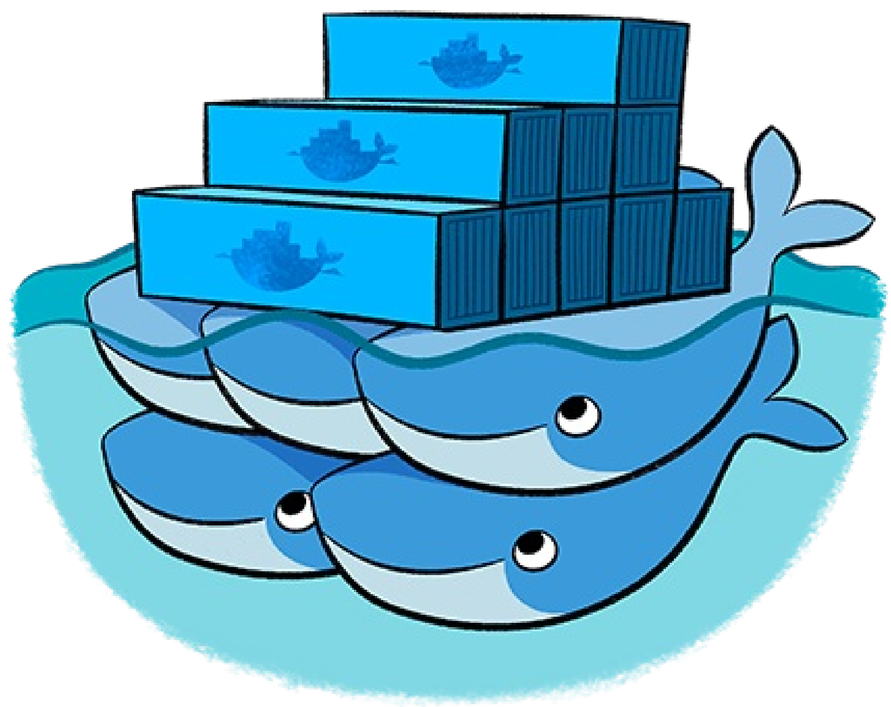
Introduction à Docker Swarm (> 1.12)
Préparation séance de TP

Jean-Patrick GELAS

Dpt. Informatique
Université Claude Bernard Lyon 1



Sources

- https://docs.docker.com/engine/swarm/
- http://zoneadmin.fr/forum/threads/cluster-swarm-avec-docker-1-12.1460/
- https://lostechies.com/gabrielschenker/2016/09/11/docker-and-swarm-mode-part-2/

Docker Swarm

- Solution native (officiel) de Docker pour faire du clustering
  - Transformer un ensemble d'hôtes Docker en un **unique** hôte virtuel Docker.
  - Utilise la même API
    - Tous les outils qui communiquent avec un *daemon* Docker peuvent utiliser Swarm (ex: Dokku, Compose, Machine, Jenkins et bien sur le **client Docker**).
- Alternatives : *Shipyard*, *Mesos*, *Kubernetes*, *Rancher*, ...

Remarque : Swarm avant Docker 1.12

- Le déploiement d'un cluster de serveur de Docker avec Swarm était lourd
  - Générer des certificats
  - Utiliser un *service discovery*
  - Configurer chaque noeuds

Docker Swarm

- Permet d'héberger et d'ordonnancer une grappe de container Docker :
  - Le *scheduling backend* peut être modifié/remplacé facilement (principe du *Swap plug'n play*). Par défaut : *Bin packing*.

