**Outils de conteneurisation**

Avec les nouvelles tendances de la technologie, l'utilisation des conteneurs est présente presque partout dans le monde .

Cette technologie de conteneurisation a été élaborée grâce à des projets open source tels que LXC et son évolution en LXD, DOCKER, RKT de CoreOs

Dans le tableau ci-dessous nous allons comparer entre les technologies les plus utilisées pour la conteneurisation docker et LXD et rkt qui sont les trois open source.

| LXD | Docker | RKT |
| --- | --- | --- |
| * Les conteneurs sont vus comme des machines virtuelles mais sans système d’exploitation * Ne supporte que les distributions linux * Peut être integré en cloud avec OpenStack ou OpenNebula * Exécute plusieurs processus sur le même conteneur * N’est pas portable | * Peut créer plusieurs conteneur qui exécute une application * Est indépendant de la plateforme utilisée, cependant on ne peut pas exécuter les images linux sur windows et inversement. * Peut être intégré avec Kubernetes ou Jenkins. * Nécessite l'exécution d’un conteneur pour chaque processus * Est portable | * virtualisation des applications. * Est utilisé sous linux uniquement * Compatible avec les orchestrateurs tels que Kubernetes et Nomad * Est portable * Supporte les images docker * Est plus sécurisé |

**DOCKER**

**Définition**

Docker est une plateforme logicielle Open source qui permet de concevoir, tester et déployer des applications conteneurisées. Ce dernier regroupe le code et les dépendances d’une application comme un environnement de calcul isolé indépendamment du système d'exploitation et de l’infrastructure utilisés.

**Terminologie :**

**image docker:** Est un fichier utilisé pour exécuter le code dans un conteneur, ce dernier regroupe toutes les informations et les dépendances, ainsi que le code source et les configurations de déploiement et d’exécution à utiliser.

**Fichier dockerfile:** Fichier texte contenant toutes les instructions pour la création d’images docker.

**Création (build)** : Processus de création d’une image à partir du fichier Dockerfile, se lance avec la commande docker build.

**Registre (registry)** : Permet le stockage et la gestion des images docker. Il est comparable à un répertoire où l’on peut extraire ou héberger les images . Le registre par défaut utilisé pour la plupart des images publiques est Docker Hub et Docker Cloud mais les développeurs peuvent avoir leur propre registre privé.

**Conteneur Docker:** Est une instance de l’image docker. Un conteneur fonctionne avec une image docker, un environnement d'exécution et un ensemble de commandes. Un conteneur est lancé très rapidement et consomme peu de ressources, ce dernier est éphémère ce qui veut dire qu’il peut être détruit et reconstruit autant de fois que l’on souhaite.

**Volumes:** Permettent d'avoir un système de fichiers accessible en écriture que le conteneur peut utiliser. Ces fichiers sont stockés au niveau du système hôte et contiennent des données persistantes ou des données partagées.

**Docker Compose**: Est un outil de commande fournissant des métadonnées pour la définition et l’exécution d’applications multi-conteneurs. Il se base sur un fichier YAML pour la configuration et assure la création d’un conteneur par image en utilisant la commande docker-compose up

**Architecture de la technologie:**

Docker utilise une architecture client serveur avec trois composants principaux: le client, le docker daemon (serveur) et l’API rest et utilise le registre docker ainsi que les composants de réseaux et les volumes de stockages.

Client Docker: Communément appelé Docker, ce dernier joue le rôle client, il interagit avec le docker daemon à travers l’invité de commandes (CLI).

