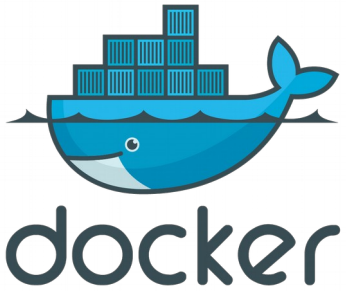
Déploiement d'application avec



William Bartolini

18/05/2017

xsalto

Sommaire

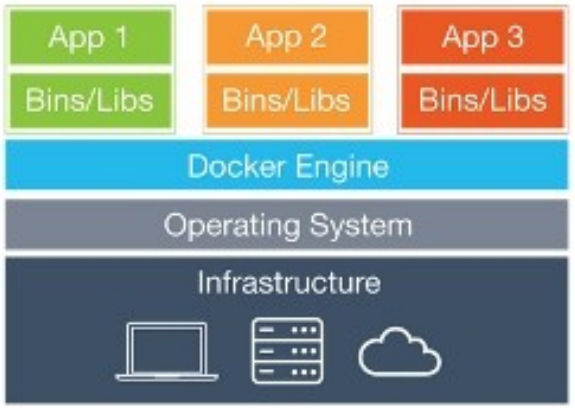
Qu’est-ce que Docker ?

- Encapsule les applications dans des conteneurs
- Virtualise depuis les ressources de la machine hôte
- Est multi-plateforme (Linux, Mac OS et Windows)

Solomon Hykes

Comment ça marche ?

