Docker Swarm en local

Folie ou outil de développement insoupçonné ?

Jonathan Schoreels

Oniryx

DevDay, 27 Novembre 2018

Plan

Contexte

- **Docker, Inc** : 2008
- Docker est **Open Source** depuis Mars 2013. Lien : `https://github.com/moby/moby`.
- **Alternatives :** Rocket (CoreOS), runC (Linux Foundation)
- "**OCI**" (Open Container Initiative) : **spécification commune**.
  `https://github.com/opencontainers`

Contexte

Pourquoi utiliser des conteneurs ?

- Contrôle du runtime

Contexte

Pourquoi utiliser des conteneurs ?

- Contrôle du runtime $\Rightarrow$ Un livrable incluant le système et l'application.

Contexte

Pourquoi utiliser des conteneurs ?

- Contrôle du runtime $\Rightarrow$ Un livrable incluant le système et l'application.
- Systèmes moins encombrés

Contexte

Pourquoi utiliser des conteneurs ?

- Contrôle du runtime ⇒ Un livrable incluant le système et l'application.
- Systèmes moins encombrés ⇒ Scripts d'installations et nettoyage

Contexte

Pourquoi utiliser des conteneurs ?

- Contrôle du runtime ⇒ Un livrable incluant le système et l'application.
- Systèmes moins encombrés ⇒ Scripts d'installations et nettoyage
- Isolation

Contexte

Pourquoi utiliser des conteneurs ?

- Contrôle du runtime $\Rightarrow$ Un livrable incluant le système et l'application.
- Systèmes moins encombrés $\Rightarrow$ Scripts d'installations et nettoyage
- Isolation $\Rightarrow$ un hôte $=$ une application

Contexte

Pourquoi utiliser des conteneurs ?

- Contrôle du runtime $\Rightarrow$ Un livrable incluant le système et l'application.
- Systèmes moins encombrés $\Rightarrow$ Scripts d'installations et nettoyage
- Isolation $\Rightarrow$ un hôte $=$ une application
- Gestion homogène : démarrage d'apps, ports exposés ...

Contexte

Pourquoi utiliser des conteneurs ?

- Contrôle du runtime $\Rightarrow$ Un livrable incluant le système et l'application.
- Systèmes moins encombrés $\Rightarrow$ Scripts d'installations et nettoyage
- Isolation $\Rightarrow$ un hôte $=$ une application
- Gestion homogène : démarrage d'apps, ports exposés ... $\Rightarrow$ client commun

Docker : Virtualisation, ou non ?

Une machine virtuelle $\neq$ un conteneur.

- Un conteneur utilise le kernel de l'hôte, une machine virtuelle peut l'émuler. Exécuter un conteneur linux sur un autre système requiert de la virtualisation, une "Docker Machine" (Hyper-V pour Windows, HyperKit pour Mac).

Docker : Virtualisation, ou non ?

Une machine virtuelle ≠ un conteneur.

- Un conteneur utilise le kernel de l'hôte, une machine virtuelle peut l'émuler. Exécuter un conteneur linux sur un autre système requiert de la virtualisation, une "Docker Machine" (Hyper-V pour Windows, HyperKit pour Mac).

- Un conteneur est **isolé d'un point de vue système, réseau et processus**. Il peut être limité en ressources : CPU, RAM, ...

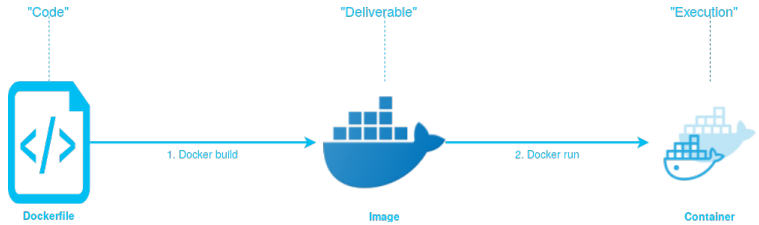
Docker : Virtualisation, ou non ?

