Création d’une image à partir d’un container

Enoncé

1. Lancez une container basé sur une image *alpine:3.8*, en mode interactif, et en lui donnant le nom *c1*

2. Lancez la commande *curl google.com*

Qu'observez-vous ?

3. Installez *curl* à l’aide du gestionnaire de package *apk*

4. Quittez le container avec CTRL-P CTRL-Q (pour ne pas killer le processus de PID 1)

5. Créez une image, nommée *myping*, à partir du container *c1*

Utilisez pour cela la commande *commit* (*docker commit --help* pour voir le fonctionnment de cette commande)

6. Lancez un shell intéractif dans un container basée sur l’image *myping* et vérifiez que *curl* est présent

Correction

1. La commande suivante permet de créé le container demandé

On utilise l'option --name pour spécifier le nom du container.

```
$ docker container run -ti --name c1 alpine:3.8
```

Suite à cette commande, on se retrouve dans un shell *sh* dans le container.

2. L'utilitaire *curl* n'est pas disponible dans une image alpine, il faut l'installer.

```
/ # curl google.com
/bin/sh: curl: not found
```

3. La commande suivante permet de mettre à jour la liste des packages et d'installer l'utilitaire curl:

```
/ # apk update && apk add curl
fetch http://dl-cdn.alpinelinux.org/alpine/v3.8/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.8/community/x86_64/APKINDEX.tar.gz
v3.8.2-13-g106f36ecbb [http://dl-cdn.alpinelinux.org/alpine/v3.8/main]
v3.8.2-8-g684f341f68 [http://dl-cdn.alpinelinux.org/alpine/v3.8/community]
OK: 9545 distinct packages available
(1/5) Installing ca-certificates (20171114-r3)
(2/5) Installing nghttp2-libs (1.32.0-r0)
(3/5) Installing libssh2 (1.8.0-r3)
(4/5) Installing libcurl (7.61.1-r1)
(5/5) Installing curl (7.61.1-r1)
Executing busybox-1.28.4-r2.trigger
Executing ca-certificates-20171114-r3.trigger
OK: 6 MiB in 18 packages
```

5. Afin de créer l'image *myping* à partir du container *c1*, il faut utiliser la commande suivante

```
$ docker container commit c1 myping
```

6. La commande suivante permet de lancer un shell *sh* dans un container basé sur l'image *myping*.

Note: même si l'on ne précise pas la commande *sh*, la commande utilisée par défaut est celle de l'image à partir de laquelle *myping* a été créée, c'est à dire *alpine*.

```
$ docker container run -ti myping
```

Depuis ce shell, nous pouvons relancer la commande de la question 2.

```
/ # curl google.com
<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>302 Moved</TITLE></HEAD><BODY>
<H1>302 Moved</H1>
The document has moved
```

```
<A HREF="http://www.google.fr/?
gfe_rd=cr&dcr=0&ei=cPXFWsz_N4zUXoGQgrAL">here</A>.
</BODY></HTML>
```

Pour résumer

Nous avons donc lancé un container, ajouté un binaire dans ce container et commité le tout en une nouvelle image. Le binaire est donc présent dans cette image. Commiter un container pour créer une image n'est pas l'approche recommandée. La création d'une image se fait à partir d'un Dockerfile, fichier texte contenant l'ensemble des commandes nécessaires.