TP 3 : Manipulation des conteneurs avec le CLI

Exercice 1 : Hello from Alpine

Le but de ce premier exercice est de lancer des containers basés sur l'image alpine

1. Lancez un container basé sur **alpine** en lui fournissant la commande **echo hello**

```
docker container run alpine echo hello
```

2. Quelles sont les étapes effectuées par le docker daemon ?

3. Lancez un container basé sur alpine sans lui spécifier de commande. Qu'observez-vous ?

```
docker run alpine
```

Exercice 2 : manipulation des commandes récurrentes

Le but de cet exercice est de manipulern des commandes récurrentes

Exécutez les commandes suivantes

```
$ docker run -it --name=test1 alpine:latest date
$ docker run -it --name=test1 alpine:latest date
```

1. Pourquoi ça ne marche pas?

Le problème est que le même nom de conteneur a été utilisé pour les deux exécutions et que le premier conteneur n'a pas été supprimé.

2. Que pouvez-vous faire pour les faire fonctionner tous les deux (il y a au moins deux façons)?

Vous pouvez supprimer le conteneur (en utilisant --rm) avec la première commande d'exécution ou vous pourrez peut-être utiliser un nom unique pour la deuxième exécution.

```
$ docker run -it --name=test1 --rm alpine:latest date
$ docker run -it --name=test1 alpine:latest date
```

Vous pouvez supprimer manuellement le premier conteneur après l'avoir exécuté.

```
$ docker run -it --name=test1 alpine:latest date
$ docker rm test1
$ docker run -it --name=test1 alpine:latest date
```

Changez le nom du conteneur.

```
$ docker run -it --name=test1 alpine:latest date
$ docker run -it --name=test2 alpine:latest date
```

Exercice 3 : shell intéractif

Le but de cet exercice est de lancer des containers en mode interactif

1. Lancez un container basé sur alpine en mode interactif sans lui spécifier de commande

```
docker run -it alpine
```

2. Que s'est-il passé ?
3. Quelle est la commande par défaut d'un container basé sur alpine

```
sh
```

4. Naviguez dans le système de fichiers
5. Utilisez le gestionnaire de package d'alpine (apk) pour ajouter un package

```
$ apk update
$ apk add curl
```

Exercice 4 : foreground / background

Le but de cet exercice est de créer des containers en foreground et en background

1. Lancez un container basé sur alpine en lui spécifiant la commande **`ping 8.8.8.8`**

```
docker run alpine ping 8.8.8.8
```

2. Arrêter le container avec CTRL-C

Le container est-il toujours en cours d'exécution ?

```
docker ps
```

⇨ Non le conteneur est arrêté

Note: vous pouvez utiliser la commande **`docker ps`** que nous détaillerons dans l'une des prochaines lectures), et qui permet de lister les containers qui tournent sur la machine.

3. Lancez un container en mode interactif en lui spécifiant la commande **`ping 8.8.8.8`**

```
docker run -ti alpine ping 8.8.8.8
```

4. Arrêter le container avec CTRL-P CTRL-Q

Le container est-il toujours en cours d'exécution ?

⇨ Oui le conteneur est toujours fonctionnel

5. Lancez un container en background, toujours en lui spécifiant la commande **`ping 8.8.8.8`**

```
docker run -d alpine ping 8.8.8.8
```

Le container est-il toujours en cours d'exécution ?

⇨ Oui le conteneur est toujours fonctionnel

Exercice 5 : liste des containers

Le but de cet exercice est de montrer les différentes options pour lister les containers du système

1. Listez les containers en cours d'exécution

```
docker ps
```

2. Est-ce que tous les containers que vous avez créés sont listés ?

Non il y a des conteneurs arrêtés

3. Utilisez l'option -a pour voir également les containers qui ont été stoppés

```
docker ps -a
```

4. Utilisez l'option -q pour ne lister que les IDs des containers (en cours d'exécution ou stoppés)

```
docker ps -a -q
```

Exercice 6 : inspection d'un container

Le but de cet exercice est l'inspection d'un container

1. Lancez, en background, un nouveau container basé sur nginx:1.14 en exposant le port 80 du container.

```
docker run -d --expose=80 nginx
```