Une machine virtuelle ≠ un conteneur.

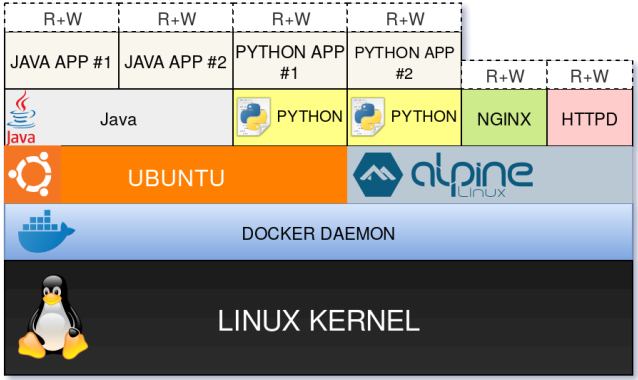
- Un conteneur utilise le kernel de l'hôte, une machine virtuelle peut l'émuler. Exécuter un conteneur linux sur un autre système requiert de la virtualisation, une "Docker Machine" (Hyper-V pour Windows, HyperKit pour Mac).
- Un conteneur est **isolé d'un point de vue système, réseau et processus**. Il peut être limité en ressources : CPU, RAM, ...
- Il ne contient qu'**un et un seul process** (Linux).

Terminologie

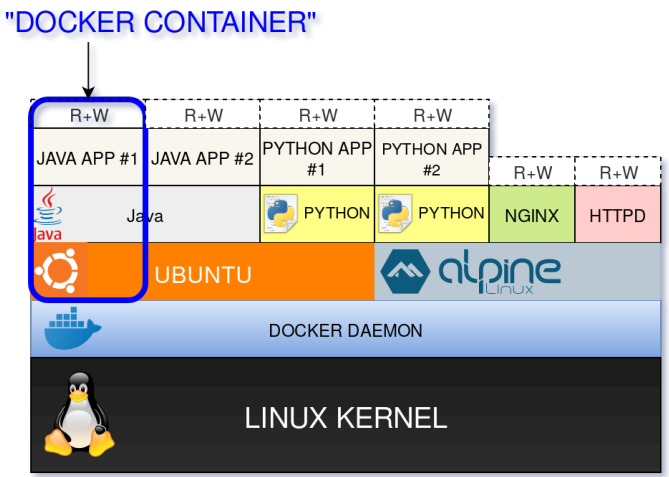
- Un **Dockerfile** contient les instructions nécessaires au **build** de l'**image**.
- Une **image** docker est constituée d'une série de **layers**. .
- Chaque **layer** est **read-only**. Un **layer** n'enregistre que les fichiers ajoutés, édités ou masqués aux couches précédentes.
- Un **container** est lancée par le **run** d'une **image**. Un **layer** en **read-write** est créé au dessus des **layers** de l'**image**.



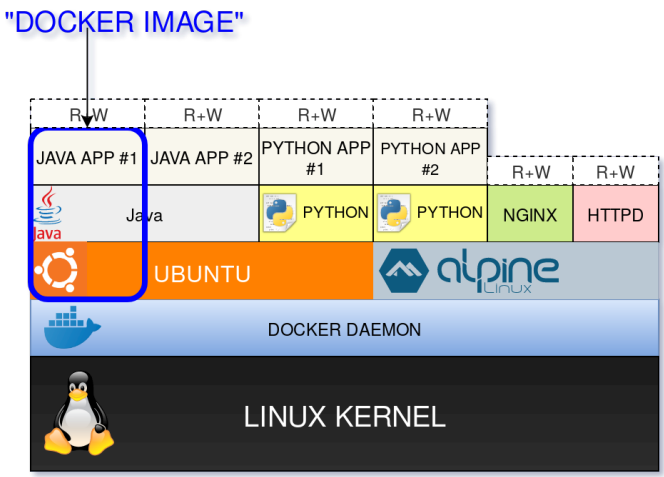
Docker : Virtualisation, ou non ?