Création d'un cluster Swarm

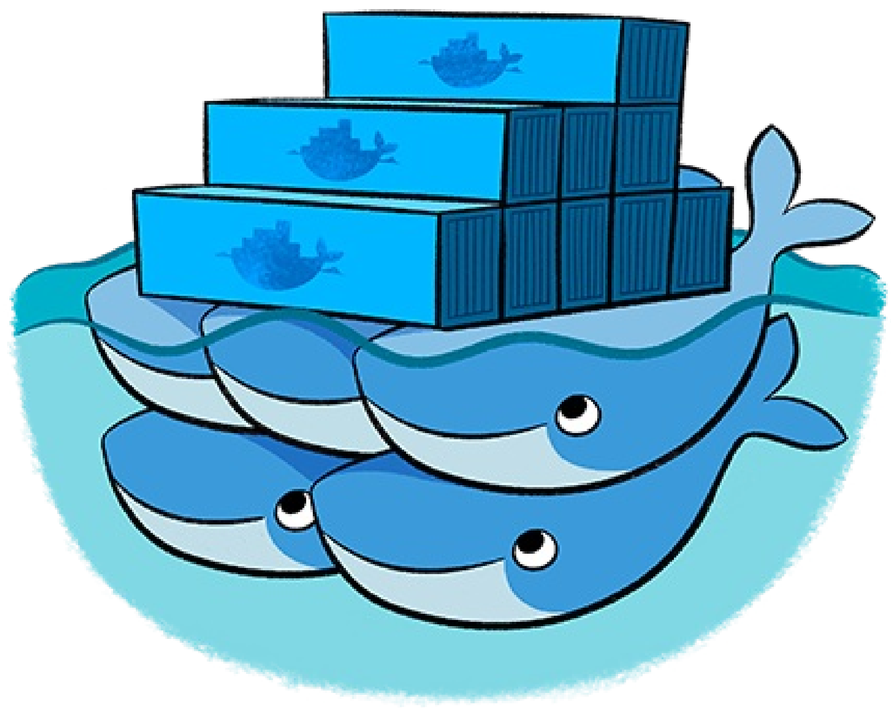
- *Pull* d'une image « Docker Swarm »
- Configurer le Swarm *Manager* et les *Workers* (nœuds physique apte à héberger des containers).
  - Ouvrir un port TCP sur chaque nœud pour communiquer avec le *Swarm manager*.
  - Installer Docker sur chaque nœuds (>1.12)
  - Créer et gérer des certificats TLS pour sécuriser son cluster.

Installation d’un Docker Swarm Cluster

- Deux méthodes :
  - Exécuter une image Swarm dans un container
  - Installer et utiliser le binaire sur son système.
- Avantages de la première méthode
  - Image construite par Docker et mise à jour régulièrement (docker run … swarm:latest)
  - Pas d’installation de binaire sur son système pour utiliser l’image
  - Une seule commande (docker run) pour obtenir et exécuter la version la plus récente
  - Le container isole Swarm de votre environnement.

Mise en oeuvre

Création d’un Docker Swarm



Docker Swarm init

- Initialisation d'un Swarm. Le nœud physique devient un nœud *Manager*.
- Docker *Swarm init* génére deux token aléatoires
  - Un Worker token
  - Un Manager token
- Utiliser l'option `--advertise-addr` si la machine à plusieurs adresses IP.
- Pour attacher un nouveau *Worker* au cluster Swarm on utilisera le *Worker token*.

```
$ docker swarm init --advertise-addr 192.168.99.121
Swarm initialized: current node (bvz81updecsj6wjz393c09vti) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join \
    --token SWMTKN-1-3pu6hszjas19xyp7ghgosyx9k8atbfcr8p2is99znpy26u2lkl-1awxwuwd3z9j1z3puu7rcgdbx \
    172.17.0.2:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

Docker Swarm join

- Permet d’ajouter un nœud à un Swarm.
- Passer en paramètre le Worker token.

```
$ docker swarm join --token SWMTKN-1-3pu6hszjas19xyp7ghgosyx9k8atbfcr8p2is99znpy26u2lkl-1awxwuwd3z9j1z3puu7rcgdbx 192.1
68.99.121:2377
This node joined a swarm as a worker.
$ docker node ls
ID                            HOSTNAME   STATUS   AVAILABILITY   MANAGER STATUS
7ln70fl22uw2dvjn2ft53m3q5     worker2    Ready    Active
dkp8vy1dq1kxleu9g4u78tlag     worker1    Ready    Active         Reachable
dvfxp4zseq4s0rih1selh0d20 *   manager1   Ready    Active         Leader
```

Docker Swarm leave

- Permet à un *Worker* de quitter le Swarm.
- A partir du manager :

