# En avant pour la reproductibilité en science des données avec Docker

## Introduction

Les conteneurs sont semblables à une forme de virtualisation qui permet d'isoler les applications. Néanmoins leur principe diffère de la virtualisation.

Toutes les fonctionnalités de la virtualisation du système d'exploitation ne sont pas disponibles, et nous n'en avons pas besoin non plus.

Il en résulte un temps de démarrage plus rapide !

Un développeur peut spécifier les ressources dont il a besoin, le système d'exploitation dont il a besoin pour exécuter l'application et il n'a pas à se soucier des dépendances puisque le conteneur s'en occupe.

## Comparaison entre les conteneurs et les machines virtuelles.

Une machine virtuelle (VM) est comme une copie d'un ordinateur physique réel utilisant une technologie d'hyperviser proche de l'émulation.

Plusieurs VM peuvent fonctionner sur le même matériel physique (environnement "multi-tenant"). En revanche, les conteneurs se trouvent installés sur un serveur physique et son système d'exploitation hôte, voir dans une VM.
Chaque conteneur partage le noyau de l'OS hôte cependant les binaires et les bibliothèques nécessaires à l'exécution de ces applications lui sont spécifiques.

Un conteneur Docker est semblable à un ordinateur dans votre ordinateur. On peut aussi le voir comme un serveur distant. On peut partager l'image d'un conteneur (une copie à un instant t, re-executable). On peut donc diffuser des logiciels, ou des données via un conteneur dans une version fixe. (Ça fait penser au tag de git non ?) Ainsi à partir des mêmes sources répliquées, les autres utilisateurs obtiendront les mêmes résultats. En avant pour la reproductibilité en science des données !
Ainsi, lors du déploiement, la gestion de toutes les dépendances complexes depuis le système d'exploitation jusqu'aux versions des paquets sont supprimées. Par conséquent, cela facilite la portabilité et le partage.

## Architecture

Docker utilise une architecture client-serveur, qui implique les trois principaux composants

- le client Docker,
- le démon Docker
- le registre Docker (dockerhub par défaut).

Le client Docker communique avec le démon Docker, qui se charge de construire, d'exécuter et de distribuer les conteneurs Docker.
Le client et le démon Docker peuvent fonctionner sur le même système ou connecter un client à un démon Docker distant.

Nos conteneurs fonctionnent de la même manière qu'un processus, mais la différence est que les conteneurs sont traités avec leur environnement complet. Les conteneurs peuvent avoir les états décrits ci-dessus : créé, en cours d'exécution, en pause, arrêté et supprimé.

Les données stockées à l'extérieur du conteneur peuvent être utilisées même si le conteneur n'existe plus. Cependant, pour cela il est nécessaire de monter les données à l'extérieur du conteneur.

Remarque :

Autres Solution techniques pour le déploiement d'application reproductibilité.
Installation depuis les sources git
Package Conda
VM : en local virtualbox ou dans le cloud
Autre système de gestion de Container comme Singularity

**Installation de docker**

Linux:
https://docs.docker.com/engine/install/ubuntu/

Windows:
voir https://docs.docker.com/docker-for-windows/install/

MacOs:
voir https://docs.docker.com/docker-for-mac/install/

Installation possible dans une VM par exemple avec Virtualbox(Windows, MacOS, Linux)
voir virtualbox