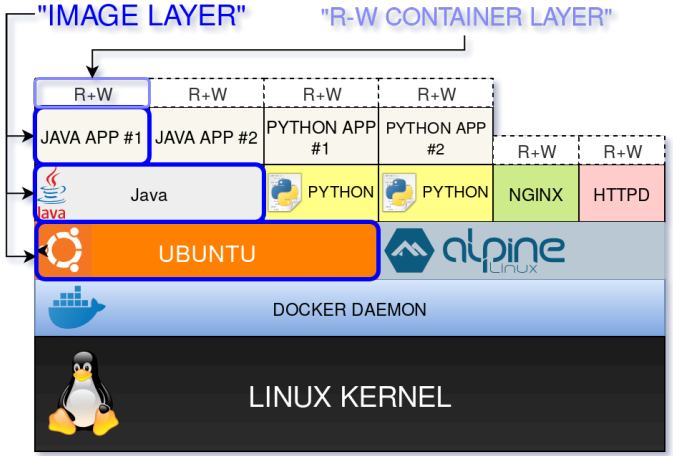
Docker : Virtualisation, ou non ?



Docker : Virtualisation, ou non ?



Docker : Virtualisation, ou non ?



Plan

Déploiement d'applications tiers

De nombreuses images officelles existent aujourd'hui :

- **HTTP Servers** : httpd, nginx, ...
- **Base de données** : mysql, postgres, mongo, ...
- **Systèmes** : Alpine, Centos, Ubuntu, Fedora, ...
- **... + Runtime** : java, python, node, "microsoft/dotnet", ...
- **Outils de Développements** : Jenkins, Sonarqube, ...
- **Infrastructure** : logstash, elasticsearch, kibana ...

Déploiement d'applications tiers

De nombreuses images officelles existent aujourd'hui :

- **HTTP Servers** : httpd, nginx, ...
- **Base de données** : mysql, postgres, mongo, ...
- **Systèmes** : Alpine, Centos, Ubuntu, Fedora, ...
- **... + Runtime** : java, python, node, "microsoft/dotnet", ...
- **Outils de Développements** : Jenkins, Sonarqube, ...
- **Infrastructure** : logstash, elasticsearch, kibana ...

Mais aussi de la communauté :

- *spotify*/**kafka**
- *plexinc*/**pms-docker**
- *grafana*/**grafana**

Déploiement d'applications tiers

De nombreuses images officelles existent aujourd'hui :

- **HTTP Servers** : httpd, nginx, ...
- **Base de données** : mysql, postgres, mongo, ...
- **Systèmes** : Alpine, Centos, Ubuntu, Fedora, ...
- **... + Runtime** : java, python, node, "microsoft/dotnet", ...
- **Outils de Développements** : Jenkins, Sonarqube, ...
- **Infrastructure** : logstash, elasticsearch, kibana ...

Mais aussi de la communauté :

- *spotify*/**kafka**
- *plexinc*/**pms-docker**
- *grafana*/**grafana**

Comment en exécuter ? **docker run <image>**

Déploiement d'applications tiers

Par défaut, aucun port n'est exposé sur l'hôte. Plusieurs solutions possibles :

- Attaquer l'IP du conteneur directement (uniquement à usage local).

Déploiement d'applications tiers

Par défaut, aucun port n'est exposé sur l'hôte. Plusieurs solutions possibles :

- Attaquer l'IP du conteneur directement (uniquement à usage local).

Déploiement d'applications tiers

Par défaut, aucun port n'est exposé sur l'hôte. Plusieurs solutions possibles :

- Attaquer l'IP du conteneur directement (uniquement à usage local).
- Briser l'isolation et tout exposer avec "**--network host**".