Virtual Machines

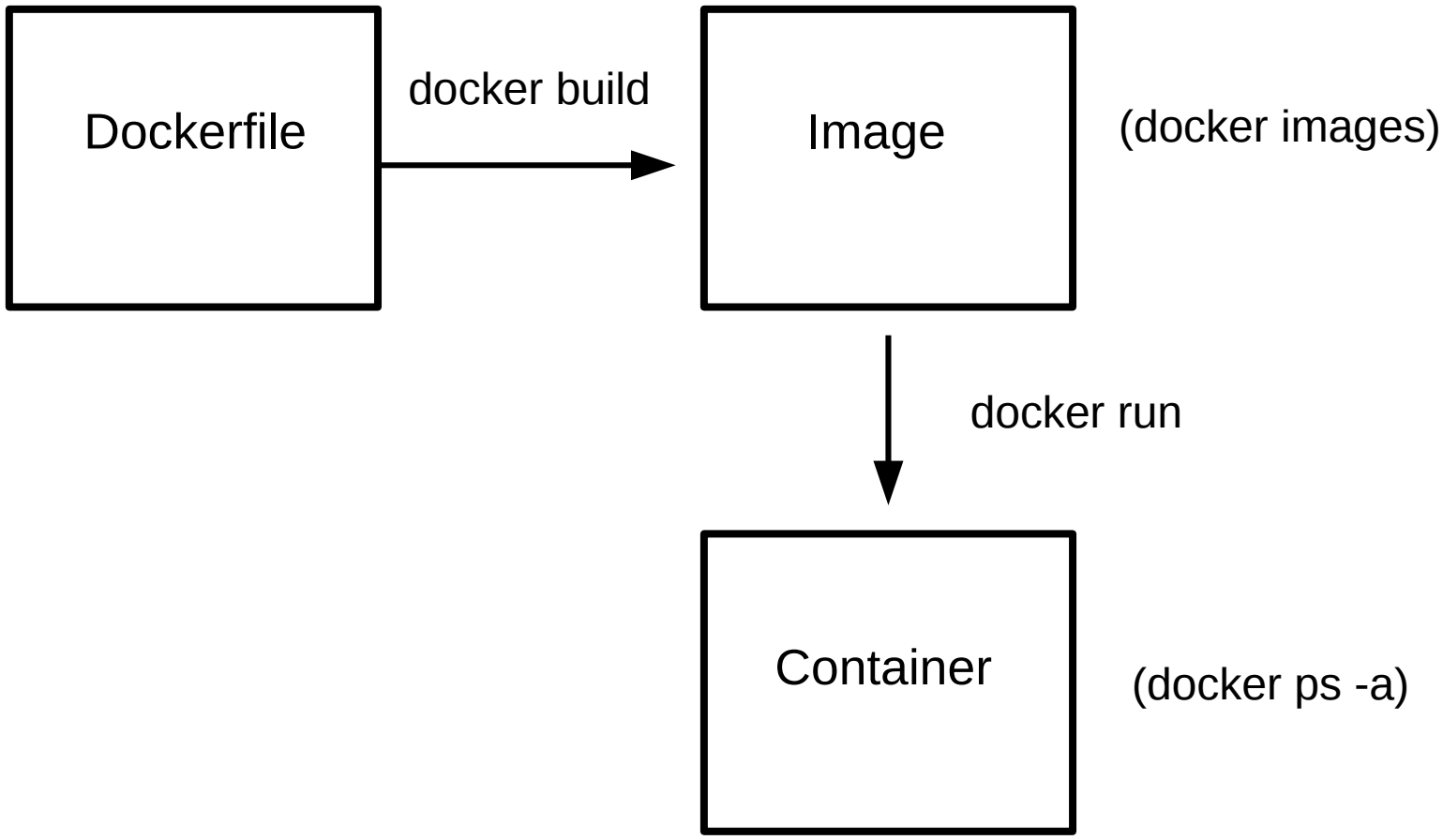


Containers

Qu’est qu’un conteneur ?

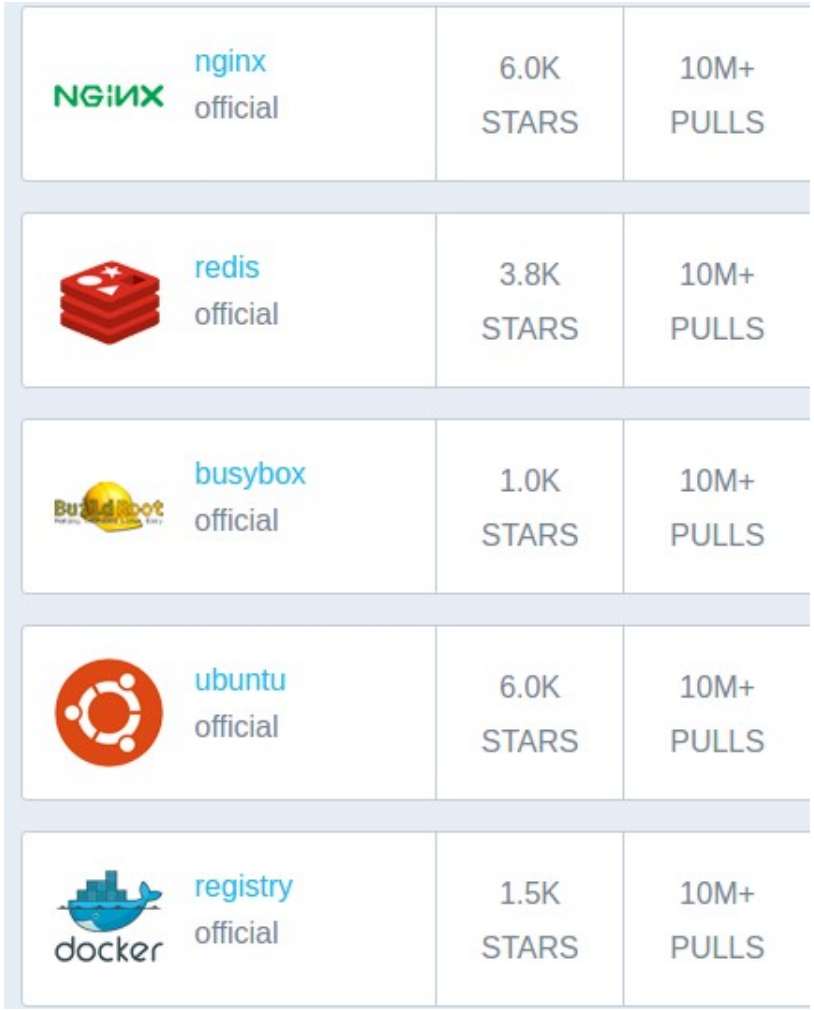
- Isolation
- Aucune dépendance
- Natif à linux (extension du standard de conteneur Linux, LXC)
- Est l’instance d’une image

Création d'un conteneur



Docker Registry

https://hub.docker.com/



- +100 000 applications
- Source par défaut
- Accès libre
- 1 repository privé gratuit
- Compatible webhook bitbucket et github

