# Cours : Présentation de Docker

## Qu'est-ce que Docker ?

Docker est une **plateforme open source** qui **automatise** le déploiement, la gestion et l'exécution d'applications dans des **conteneurs légers**.

Un conteneur est :

- une **unité logicielle**
- qui **encapsule** une application et ses dépendances,
- garantissant ainsi sa **portabilité** et son exécution cohérente **quel que soit l'environnement** (Windows, Lonux, MacOsX, etc.).

Docker est une **technologie de conteneurisation** reposant sur le **noyau Linux** et ses fonctionnalités de **virtualisation par conteneurs** (**LXC** pour Linux Containers), notamment :

- le composant **cgroups** pour contrôler et limiter l'utilisation des ressources pour un processus ou un groupe de processus (utilisation de la RAM, CPU entre autres) associé au système d'initialisation **systemd** qui permet de définir l'espace utilisateur et de gérer les processus associés ;
- les **espaces de noms** ou **namespaces** qui permettent de créer des environnements sécurisés de manière à isoler les conteneurs et empêcher par exemple qu'un groupe puisse **voir** les ressources des autres groupes.

Docker offre des outils pour utilise ces fonctionnalités de manière simplifiée pour permettre, entre autres :

- la **duplication** et la **suppression** des conteneurs ;
- l'**accessibilité** des conteneurs à travers la gestion des **API** et **CLI** ;
- la **migration** (à froid ou à chaud) de conteneurs.

Un conteneur Docker **se construit** à partir d'une **image**.

Une image Docker est un **package léger, autonome et exécutable** d'un logiciel qui inclut tout ce qui est nécessaire pour l'exécuter :

- **code** de l'application,
- **environnement d'exécution** (runtime),
- **outils système et librairies**, etc.

De nombreuses images sont disponibles sur :

- le **registre officiel** (appelé hub): https://hub.docker.com ;
- de nombreux dépôts initiés par de **simples** utilisateurs.

Il est bien sûr possible de proposer des images, d'en modifier d'autres et de déposer la modification sur le dépôt officiel.

## Comparaison de Docker avec une virtulisation classique

Lien : https://www.docker.com/what-container

**Utilisation d'un hyperviseur :**

- virtualisation d'une ou plusieurs machines physiques,
- Les VM intègrent elles-mêmes un système d'exploitation complet sur lequel les applications qu'elles contiennent sont exécutées.
- hyperviseur responsable de tous les échanges de données. L'exécuttion de plusieurs machines virtuelles sur un même serveur demande de grosses performances et un nombre suffisant de ressources pour assumer plusieurs machines virtuelles.

Le démarrage d'une machine virtuelle peut être plus ou moins long en fonction aussi de la technique utilisée (virtualisation complète ou paravirtualisation).

**Utilisation de Docker :**

Les conteneurs Docker s'exécutent sur une machine hôte :

- partagent le noyau du système d'exploitation de cette machine
- font directement appel à celui-ci pour exécuter les applications. Les conteneurs démarrent ainsi très rapidement et utilisent peu de ressources (processeur, mémoire vive, etc.).
- très légers : n'embarquent pas de système d'exploitation et qu'ils peuvent partager des fichiers communs
    - réduit l'utilisation du disque,
    - migrent plus facilement d'une machine physique à une autre

- téléchargements d'images sont beaucoup plus rapides.

Par ailleurs, la technologie mise en œuvre isole les applications les unes des autres et de l'infrastructure sous-jacente.

## La virtualisation complète d'un ordinateur

La virtualisation classique permet, via un hyperviseur, de simuler une ou plusieurs machines physiques, et de les exécuter sur le serveur hôte sous forme de machines virtuelles (VM). Ces VM intègrent elles-mêmes un système d'exploitation complet sur lequel les applications qu'elles contiennent sont exécutées. L'hyperviseur est donc responsable de tous les échanges de données. Exécuter plusieurs machines virtuelles sur un même serveur demande de grosses performances et un nombre suffisant de ressources pour assumer plusieurs machines virtuelles. Le démarrage d'une machine virtuelle peut être plus ou moins long en fonction aussi de la technique utilisée (virtualisation complète ou paravirtualisation).

La virtualisation complète, la création d'une VM nécessite :

- l'**installation complète** d'un système d'exploitation qui intègre un grand nombre de couche logicielles afin de permettre l'exécution d'une diversité de services ou d'applications,
- l'installation des **aplications.**

Cependant l'application ou le service qui sera configuré ne nécessite qu'**une partie des fonctionnalités de l'OS**. Par exemple la mise en place d'un serveur DNS ne nécessite pas la présence de composants multimédia ou de programmation comme python.

## La virtualisation avec Docker

Les conteneurs Docker s'exécutant sur une machine hôte partagent le noyau du système d'exploitation de cette machine et font directement appel à celui-ci pour exécuter leurs applications. Ils démarrent ainsi très rapidement et utilisent peu de ressources (processeur, mémoire vive, etc.). Du fait que les conteneurs n'embarquent pas de système d'exploitation et qu'ils peuvent partager des fichiers communs, ils sont très légers : cela réduit l'utilisation du disque, ils se migrent plus facilement d'une machine physique à une autre et les téléchargements d'images sont beaucoup plus rapides Par ailleurs, la technologie mise en œuvre isole les applications les unes des autres et de l'infrastructure sous-jacente.

