Docker : LAB 1 - Créer des images Dockers from scratch

Objet	Création d'une image from scratch
Niveau requis	débutant, avisé
Débutant, à savoir	
Suivi	:DRAFT:

Docker est le moyen de disposer de plein d'applications même si celles-ci tournent sur un système d'exploitation différent. Sa force réside dans le fait qu'il n'y a pas besoin d'installer/configurer pour chacune des applications, le système qui lui correspond. Les applications seront donc directement prêtes à l'emploi !

On peut via Docker, construire l'environnement dans lequel on veut faire tourner les applications. L'idée étant de distribuer des applications qui embarquent ce dont elles ont besoin pour fonctionner.

On peut construire tout un environnement pour une petite application (système d'exploitation, configuration, ajout des dépendances, ….), mais en plus on peut choisir comment la démarrer : en mode statique, en mode dynamique, en mode lecture seule, en mode lecture - écriture, en redirigeant des ports, en définissant une ip, en partageant le réseau…

- Une machine virtuelle, c'est l'installation d'un système d'exploitation complet dans lequel on peut faire tourner plusieurs applications.
- Une image docker, c'est une application qui embarque avec elle uniquement les bouts du système d'exploitation dont elle a besoin.

Créer une image Docker hors ligne

Le dockerfile

La première étape d'une image Docker c'est la création d'un dossier contenant un fichier “Dockerfile”

```
mkdir MonProjetDocker
vim MonProjetDocker/Dockerfile
```

Le Dockerfile est le fichier qui va “construire” l'image, il faut lui indiquer les actions qu'il faudra faire: copie de fichier, exécution de commande, ajout de variable d'environnement …

Les premières lignes du fichier Dockerfile seront les suivantes

```
FROM scratch
MAINTAINER Dave Hill < dave-hill [ at ] docker.dyrk.org >
```

FROM sert à indiquer ce que vous souhaitez utiliser comme environnement pour Docker.

Exemple :

```
FROM debian:jessie
```

Docker construira l'image en s'appuyant sur l'image système minimale d'une Debian dans la version Jessie (Version 7).

Bien entendu, pour récupérer l'image de Debian, docker aura besoin que la machine soit connectée à internet, c'est donc pour ça qu'on va utiliser plutôt “scratch” !

Le terme “scratch” indiquera à Docker, que l'on souhaite pas utiliser d'environnement et qu'on veut le construire.

MAINTAINER: Permet de spécifier l'auteur ... vous !

Ajouter un système à l'image Docker

Comme on part sans “image base”, “from scratch”, il est donc important d'ajouter manuellement un système à l'image docker

Il est possible d'en créer soi-même, ou bien d'en télécharger !

Exemple :

Pour créer un environnement Centos 7, il faut aller récupérer l'image de Centos 7 sur le site de Centos

Une fois récupérée, la copier dans mon dossier, et ajouter une nouvelle ligne au dockerfile:

```
ADD CentOS-7-20140625-x86_64-docker_01.img.tar.xz /
```

En clair, cette ligne va dire, de prendre le fichier “CentOS-7-20140625-x86*64-docker*01.img.tar.xz”, et de l'extraire à la racine “/”

ADD permet d'importer des fichiers dans l'image, dans le cas des tar.xz, tar.gz ... elle prendra même l'initiative de les extraire directement.

```
ADD script/SuperScript.sh /root
```

Indique à Docker de prendre le fichier “SuperScript.sh” situé dans le dossier “script” du dossier docker, et de le mettre dans l'image à l'emplacement /root

Exécuter des commandes dans l'image

Par exemple pour ajouter un utilisateur

```

```

```
RUN useradd dyrk
```

RUN va lancer la ligne de commande “useradd dyrk”, qui aboutira sur la création de l'utilisateur Dyrk !

```
RUN chmod +x /root/SuperScript.sh
```

Ici, ons donne des droits d’exécutions au fichier, en exécutant la commande `chmod +x /root/SuperScript.sh`.

Ces commandes-là, seront exécutées PENDANT la construction de l'image, elles ne seront pas exécutées lorsqu'on démarrer/utilise l'image.

Définir des variables d'environnements

Pour ajouter des variables d'environnement, il suffit d'utiliser l'instruction ENV

```
ENV MaVariable "Hello World"
```

Exécuter des commandes au démarrage de votre Image

L'instruction “CMD” permet, lors du démarrage de l'image, d'amorcer directement un script, ou d'exécuter une commande.

Un script pour démarrer une application par exemple.

Chacun des paramètres doit être entre guillemets et séparé par une virgule

```
CMD ["ls", "-la", "/root"]
```

Exécutera au démarrage de l'image la commande suivante :

```
ls -la /root
```

Exemple complet

Voici un basique exemple de “Dockerfile”

```
    FROM scratch
    MAINTAINER Dave Hill < dave-hill [ at ] docker.dyrk.org >
    ADD CentOS-7-20140625-x86_64-docker_01.img.tar.xz /
    ADD script/SuperScript.sh /root
```

```
    RUN chmod +x /root/SuperScript.sh
    CMD ["ls", "-la", "/root"]
```

Dans un premier temps, on va charger les fichiers de CentOs 7, puis on ajoute un script dans /root. On lui donne les droits d'exécution, et enfin, je fait en sorte qu'à chaque démarrage ce script soit lancé.

Les images

Construction de l'image

Pour fabriquer l'image, il faut se positionner dans le dossier ou se trouve le Dockerfile et lancer la commande suivante

```
docker build .
```

A l'issue de cette commande on obtiendra un id du genre 8db3545acd

Il est possible de donner un nom au build, de manière à ne pas avoir à mémoriser par coeur les id.

```
docker build -t nomDeLimage .
```

Lister les images créées

Lorsqu'on fabrique des images dockers, chacune des modifications sera sauvegardée, on peut à tout moment consulter les images dockers enregistrées avec la commande suivante :

```
docker images
```

Démarrer une image

Démarrage standard

Aucun paramètre particulier, l'image va démarrer de manière statique, et s'arrêter.

```
docker run <id ou nom de l'image>
```

Démarrage avec une commande

En spécifiant une commande à exécuter

```
docker run -i <id ou nom de l'image> "ls /"
```

Démarrage dynamique

```
    docker run -i -t <id ou nom de l'image> "ls /"
```

Démarrage en mode écriture

Sans ce paramètre, lesystème sera en ReadOnly, certaines actions seront donc impossible.

```
docker run -i -t --privileged  <id ou nom de l'image> "ls /"
```

Les Conteneurs

Lorsqu'on démarre une image, docker va automatiquement créer ce que l'on appelle un “container”. Ce conteneur sera le récipient dans lequel on exécute les images.

Aussi, pour chaque image que l'on démarrere avec la commande “docker run”, on aura un “container”.

Lister les "containers" démarrés

Il est possible lorsqu'une ou plusieurs images sont démarrées, de lister les containers actifs.

```
docker ps
```

Arrêter le container d'une image en cours d’exécution

```
docker stop <id du container>
```

Accéder à un container en cours d'exécution

Via la commande “docker ps”, on a la possibilité de récupérer l'id du container, pour le rejoindre, il suffira de saisir la ligne de commande suivante

```
docker exec -i -t <id du container> bash
```

Les paramètres sont sensiblement les mêmes que ceux de la commande “docker run”. Comme on peut le constater on a spécifié bash, mais on aurait pu exécuter n'importe quelle autre commande.

pour utiliser internet, ou accéder au réseau depuis l'image docker, il faudra la démarrer en mode écriture, et une fois à l'intérieur, utiliser la commande suivante :

```
sysctl -w net.ipv4.ip_forward=1
```