Kit d'orchestration avec Docker Swarm mode

Speaker : Vincent Demeester (Docker Inc.)
Format : Conférence
Date : 6 avril 2017

Vincent est développeur sur Docker Core depuis 1 an et demi.

L'orchestration

On parle souvent de cluster de machines. L'orchestration va permettre de scheduler des containers. Nécessaire lorsque l'infrastructure comporte trop de serveurs.
Facilité la maintenance, la scalabilité. Plus d'agilité. Permet de ne plus avoir à penser le « comment je dois déployer mon application » => rôle délégué à l'orchestrateur

Fonctionnalités d'un orchestrateur :

- Placé, démarré, déployé une application
- Respecter des contraintes (ex : utilisation d'un disque SSD)
- Faire du load-balancing
- Pas de coupure de service
- Self-healing : l'orchestrateur s'occupe de redéployer l'application à un autre endroit lorsque c'elle ci est KO
- Flux sécurisés
- Provisionning de nouvelles machines

Plusieurs outils :

- Containers bases
  - Swarm (Docker)
  - Kubernetes (Google)
  - Rancher (Rancher)
  - Fleet (Coreos) -> mort car passe sur Kubernetes
- Plus généraux
  - Nomad (Hashicorp)
  - Mesos (Apache)
- O.V.N.I => lock-in sur les providers
  - Amazon web services
  - Google Cloud (GCE)

Derrière les orchestrateurs, on retrouve des outils :

- Service Discovery
  - etcd
  - consul
  - zookeeper
- Provisionning

    - terraform
    - infrakit
    - chef, puppet, ansible, saltstack
- Monitor
    - Prometheus

Docker

Docker, c'est
- Une société : Docker Inc.
- Une plateforme : dockerd (engine)
- Des outils : compose …

Architecture de Docker

3 notions principales :
1. Conteneur (runtime) : basé sur une image
2. Image : template en read-only
3. Registry : similaire à Maven Central pour les images Docker

Docker Swarm

En anglais, Swarm est un « nombre élevé d'abeilles qui se déplacent ».
Confusion : dans Docker, il y'a 2 projets : swarm mode et docker/swarmkit project

Swarm V1

Outil d'orchestration utilisant les API de Docker.
Désavantages : scaling, discovery, load-balancing … non faisables

Swarm mode

Extension de Docker pour que ce dernier devienne un orchestrateur et que la partie orchestrateur de Docker puisse être utilisée avec d'autres types de conteneurs (ex : Rocket de CoreOS).
Swarm mode = docker/swarmkit + docker engine, à partir de la version 1.12

Sécurisé : tous les nœuds discutent en TLS. Docker Swarm met régulièrement à jour les clés.

Docker Swarm part d'un modèle déclaratif pour arriver à l'état souhaité. Load-balancing.

Cluster : un nœud ou plus. Système d'élection
Autres notions :
- Services : état désiré, va créer des tâches
- Tasks : correspond à un container spécifique, immutable
- Secrets, Networks, Node, Volumes, Stack

Démo

6 nœuds docker

docker-machine ssh vdedemo1
docker swarm init

Génère une ligne de commande (contenant un token) afin que les autres nœuds puissent rejoindre le cluster.
Nécessité d'exécuter cette ligne de commande sur les 5 autres nœuds

Lister les nœuds :
docker node ls

Il faut désigner un autre manager au cas ou le nœud 1 tombe.
docker node update –role-manager vdedemo2

Utilisation de Docker Swarm visualize : https://github.com/dockersamples/docker-swarm-visualizer

Démarrage d'un ping sur une image alpine. Le cluster choisi un nœud pas trop chargé et crée une tâche exécutant le ping.

Création d'un serveur web sur le port 80. On peut y accéder de n'importe quelle adresse IP. Nul besoin de savoir sur quel serveur a été déployé l'application.

Création d'un réseau :
docker network –driver overlay –opt encrypted exquisite

Création d'une base Mongo sur le réseau :
docker create –name mongo –network exquisite mongo :3.3.8

La base Mongo est démarrée sur le worker 6 une fois l'image téléchargée.

Création d'un back Java rattaché au réseau avec limitation de mémoire :
Docker service create –name back –network exquisite –limit-memory 64M –reserve-memory 64M vdemeester/exquisite-words-java:v1

Création d'un frontend
docker service create –name front –network exquisite 80 :80 vdemeester/exquisite-web :v1

Démarre 15 back :
docker service scale back=15

Démarre 2 fronts :
docker service scale front=2

Indique à Docker de faire des upgrade 2 par 2 afin de ne pas avoir d'interruption de services
docker service update –update-parallelism –update-delay 10s back

Mise à jour vers back v2. On voit les images V1 se fermer et les images v2 se lancer.
Mise à jour vers front v2. Rollback nécessaire car V2 buggées
docker service update –rollaback front

Lorsqu'un nœud pose problème, on peut demander à Docker de rescheduler les tâches sur les autres nœuds afin d'effectuer une opération de maintenance.

Possibilité de scheduler un service sur tous les nœuds.