AGUILAR_Flavien_Docker_TP

Flavien AGUILAR

30/11/2020

I - Docker sous Linux :

1) Une fois docker installé, nous pouvons vérifier la version avec la commande :

```
[miiruki@detp:~ $] docker --version
Docker version 19.03.13-ce, build 4484c46d9d
```

Dans notre cas, la version 19.03.13-ce est installée.

2) Lorsque nous tapons la commande

```
[miiruki@detp:~ $] docker run hello-world
```

Docker va récupérer le container « hello-world » et le lancer sur la machine. Dans notre cas, le tout c'est bien passé veut dire que l'image à bien été récupérée et qu'elle s'est correctement executée.

Le fichier sur le docker hub permettant à la création du container est le fichier Dockerfile

3) Nous recherchons les images Debian :

```
[miiruki@detp:~ $] docker search Debian
NAME                                                      DESCRIPTION
                                                STARS               OFFICIAL
               AUTOMATED
ubuntu                                                    Ubuntu is a Debian-
    based Linux operating sys..   11552                 [OK]
debian                                                    Debian is a Linux
    distribution that's compos..  3668                  [OK]
arm32v7/debian                                            Debian is a Linux
    distribution that's compos..  68
itscaro/debian-ssh                                        debian:jessie
                                      28
                                          [OK]
arm64v8/debian                                            Debian is a Linux
    distribution that's compos..  23
samueldebruyn/debian-git                                  a minimal docker
    container with debian and g..  22
                                          [OK]
multiarch/debian-debootstrap                              multiarch ports of
    debian-debootstrap            13
i386/debian                                               Debian is a Linux
    distribution that's compos..  12
```

```
11 eboraas/debian                                          Debian base images,
      for all currently-availa..  8
      [OK]
12 smartentry/debian                                       debian with
      smartentry                              5
                                                 [OK]
13 amd64/debian                                            Debian is a Linux
      distribution that's compos..  4
14 ppc64le/debian                                          Debian is a Linux
      distribution that's compos..  4
15 vicamo/debian                                           Debian docker images
       for all versions/archit..  3
16 s390x/debian                                            Debian is a Linux
      distribution that's compos..  2
17 vpgrp/debian                                            Docker images of
      Debian.                          2
18 arm32v5/debian                                          Debian is a Linux
      distribution that's compos..  2
19 spritsail/debian-builder                                A Docker image based
       on debian:slim ideal fo..  1
      [OK]
20 dockershelf/debian                                      Repository for
      docker images of Debian. Test..  1
                                                 [OK]
21 holgerimbery/debian                                     debian multiarch
      docker base image                1
22 jdub/debian-sources-resource                            Concourse CI
      resource to check for updated D..  0
                                                 [OK]
23 fleshgrinder/debian                                     Debian base images
      for production and multis..  0
                                                 [OK]
24 mdoerges/debian-buster-nginx                            Debian Buster with
      Nginx                            0
25 casept/debian-amd64                                     A debian image built
       from scratch. Mostly fo..  0
26 1and1internet/debian-9-nginx-php-7.2-wordpress-4        debian-9-nginx-php
      -7.2-wordpress-4                  0
                                                 [OK]
27 mdoerges/debian-buster-php73                            Debian Buster with
      PHP 7.3                          0
```

4) Nous pouvons créer notre premier container à partir de l'image debian avec la commande :

```
1 [miiruki@detp:~ $] docker run -d debian:latest
2 Unable to find image 'debian:latest' locally
3 latest: Pulling from library/debian
4 756975cb9c7e: Pull complete
```

```
5 Digest: sha256:
     e2cc6fb403be437ef8af68bdc3a89fd58e80b4e390c58f14c77c466002391193
6 Status: Downloaded newer image for debian:latest
7 85c19ea4053b9fc6098e056c248c65fb1c5ec97a1ec1683512898525aef09ffd
```

5) Nous pouvons vérifier que le container est bien créer avec la commande :

```
1 [miiruki@detp:~ $] docker ps -a
2 CONTAINER ID        IMAGE               COMMAND             CREATED
                 STATUS                      PORTS               NAMES
3 85c19ea4053b        debian:latest       "bash"              7 seconds
     ago       Exited (0) 6 seconds ago
     nice_perlman
4 87039a4a8eff        hello-world         "/hello"            3 hours ago
             Exited (0) 3 hours ago
     affectionate_almeida
```

