## Table des matières



# 1) Docker sous Linux

## 1) Installation de docker sous Linux

1) Pour savoir quelle est notre version de docker on utilise la commande suivante :

```
test@203-5:~$ docker -v
Docker version 19.03.2, build 6a30dfca03
```

On peut voir que notre version est la 19.03.2.

2) On vérifie que notre installation fonctionne bien avec la commande suivante :

```
test@203-5:~$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working
correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the
Docker Hub.
```

```
 3. The Docker daemon created a new container from that image
which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client,
which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container
with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://cloud.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/
```

a) Le retour de cette commande nous explique que docker répond bien à notre hello (en vert) et que ce message est pour vérifier que notre installation fonctionne bien correctement.

La suite de ce message nous explique la procédure utilisée pour vérifier que l'installation est bonne.

b) Je vais maintenant sur https://hub.docker.com/ pour retrouver l'image hello-world

c) Voir ls informations sur le site https://hub.docker.com/_/hello-world

3) Pour rechercher les images officielle Debian on utilise la commande suivante : seules celles avec le [OK] dans la colonne official sans bien des images officielles.

```
test@203-5:~$ docker search debian
NAME                                                     DESCRIPTION
STARS               OFFICIAL            AUTOMATED
ubuntu                                                   Ubuntu is a
Debian-based Linux operating sys…   10010                  [OK]
debian                                                   Debian is a
Linux distribution that's compos…   3235                   [OK]
arm32v7/debian                                           Debian is a
Linux distribution that's compos…   61
itscaro/debian-ssh                                       debian:jessie
26                                      [OK]
...
```

Ensuite on les installes ainsi que les images officielle busybox on utilise les commandes suivantes :

```
test@203-5:~$ docker pull debian:latest
latest: Pulling from library/debian
4a56a430b2ba: Pull complete
```

```
Digest:
sha256:e25b64a9cf82c72080074d6b1bba7329cdd752d51574971fd37731ed164
f3345
Status: Downloaded newer image for debian:latest
docker.io/library/debian:latest
```

```
test@203-5:~$ docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
7c9d20b9b6cd: Pull complete
Digest:
sha256:fe301db49df08c384001ed752dff6d52b4305a73a7f608f21528048e8a0
8b51e
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest
```

4) Maintenant je crée mon premier container Debian StretchOne grâce à la commande suivante :

```
test@203-5:~$ docker run -it --hostname JessieOne --name JessieOne
debian:latest bash
```

Expliquons la commande petit à petit :

- En premier le -it veut dire interactif (il permet donc à docker d'interagir avec nous)

- Ensuite le –hostname permet de définir le nom d'hôte de notre container

- Pour continuer le –name permet de définir le nom du container

- Pour terminer la commande bash permet d'exécuter le container

5) Je quitte maintenant mon container et je liste mon container nouvellement crée :

```
test@203-5:~$ docker ps -l
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS                      PORTS
NAMES
9ac8859a6081        debian:latest       "bash"              10
minutes ago      Exited (0) 2 minutes ago
```

Maintenant je ne liste que le dernier container ID :

```
test@203-5:~$ docker ps -l
```

```
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS                      PORTS
NAMES
9ac8859a6081        debian:latest       "bash"              14
minutes ago      Exited (0) 6 minutes ago
```

# 2) Création d'une image Docker

1) Je commence par cloner le container du prof grâce à la commande suivante :

```
git clone
https://registry.iutbeziers.fr:5443/pouchou/debianiut.git
```

2) La commande RUN permet de faire comme si on était sans le terminal et que nous lancions une commande

La commande ENV permet de sélectionner ce que l'on veut de base (par exemple avec ENV on peut choisir la langue : FR) il

La commande FROM permet de dire de quel container on part

3) L'intérêt de tout faire en une seule fois permet de ne pas créer plusieurs couches et donc de surcharger le fichier ce qui le rendra lus lent à l'exécution.

4)

Une fois fait je me déplace dans le dossier debianiut et je crée ma propre image via la commande suivante :

```
test@203-5:~$ docker build -t monping .
```

Une fois fait j'édite le fichier Dockerfile via la commande vim :

```
FROM registry.iutbeziers.fr/debianiut:latest
CMD ["/bin/ping","-c","4","www.iutbeziers.fr"]
```

Maintenant je construis l'image via la commande docker build :

```
test@203-5:~/debianiut$ docker build -t monping .
Sending build context to Docker daemon  73.73kB
Step 1/2 : FROM registry.iutbeziers.fr/debianiut:latest
 ---> d04524022501
```

```
Step 2/2 : CMD ["/bin/ping","-c","4","www.iutbeziers.fr"]
 ---> Running in 02c985ee1ff8
Removing intermediate container 02c985ee1ff8
 ---> e91695adb989
Successfully built e91695adb989
Successfully tagged monping:latest
```

Et maintenant je n'ai plus qu'à lancer l'image avec la commande docker run :

```
test@203-5:~/debianiut$ docker run monping
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=1.10 ms
64 bytes from www.iutbeziers.fr (194.199.227.80): icmp_seq=2
ttl=62 time=1.28 ms
64 bytes from www.iutbeziers.fr (194.199.227.80): icmp_seq=3
ttl=62 time=1.95 ms
64 bytes from www.iutbeziers.fr (194.199.227.80): icmp_seq=4
ttl=62 time=1.38 ms

--- www.iutbeziers.fr ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 1.108/1.433/1.952/0.315 ms
```

5) Le -rm sert à supprimer l'image juste après avoir fait la commande.