Déploiement d'applications tiers

Par défaut, aucun port n'est exposé sur l'hôte. Plusieurs solutions possibles :

- Attaquer l'IP du conteneur directement (uniquement à usage local).
- Briser l'isolation et tout exposer avec "**--network host**".

Déploiement d'applications tiers

Par défaut, aucun port n'est exposé sur l'hôte. Plusieurs solutions possibles :

- Attaquer l'IP du conteneur directement (uniquement à usage local).
- Briser l'isolation et tout exposer avec "**--network host**".
- Publier certains ports sur l'hôte avec l'option "**-p HOST_PORT:CONTAINER_PORT**".

Déploiement d'applications tiers

Par défaut, aucun port n'est exposé sur l'hôte. Plusieurs solutions possibles :

- Attaquer l'IP du conteneur directement (uniquement à usage local).
- Briser l'isolation et tout exposer avec "**--network host**".
- Publier certains ports sur l'hôte avec l'option "**-p HOST_PORT:CONTAINER_PORT**".

Comment trouver une liste exhaustive des applications ?

Registres en ligne :

- `https://hub.docker.com`
- `https://quay.io/repository`
- `https://container-registry.oracle.com/`

Comment trouver une liste exhaustive des applications ?

Registres en ligne :

- `https://hub.docker.com`
- `https://quay.io/repository`
- `https://container-registry.oracle.com/`

Registres embarqués :

- GitLab : `https://docs.gitlab.com/ee/user/project/container_registry.html`
- Artifactory : `https://www.jfrog.com/confluence/display/RTF/Docker+Registry`

Comment trouver une liste exhaustive des applications ?

Registres en ligne :

- `https://hub.docker.com`
- `https://quay.io/repository`
- `https://container-registry.oracle.com/`

Registres embarqués :

- GitLab : `https://docs.gitlab.com/ee/user/project/container_registry.html`
- Artifactory : `https://www.jfrog.com/confluence/display/RTF/Docker+Registry`

Ou même en local grâce à l'image docker "*registry*" :
`https://hub.docker.com/_/registry/`

Dockerfile

Un **Dockerfile** contient les instructions nécessaires au **build** de l'**image**.

```
FROM ubuntu
RUN [...] apt install python3
COPY host-app.py /container-app.py
CMD ["python3", "/container-app.py"]
```

Pour **builder** l'image depuis le dossier contenant le **Dockerfile**: docker build -t my-app .

Dockerfile

Mais aussi à partir d'une autre image "applicative" !

```
FROM nginx
COPY hello.html /usr/share/nginx/html/index.html
```

Il est donc possible de créer des images génériques ne nécessitant plus que la copie de l'exécutable !

Plan

Topologie

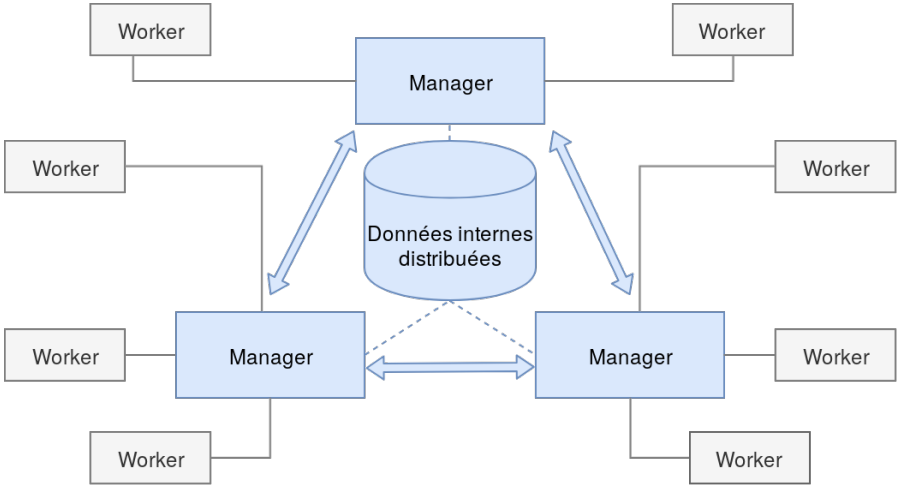
Docker Swarm est un outil de clustering et un orchestrateur de conteneurs. Un Node du cluster Swarm peut être de deux types différents :

