Ici on passe `MYSQL_ROOT_PASSWORD` qui d'après la documentation de l'image permet de paramétrer le mot de passe de l'utilisateur `root` de la base de donnée. Les variables d'environnement suivantes de cette image peuvent aussi être intéressantes :
  - `MYSQL_DATABASE` : crée automatiquement dans le conteneur une base de donnée au nom correspondant à la valeur de cette variable
  - `MYSQL_USER` et `MYSQL_PASSWORD` : créer automatiquement dans le conteneur un utilisateur avec le login et password correspondant à la valeur de ces variables
- `-d` : permet de préciser l'image à utiliser et le tag, sous la forme `image:tag`. Le tag permet de préciser la version. On met souvent `latest` pour avoir le plus récent.

Et voilà, le tour est joué ! Le conteneur est lancé. Pour voir vos conteneurs actifs, vous pouvez faire :

```
docker ps
```

Pour voir les logs du conteneur vous pouvez faire :

```
docker logs -f mon-mysql
```

# Construire une image

Pour construire une image docker, il faut définir un `Dockerfile`. Pour démonstration, nous allons créer une image pour une application web Java faisant juste un "Hello World".

## Setup du projet

Pour commencer, récupérez le projet :

```
git clone https://github.com/Ombrelin/java-springboot-helloworld.git
cd java-springboot-helloworld
```

Pour le construire (Java 11 requis):

Windows	Linux
<code>./mvnw.cmd clean package</code>	<code>./mvnw clean package</code>

Cela va vous construire un JAR exécutable dans `target/build`.

## Le Dockerfile

Pour commencer à la racine du projet un dossier `Dockerfile`.

On commence par hériter notre image d'une autre, pour ré-utiliser ce qu'elle contient :

```
FROM adoptopenjdk/openjdk11
```

Cette image contient un JDK 11 déjà installé. On va ensuite exposer le port de l'application web, pour pouvoir y accéder en dehors de l'image :

```
EXPOSE 8080
```

Ensuite on copie notre fichier JAR de notre application dans l'image :

```
ADD ./target/demo-0.0.1-SNAPSHOT.jar app.jar
```

Enfin, on précise le point d'entrée de notre processus, tout simplement la commande qui exécute l'application :

```
ENTRYPOINT ["java","-jar","/app.jar"]
```

# Constuction à partir du Dockerfile

Pour construire une image à partir de votre Dockerfile :

```
docker build -t mon-image .
```

Vous pouvez ensuite lister vos images locales pour y voir apparaitre la votre :

```
docker images
```

Pour lancer votre image pour la tester, un simple `docker run` :

```
docker run -p 8080:8080 -t mon-image
```

Voilà, vous avez lancé un conteneur à partir de votre image, vous pouvez lancer votre navigateur à <http://localhost:8080> pour vérifier que l'application est bien lancée.

# Persister les données des conteneurs

Pour l'instant les données de vos conteneurs ne sont pas persistées à l'extérieur, elle disparaîtront si vous supprimez le conteneur, ce qui n'est pas très pratique pour une base de donnée par exemple.

Pour remédier à cela, on utilise la notion de volumes, afin de monter un dossier de notre système hôte, dans le conteneur docker, permettant à celui-ci de persister des fichiers.

Pour ce faire on va utiliser l'option `-v DossierMachineHote:DossierConteneur` . Par exemple pour persister les données de MySQL:

```
docker run -p 3306:3306 --name mon-mysql -v /un/dossier/sur/mon/ordi:/var/lib/mysql -e  
MYSQL_ROOT_PASSWORD=monMdpTresSecret -d mysql:latest
```

# Pour aller plus loin

Les sujets suivants peuvent être intéressants :

- Docker Compose
  - Publier ses conteneurs sur Docker Hub
- 

Revision #17

Created 16 November 2021 21:15:13 by Arsène Lapostolet

Updated 27 November 2021 22:32:11 by Arsène Lapostolet