Docker Tutorial PDF

Qu'est-ce que Docker (en français) ?

Si vous vous êtes un minimum intéressé à l'univers devops, vous avez probablement entendu parler de Docker qui est probablement le logiciel libre qui s'est le plus popularisé ces dernières années. Nous allons profiter de ce tutorial docker pour en savoir plus sur cet outil génial.

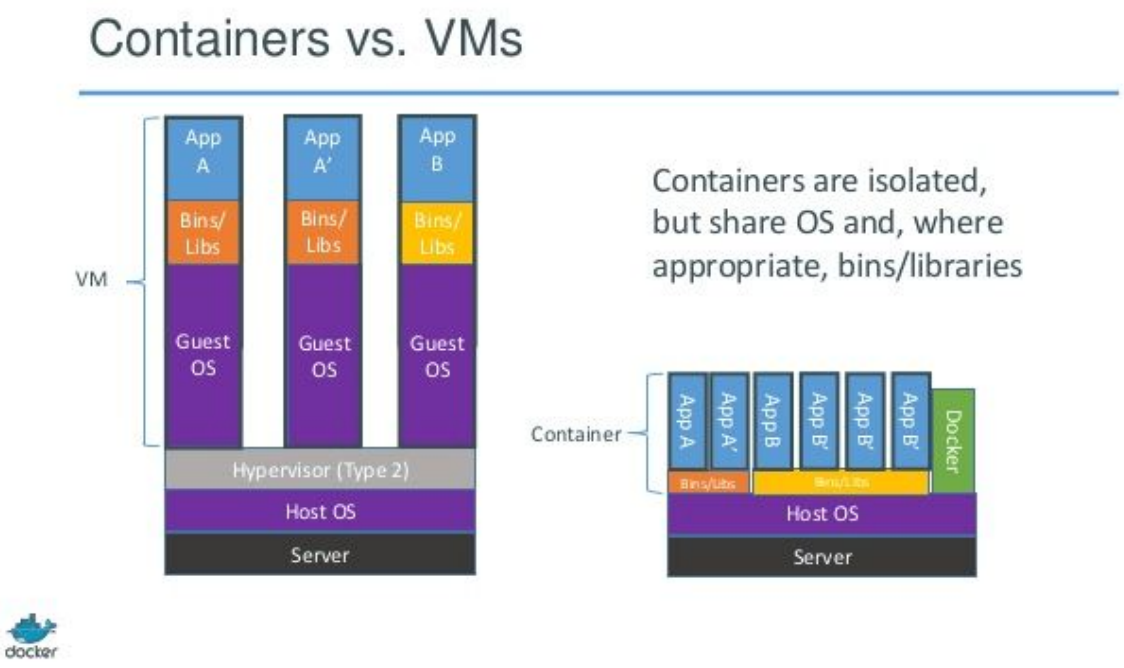
Il permet d'automatiser le déploiement d'applications et de leurs dépendances dans différents conteneurs isolés sur un n'importe quel serveur Linux.

Il a été développé par Solomon Hykes à la base comme projet interne de DotCloud en tant qu'évolution de solutions open-source déjà existantes au sein de la société. Docker a été distribué en open-source la première fois le 13 mars 2013. Début 2017, il y a 1600 contributeurs sur le Github de l'application.

Définition d'un conteneur

Docker a la particularité de fonctionner avec des conteneurs. Cependant que veut dire concrètement ce terme ?

Commençons par un petit schéma :



Docker conteneur versus machine virtuelle – tutorial docker

Comme le montre ce schéma, une machine virtuelle va recréer un serveur complet pour chaque application (en se réservant des ressources) avec son propre système d'exploitation. Hors Docker va isoler l'application tout en utilisant le système d'exploitation de son hôte.

Avec Docker, on pourrait du coup avoir plusieurs conteneurs différents pour gérer des applications différentes mais qui utiliseront tous le même système d'exploitation :

1. Apache en serveur web
2. MySQL en base de données
3. Redis en serveur de cache

En plus de tout ça, contrairement aux machines virtuelles, Docker partage l'ensemble des ressources (RAM, CPU…) avec le système d'exploitation de l'hôte. C'est un sacré avantage.

Il est utilisé pour faire des conteneurs sur Linux mais des versions beta existent pour faire la même chose sous Windows. Cependant, nous allons rester sur la partie Linux pour le moment.

Tutorial docker – installons Docker

Sur Ubuntu il est très simple d'installer Docker.

```
sudo apt-get install -y docker.io
```

N'hésitez pas à vous référer à la documentation officielle pour installer Docker sur une autre distribution Linux.

Sous Windows et Mac OS, il faudra installer le logiciel Docker Toolbox qui sera une fenêtre Terminal particulière. Cet outil utilisera une Virtual Box pour faire tourner le Linux hôte. [lien]

Je vous laisserais vous référer à la documentation officielle en cas de soucis d'installation et d'utilisation de Docker Toolbox.

Les premières fonctions à connaitre

Commençons par lancer Docker pour pouvoir l'utiliser :

```
/etc/init.d/docker start
```

Si vous avez le message « Cannot connect to the Docker Deamon. Is the Docker Deamon running on this host? » sur Ubuntu alors que le démon tourne bien, ajoutez sudo devant chaque commande Docker.

Si un jour vous en avez besoin, sachez que vous pourrez voir sur quelle version de l'application vous êtes actuellement :

```
docker version
```

Nous commençons par lancer un premier Docker qui sert uniquement à afficher « Hello world » pour apprendre la commande de lancement :

```
docker run hello-world
```

La première fois, Docker vous prévient qu'il ne trouve pas l'image docker du nom de hello-world ; mais comme il arrive à la trouver, il la télécharge automatiquement et lance ce premier conteneur.