- Les Managers gèrent et inspectent le swarm.
- Les Workers exécutent leurs tâches assignées

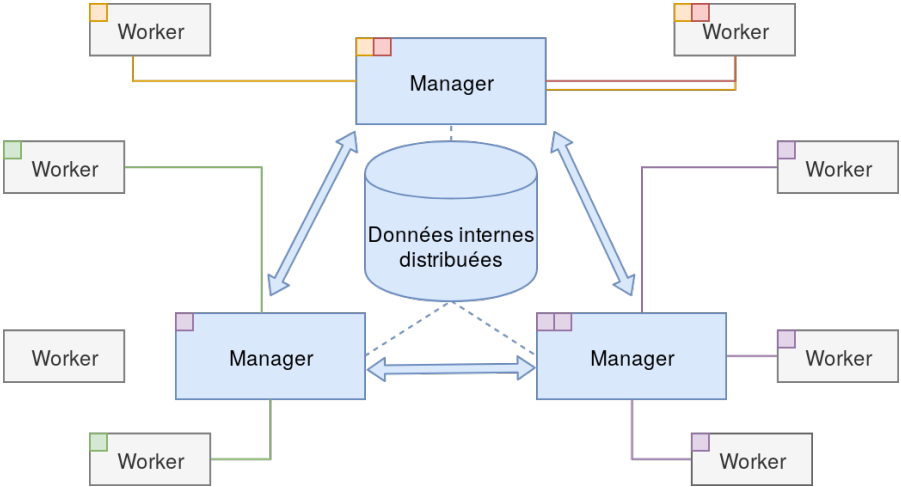
Toutes les communications et les données internes sont sécurisées (TLS).

Pour activer le mode Swarm : **docker swarm init**

Topologie


Worker
Worker
Manager
Worker
Worker
Données internes distribuées
Worker
Manager
Manager
Worker
Worker
Worker

Topologie


Worker
Manager
Worker
Worker
Worker
Données internes distribuées
Worker
Manager
Manager
Worker
Worker
Worker

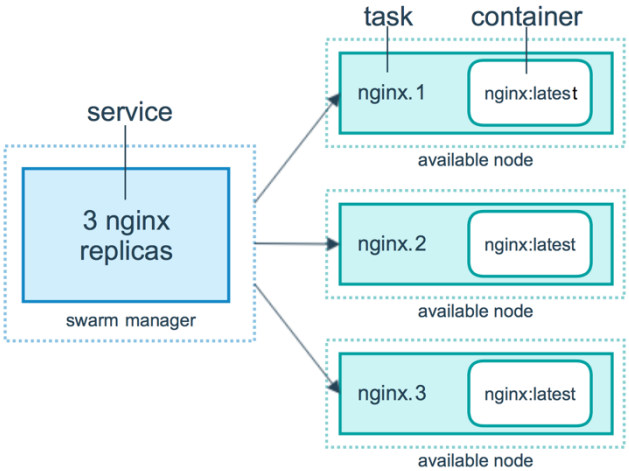
Docker Service

Un Service est un groupe de conteneurs d'une même image partageant une même configuration, permettant :

- Distribuer automatiquement les instances dans le Swarm
- Redémarrer automatiquement les instances tombées
- Augmenter le nombre de réplicas
- D'effectuer des *rollings updates*.

