TP 3 : Prise en main de Docker

Introduction

Ce TP permet de se familiariser avec la CLI Docker et sur les fonctions de base des conteneurs Docker.
Nous montrerons comment :

- Prise en main
- Créer une image
- Dockeriser une application

Installation

Plusieurs méthodes d'installation sont disponibles pour Docker. La plupart des distributions possèdent un package officiel. Néanmoins ce package n'est pas tout le temps à jour, il peut apparaître utile d'utiliser les dépôts fournis par Docker afin de disposer d'un package à jour.

Exercice 1 : installation et prise en main

1. Affichez la version de Docker installé :
   sudo docker version
2. Lancez votre premier conteneur (hello-world image):
   sudo docker run hello-world
   Docker va automatiquement chercher l'image sur Internet si elle n'est pas déjà installée.
3. Vérifiez que le serveur docker (the docker daemon) est bien en cours d'exécution
   ps aux | grep dockerd
4. Listez les conteneurs en cours d'exécution et les exécutions passés :
   docker ps docker ps -a
5. Listez les images installées sur votre système :
   docker images
6. Téléchargez l'image ubuntu sans l'exécuter
   docker pull ubuntu

7. Lancez le conteneur ubuntu

docker run ubuntu

Le conteneur termine immédiatement car aucun processus n'est en cours d'exécution.

8. Lancez plusieurs commandes dans le conteneur ubuntu (sleep, ls, etc.) :

docker run ubuntu ls

docker run ubuntu sleep 10

9. Pour lancer des processus/commandes dans un conteneur en cours d'exécution, on utilise la commande docker exec. Ci-dessous, vous lancez le conteneur ubuntu avec un sleep suffisamment long pour que vous ayez le temps de lancer d'autres commandes puis vous lancez des commandes grâce au nom de votre conteneur :

docker run ubuntu sleep 300 //Le conteneur est partie pour 300sec

docker ps //on visualise le nom du conteneur en cours d'exécution (c'est le dernier champ)

docker exec ls

10. Lancez votre conteneur en background (option –d)

docker run –d ubuntu sleep 200 //Effectuée plusieurs fois vous aurez plusieurs instances du conteneur

11. Remettez un de vos conteneurs en foreground (commande docker ps pour récupérer le nom):

docker attach

Exercice 2 : image et gestion des conteneurs

Dans le cadre de cet exercice, on utilisera un conteneur sous Ubuntu.

- Étape 1 : exécuter la commande suivante :

docker container run –ti ubuntu bash

Cette commande vous permet d'importer une image depuis le Docker Store, et de faire fonctionner un shell bash à l'intérieur de ce conteneur.

- Etape 2 : personnalisation du conteneur en installant de nouveaux paquets à l'aide la commande suivante :

apt-get update
apt-get install –y figlet
figlet <nom figlet>

Vous devrez normalement avoir les mots correspondant au nom de votre figlet affichés sur votre écran. Vous pouvez quitter le conteneur Ubuntu.

- Etape 4 : Récupération de l'ID du conteneur. Exécuter la commande suivante :

docker container ls –a

- Etape 5 : Création d'une image locale avec la commande commit :

docker container commit ID_de_vote_conteneur

- Etape 6 : Vérification de la création de l'image à l'aide de la commande :

docker image ls

Vous devez normalement voir apparaître dans la liste de vos images, l'image Ubuntu de votre conteneur ainsi que l'image que vous venez de créer. Cette dernière, comme vous pourrez le constater, ne possède ni de repository, ni de tag.

- Etape 7 : Attribution d'un tag à votre image à l'aide de la commande suivante :

docker image tag

Vous pouvez ensuite vérifier que l'attribution du tag à bien été réalisée à l'aide de la commande :

docker image ls

- Etape 8 : vous allez maintenant essayer de faire fonctionner un conteneur comportant l'image que vous venez de créer à l'aide de la commande suivante :

docker container run

Exercice 3: Création d'une image à partir d'un dockerfile

Au lieu de créer une image binaire statique, nous pouvons utiliser un fichier appelé un Dockerfile pour créer une image.

On va commencer cet exercice, en créant un fichier dans lequel on récupérera le hostname et on l'affiche. ATTENTION ! Si vous êtes encore dans votre conteneur Ubuntu, vous devez le quitter pour revenir à l'invite de commande de base de Docker. Pour cela, entrez la commande exit.