6) Nous pouvons attacher un shell à notre container et exécuter un script simple :

```
1 [miiruki@detp:~ $] docker run debian:latest bash -c "while : ; do echo
     "coucou"; sleep 1 ; done "
2 coucou
3 coucou
4 coucou
5 coucou
6 coucou
7 coucou
8 coucou
9 coucou
10 coucou
11 coucou
12 coucou
13 coucou
14 coucou
15 coucou
```

7) Nous pouvons stopper et redémarrer le container avec la commande :

```
1 [miiruki@detp:~ $] docker ps
2 CONTAINER ID        IMAGE               COMMAND
     CREATED             STATUS              PORTS               NAMES
```

```
3 c73dc2250d6a        debian:latest       "bash -c 'while : ;.."   2
     minutes ago       Up 2 minutes
     romantic_ishizaka
4 [miiruki@detp:~ $] docker restart c73dc2250d6a
5 c73dc2250d6a
6 [miiruki@detp:~ $] docker ps
7 CONTAINER ID        IMAGE               COMMAND
     CREATED             STATUS              PORTS               NAMES
8 c73dc2250d6a        debian:latest       "bash -c 'while : ;.."   2
     minutes ago       Up 5 seconds
     romantic_ishizaka
```

8) Nous pouvons supprimer le container avec la commande :

```
1 [miiruki@detp:~ $] docker rm c73dc2250d6a
2 c73dc2250d6a
```

9) Nous pouvons rentrer dans le container à son démarrage avec la commande :

```
1 [miiruki@detp:~ $] docker run -it debian:latest
2 root@0bad82c3c124:/# ls
3 bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run
       sbin  srv  sys  tmp  usr  var
```

10) Nous pouvons nommer le container et son hostname DebianOne avec la commande :

```
1 [miiruki@detp:~ $] docker run -it -h DebianOne --name DebianOne debian:
     latest
2 root@DebianOne:/#
```

11) Pour nous attacher au container, nous allons commencer par le démarrer :

```
1 [miiruki@detp:~ $] docker start DebianOne
2 DebianOne
```

Nous allons ensuite devoir nous attacher au container avec la commande :

```
1 [miiruki@detp:~ $] docker attach DebianOne
```

Pour se détacher du container nous avons juste à taper la commande :

```
root@DebianOne:/# exit
exit
```

12) Nous pouvons exécuter une commande shell avec la commande :

```
[miiruki@detp:~ $] docker exec DebianOne ls
bin
boot
dev
etc
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
```

13) Nous pouvons lister le dernier container créer :

```
[miiruki@detp:~ $] docker ps -l
CONTAINER ID        IMAGE               COMMAND             CREATED
                STATUS              PORTS               NAMES
c1bd73a6ed55        debian:latest       "bash"              14 minutes
   ago      Up 6 minutes                                DebianOne
```

14) Nous pouvons attacher un volume à notre container avec la commande :

```
[miiruki@detp:~ $] docker run -it -v $(pwd)/flavien:/home debian:latest
```

Cette pratique peut être utile si nous voulons copier des fichiers de configuration par exemple.

15) Pour supprimer le container, nous allons taper la commande :

```
[miiruki@detp:~ $] docker rm 0d0de3a2dd4d d2980a799893 cab0830cda54 3b4a05863804 a065a0347a89 0bad82c3c124 852161a84794 b86b45608f7a 629d9f7409c0 42b04dacdaf2 85c19ea4053b 87039a4a8eff
0d0de3a2dd4d
d2980a799893
cab0830cda54
3b4a05863804
a065a0347a89
0bad82c3c124
852161a84794
b86b45608f7a
629d9f7409c0
42b04dacdaf2
85c19ea4053b
87039a4a8eff
```

16) Pour supprimer une image docker, nous allons utiliser la commande :

```
[miiruki@detp:~ $] docker rmi hello-world
Untagged: hello-world:latest
Untagged: hello-world@sha256:e7c70bb24b462baa86c102610182e3efcb12a04854e8c582838d92970a09f323
Deleted: sha256:bf756fb1ae65adf866bd8c456593cd24beb6a0a061dedf42b26a993176745f6b
Deleted: sha256:9c27e219663c25e0f28493790cc0b88bc973ba3b1686355f221c38a36978ac63
```

17) Nous pouvons purger notre système avec la commande :

```
[miiruki@detp:~ $] docker system prune
WARNING! This will remove:
  - all stopped containers
  - all networks not used by at least one container
  - all dangling images
  - all dangling build cache

Are you sure you want to continue? [y/N] y
Total reclaimed space: 0B
```

II - Création Image Docker

1) Pour créer notre image à partir du Dockerfile, nous allons utiliser la commande :

```
[miiruki@detp:~/Documents/tpdocker $] docker build -t debian:fa .
```