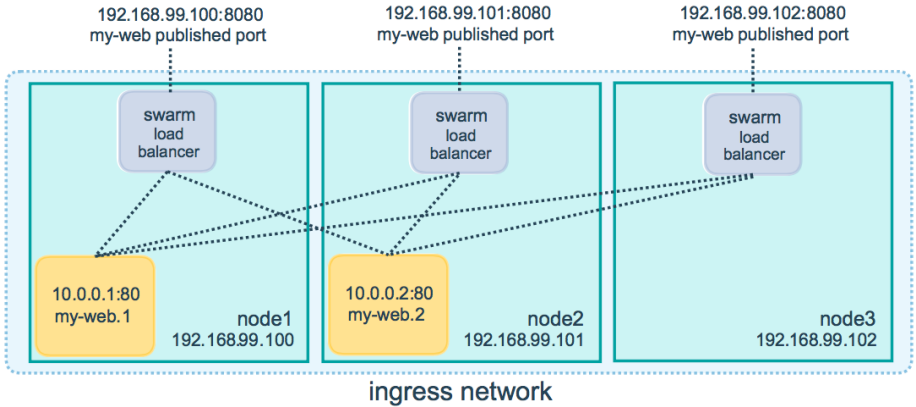
Pour créer un service : **docker service create <image>**

Docker Service



Load Balancing, Ingress Networking

Peu importe où un service est instancié, toute requête reçue par un membre du Swarm est routée vers le bon membre de façon transparente.



Docker Stack

Une Stack est un ensemble de services décrit par un "docker-compose.yml".

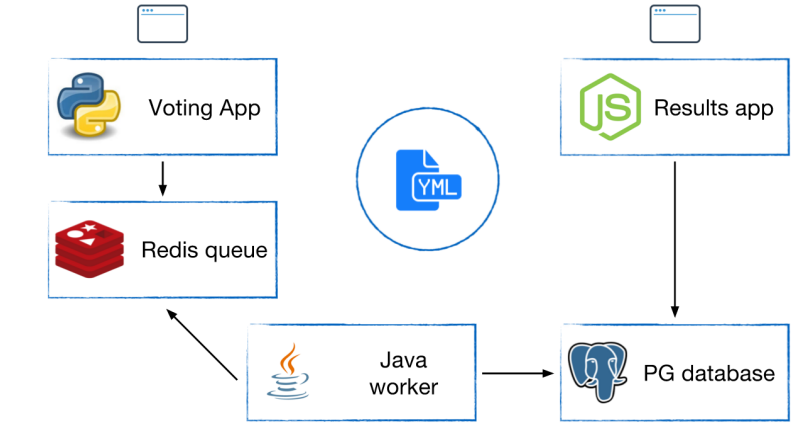
```yaml
version: '3.1'
services:
  wishlistmanager:
    image: jschoreels/wishlist-manager-springboot:1.0.0-SNAPSHOT
    ports:
      - 2000:8080
    environment:
      SPRING_DATASOURCE_URL: jdbc:postgresql://wishlist_db:5432/wishlistmanager
      SPRING_DATASOURCE_USERNAME: wishlistmanager-app
      SPRING_DATASOURCE_PASSWORD: secret

  wishlist_db:
    image: postgres:9.6.8
    environment:
      POSTGRES_DB: wishlistmanager
      POSTGRES_USER: wishlistmanager-app
      POSTGRES_PASSWORD: secret
```

Pour déployer : **docker stack deploy --compose-file docker-compose.yml <stack_name>**

Docker Stack

"Voting App".