6) Non on ne pas changer la destination ou le nombre de ping

7)

# 3) Orchestration avec Ansible

Je commence par récupérer le script du prof via la commande suivante :

```
test@203-5:~/debianiut$ git clone
https://registry.iutbeziers.fr:5443/pouchou/tp3automatisation.git
Clonage dans 'tp3automatisation'...
remote: Enumerating objects: 106, done.
remote: Counting objects: 100% (106/106), done.
remote: Compressing objects: 100% (83/83), done.
remote: Total 106 (delta 53), reused 50 (delta 20)
Réception d'objets: 100% (106/106), 21.17 KiB | 0 bytes/s, fait.
Résolution des deltas: 100% (53/53), fait.
```

Je dois passer en su maintenant.

## 3.1) Installation de l'environnement du TP via un script

## 3.2) Installation et paramétrage d'Ansible

J'installe Ansible grâce au script via la commande suivante :

```
root@203-5 : /home/test/debianiut/tp3automatisation
# bash install_ansible.sh
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
build-essential is already the newest version (12.3).
python-dev is already the newest version (2.7.13-2).
python-setuptools is already the newest version (33.1.1-1).
python-setuptools passé en « installé manuellement ».
Les paquets suivants ont été installés automatiquement et ne sont
plus nécessaires :
  geoip-database-extra icedtea-netx icedtea-netx-common libjs-
openlayers
  libwireshark8 libwiretap6 libwscodecs1 libwsutil7
  linux-headers-4.9.0-3-amd64 linux-headers-4.9.0-3-common
...
```

## 3.3) Installation des containers SSH

Je commence par générer une clé SSH pour mon utilisateur root :

```
root@203-5 : /home/test/debianiut/tp3automatisation
# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
...
```

Je lance ensuite le script create-cont.sh via la commande suivante :

```
root@203-5 : /home/test/debianiut/tp3automatisation
# bash create-cont.sh
effacement des containers existants
################################
ff90ef6162e9
d5929b9dcc1a
44bc805b5072
60c32372ca16
...
```

## 3.4) Prise en main d'Ansible

1) Le fichier /etc/ansible/hosts sert à gérer tous les hôtes qui se situent dans ansible

2) Pour créer un groupe container qui regroupe tous les containers j'édite le fichier hosts :

```
[container]
debian
centos
```

3) Le fichier ansible.cfg sert à configurer ansible

4) Je vérifie avec la commande suivante que mes cibles ansible sont bien vivantes :

```
root@203-5 : /home/test/debianiut/tp3automatisation
# ansible all -m ping
 [WARNING]: Found both group and host with same name: debian

 [WARNING]: Found both group and host with same name: centos
```

L'option «host_key_checking = false» sert à ne pas demander des clés SSH aux hôtes. Il ne regarde pas les clés SSH des hôtes.

5) La couche applicative utilisée par ansible est SSH (7émé couche du modèle OSI)

6) Pour débugger ansible il faut utiliser l'option -vvv. La commande se présentera sous la façon suivante :

[commande] [option] [argument]

Ici nous avons fait

```
ansible all -vvv -m ping
```

## 3.5) Installation d'apache via la modules yum et apt d'Ansible

1) J'installe le serveur apache2 sous Debian à l'aide de la commande ansible sur mes containers.

Je commence par créer un fichier .yml et je l'édite de la façon suivante :

```
- hosts: debian
  tasks:
  - name: Installation d'Apache sur Debian
    apt: name=apache2 update_cache=yes
```

Une fois le fichier édité j'utilise la commande suivante pour installer le serveur apache sur les containers Debian :

```
root@203-5 : /home/test/debianiut/tp3automatisation
# ansible-playbook -i hosts playbook.yml

PLAY [debian]
*********************************************************************
*********************************************************************

TASK [Gathering Facts]
*********************************************************************
*********************************************************
ok: [debian-1]
ok: [debian-2]
ok: [debian-0]
ok: [debian-3]
ok: [debian-4]

TASK [Installation d'Apache sur Debian]
*********************************************************************
************************************
 [WARNING]: Updating cache and auto-installing missing dependency:
python-apt

 [WARNING]: Could not find aptitude. Using apt-get instead

changed: [debian-1]
changed: [debian-0]
changed: [debian-3]
changed: [debian-4]
changed: [debian-2]

PLAY RECAP
*********************************************************************
*********************************************************************
**
debian-0                   : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
debian-1                   : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
debian-2                   : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
debian-3                   : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
debian-4                   : ok=2    changed=1    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

Idinpotance : Ansible y est

rm et ls n’y sont pas

Indinpotance veut dire que si l’on refait plusieurs fois la même commande rien ne change entre les deux.

**Ansible copie les commandes au fur et à mesure. Il se connecte à distance aux hôtes**