2) Dans le Dockerfile, les différentes commandes servent à :

RUN : Permet d'éxecuter des commandes à l'intérieur du container ENV : Permet de définir les paramètres sytsème FROM : Permet d'importer une image existante

3) Le fait de limiter le nombre de commande permet de réduire la taille du container. En faisant tous les apt update en une seule commande, nous allons de ce fait optimiser la taille du container.
4) Pour que notre container envoi un script, nous allons devoir rajouter les commandes suivantes à la fin du Dockerfile :

```
ENTRYPOINT ["/bin/ping"]
CMD ["-c4","www.iutbeziers.fr"]
```

Le entrypoint va permettre d'initialiser la commande et le cmd de passer les options.

Nous pouvons le vérifier le résultat :

```
[miiruki@detp:~/Documents/tpdocker $] docker run pingfour
PING www.iutbeziers.fr (194.199.227.80) 56(84) bytes of data.
64 bytes from www.iutbeziers.fr (194.199.227.80): icmp_seq=1 ttl=62
    time=0.990 ms
64 bytes from www.iutbeziers.fr (194.199.227.80): icmp_seq=2 ttl=62
    time=1.21 ms
64 bytes from www.iutbeziers.fr (194.199.227.80): icmp_seq=3 ttl=62
    time=0.801 ms
64 bytes from www.iutbeziers.fr (194.199.227.80): icmp_seq=4 ttl=62
    time=1.19 ms

--- www.iutbeziers.fr ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3026ms
rtt min/avg/max/mdev = 0.801/1.049/1.211/0.171 ms
```

5) Lorsque nous lançons le container avec l'option –rm , le container va s'éxécuter et se supprimer tout seul une fois sa tâche accomplie.

6) Nous pouvons modifier la destination et le nombre de ping en modifiant le dockerfile et en reconstruisant l'image.

7) Nous pouvons rajouter l'option –entrypoint au docker run pour pouvoir faire un traceroute à la place du ping :

```
[miiruki@detp:~/Documents/tpdocker $] docker run --entrypoint
   tracreroute -it pingfour -I google.fr
```

8) Nous pouvons transformer notre container en image avec la commande :

```
[miiruki@detp:~/Documents/tpdocker $] docker commit e797af06ecd2
sha256:cf43b9c386d20ce110582123fe949aa57e9f8a49392245366b433294a50f00fb
```

9) Une fois l'image créée, nous allons la tagger :

```
[miiruki@detp:~/Documents/tpdocker $] docker tag cf43b9c386d2 pingfour:
   fa
```

11) Une fois l'image créée et la visibilitée modifiée, nous pouvons pull l'image docker. Dans mon cas, ce sera Mr Pouchoulon qui réalisera le pull de mon image depuis sa machine.

12) Pour créer un container qui vas faire office de serveur ssh en écoute sur le port 2222, nous allons utiliser le Dockerfile suivant :

```
FROM registry.iutbeziers.fr/debianiut

RUN apt update && apt upgrade -y && apt install -y ssh
EXPOSE 2222
RUN mkdir -p /run/sshd
RUN echo 'root:root' | chpasswd
RUN sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin yes/'
   /etc/ssh/sshd_config
ENTRYPOINT ["/usr/sbin/sshd"]
CMD ["-D","-p2222"]
```

Nous allons maintenant build notre image Docker :

```
[miiruki@detp:~/Documents/tpdocker/ssh_serv $] docker build -t ssh:fa .
Sending build context to Docker daemon  2.048kB
Step 1/8 : FROM registry.iutbeziers.fr/debianiut
 ---> 5437d6e1db42
Step 2/8 : RUN apt update && apt upgrade -y && apt install -y ssh
 ---> Using cache
 ---> a8ad29aef3c1
Step 3/8 : EXPOSE 2222
 ---> Using cache
 ---> a9b36f1ae347
Step 4/8 : RUN mkdir -p /run/sshd
 ---> Using cache
 ---> 99e7c7929a51
Step 5/8 : RUN echo 'root:root' | chpasswd
 ---> Using cache
 ---> f767642b5093
Step 6/8 : RUN sed -i 's/#PermitRootLogin prohibit-password/
   PermitRootLogin yes/' /etc/ssh/sshd_config
 ---> Running in 40bd461c171b
Removing intermediate container 40bd461c171b
 ---> b0e724a43348
Step 7/8 : ENTRYPOINT ["/usr/sbin/sshd"]
 ---> Running in 2723522cf16b
Removing intermediate container 2723522cf16b
 ---> f52b60f5d125
Step 8/8 : CMD ["-D","-p2222"]
 ---> Running in 964cbc831bc3
Removing intermediate container 964cbc831bc3
 ---> 85e043b41eaf
Successfully built 85e043b41eaf
Successfully tagged ssh:fa
```