Une commande permet de voir tous les conteneurs qui sont en cours d'exécution :

```
docker ps
```

En ajoutant l'option -a, vous pourrez également voir tous les conteneurs qui ont été lancé mais qui se sont arrêtés.

Travaillons avec des images Docker

Une image Docker est une configuration d'un conteneur que nous pouvons récupérer ou que nous pouvons partager. Docker propose le site Docker Hub qui est un service de partage d'images qui s'est fortement inspiré de GitHub.

Vous pourrez y récupérer un grand nombre d'images dont certaines directement proposées par les éditeurs pour vous mâcher une partie du travail.

Je commence par chercher une image PHP afin de la télécharger :

```
docker search php
```

Le premier résultat est parfait pour mon test, il y a un conteneur du nom de php de disponible. Je décide de le télécharger

```
docker pull php
```

Si vous avez bien suivi ce tutorial docker, j'ai normalement 2 images à présent de téléchargé : hello-world et php. Vérifions cela grâce à la commande suivante qui me permet de connaitre le nom de toutes mes images :

```
docker images
```

Pour cette première partie, nous n'allons pas détailler les options de ces fonctions mais il sera très intéressant de le faire à la suite du tutorial docker.

Je vais créer ma première image Docker

Ce tutorial docker ne sera pas le dernier sur cet outil mais je ne veux pas terminer ce tutorial docker sans entamer ce sujet intéressant des Dockerfile. Nous allons créer notre propre image dès maintenant.

Mettez-vous dans un nouveau dossier et commencez par créer un fichier Dockerfile avec le contenu suivant :

```
FROM ubuntu:14.04

MAINTAINER Judicael paquet
```

Ce fichier va nous permettre de construire notre première image.

Je commence par définir le Linux sur lequel va se créer mon image et l'auteur de celle-ci (autant se faire de l'auto promo).

A présent, nous allons construire notre image :

```
docker build .
```

Il télécharge puis exécute les deux étapes que je lui ai définit dans le Dockerfile. Si tout va bien, le script se termine sur un Successfully Build XXX.

Nous allons rajouter à présent l'installation d'apache 2 en mettant à jour la liste de packages et en installant le package apache2. Voici le Dockerfile mis à jour :

```
FROM ubuntu:14.04

MAINTAINER Judicael paquet



RUN apt-get update && \

    apt-get install -y apache2
```

La commande RUN nous permettra comme vous devez le deviner de lancer des commandes lors de la création de notre image et le caractère \ permet d'écrire notre commande sur plusieurs lignes.

A présent, nous allons reconstruire notre image :

```
docker build .
```

Sans option, il a automatiquement mis en cache le travail précédent d'où le fait qu'il ne prenne du temps que pour l'étape que nous venons de rajouter.

Quand vous commencerez à beaucoup manipuler vos Dockerfile, vous aurez parfois besoin de refaire vos images intégralement sans cache. Vous pourrez alors lancer cette commande :

```
docker build --no-cache .
```

Pour donner un nom à votre image, il vous faudra ajouter l'option -t comme ceci :

```
docker build -t monapache .
```

Cela permettra quand vous listerez vos images d'avoir écrit « monapache » au lieu de « <none> » dans la colonne REPOSITORY.

docker run

Nous allons à présent lancer un conteneur avec apache2. J'ai précisément pris cet exemple car en lançant votre conteneur de façon classique cela ne fonctionnera pas. Il faut lancer Apache en même temps que de lancer le conteneur avec apachetcl et le faire dans le mode -D FOREGROUND :

```
docker run -d -p 8000:80 monapache /usr/sbin/apache2ctl -D FOREGROUND
```

-d : le mode détaché permet de ne pas bloquer votre terminal. Dès que le conteneur est lancé, docker vous rend la main

-p : permet de définir le port à appeler pour qu'il atteigne le port 80 sur le conteneur. C'est très pratique car nous pourrons du coup avoir un conteneur par port.

-D FOREGROUND : permet de lancer le processus dans le conteneur et d'attacher la console au processus d'entrée classique.

Vous pouvez dès maintenant tenter de mettre http://localhost:8000 dans votre navigateur. En effet, on arrive bien à afficher la page par défaut de Linux sur ubuntu.

Vous pouvez à présent vérifier que votre conteneur est également bien lancé avec docker ps. D'ailleurs cette commande vous retourne le nom de votre conteneur qui est mis par hasard quand vous n'indiquez aucun nom lors du lancement de celui-ci.

docker stop

Maintenant vous pourrez arrêter votre conteneur en utilisant la méthode suivante :

```
docker stop [nom du conteneur]
```

Sachant que Apache n'est pas installé sur l'OS hôte mais uniquement sur le conteneur, l'arrêt de ce conteneur rendra Apache indisponible.

Si vous désirez mettre un nom à votre conteneur, vous pourrez utiliser l'option –name comme ceci :

```
docker run -d  --name monconteneur -p 8000:80 monapache /usr/sbin/apache2ctl
-D FOREGROUND
```

Conclusion tutorial docker

Ce premier tutorial Docker vous permet de faire vos premiers pas sur un outil vraiment génial et qui est presque devenu indispensable dans le monde du devops.

Je proposerai des tutoriels encore plus approfondis sur le sujet en espérant que ces premiers pas dans le monde de Docker vous ont intéressés.

Suite : Tutoriel Docker 2 : Maitrisez les Dockerfile

[docker tutorial français – docker tuto]

==========

Docker tutorial PDF est proposé par Myagile Partner.
Ce support est proposé pour vous aider dans votre agilisation de tous les jours.