Exemple tiré de : `https://github.com/dockersamples/example-voting-app`



Docker Stack

"Voting App".

```yaml
version: '3'
services:
  redis:
    image: redis:alpine
  db:
    image: postgres:9.4
  vote:
    image: dockersamples/examplevotingapp_vote:before
    ports:
      - "5000:80"
    deploy:
      replicas: 2
  result:
    image: dockersamples/examplevotingapp_result:before
    ports:
      - "5001:80"
  worker:
    image: dockersamples/examplevotingapp_worker
```

Docker Stack

Déploiement d'une stack :

```
$  docker stack deploy vote --compose-file docker-compose.yml
Creating network vote_default
Creating service vote_worker
Creating service vote_redis
Creating service vote_db
Creating service vote_vote
Creating service vote_result
```

Tous les services sont préfixés du nom de la stack, mais le DNS contient également le nom tel que défini dans la définition.

```
$  docker service ls
ID                  NAME                MODE                REPLICAS
0o0cq1el08va        vote_db             replicated          1/1
7jwgurqosa4s        vote_redis          replicated          1/1
sixa5wrfb9zc        vote_result         replicated          1/1
z4vrksycyct8        vote_vote           replicated          2/2
ygbqlg4q727v        vote_worker         replicated          1/1
```

Plan

Cas d'utilisations :

- ♥♥♥ Réplication d'un environnement de développement
- ♥♥♥ Fournir des images standards pour les nouveaux collègues

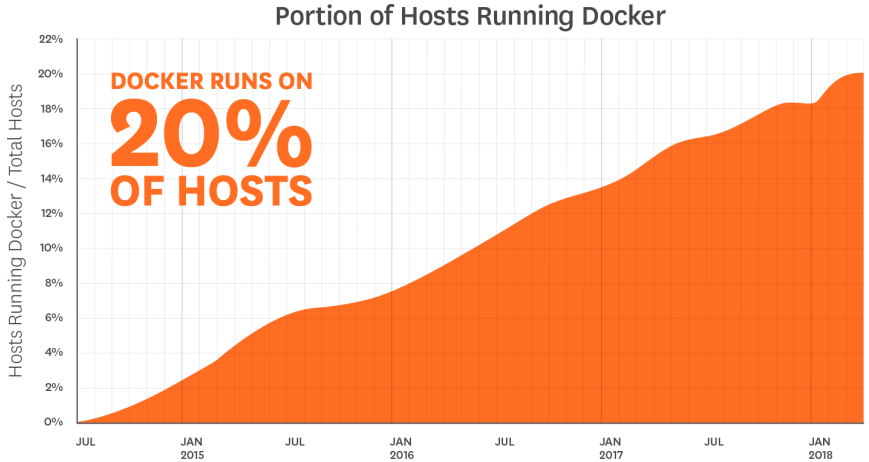
Cas d'utilisations :

- ♥♥♥ Réplication d'un environnement de développement
- ♥♥♥ Fournir des images standards pour les nouveaux collègues
- ♥♥ Tests mise à l'échelle
- ♥ Tests de charge

Cas d'utilisations :

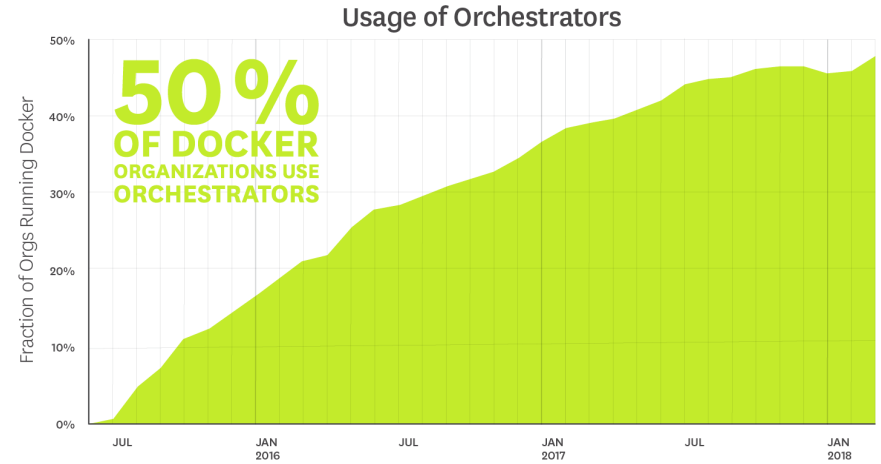
- ♥♥♥ Réplication d'un environnement de développement
- ♥♥♥ Fournir des images standards pour les nouveaux collègues
- ♥♥ Tests mise à l'échelle
- ♥ Tests de charge
- ♥♥♥♥♥♥♥ Investiguer de nouvelles technologies rapidement