Nous allons maintenant lancer le container :

```
[miiruki@detp:~/Documents/tpdocker/ssh_serv $] docker run -d --name
   flavien ssh:fa
5f7e45f96b3c730edbbb2f71f6c46c4424c223b95f5c36c98f546235df9e1dd5
```

Nous pouvons maintenant récupérer l'adresse IP du container :

```
[miiruki@detp:~/Documents/tpdocker/ssh_serv $] docker inspect flavien |
    grep IPAd
            "SecondaryIPAddresses": null,
            "IPAddress": "172.17.0.2",
                    "IPAddress": "172.17.0.2",
```

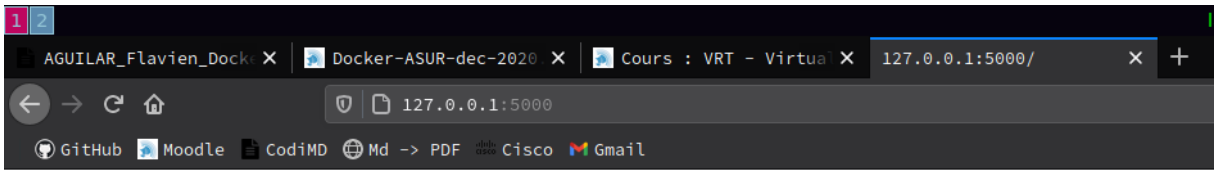
Nous allons maintenant nous connecter sur le container :

```
[miiruki@detp:~/Documents/tpdocker/ssh_serv $] ssh root@172.17.0.2 -p
    2222
root@172.17.0.2's password:
Linux 5f7e45f96b3c 5.8.18-1-MANJARO #1 SMP PREEMPT Sun Nov 1 14:10:04
    UTC 2020 x86_64

The programs included with the Debian GNU/Linux system are free
    software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@5f7e45f96b3c:~# pwd
/root
root@5f7e45f96b3c:~# exit
déconnexion
Connection to 172.17.0.2 closed.
```

13) Nous allos éxecutter le script python :

```
[miiruki@detp:~/Documents/tpdocker/ssh_serv $] python3 app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: development
 * Debug mode: on
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 214-226-594
```

Nous pouvons accéder à la page WEB depuis le navigateur :



Le python c'est bon mangez en

14) Nous allons maintenant faire tourner cette apli dans un container docker en utilisant le Dockerfile suivant :

```
[miiruki@detp:~/Documents/tpdocker/python $] cat Dockerfile
FROM python:3

WORKDIR /code
COPY ./app.py .
RUN pip install flask
RUN export ENV FLASK_APP=app.py
RUN export ENV FLASK_ENV=development
EXPOSE 9999
ENTRYPOINT ["python3"]
CMD ["app.py"]
```

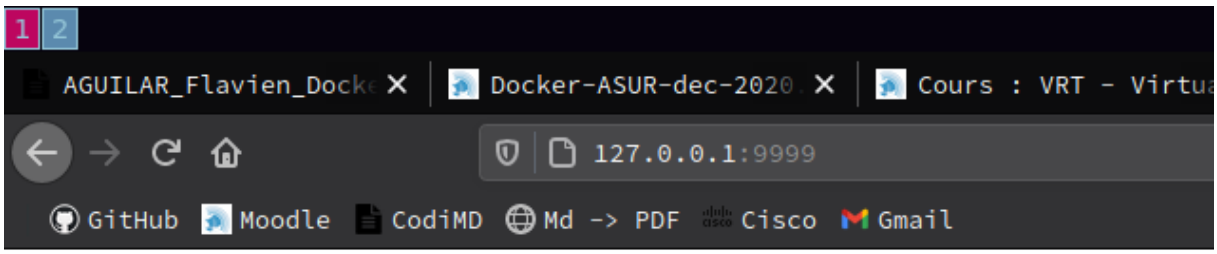
Nous allons maintenat build l'image :

```
[miiruki@detp:~/Documents/tpdocker/python $] docker build -t python:fa
    .
```

Nous pouvons lancer le container :

```
[miiruki@detp:~/Documents/tpdocker/python $] docker run -p 9999:5000 -d
    --name python python:fa
```

L'appli est accessible depuis le port 9999 de la machine hôte :



Le python c'est bon mangez en