Docker est multi-plateforme

- Windows 10 Professional et Enterprise 64-bit
  https://www.docker.com/docker-windows
- Mac OSX Yosemite 10.10.3 ou supérieur
  https://www.docker.com/docker-mac
- Linux, support natif
  https://www.docker.com/docker-ubuntu
- Windows serveur 2016
  https://www.docker.com/docker-windows-server
- Autres versions de Windows et Mac
  https://www.docker.com/products/docker-toolbox

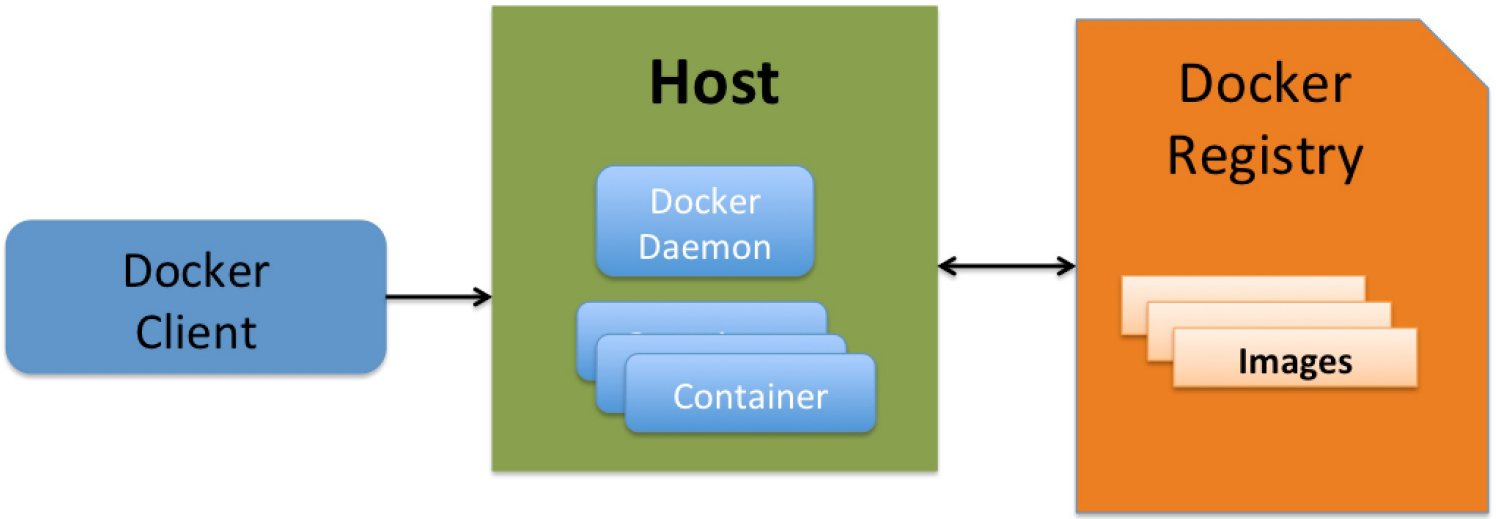
Les avantages

Légèreté, Isolation et multi-plateformes permettent :

- Un déploiement simplifié
- Une unicité de l'environnement
- Grande modularité, chaque conteneur étant isolé
- Une « scalabilité » grâce aux déploiements automatique
- Partage des applications simplifié

Sommaire

Docker CLI



Documentation :

https://docs.docker.com/engine/reference/commandline/cli/

Manipulation des images

Exécuter une image

```
docker run <image>:<tag>
docker run <image_id>
```

Liste des images installées

```
docker images
```

Transférer des images

```
docker pull <image>:<tag>
docker pull myregistry.local:5000/testing/test-image
docker push <image>:<tag>
```

Chercher une image sur le hub

```
docker search <termes>
```

Manipulation des conteneurs

Lister tout les conteneurs

```
docker ps -a
```

Contrôle du conteneur

```
docker start <container_id>
docker restart <container_id>
docker stop <container_id>
docker kill <container_id>
```

Exécuter une commande à l'intérieur d'un conteneur :

```
docker exec <container_id> <commande>
```

Ouvrir une session bash dans le container

```
docker exec -ti <container_id> /bin/bash
```

Sauvegarde/Restauration

Sauvegarder les modifications faites dans un conteneur

```
docker commit -p <container_id> <nom_image>
```

Sauvegarder une image dans un fichier

```
docker save <chemin_fichier> <nom_image>
```