Quelques statistiques par datadog



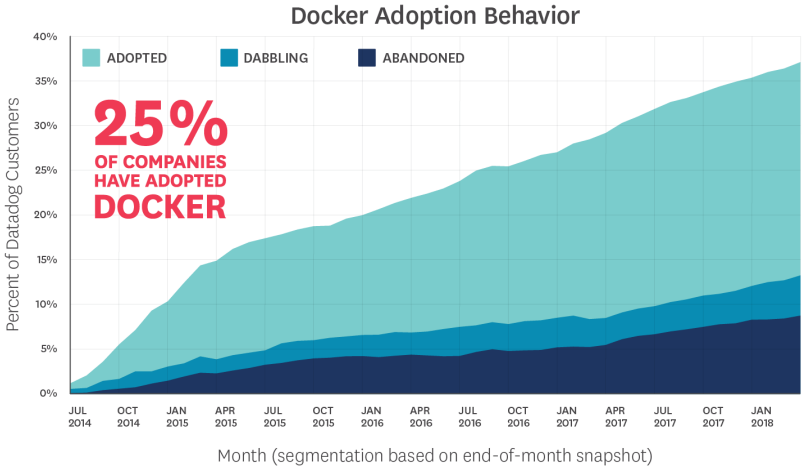
source : `https://www.datadoghq.com/docker-adoption/`

Quelques statistiques par datadog



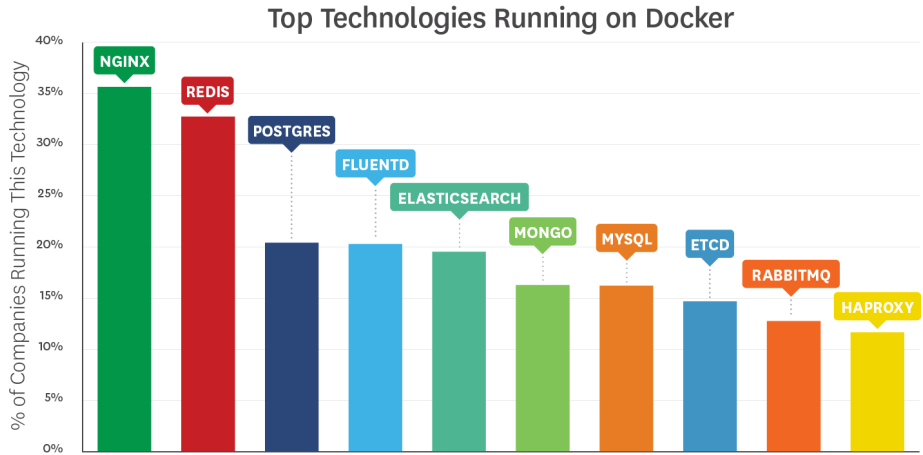
source : `https://www.datadoghq.com/docker-adoption/`

Quelques statistiques par datadog



source : `https://www.datadoghq.com/docker-adoption/`

Quelques statistiques par datadog



source : https://www.datadoghq.com/docker-adoption/

Conclusion

En conclusion :

- Docker est une technologie mature
- Docker Swarm est un orchestrateur facile à mettre en place
- Certains frameworks ou logiciels sont mieux adaptés aux containers

Pour aller plus loin :

- **The Twelve-Factor** / **Cloud-Native** : techniques de développement
- **Kubernetes** : alternative populaire à Docker Swarm

Question time !