2. Notez l'identifiant du container retourné par la commande précédente.

3. Inspectez le container en utilisant son identifiant

```
docker inspect c09d806264c4
```

4. En utilisant le format Go template, récupérez le nom et l'IP du container

```
docker inspect --format '{{.NetworkSettings.IPAddress}}' c09d806264c4
```

5. Manipuler les Go template pour récupérer d'autres informations

6. Pour plus de pratique, comment déterminer rapidement et succinctement l'identifiant de l'image et la date de création d'une image Alpine?

```
docker inspect alpine:latest --format='{{.Id}} {{.Created}}'
```

Exercice 7 : exec dans un container

Le but de cet exercice est de montrer comment lancer un processus dans un container existant.

1. Lancez un container en background, basé sur l'image alpine. Spécifiez la commande **`ping 8.8.8.8`** et le nom ping_container avec l'option --name

```
docker run -d --name ping_container alpine ping 8.8.8.8
```

2. Observez les logs du container en utilisant l'ID retourné par la commande précédente ou bien le nom du container

```
docker logs ping_container
```

Quittez la commande de logs avec CTRL-C

3. Lancez un shell sh, en mode interactif, dans le container précédent

```
docker exec -it ping_container sh
```

4. Listez les processus du container

```
ps
```

Qu'observez vous par rapport aux identifiants des processus ?

Exercice 7 : cleanup

Le but de cet exercice est de stopper et de supprimer les containers existants

1. Listez tous les containers (actifs et inactifs)
1. Stoppez tous les containers encore actifs en fournissant la liste des IDs à la commande stop
2. Vérifiez qu'il n'y a plus de containers actifs
3. Listez les containers arrêtés
4. Supprimez tous les containers
5. Vérifiez qu'il n'y a plus de containers

Exercice 8 : Installer LAMP dans un conteneur ubuntu

Le but de cet exercice est de'installer LAMP dans un conteneur ubuntu

L'acronyme LAMP désigne un ensemble de quatre technologies open source : un système d'exploitation Linux, un serveur web Apache, un système de bases de données MySQL et le langage de programmation PHP. L'objectif de cet exercice est d'installer et configurer ces quatre briques dans un conteneur Docker en se basant sur l'image **Ubuntu 18.04** ainsi que d'installer phpMyAdmin pour l'administration graphique de la base de données MySQL.

1. Vérifier que le moteur Docker est en cours d'exécution
2. Chercher l'image de conteneur en question
3. Télécharger l'image en local
4. Lancer un conteneur en se basant sur l'image déjà téléchargé
5. Mettre à jour les sources listes du conteneur
6. Installer le serveur web Apache 2 dans le conteur en question
7. Démarrer le serveur web Apache
8. Consulter la page index.php en utilisant l'adresse IP de conteneur

9. Installer le SGBD MySQL dans le conteneur en question
10. Sécuriser l'accès au SGBD MySQL
11. Installer le gestionnaire graphique PHPMyAdmin dans le conteneur en question
12. Configurer le serveur web Apache pour supporter PHPMyAdmin
13. Accéder à MySQL en utilisant la page à la page phpmyadmin.php

Correction

```
[root@localhost ~]# systemctl start docker

[root@localhost ~]# systemctl status docker

[root@localhost ~]# docker search bionic

[root@localhost ~]# docker pull docker.io/cleardata/bionic

[root@localhost ~]# docker images

[root@localhost ~]# docker run -it ba274ec1e096 /bin/bash

root@747b7ad47497:/# apt-get update

root@747b7ad47497:/# apt-get install apache2

root@747b7ad47497:/# service apache2 start

root@747b7ad47497:/# service apache2 status

[root@localhost ~]# docker ps

[root@localhost ~]# docker inspect jovial_keller | grep "IPAddress"

root@747b7ad47497:/# apt-get install software-properties-common

root@747b7ad47497:/# add-apt-repository ppa:ondrej/php

root@747b7ad47497:/# apt-get update

root@747b7ad47497:/# apt-get install -y php7.4

root@747b7ad47497:/# apt-get install mysql-server

root@747b7ad47497:/# service mysql start

root@747b7ad47497:/# mysql_secure_installation

root@747b7ad47497:/# mysql -u root -p
```