TP2 – Découverte de Docker

Ce TP a pour objectif de présenter une autre forme de virtualisation appelée conteneurisation. Au lieu de faire appel à un OS complet comme dans une VM, le conteneur est un silo basé sur le même système que celui de l'hôte et permet de se concentrer sur l'application qu'il héberge.

Il existe plusieurs type de conteneurs, depuis le chroot traditionnel du monde Unix, jusqu'aux Jails de BSD et la technologie OpenVZ. Dans le cadre de ce TP, on étudiera le type de conteneurs qui suscite le plus grand intérêt actuellement : Docker.

Docker se base à l'origine sur les nouveaux conteneurs Linux nommés LxC. Mais Docker s'est surtout popularisé pour sa possibilité de gérer toutes les modifications d'une application, au travers du versionning du conteneur l'hébergeant.

Installation de Docker

On suppose que Docker est déjà préinstallé sur les stations de travail du CNAM.

Première utilisation de Docker

Hello World

Pour obtenir un simple « Hello World » avec Docker, tapez la commande suivante :

```
docker run hello-world
```

- quels messages obtenez-vous ?
- votre installation de Docker est-elle opérationnelle ?

Essayez la commande proposée :

```
docker run -it ubuntu bash
```

Relance d'un conteneur

Esayez les commandes suivantes :

```
docker ps -a
```
(récupérer l'ID du conteneur Ubuntu → `857f4160e7c3`)

```
docker start 857f4160e7c3

docker ps

docker exec -it 857f4160e7c3 bash
```
(vérifiez dans quel système vous vous trouvez)

```
docker stop 857f4160e7c3

docker ps
```

Nettoyage des anciens conteneurs

```
docker ps -a
```
(notez l'ID de tous les conteneurs présents)

```
docker rm  857f4160e7c3
```
(faites de même pour tous les ID présents)

Nettoyage des images précédentes

```
docker images

docker rmi hello-world
docker rmi ubuntu

docker images
```

Recherche et récupération d'une image depuis le Hub

On se propose d'installer le serveur web Nginx. On peut vérifier la présence d'un conteneur image de Nginx sur le hub central de Docker. Pour ce faire, tapez la commande suivante :

```
docker search nginx
```

Récupérez l'image en local avec :

```
docker pull nginx
```

> NOTA :
> - la commande `docker pull` fait écho à la commande `git pull` dont elle est inspirée.
> - De la même façon que pour Git, une image Docker est faite d'une base et de différences incrémentales qui intègrent les différentes modifications et versions.

Vérifiez la présence de l'image récupérée :

```
docker images
```

Création d'un conteneur à partir de l'image téléchargée

```
docker run -dit --name myapp nginx

docker ps
```

« Inspection » du conteneur

```
docker inspect myapp | more
```

Retrouvez l'IP du conteneur à partir de cette commande, ainsi que les ports réseaux ouverts.

Redirection de ports réseaux : application à un site web

Arrêtez d'abord l'application web

```
docker stop myapp
docker rm myapp
```

Relancez-la en faisant du NAT sur le ports 80 et 443 du conteneur (80 → 8000 sur la VM, 443 → 8443 sur la VM)

```
docker run -dit --name newapp -p 8000:80 -p 8443:443 nginx
```

Ouvrez la bonne URL (IP:port) pour avoir le message de Nginx.

Modification d'une application en conteneurisation

Entrez dans le conteneur et modifiez le message de Nginx

(installer vim ou nano : `apt-get update && apt-get -y install vim` ou `apt-get -y install nano` dans le conteneur, pour éditer le fichier html)

Build automatisé d'un conteneur

Récupération d'une image Debian de base

```
docker pull debian
```

```
docker images
```

Création d'un fichier de build (nommé `dockerfile`) :

```
FROM debian:latest

RUN apt-get -y update && apt-get install -y fortunes cowsay

CMD /usr/games/fortune  | /usr/games/cowsay -f stegosaurus
```

Création d'une nouvelle image locale à partir du fichier `dockerfile` :

```
docker build -t cnam .
```
(le point final est important)

```
docker images
```

Création d'un nouveau conteneur à partir de cette image et exécution de la commande fournie à « CMD »

```
docker run cnam
```

Modifiez le fichier dockerfile et relancer un conteneur basé sur la nouvelle image.