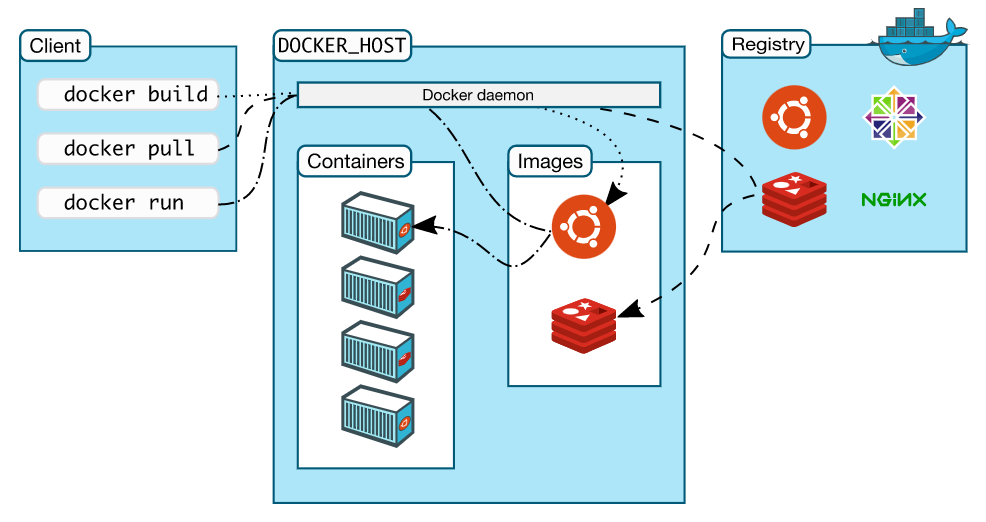
Les commandes les plus utilisées sont

* docker run: Pour instancier un nouveau conteneur
* docker pull: Pour récupérer une image du registre
* docker build: Pour construire une nouvelle image

Un Docker daemon appelé dockerd joue le rôle de serveur. Il se charge d’exécuter les requêtes des clients docker et est responsable de la création et de la gestion d’objets docker tels que les conteneurs, les images, volumes et réseaux. Il peut également communiquer avec les autres dockerd.

Rest API: Interface utilisée pour la communication client/serveur à travers des sockets UNIX ou l’interface réseau.

Lors de la création d’un container à partir des commandes clients comme docker run ou bien docker pull, docker utilise le registre pour récupérer les images configurées, par défaut, il utilise le docker hub qui héberge les images docker de toute la communauté, ou bien à partir d’un registre privé au niveau local.



architecture docker

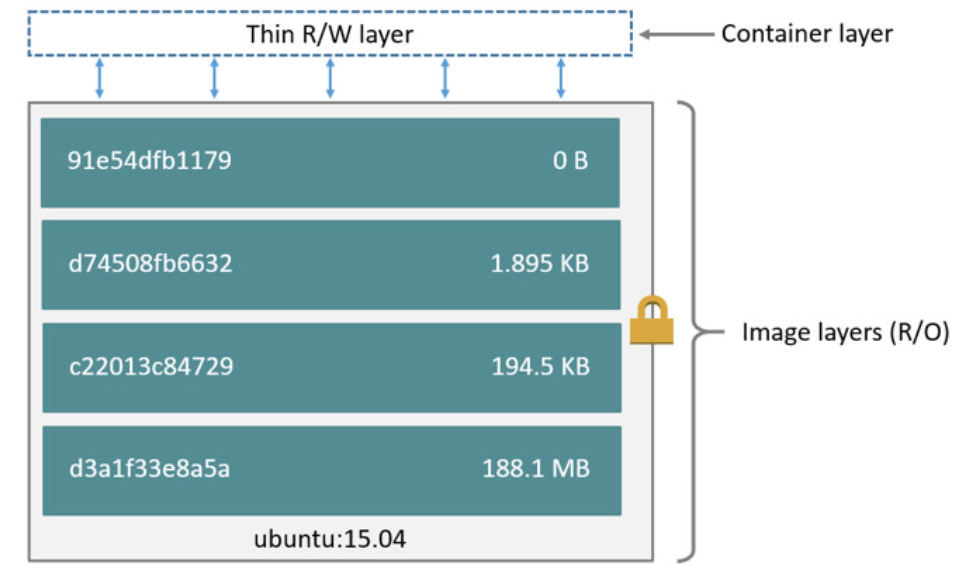
**liens vers l image: https://docs.docker.com/get-started/overview/**

.

**Principe de fonctionnement:**

Docker repose sur le principe d’isolation qui sépare les ressources du système d'exploitation hôte. Il utilise le noyau de linux et ses fonctionnalités comme les espaces noms (namespace) et les espaces de contrôles (Cgroups) et ainsi chaque conteneur docker aura sa propre adresse ip, son propre système de fichiers avec des limitations des ressources telles que le processeur et la mémoire**.**

Les images Docker sont basées sur un système de fichiers en couches. une couche représente un ensemble de changement que l’on effectue sur le système de fichiers de base. Ces changements sont effectués à partir des instructions au niveau du dockerfile. Ces couches sont empilées dans un ordre précis afin de construire une image, et peuvent être partagées entre plusieurs images. Ainsi, lorsque l'on récupère une image à partir du registre, cette dernière se télécharge couche par couche. Ces couches sont de type lecture seule, pour créer un conteneur une couche supplémentaire de lecture/écriture est ajoutée au sommet de la pile.



conteneur base sur image ubuntu 15.04