## A chaque service une VM dédiée

Les pratiques courantes consistent à créer de nouvelles VM par duplication d'une image contenant un OS personnalisé et à assurer une **isolation forte** entre les VMs. Dans ce cas de figure on virtualise des systèmes d'exploitation identiques sur un même hôte physique.

Mais cela représente une consommation importante de ressources disque et de calcul pour implémenter l'application ou le service.

## Docker pour quoi faire ?

La technologie de conteneur mise en œuvre par Docker est une réponse à ces problématiques :

- Permettre la **création** de service s'exécutant sur un **système d'exploitation identique**,
- Partager une grande partie des **fonctionnalités** et des librairies de base du système d'exploitation,
- Assurer une **isolation** entre les VMs.
- **Limiter** la consommation de ressources : espace disque, mémoire, temps de mise en œuvre très réduit , administration et maintenance lors des mises à jour notamment.

Docker permet des usages qu'il était difficilement envisageable auparavant.

**Exemple** : un développeur web peut :

- très rapidement mettre en œuvre un **environnement vierge** pour réaliser ses **tests** ;
- puis **supprimer** cet environnement une fois les tests terminés afin de **libérer** des ressources.
- De plus il pourra tester son application dans des **environnements différents**.
- Intéressant s'il doit faire plusieurs fois.

C'est le principe des **micro-services** en facilitant la création de conteneurs Docker pour chaque service ou applications à provisionner selon les besoins.

Docker n'est ni un émulateur ni une solution de virtualisation. C'est un **logiciel client-serveur** basé sur des fonctionnalités de bas niveau du noyau Linux et ou de Windows. Si la virtualisation classique consiste à simuler le fonctionnement d'une machine physique en isolant les applications dans des systèmes d'exploitation situés dans des VMs différentes.

L'approche de Docker consiste :

- à assurer l'isolation entre application
- avec les fonctionnalités du système d'exploitation lui-même.

L'ordinateur contenant le système d'exploitation est central dans cette architecture et l'isolation est moins stricte.

Docker est ainsi associé à de la **virtualisation légère** et de ce fait est perçu comme fournissant moins de garantie d'isolation que la virtualisation classique. Docker est probablement à considérer comme **complémentaire** à la virtualisation classique.

Le terme **conteneur** décrit :

- une **image** de l'application ou du service **en lecture seule**,
- à laquelle on ajoute un espace accessible en **écriture**.
- L'arrêt du conteneur **efface les données écrites** : pas de persistance des données.

> Le **montage de volume** permet la **persistance des données** des conteneurs.

## Les limites des conteneurs Docker

Les conteneurs Docker ont quelques limites :

- **isolement relatif avec le système d'exploitation hôte**, les conteneurs peuvent être plus vulnérables, car ils partagent un noyau et des composants systèmes et leur fonctionnement exige déjà un niveau d'autorisation élevé (généralement l'accès root dans les environnements Linux) : si toute l'architecture est basée sur Docker et si le système hôte est attaqué tous les services seront « accessibles » et exposés plus facilement et rapidement aux attaques. À noter que les plateformes de conteneurs évoluent dans le sens d'une plus grande sécurisation en matière d'isolement et de séparation des droits des OS ;
- **attribution moins fine et stricte des ressources système et multiplication facile des conteneurs**, ce qui rend possible une consommation d'une grande quantité de ressources sans s'en rendre compte ;
- **très forte dépendance entre les conteneurs et le système hôte** qui fait qu'un conteneur Linux ne peut être exploité nativement que sur Linux (idem avec un conteneur Windows qui ne peut être exécuté que sur Windows) même si les choses commencent à évoluer.

## Comment fonctionne Docker

Docker se compose :

- d'un client,
  - d'une API,
  - et d'un serveur sous forme de démon.

  Docker dispose d'une fonctionnalité de dépôt central qui **Docker Hub**. Cette fonctionnalité est essentielle et consiste en un dépôt public, accessible avec une simple connexion Internet, qui permet de favoriser la réutilisation des images déjà préparées par des éditeurs de logiciels, des entreprises ou des particuliers. Ces contributeurs constituent la communauté qui s'est développée autour de la technologie Docker et c'est une des raisons principales de sa rapide adoption.

Site de Docker hub : https://hub.docker.com/

Sur Docker Hub :

- les images communautaires sont identifiables par leur nom préfixé par l'identifiant du propriétaire de l'image et séparé par le symbole /.
- les images officielles ne sont pas préfixées.

Vous pouvez également faire partie de cette communauté et partager vos propres images Docker en créant au préalable **votre compte** sur Docker Hub.

> **INFORMATION**
> Pour prendre connaissance des images officielles maintenues sur le dépôt GitHub nommé docker-library/official-images, accédez au site https://github.com/docker-library/official-images au répertoire library.

Docker c’est aussi :

- **Docker Compose** pour automatiser la mise en œuvre de plusieurs conteneurs en lien les uns avec les autres.
- **Docker Machine** pour provisionner des machines de déploiement de manière unifiée, et ce quel que soit le support utilisé, local ou dans un cloud. Il permet de réserver rapidement des machines pour déployer des conteneurs sur Amazon, Azure ou d’autres prestataires de Cloud.
- **Docker Swarm** pour administrer un cluster de machines Docker, le faisant apparaître comme une seule machine prête à recevoir de multiples conteneurs.

# Découvrir Docker

Docker propose un environnement de découverte appelé **Play with Docker (PWD)** :

- https://labs.play-with-docker.com/

Création d'une instance et vérification de la version de Docker

# Retour Accueil Docker

- Docker