```
$ docker node ls
ID                            HOSTNAME   STATUS   AVAILABILITY   MANAGER STATUS
7ln70fl22uw2dvjn2ft53m3q5     worker2    Ready    Active
dkp8vy1dq1kxleu9g4u78tlag     worker1    Ready    Active
dvfxp4zseq4s0rih1selh0d20 *   manager1   Ready    Active         Leader
```

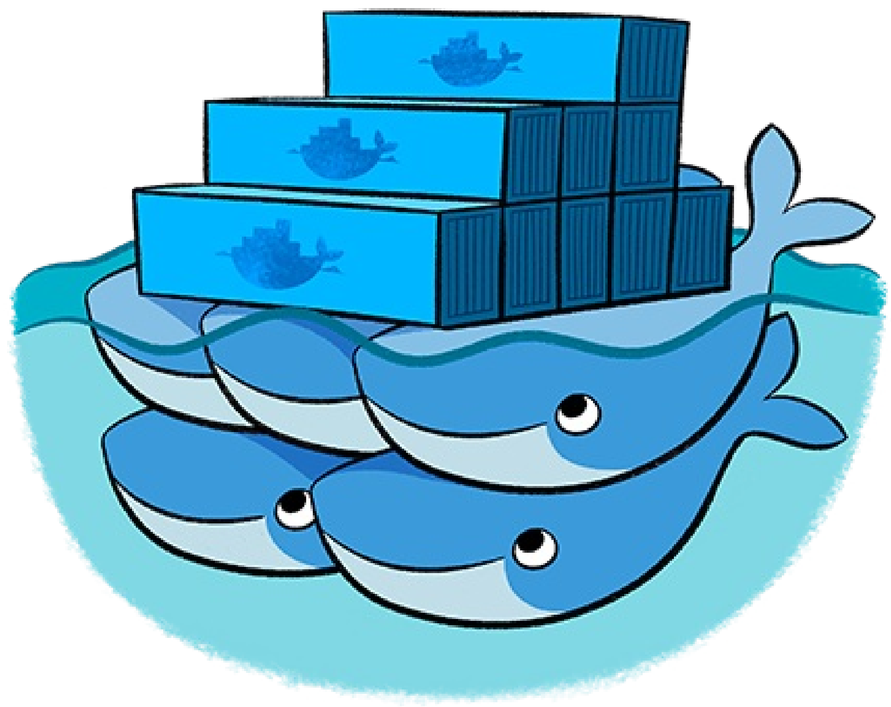
- A partir d’un worker :

```
$ docker swarm leave
Node left the default swarm.
```

- Pour supprimer un nœud inactif *docker node rm*

Mise en oeuvre

Usage d’un Docker Swarm (déploiement de services)



Déploiement

- Déploiement d'un service exposé sur nos nœuds et *load balancé* dans notre cluster.
  - Création d'un service qui expose le port 8001 de notre serveur vers 5 containers nginx.

```
docker service create --name nginx --replicas 5 -p 8001:80/tcp nginx
```

  - Lister les containers

```
root@kube:~# docker service tasks nginx
ID                          NAME     SERVIC
cmmni6a83fmqnk7h1wqb7yrdl   nginx.1  nginx
5o26sravvboq4d6femhmcjvhh   nginx.2  nginx
1gnokp6s37fhrxn8warwu63tt   nginx.3  nginx
at7lmzqd4pq8e8f3bv5yt4sxl   nginx.4  nginx
dcyj70t557tz7o63cyvwps1v0   nginx.5  nginx
root@kube:~#
```

Besoin de plus : Scale !

- On peut *scaler* le service selon nos besoins

```
docker service scale nginx=40
```

```
2l6bzwgp1lzutlmffputvw203  nginx.24
88wlk3s0l0lziaa3v9wclxmpe  nginx.25
ec5b6zbsd7i7t90m21veufh58  nginx.26
08kqc6q4301o0wgp494zr1i3r  nginx.27
bajmwp0766s32dpqj2uac5drb  nginx.28
d7onzqls9srifs72bqxfu6r2c  nginx.29
6er79s2g5qa4mmfk5efqc5mon  nginx.30
5u5dbzezne61mwuwc1qqtd7dc  nginx.31
9x9y0e95suuty7xrrf2lofl3j  nginx.32
4wgpj14j9f3xfe0es6hl1quna  nginx.33
4nouu3najhvx8yvsa8sxt6t67  nginx.34
de0ks32qnbs290k8iiqczs6li  nginx.35
9mycc8dq8vgj7iurgqhm3crf4  nginx.36
01aqcgzbl3ynu5eiboxp104fj  nginx.37
dnveqceuty4fe0ta3bs12buqm  nginx.38
8h4hh1dtt8mq2jecnyot02l4b  nginx.39
1l9o7rconzta7huqaubnggqh1  nginx.40
root@kube:~# _
```