Notes de cours: Les conteneurs

Master M1: DevOps – Méthodes et Outils

2020

Ces notes de cours sont un complément aux slides fournit sur les conteneurs.

1 Cloud et conteneurs

Les slides commencent par présenter le concept de Cloud Computing (qui peut être traduit par *Informatique dans les nuages* en Français). Le Cloud computing est présenté ici car il est le contexte dans lequel les technologies de conteneurs se sont développées.

Les slides présentent les différents modèles de Cloud et en particulier le modèle *Infrastructure-as-a-service*, où les clients déploient leur propre infrastructure logicielle au dessus des ressources matérielles obtenues au près d'un fournisseur.

Si une infrastructure Cloud offre de nombreux avantages, il y a aussi des problèmes à résoudre:

- Du point de vue du fournisseur:
  - Comment contrôler finement la consommation de ressources des utilisateurs?
  - Comment s'assurer que des utilisateurs, malintentionnés ou non, ne compromettent pas le bon fonctionnement de la plateforme?
- Du point de vue des utilisateurs:
  - Comment déployer facilement leur application sur une infrastructure distante sur laquelle ils ont peu de contrôle?
  - Comment assurer un bon niveau d'isolation par rapport aux autres utilisateurs de la plateforme?

La réponse a toutes ces questions est d'utiliser une technique de *conteneurisation*. La forme la plus courante de *conteneurisation* sont les machines virtuelles. Les slides fournissent des détails sur les machines virtuelles. Plus récemment, des solutions appelées *conteneurs* ont été proposées.

Le point central qui différencie un conteneur d'une machine virtuelle est que la machine virtuelle exécute un système d'exploitation complet alors qu'un conteneur utilise le noyau du système d'exploitation de la machine hôte.

De part leur légèreté, il est très facile de créer et démarrer des conteneurs. C'est pourquoi leur intérêt va bien au delà du seul contexte du Cloud Computing. Ils sont très largement utilisés dans les approches DevOps.

2 Les conteneurs Docker

La suite du cours présente les technologies de conteneurs en se concentrant sur la technologie Docker. Toutes les technologies de conteneurs se ressemblent. Ainsi, la plupart des points présentés dans le cours resteraient valide dans une certaine mesure avec d'autres technologies. Docker a été choisi pour ce cours car c'est une technologie de conteneurs très populaire, avec beaucoup de documentations disponibles en ligne.

Il est à noter cependant que la technologie Docker évolue très vite, que ce soit au niveau des fonctionalités fournies, des APIs, ou du fonctionnement interne. Ainsi, il est fort probable que certaines parties du cours ne sont pas complètement à jour. Les concepts généraux restent cependant les mêmes.

2.1 Illustration du fonctionnement de Docker

Les slides 40 et 41 illustrent le fonctionnement de Docker:

- Le Dockerfile (décrit plus en détails dans la suite des slides) est le fichier texte qui permet de décrire comment construire une image docker. Ce fichier texte peut être stocké dans un dépôt git (notion de *infrastructure-as-code*).
- Docker permet de créer automatiquement une image à partir d'un Dockerfile (attention l'image de la figure 40 n'est pas claire: le résultat de l'exécution de *Build* est une image et pas un conteneur).
- L'image créée peut être publiée vers un registre
- Sur une autre machine (host 2), le même utilisateur ou un autre utilisateur peut récupérer l'image depuis le registre (commande `Pull`) et créer des conteneurs à partir de cette image (commande `Run`).

La figure du slide 41 illustre le fait que:

- Le client docker est l'agent en charge de récupérer les commandes émises par l'utilisateur et de les transmettre au *daemon* docker.
- Le *daemon* docker assure la création et la gestion des images et des conteneurs sur la machine hôte, et gère aussi automatiquement les interactions avec le ou les registres.

Le slide 42 illustre l'intérêt de l'approche fondée sur des couches dans la diffusion des images:

- On suppose qu'on part d'une image pour l'application A.
- On modifie cette image pour créer une nouvelle version de l'application. Pour cela, 2 nouvelles couches sont créées: une couche avec des modifications au niveaux des librairies du système, et une couche avec des modifications au niveau applicatif.
- La figure montre que pour déployer la nouvelle version de l'image, seules les modifications sont envoyées vers le registre et copiées vers les machines cibles.

3 Utilisation de Docker

La présentation des commandes Docker fournie dans les slides est loin d'être exhaustive mais elle couvre une grande partie des utilisations classiques de Docker

Un point (parmi d'autres) peut être difficile à comprendre à partir des informations fournies dans les slides, la notion de port et la gestion des ports avec Docker.

Notion de port: La notion de port apparaît très rapidement quand on manipule des conteneurs Docker. En effet, très souvent Docker va permettre de déployer des services auxquels il est possible d'accéder à distance en envoyant des requêtes. L'identification d'un point de connexion réseau se fait à l'aide d'un numéro de port (voir vos cours de réseau pour plus de détails).

Sur un serveur classique, on ne peut pas avoir deux services qui écoutent sur le même port. Ainsi si un utilisateur veux démarrer deux instances d'un même service, il doit définir un numéro de port différent pour chaque instance du service.

Avec Docker le problème devient différent. En effet, le port sur lequel écoute un service exécuté au sein d'un conteneur est le port du conteneur et pas de la machine hôte. On pourrait imaginer que Docker associe automatiquement le port du conteneur avec le même numéro de port sur la machine hôte. Cependant cette solution a un inconvénient principal: si le port est déjà utilisé sur la machine hôte, on ne peut plus démarrer le conteneur. Et ce scénario peut être fréquent car il correspond au démarrage de deux instances du même conteneur sur la même machine.

Pour résoudre ce problème, Docker offre la possibilité de définir au démarrage d'un conteneur, l'association entre un port du système hôte et un port d'un conteneur. Par exemple, considérons une image d'un serveur Web écoutant sur le port 80. Il est possible de démarrer 2 instances de cette image sur la même machine, avec une instance `A` pour laquelle le port 80 du conteneur est associé au port 8000 de l'hôte et une instance `B` pour laquelle le port 80 esy associé au port 8001 de l'hôte. Tout requête envoyée vers le port 8000 (resp. 8001) de la machine hôte sera alors redirigée automatiquement vers le port 80 de l'instance `A` (resp. `B`).