Docker pour le développement logiciel et la recherche reproductible

1ère session LIFTech'

Auteur: Florent Jaillet

Date: 5 novembre 2015

Motivation

- Intérêt pour Docker suite à des présentations
- Volonté de tester car semblait utile pour le développement et la reproductibilité
- LIFTech' : une bonne occasion de s'y mettre et de vous remonter ce que j'ai repéré
- Je ne suis pas du tout spécialiste de Docker, toute précision ou correction est bienvenue
- Je compte utiliser Docker par la suite et propose de former un groupe de travail au LIF si d'autres sont intéressés

Plan

- Présentation de Docker (Build, Ship, Run)
- Intérêt de Docker pour le développement logiciel
- Intérêt de Docker pour la recherche reproductible

Qu'est-ce que Docker ?

- Une plate-forme au développement et au succès très rapide qui fournit de nombreux outils pour les conteneurs.
- Dans cette présentation, on ne s'intéresse qu'à l'outil principal : **Docker Engine**
- D'après l'aide de Docker (docker --help) :

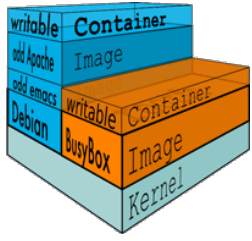
  A self-sufficient runtime for containers.
- Le site docker.com précise pourquoi il a été conçu :

  Build, Ship, Run

  An open platform for distributed applications for developers and sysadmins

Qu'est-ce qu'un conteneur logiciel ?

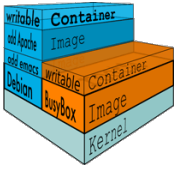
- Virtualisation de l'environnement d'exécution (processeur, mémoire vive, réseau, système de fichier…) et non de la machine
- Isolation de l'espace utilisateur, mais noyau partagé par le système et l'ensemble des conteneurs
- Permet notamment le partage souple des ressources avec un surcoût très faible comparé à une machine virtuelle
- Une longue histoire (Unix chroot 1979, FreeBSD jails 2000, OpenVZ 2005, LXC 2008, Docker 2013)



Quelques caractéristiques de Docker

- Combinaison et adaptation de différents concepts et outils éprouvés (conteneurs, scripts, gestion de version, gestion de paquets) avec une approche DevOps
- Principal intérêt : simplicité d'utilisation et auto-suffisance
- Logiciel libre écrit en Go
- Pour l'instant uniquement des conteneurs pour le noyau Linux
- Disponible sous Linux, sous Windows et Mac via la **Docker Toolbox** fournissant une machine virtuelle Linux minimaliste, et sous de nombreuses infrastructures nébuleuses

Création de conteneurs (Build)



- Notion d'image permettant une construction multicouche des conteneurs
- Possibilité de créer une nouvelle image en modifiant une image existante

build	Génération automatisée d'images avec langage de script spécialisé dans un **Dockerfile**, exemple : FROM docker/whalesay:latest RUN apt-get -y update && apt-get install -y fortunes CMD /usr/games/fortune -a | cowsay
commit	Création d'une image à partir des modifications lors de l'exécution d'un conteneur
history	Affichage de l'historique d'une image (couches avec possibilité de dépiler)
save, **load**, **export**, **import**	Sauvegarde et rechargement d'images et de conteneurs

Dépôts de conteneurs (Ship)

search	Recherche d'images sur le **Docker Hub** (hub.docker.com)
pull, **push**	Récupération / envoi d'image depuis / sur un dépôt (par défaut le Docker Hub)

- Nombreuses images officielles (gcc, java, python, golang, ruby, haskell, perl, centos, fedora, ubuntu, debian…)
- Nombreuses images publiques d'utilisateurs ou d'organisations
- Versionnage des images
- Possibilité d'utiliser son propre dépôt avec **Docker Registry**, notamment disponible sous forme d'image Docker officielle

Exécution et supervision de conteneurs (Run)

daemon	Lancement de la partie serveur (démon) de l'architecture client/serveur de Docker
run	Lancement d'une commande dans un nouveau conteneur construit à partir d'une image
start, **stop**, **restart**, **pause**, **unpause**, **kill**	Contrôle de l'exécution des conteneurs
ps, **logs**, **top**, **stats**	Surveillance des conteneurs

- Le lien entre plusieurs conteneurs se fait via des liens réseau
- Utilisation de volumes pour le partage et la persistance de données
- Possibilité de coordonner l'exécution de multiples conteneurs (**Docker Compose**)

Intérêt pour le développement logiciel

Possibilité de tirer profit de Docker à de nombreux niveaux du développement :

- programmation et débogage : partage d'un environnement entre développeurs, résout les difficultés de configuration des dépendances et facilite notamment l'arrivée de nouveaux développeurs et l'utilisation de machines multiples
- test : possibilité de tester facilement avec diverses versions de composants (compilateurs, versions de bibliothèques, distributions Linux…)
- intégration continue : automatisation très facile
- exécution : facile à reconstruire et faire évoluer, peut servir de base pour environnement d'utilisation ou de démonstration facile à déployer, diffuser et utiliser

Limites :

- Actuellement uniquement pour le développement sous Linux
- Utilisation d'interfaces graphiques pas immédiate (mais possible)

Approches possibles pour le développement logiciel

- Construction hiérarchique d'images (exemple : couche d'exécution + couche de test + couche d'intégration continue)
- Utilisation de plusieurs images coordonnées (exemple : image d'exécution, image de compilation, image avec environnement de développement…)
- Pour la programmation, possibilité de choisir jusqu'à quel niveau les outils sont placés dans Docker, par exemple possibilité d'utiliser un environnement de développement intégré en local sur la machine avec compilation et débogage distant dans un conteneur Docker

Intérêt pour la recherche reproductible

- Facilité d'intégration dans le flux de travail (utilisation simple, surcoût en temps de calcul négligeable, déploiement facilité sur tout système dont grappe de calcul...)
- Sauvegarde simple de l'environnement d'exécution lié à une publication (**export** ou **save**) et des informations pour le reconstruire et le faire évoluer (**Dockerfile**)
- Partage, distribution, réutilisation, démonstration grandement facilités

Limites :

- Résultats dépendants du noyau Linux utilisé
- Actuellement uniquement pour Linux, nécessité d'une machine virtuelle sous Windows et Mac
- Nécessite une certaine discipline

Plus d'informations : Carl Boettiger. 2015. An introduction to Docker for reproducible research. SIGOPS Oper. Syst. Rev. 49, 1 (January 2015), 71-79.