DM Docker

Mastère DevOps

Par : Antoine FERRO

Intervenant : S.LAKHDARI

Année : 2020

Consignes :
Créer un serveur web qui affiche le fichier index.html
Le fichier index.html est produit par un autre conteneur qui insère dans le fichier la date toutes les 60 mn

Environnement :

- Système d'exploitation : Ubuntu 20.04.2 LTS
- Dossier de travail : */home/aferro/docker/web*
- Logiciels : Docker Engine - Community version 20.10.2
- Image : nginx & httpd

Description :
Pour ce projet, je n'ai pas effectué de fichier yml ou de docker-compose. Le résultat est le même que si j'en avais utilisé, vous verrez donc, au fil de ce TP, la façon dont j'ai procédé.

Pour commencer, j'ai créé un nouveau réseau de type bridge nommé "web_server" :
$ docker network create web_server
1c28e76230fe939fe7ff38482daa763d4ab56e4fc919e615c3899331ba59d08b

Ce dernier apparaît bien dans la liste des network :
$ docker network ls

NETWORK ID	*NAME*	*DRIVER*	*SCOPE*
f05364d14f24	*bridge*	*bridge*	*local*
eb78d07e5d52	*host*	*host*	*local*
acfe9fb1033b	*mybridge*	*bridge*	*local*
9c60976e0dd5	*web*	*bridge*	*local*
1c28e76230fe	*web_server*	*bridge*	*local*

1) Création d'un nouveau volume

Dans la consigne, il est clairement demandé de créer deux conteneurs, il faut donc un nouveau volume dans lequel les conteneurs seront présents. Ce volume va permettre d'échanger différentes données entre les deux conteneurs.

$ docker volume create --name web_share
web_share

Je vérifie que tout s'est bien passé :

$ docker volume ls

DRIVER	*VOLUME NAME*
local	*nginx_logs*
local	*registry*
local	*web_share*

2) Création des conteneurs

a. nginx

Je commence par exécuter la commande suivante, afin de run une image nginx, et accessible depuis le volume web_share :

$ docker run -tid -v web_share:/home/aferro/docker/web --net web_server nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
a076a628af6f: Pull complete
0732ab25fa22: Pull complete
d7f36f6fe38f: Pull complete
f72584a26f32: Pull complete
7125e4df9063: Pull complete
Digest: sha256:10b8cc432d56da8b61b070f4c7d2543a9ed17c2b23010b43af434fd40e2ca4aa
Status: Downloaded newer image for nginx:latest
5ccea7897dfb634ebee7ddc3a07df2dcd1eb4dab2cf96d9273067b88f1944e6b

b. httpd

Je crée ensuite httpd, qui recevra les requêtes HTTP (port 80) des clients et qui redirigera vers nginx. Httpd est le programme du serveur HTTP d'apache.

```
$ docker run  -tid -v web_share:/home/aferro/docker/web --net web_server -p 80:80 httpd
Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd
a076a628af6f: Already exists
e444656f7792: Pull complete
0ec35e191b09: Pull complete
4aad5d8db1a6: Pull complete
eb1da3ea630f: Pull complete
Digest: sha256:2fab99fb3b1c7ddfa99d7dc55de8dad0a62dbe3e7c605d78ecbdf2c6c49fd636
Status: Downloaded newer image for httpd:latest
c1834626ea1111dde16662281fcb27d107dd78a3c3df10a554099f845ebc455a
```

A présent, je me connecte sur le conteneur httpd afin de créer un fichier index.html dont voici le contenu :

```
$ cat >> web/index.html
123456789
```

3) Configuration Apache

Je dois à présent configurer un fichier de configuration apache situé à l'endroit suivant :

```
$ nano /usr/local/apache2/conf/httpd.conf
```

Avant d'effectuer cette commande, il faut savoir que j'ai dû mettre à jour les paquets et installer nano.

Je descends jusqu'au *DocumentRoot* car les fichiers situés en dessous de ce dernier constituent l'arborescence de base des documents qui seront visibles depuis le web. Je modifie donc les valeurs entrées :

```
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "/home/aferro/docker/web"
<Directory "/home/aferro/docker/web">
```

Un redémarrage est nécessaire afin que ces changements s'appliques :

```
$ docker restart c183
c183
```

Je vérifie les différentes adresses IP des conteneurs.

$ docker inspect network web_server

```
…
      "Containers": {
         "5ccea7897dfb634ebee7ddc3a07df2dcd1eb4dab2cf96d9273067b88f1944e6b": {
            "Name": "tender_meninsky",
          "EndpointID": "490c19ddc3841923038fdd3f42d083ff45285009765075ee9d0c13e60139983d",
            "MacAddress": "02:42:ac:12:00:03",
            "IPv4Address": "172.18.0.3/16",
            "IPv6Address": ""
         },
         "c1834626ea1111dde16662281fcb27d107dd78a3c3df10a554099f845ebc455a": {
            "Name": "great_sinoussi",
         "EndpointID": "d4c9ed1ce512ec5a25fbbf60446eb4b4c3b45075456a491475448b4fd5a11945",
            "MacAddress": "02:42:ac:12:00:02",
            "IPv4Address": "172.18.0.2/16",
            "IPv6Address": ""
         }
...
```

Voici à quel conteneur correspond quel ID :

$ docker ps

CONTAINER ID	*IMAGE*	*COMMAND*	…
c1834626ea11	*httpd*	*"httpd-foreground"*	…
5ccea7897dfb	*nginx*	*"/docker-entrypoint.…"*	…

4) Création du script bash

Il est maintenant temps de créer, dans le conteneur nginx, le script bash qui va nous permettre d'insérer dans le fichier index.html la date toutes les 60min (je commence par 120 secondes afin de vérifier si le script est fonctionnel :

```
$ docker exec -it c1834 bash
$ cat /home/aferro/docker/script.sh
#!/bin/bash
while true
do
echo "" > /home/aferro/docker/nginx_web/index.html
date +"%D %R" >> /home/aferro/docker/web/index.html
sleep 120
done
```

Comme vous pouvez le voir, ce script va initialiser le fichier index.html toutes les 120 secondes (soit 2min) en ajoutant la date et l'heure :

```
$ curl 172.18.0.2
02/07/21 12:16
```

2min plus tard :

$ curl 172.18.0.2
02/07/21 12:18

Par la suite, je modifie le script pour qu'il sleep 3600 secondes (1h) :

$ cat /home/aferro/docker/script.sh
#!/bin/bash
while true
do
echo "" > /home/aferro/docker/nginx_web/index.html
date +"%D %R" >> /home/aferro/docker/web/index.html
sleep 3600
done