Le site officiel de Docker définit son outils comme tel: « Docker est un outil qui peut empaqueter une application et ses dépendances dans un conteneur virtuel, qui pourra être exécuté sur n'importe quel serveur Linux ». Le mot clé de cette définition est conteneur. Qu'est ce qu'un conteneur?

LE CONTENEUR

Le conteneur n'est pas une notion nouvelle, ce n'est pas une notion inventé par Docker. Linux a un système de conteneur qui s'appelle LXC (LinuX Container) qui permet de gérer cet empaquetage.

Un conteneur est globalement une sorte de boite, un peu comme une machine virtuelle, qui va être complètement isoler du système d'exploitation dans laquelle nous allons pouvoir installer toutes les librairies dont a besoin notre application pour

fonctionner ainsi que notre application. Le conteneur étant complètement isolé du reste nous allons pouvoir la distribuer un peu partout, indépendamment du système d'exploitation.

Pourquoi c'est bien?

Actuellement nos applications ont besoin de plus en plus de technologies pour fonctionner. Prenons un cas concret avec une application PHP.

Une application PHP a besoin d'une certaine version de PHP, d'images magiques (pour la conversion d'images), d'une base de données, d'un système de cache, d'un serveur web (Nginx ou Apache), d'un indexeur (Elasticsearch)… Pour au final nous retrouver avec une application nécessitant de nombreuses dépendances.

Le problème est que lorsque nous travaillons avec un administrateur système ou un hébergeur tier le déploiement de notre application peut rapidement s'avérer fastidieuse. Ces derniers devront installer sur chaque serveur de déploiement les dépendances requises pour le bon fonctionnement de notre application. En tant que développeur nous ne sommes pas forcément sensible à toutes les problématiques liées aux serveurs ce qui créé un climat hostile entre les développeurs et les administrateurs systèmes.

Le gros avantage du conteneur est que ce dernier va pouvoir être livré avec l'intégralité des dépendances liées à notre application.

Conteneur VS Machine virtuelle (VM)

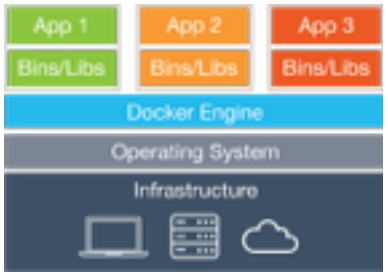
Ce que nous venons de décrire est exactement le principe de fonctionnement d'une VM. Pour comprendre la différence entre ces deux technologies voici un petit schéma issu du site officiel de Docker.

Virtual Machines

Voici la structure actuelle d'un serveur que l'on a avec des machines virtuelles. Nous nous retrouvons avec une infrastructure et par dessus un système d'exploitation qui va ensuite faire fonctionner diverses machines virtuelles. Pour schématiser nous

avons un ordinateur dans un ordinateur. Le problème de cette structure est la redondance des système d'exploitation. Si notre serveur héberge x machine(s) virtuelle(s), nous aurons x système(s) d'exploitation(s) installer sur notre serveur, ce qui consomme beaucoup de ressources mémoires et processeurs.

Le système de conteneur nous permet de nous absoudre de cette contrainte et te faire fonctionner nos application directement sur le système d'exploitation du serveur hôte. Nous avons ainsi plus besoin de virtualiser les différents systèmes d'exploitations de nos applications. Ce qui va alléger notre structure serveur.



Conteneur

Les avantages

Les avantages de l'utilisation de conteneur sont nombreux. Nous allons nous intéressez aux quatre principaux, les gains en performance, la probabilité des conteneurs, leur scalabilité et les facilités de déploiement.

Un système d'exploitation s'appuie sur de nombreux processus coûteux en mémoire et en CPU (temps de calcul d'un ordinateur). La réduction du nombre de systèmes d'exploitation à un unique OS tournant sur notre serveur hôte augmente considérablement les performance de ce dernier.

