Docker & DevoOs : adopter les mêmes pratiques sous Windows et Linux !

Speaker : Julien Corioland (Microsoft), Adrien Blind (SG)
Format : Conférence
Date : 7 avril 2017

Sondage : une trentaine d'Ops dans la salle ☺

Rappels sur Docker

A la base, Docker est une technologie d'Ops mais pour les développeurs. C'est pourquoi cela a bien pris. Docker = container + tout l'écosystème.

Propriétés intéressantes :

- Immutabilité. On reconstruit une image. On ne la met pas à jour.
- Full Stack versionning : dans un même artefact, on a de l'OS, du middleware et d'application. Artefact auto-portant. Tout est packagée. Passe sur tous les environnements. Artefact construit automatiquement et versionné. Le cycle de vie de l'application et de l'environnement sont réconciliés.
- Réorganise les rôles et les responsabilités : pour construite une image Docker, il y'a un travail conjoint entre Ops et Dév.
- Le Docker container intègre le paradigme du Continuous Delivery avec Jenkins et le repo d'entreprise Nexus.

Virtualisation vs Isolation

Docker est de l'isolation au sein d'un même OS.
Depuis Windows Server 2016 et la dernière maj de Windows 10, on a des conteneurs.
2 types de container : Windows Server containers (comme Docker) et Hyper V-containters (embarque une machine virtuelle). Ce n'est qu'au déploiement qu'on choisit le type de container (Windows Server vs Hyper-V).

Le docker engine a été porté sur Windows (travail conjoint entre Microsoft et Docker inc.).
Même Dockerfile, même tooling.
Le format d'image est le même sous Linux et Windows. Dans la Registry Docker, on retrouve par exemple l'image microsoft/dotnet.

Démo depuis une session Powershell
> docker version
> type ./Dockerfile
> docker build –t devoxx/iis
> docker images
> docker run –d –p 80 :80 devoxx/iis ping localhost –t
Prend du temps car lancé depuis Windows 10 : Windows Server Core doit tout d'abord booter avant de démarrer le conteneur
> docker ps

La tendance est aux microservices : une application est décomposée en plusieurs composants qui dialoguent entre eux. Colle bien avec le conteneur qui n'est pas fait pour les gros monolithes (même si on peut).

La data n'est plus montée dans le conteneur avec les points de montage. Privilégier l'utilisation de volumes Docker.
Ces composants parlent entre eux au travers du Docker Networks. Permet de ne pas faire sortir leurs échanges à l'extérieur. Arrivée des topologies décrivant la stack et les composants.

CaaS platform avec Docker Swarm pour l'orchestration.
Dans Docker 1.12 l'orchestration a beaucoup changé : il y'a plusieurs managers et workers dans le cluster. Managers et workers dialoguent de manière sécurisée en TLS.
On peut également créer des services : plusieurs instances de la même image => on peut ordonner la création de 4 réplicas du même service.
Tous les conteneurs du même réseau peuvent dialoguer entre eux.
Load balancer intégré à Docker (IPVS – load-balancer de niveau 4). Il suffit de connaître l'IP que d'un seul nœud du service.
Mécanisme de DNS dynamique intégré à Docker : afin de dialoguer entre eux, les microservices peuvent utiliser le nom de l'image comme host.
Lorsqu'un nœud du cluster tombe, les conteneurs sont déployés sur d'autres nœuds.

Démo d'un cluster hybride avec nœuds Windows et Linux