TP4 : DOCKERFILE

Exercice 1 : Gestion des images

1. Commençons par lister les images disponibles : docker images.
2. Pour ajouter une image localement, il suffit de faire un simple docker pull xxx. Allez sur le hub docker et cherchez une image Apache officielle (aussi connu sous le nom httpd) puis téléchargez-la.
3. Nous allons maintenant créer notre propre image qui va nous permettre de lancer un serveur Web apache. Pour cela, il faut définir la méthode de construction du container dans un fichier.
   a. Créez un répertoire
      ~/Docker/Apache
   b. Dans ce répertoire, créez un fichier Dockerfile. Dans ce fichier, mettez les commandes :

```
FROM ubuntu:latest
        MAINTAINER Votre nom
        ENV DEBIAN_FRONTEND=noninteractive

        RUN apt-get update && apt-get install -y tzdata && apt-get install -y apache2

        WORKDIR /var/www/html

        ENV APACHE_RUN_USER www-data
        ENV APACHE_RUN_GROUP www-data
        ENV APACHE_LOG_DIR /var/log/apache2
        ENV APACHE_PID_FILE /var/run/apache2.pid
        ENV APACHE_RUN_DIR /var/run/apache2
        ENV APACHE_LOCK_DIR /var/lock/apache2

        RUN mkdir -p $APACHE_RUN_DIR $APACHE_LOCK_DIR $APACHE_LOG_DIR

        ENTRYPOINT [ "/usr/sbin/apache2" ]
        CMD ["-D", "FOREGROUND"]
        EXPOSE 80
```

   c. Créons l'image sudo docker build -t="votre nom/apache". Attention à ne pas l'oublier le point '.' à la fin de la commande (qui indique que le fichier Dockerfile est dans le répertoire local. Le nom du container est nécessairement du votre nom/apache et non apache car apache serait interprété comme un répertoire officiel.
   d. Listez vos images : docker images

Exercice 2 : Faire tourner des applications avec Docker

1. Jusqu'à présent, nous avons fait tourner des containers ubuntu en mode interactif. C'est intéressant, mais dans un contexte de production où un opérateur veut démarrer beaucoup de container sur une machine, on veut démarrer les containers en mode

démon, c'est-à-dire en tâche de fond. Cela se fait simplement avec l'option -d.

a. Commençons par un simple docker run -d --name=demon ubuntu /bin/bash. Que se passe-t-il ? Faites un docker ps et docker ps -a pour comprendre

b. Détruisez le container précédent. Nous allons corriger le problème précédent en faisant quelque chose dans le container :
docker run -d --name=demon ubuntu /bin/sh -c "while true ; do echo hello world ; sleep 1 ; done"

c. Montrez ce qui se passe dans le container depuis la machine hôte puis détruisez le container.

2. Démarrez un container votre nom/apache en mode démon en exposant le port 80 :
docker run -d -p 80 --name=apache votre nom/apache
3. La dernière commande du fichier Dockerfile précédent permet d'exposer un port interne du container (le port 80 du serveur apache) depuis l'hôte debian. Pour trouver quel est le port choisi par Docker, il suffit de taper docker port apache 80. Faites un test en ouvrant votre navigateur et vous connectant sur la machine locale (l'hôte Debian) sur le port docker.
4. On peut mieux controller le port choisi. Si par exemple on veut que cela soit le port 80, il suffit de modifier la commande précédente :
docker run -d -p 80:80 --name=apache votre_nom/apache
Stoppez, détruisez et recréez votre container pour vérifier que cela fonctionne.
5. On veut maintenant en plus contrôler le contenu du serveur Web depuis l'hôte debian. Pour cela on va :

(a) créer un répertoire mkdir website dans le répertoire Apache crée précédemment.

(b) mettre dans ce répertoire un fichier index.html avec le contenu suivant :

```
<html>
Hello world
</html>
```

(c) démarrer votre container en montant le répertoire apache là où le serveur Apache dans le container va chercher ses données :
docker run -d -p 80:80 -v ~/Docker/Apache/website:/var/www/html --name=apache votre_nom/apache

(d) vérifiez que le container offre le service attendu.