Présentation de Docker

Qu’est-ce que Docker ?

« Docker est un logiciel libre destiné aux développeurs et administrateurs systèmes, dont l’objectif est de faciliter le développement, la diffusion et le déploiement d’applications web autonomes. Son principal intérêt est d’assembler les briques d’une application en conteneurs pouvant être partagés, sous forme d’images, au travers d’un registre (“docker registry”) et de les exécuter quels que soient la plateforme et l’environnement. Grâce à Docker, les applications deviennent portables et exécutables n’importe où : sur des ordinateurs portables Mac ou Windows, des machines virtuelles, de serveurs de production, etc. » (source https://www.disko.fr/reflexions/technique/introduction-docker/)

Docker a été initialement développé par Solomon Hykes pour un projet interne à la société de plateforme en tant que service (platform as a service - PaaS) qu’il a créée en 2008, dotCloud. Docker a été distribué en tant que projet open source en 2013, projet qui a rapidement été très actif avec de nombreux contributeurs. Selon Wikipédia « En décembre 2017, le projet a été mis en favoris plus de 46 000 fois sur GitHub, avec plus de 13 500 forks et 1 700 contributeurs ». Vous trouverez un entretien très intéressant de Solomon Hykes qui parle de son produit ici.

Le nom Docker recouvre le logiciel lui-même, la technologie et les outils qu’il embarque, le projet d'une communauté Open Source (communauté Open Source Docker), la nouvelle société créée par Solomon Hykes (*Docker inc*) en charge du développement qui s’appuie sur le travail de la communauté et le nom de la plateforme officielle en ligne.

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 utilise des fonctionnalités natives au noyau Linux, comme les cgroups ou les namespaces, mais offre les outils pour le faire 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.

Les conteneurs Docker sont construits à partir des images Docker. 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. Disponible pour les applications basées sur Linux et Windows, le logiciel conteneurisé fonctionnera toujours de la même manière, quel que soit l'environnement. Ainsi, les conteneurs Dockers offrent une grande flexibilité qui permet de les créer, déployer, mettre à jour, copier et déplacer d'un environnement à un autre.

De nombreuses images allant de la simple application comme Nextcloud à un système d’exploitation complet comme Debian sont disponibles sur : • le registre officiel (appelé hub) : https://registry.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.

Pour terminer cette rapide présentation, il est utile de préciser que les conteneurs Docker sont actuellement pris en charge par les principaux acteurs du cloud. Ainsi, un conteneur Docker peut se déployer en de multiples instances sur toutes les principales distributions Linux, Microsoft Windows, et sur toutes les infrastructures, y compris les machines virtuelles, bare-metal mais aussi dans le cloud. Les forces de Docker face à une virtualisation classique 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).

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.

Les limites des conteneurs Docker Les conteneurs Docker souffrent tout de même de 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.

Docker pour quoi faire ? Alors que Solomon Hykes, le CTO et fondateur de Docker, quitte l'entreprise, les résultats sont là et il ne se passe pas une semaine sans qu'une annonce autour de Docker ne soit faite.

On peut lire ici (article de journal du net) « Docker Inc revendique une communauté massive d'utilisateurs, avec à la clé plus de 200 groupes actifs à travers le monde. Inaugurée en 2014, en parallèle de la sortie de Docker 1.0, la forge lancée par la société de San Francisco (le Docker Hub), et via laquelle les développeurs peuvent partager leurs containers, atteint à ce jour 3,5 millions d'applications containeurisées. Depuis son ouverture, la plateforme a enregistré pas moins de 37 milliards de téléchargements de containers (ou pull en langage technique). « L'augmentation du nombre de pull sur le Docker Hub est exponentielle. Il atteint désormais le chiffre pharaonique de 1 milliard toutes les deux semaines », confie Patrick Chanezon. ».

Cet article fait également état de 450 clients payants dont plusieurs dizaines sont français, parmi lesquels une majorité de groupes du CAC40.

Pourquoi tant d'engouement ? Quelles sont les utilisations possibles, efficaces et pertinentes de Docker ? • Déployer rapidement un service lorsque l'on a besoin de le déployer plusieurs fois : cette reproductibilité est la base de docker, c'est typiquement l'utilisation que peut en faire un fournisseur de cloud. • Distribuer une application : Docker en tant que "système de distribution d'une application" est de plus en plus utilisé tant par les développeurs qui créent rapidement un serveur de développement et n'ont plus à packager une application sous différents systèmes (deb, rpm, etc) avant de la distribuer que par l'utilisateur qui utilise une image docker (en général très légère) pour tester puis garder/migrer ou finalement jeter une application (plutôt que de compiler un tarball). Souvent quelques commandes suffisent pour avoir une solution comme Nextcloud ou Ansible opérationnelle. • Développer et tester une application, Docker permet : ◦ de concevoir une architecture de test plus agile, chaque conteneur de test pouvant par exemple intégrer une brique de l'application (base de données, langages, composants, ...), le développeur pourra tester sur la même machine plusieurs versions d'un même logiciel en inter changeant le conteneur correspondant. Par exemple pour une application PHP, on pourrait facilement tester plusieurs versions de PHP comme plusieurs versions d'Apache ou de Nginx ; ◦ de développer une application selon le concept d'architecture de micro-services avec pour chaque couche des conteneurs isolant les composants de l'application ; ◦ faciliter le process de mise à jour de l'application : les images Docker sont versionnées et permettent une mise à jour simplifiée et maîtrisée. Le process de rollback est aussi simplifié : on redéploie la version précédente de l'image. ◦ d'avoir un environnement de développement identique à l'environnement de production d'autant plus que du fait de la disparition du système d'exploitation intermédiaire classique des machines virtuelles, les développeurs bénéficient d'une pile applicative plus proche de celle de l'environnement de production ce qui engendre mécaniquement moins de mauvaises surprises lors des passages en production.