Le système de boite isolée rend la technologie des conteneurs beaucoup plus portable. Si nous voulons transférer une machine virtuelle d'un serveur A à un serveur B nous devons effectuer un snapshot[1] intégral de notre VM et le transférer. Le problème est qu'une machine virtuelle pèse lourd. Du coup le transfert d'une VM prendre beaucoup de temps.

Le troisième avantage du conteneur est qu'il est beaucoup plus facilement scalable. Nous allons pouvoir bouger nos petits boîtes très rapidement d'un serveur à l'autre, et même les faire évoluer en terme de performance.

Le dernier avantage que nous allons abordé est lié à la faculté de portable et scalable de nos conteneurs est le déploiement de ces derniers. Déployer un conteneur va être très simple. Vu qu'un conteneur est léger et qu'il embarque l'intégralité des dépendances nécessaires au bon fonctionnement de notre application nous allons pouvoir envoyer à notre

[1] État d'un système sauvegardé à un instant donné.

administrateur système ou notre hébergeur l'intégralité de notre environnement de production.

DOCKER

Pourquoi le conteneur devient-il intéressant maintenant?

Le système de conteneur n'est pas une chose nouvelle, mais son utilisation se répand à vitesse grand V. Mais pourquoi maintenant? Tout simplement grâce à Docker.

Docker nous permet d'utiliser et de faire fonctionner le système de conteneur d'une manière très simple. Docker met à disposition une série d'outils en ligne de commandes permettant aux développeurs de faire abstraction du déploiement, du fonctionnement et de l'export de l'environnement de notre application. Docker va se charger de tout pour nous.

Avant Docker le système de conteneurs était une technologie très intéressante mais trop complexe pour être utilisée par le plus grand nombre.

Aujourd'hui, avec Docker, même un développeur ne connaissant que très peu l'univers de l'administration serveur va pouvoir utiliser le système de conteneur.

Son principe de fonctionnement

Son entrer trop dans les détails Docker fonctionne globalement avec deux éléments (trois pour les serveurs Mac et Windows):

- Docker Deamon
- Docker Client
- Boot2Docker (Mac et Windows)

Docker Deamon

Docker Deamon est une application qui va tourner en tâche de fond sur notre serveur et qui va nous permettre de faire fonctionner nos différents conteneurs.

Docker Client

Docker Client est comme son nom l'indique un client qui va nous permettre de communiquer avec Docker par le biais de

commande dans notre terminal (créer, supprimer, déplacer, éditer un conteneur ...).

Boot2Docker

Boot2Docker est lié à la compatibilité de Docker sur les divers systèmes d'exploitation. Actuellement Docker ne fonctionne que sur Linux. Pour les utilisateurs Mac et Windows Docker fournit l'outil Boot2Docker qui permet de lancer une micro machines virtuelles Linux qui intègre les outils de Docker.

Comment cela va nous permettre de développer de meilleur logiciel?

Lorsque votre application est dans des conteneurs Docker, vous n'avez plus à vous inquiéter à propos de la mise en place et du maintient de l'environnement ainsi que des différents outils et langages de programmation.

Afin d'éviter le casse-tête de la configuration de l'environnement de notre application Docker met à la disposition des développeurs Hub Docker, un site web collaboratif composé d'une multitude d'image système (frameworks) qui vont servir de base à notre conteneur. Ses images ne contiennent que les fichiers systèmes de notre OS. Il faut comprendre par là que nous n'aurons pas un système d'exploitation mais seulement accès aux librairies et aux binaires de notre OS). La communauté Docker st très actives et propose déjà plus d'un millier de frameworks.

L'isolation des conteneurs Docker offre une totale liberté au développeur sur ces choix de technologies et de langages. Le développeur n'a plus à ce soucier de la compatibilité de ces dernières et peut ainsi utiliser les meilleurs outils pour son application.