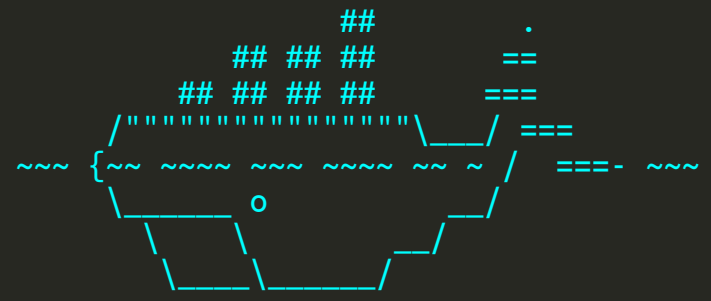
Docker.io



Vendredi 16 mai 2014

mcaneill, ppepos | Merci à Stéphane et Arthur !

Do you want some magic?

```
apt-get install docker.io && docker.io run -i -t ubuntu bash
```

Sommaire

Pésentation : Docker

- Début du projet : Mars 2013
- Première release : Avril 2014
- Forks GitHub : 1 932
- Stars GitHub : 11 858
- Langage : Go
- Crée par : DotCloud

Présentation : L'écosystème Docker

- Images
- Containers
- Dockerfile
- index.docker.io

Objectifs de Docker

- Se débarasser des contraintes d'environnement avec une installation automatisée
- Créer facilement des images transportables, archivables, réutilisables
- Isoler des programmes sans utiliser de machine virtuelle

Cas d'usage

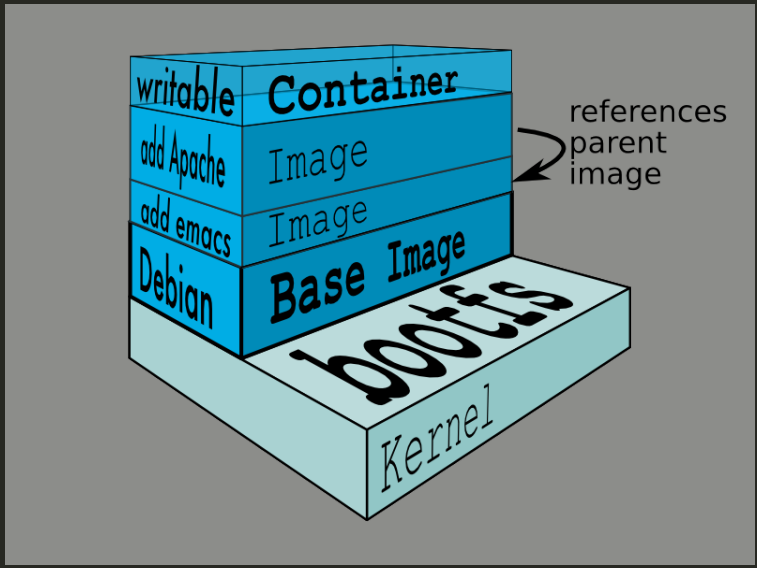
- Suites unitaires
- Intégration continue
- Platform as a service
- Production (bientôt…)

Another Union File System

- Système de fichiers par couche
- AUFS fusionne différents disques de manière transparente
- Gestion des conflits, Round Robin Policy
- Mise à jour grace à "User's Direct Branch Access"
- `# mount -t aufs -o br=/tmp/foo=rw:/tmp/bar=rw -o udba=reval -o create=rr none /tmp/aufs/`
  - Type : AUFS
  - 2 sources en lecture/écriture : /tmp/foo et /tmp/bar
  - Si les dossiers source sont mis à jour, AUFS le sera aussi
  - Round Robin Policy
  - Pas de device associé

AUFS dans Docker

- Pile de couches en lecture seule, couche du haut en lecture-écriture
- La clause `FROM foo` hérite une pile, et les autres clausent écrivent par-dessus cette pile
- Permet le partage des images de base
- Docker inclut également des pilotes pour d'autres systèmes de fichiers (VFS, DeviceMapper, BTRFS, ...)



http://docs.docker.io/terms/layer

LXC

Features

- Kernel namespaces
- Profils Apparmor, SELinux
- Poltiques Seccomp
- `chroot`
- Groupes de contrôle (`cgroups`)

Morceaux

- La lib: `liblxc`
- Bindings pour différents langages (python3, lua, ruby, Go)
- Outils standards pour contrôler les conteneurs
- Templates

LXC

Description humaine

Mi-chemin entre chroot sur les stéroïdes et une machine virtuelle.

Isoler les processus au maximum sans avoir besoin d'un deuxième kernel.

Commandes Docker

- `docker` - liste les commandes
- `docker run -i -t ubuntu bash` - télécharge l'image Ubuntu et lance un container avec Bash
- `docker ps` - liste les containers actifs
- `docker inspect id` - renvoie des informations sur le container
- `docker build -t tag .` - construit l'image à partir du Docker file dans le répertoire courant
- `docker run -i -t tag` - lance le container fraichement construit
- `docker kill id` - termine un container

Commandes Dockerfile

- `FROM` - hérite une image
- `RUN` - lance une commande
- `ADD` - ajoute un fichier dans l'image
- `EXPOSE` - expose un port
- `ENV` - modifie une variable d'environnement
- `CMD, ENTRYPOINT` - processus lancé au demarrage

```
FROM debian:testing
MAINTAINER Anne Onyme <anne.onyme@caramail.com>
RUN apt-get install -y a-great-package
ADD great.conf /etc/a-great-package/
# awesome command:
CMD /usr/bin/a-great-package
```

Commandes moins de base

- `docker logs` - Affiche stdout du conteneur
- `docker top` - Affiche top pour le conteneur
- `docker attach` - S'attache à un conteneur en exécution (screen-esque)

Lab

- Objectif : faire rouler une application Flask
- Indices :
  - Dépendances : (Debian/Ubuntu) `python python-flask python-sqlalchemy`
  - Repo de l'application Flask : `https://github.com/matthieucan/shorturl`
    - Utilisez git, l'application peut être par exemple stockée dans /opt
  - Il faut initialiser la base de données : `python init_db.py`
  - L'applications se lance avec `python --ip 0.0.0.0 --port 5000 run.py`

Solution du lab

Dockerfile

```
FROM ubuntu:trusty

MAINTAINER Spider Pig <spider.pig@cochonbiensingulier.org>


RUN apt-get install -y git python python-flask python-sqlalchemy
RUN cd /opt && git clone https://github.com/matthieucan/shorturl
RUN cd /opt/shorturl; python init_db.py


CMD cd /opt/shorturl; python run.py --ip 0.0.0.0 --port 5000
```

Commandes

```
# docker build -t my_flask_app .
# docker run -d -p 5000 my_flask_app
```

Merci !

Questions ?