Etape 1 : Créer un fichier appelé index.js dans le répertoire « app » à l'aide d'un éditeur de texte comme nano ou vi. Ici, n'importe quel éditeur de texte.
Dans ce fichier, écrivez les éléments suivants :

```
var os = require(« os ») ;
var hostname = os.hostname() ;
console.log(« hello from » + hostname) ;
```

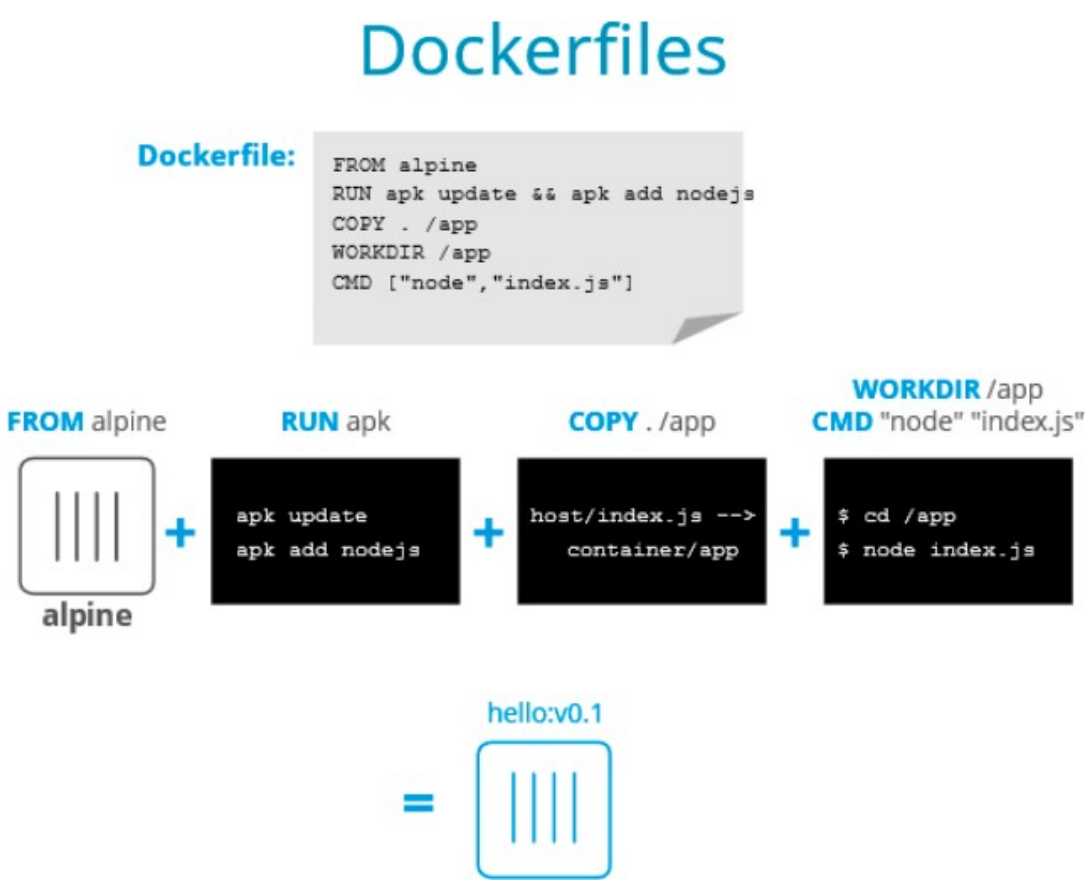
Le fichier que vous venez de créer est le code JavaScript pour votre serveur. Comme vous pouvez vous en doutez, Node.js sert simplement à afficher un message « hello ». On va « dockeriser » cette application en créant un Dockerfile. On utilisera l'alpine comme OS de base, y ajouter du Node.js et copier votre code source dans le conteneur. Nous spécifions aussi la commande par défaut à exécuter pour la création de conteneurs.

Etape 2 : Créer un fichier Dockerfile (Attention les Majuscules sont importantes) et copier le contenu suivant à l'intérieur, une nouvelle fois à l'aide d'un éditeur de texte.

```
FROM alpine
RUN apk update && apk add nodejs
COPY . /app /app
WORKDIR /app
CMD [« node », « index.js »]
```

Maintenant, passons à la création de votre première image en utilisant Dockerfile. Appelez votre image hello:v0.1. Entrez la commande suivante :

```
docker image build –t hello:v0.1 .
```



- Etape 3 : pour vous assurer que votre application fonctionne correctement, vérifier en démarrant le conteneur de votre application avec la commande suivante : **docker container run hello:v0.1**

Vous devriez obtenir en sortie quelque chose de semblable à ce qui suit (excepté l'ID qui sera différente) : hello from 25d79b6cghu