Importer une image depuis un fichier :

```
docker load -i <chemin_fichier>
```

Supprimer

Supprimer un container

```
docker rm <container_id>
```

Supprimer une image

```
docker rmi <image_id>
```

Supprimer tous les containers :

```
docker rm $(docker ps -a -q))
```

Supprimes toutes les images :

```
docker rmi $(docker images -q)
```

Autres commandes

Liste des réseaux docker

```
docker network ls
```

Liste des volumes

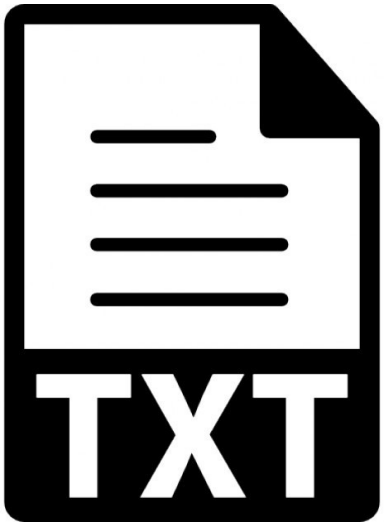
```
docker volume ls
```

Avoir des informations sur un conteneur

```
docker inspect <container_id>
```

Sommaire

Dockerfile



- Décrit comment build l'image
- Chaque ligne contient une commande

Documentation :

https://docs.docker.com/engine/reference/builder/

Pour build un Dockerfile :

```
docker build -t <image_name> :<tag> <path>
docker build -t name/exemple:latest Apache/
```

Synthèse des commandes

Définir l'image de départ de notre image

FROM <image>:<tag>

Cette commande doit être la première du Dockerfile

Exécuter une commande

RUN <command>

Définir le répertoire courant

WORKDIR /path/to/workdir

Exposer un port

EXPOSE <port>

N'ouvre pas le port mais indique uniquement à Docker qu'il sera utilisé

Synthèse des commandes

Copier des fichiers dans l'image

COPY <src> <dest>

ADD <src> <dest>

Utilisable aussi bien dans le contexte que sur des ressources distantes
Extrait les archives tar qu'il reconnait

Définir la commande de démarrage

CMD ["executable","param1","param2"]
Par défault l'ENTRYPOINT est /bin/sh -c
docker run [OPTIONS] IMAGE[:TAG|@DIGEST] [COMMAND] [ARG...]
ENTRYPOINT ["executable", "param1", "param2"]
CMD ["param1","param2"]

Utiliser les arguments

```
ARG CONT_IMG_VER
ENV CONT_IMG_VER ${CONT_IMG_VER:-v1.0.0}
RUN echo $CONT_IMG_VER
```

Synthèses des commandes

- Créer un point de montage

```
VOLUME <repertoire>
```

.dockerignore

```
# comment
    */temp*
    */*/temp*
    temp?
```

Sommaire

Docker-compose

Exemple de fichier docker-compose.yml :

```
version: '2'
services:
  web:
    build: .
    ports:
     - "5000:5000"
    volumes:
     - .:/code
  redis:
    image: "redis:alpine"
```

Installation : https://docs.docker.com/compose/install/

Que permet vraiment docker ?

+ Un déploiement rapide d'application
+ Une configuration simplifié
+ Une administration réseau simplifié
+ Une gestion accru des versions de chaque composant

- Demande beaucoup de pratique avant d'être productif
- Certains cas spécifiques n'ont pas de solution existante
- Des applications ne sont toujours qu'au stade expérimentale

Ressources & outils

Documentation :

https://docs.docker.com/

Interfaces UI pour docker :

http://portainer.io/
https://github.com/portainer/portainer

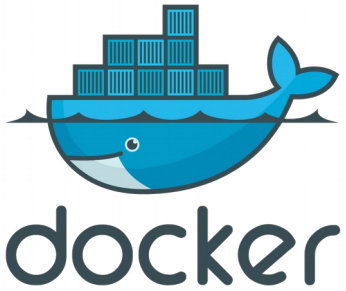
Déploiement multi-serveurs :

https://docs.docker.com/engine/swarm/

Orchestration de conteneurs :

http://rancher.com/rancher/
http://anthonykgross.fr/p/deploiement-continu-docker-gitlab-rancher

Merci pour votre attention !



William Bartolini

18/05/2017

xsalto