

**Nom** : Francisco

**Prénoms** : Gaspar da Rosa

**Groupe** : gr1-alt

**N° Étudiant** : uapv2502991

**Parcours** : M1 - ILSSEN

## **TP2 – Conteneurs Et Images**

### **Partie 2 – Installation et démarrage de Docker**

- La commande docker pour afficher la version : **sudo docker version**

Version Client = Version Server = **27.3.1**

- La commande qui permet de voir les composants du daemon Docker qui tournent :  
**sudo systemctl status docker**

- Les services utilisés par docker sont : **docker.service** (lance le daemon) et **docker.socket** (écoute les connexions à l'API Docker). Les services ont été démarré dans root, l'utilisateur est << **dockeruser@vmdocker** >>

Ainsi les commandes sont :

- **sudo systemctl status docker.service**
- **sudo systemctl status docker.socket**

- Pour arrêter docker on utilise la commande : **sudo systemctl stop docker**

Après avoir arrêté docker, le statut du socket est **inactif**.

- Pour désactiver docker : **sudo systemctl disable docker**

Et pour le réactiver : **sudo systemctl enable docker**

- Pour permettre à l'utilisateur dockeruser d'exécuter les commandes docker sans passer par sudo, je me suis servi des commandes **sudo groupadd docker** et **sudo usermod -aG docker dockeruser**. ensuite j'ai testé en regardant la version de docker (**docker version**).

### Partie 3 – Premières manipulations de conteneurs et d'images

- Le répertoire dans lequel Docker stocke ses objets c'est : **/var/lib/docker**
- Les différentes catégories d'objets Docker stockées sont : **containers, images, volumes, plugins, networks**, etc.
- En utilisant la commande **sudo ls /var/lib/docker/containers | wc -l**, je constate que pour l'instant il y en a 0.
- La commande que j'ai utilisé pour rechercher l'image hello-world est : **docker search hello-world**
- Plus étoilé : **docker search --filter=stars=10000 nginx**  
En utilisant la commande **docker run hello-world**

```
arm64v8/hello-world          Hello World! (an example of minimal Dockeriz... 3
wjimenez5271/hello-world      0
danfengliu/hello-world        0
lbadger/hello-world           0
ansibleplaybookbundle/hello-world  Simple containerized application that tests ... 0
swarna3005/hello-world         0
kousik93/hello-world          0
silver8642/hello-world         0
dockeruser@vmdocker:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:d211f485f2dd1dee407a80973c8f129f00d54604d2c90732e8e320e5038a0348
Status: Downloaded newer image for hello-world:latest

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.
    (amd64)
 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://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

dockeruser@vmdocker:~$
```

- Après exécution, il y a 1 container.

- Pour vérifier le contenu du répertoire des images, j'utilise la commande : **sudo ls /var/lib/docker/image** et je remarque qu'il y a une image. Grâce à la commande **docker images** je confirme bien que l'image **hello-world** a bien été créé.

- En utilisant la commande **docker images --digests**, le sha256 de l'image est :

```
sha256:d211f485f2dd1dee407a80973c8f129f00d54604d2c90732e8e320e5038a0348
```

- Pour lister les conteneurs en exécution, j'utilise la commande : **docker ps**

- La commande renvoie une liste vide, ce qui veut dire qu'il y a aucun container qui s'exécute.

- La commande docker qui permet de lister les images c'est : **docker images**

Son identifiant est

```
IMAGE ID  
d2c94e258dcb
```

- Docker utilise une image en cache pour les exécutions répétées.

- En tapant la commande **docker ps -a | wc -l**, je constate qu'il y a 3 containers. La commande **docker ps -a** je vérifie que 1 conteneur s'exécute et 2 sont stockés localement.

- Mon essai de suppression de l'image grâce à la commande **docker rmi hello-world** a été un échec. La raison est que, Docker empêche la suppression car l'image est en cours d'utilisation par un conteneur.

- La commande qui permet de lister tous les conteneurs et pas uniquement ceux en exécution et observez leur statut est la commande : **docker ps -a**

- Les containers s'appellent : **hello-world**, **hello-world** et **nginx**.

- Pour afficher l'information de façon non tronquée, j'utilise les commandes :

- pour les containers en exécution seulement = **docker ps --no-trunc**
- pour tous les containers = **docker ps -a --no-trunc**

- Pour exécuter une nouvelle fois l'image puis supprimer le conteneur, j'utilise les commandes : **docker run --name hello1 hello-world** et **docker rm hello1**

- Pour exécuter le conteneur en utilisant le SHA256 de l'image, j'ai utilisé la commande : **docker run --name hello2 <sha256-of-image>**

Je constate que :

- Docker crée un tout nouveau conteneur distinct même si l'image est la même.
- Le nom attribué explicitement a été pris en compte par docker.

### Conclusions :

1. Docker permet de créer plusieurs conteneurs à partir de la même image sans conflit.
2. Chaque conteneur a un identifiant unique même si plusieurs conteneurs sont basés sur la même image, qu'elle soit lancée par nom ou SHA256.

- Pour supprimer l'image (sans forcer) j'ai utilisé :

- j'ai listé les containers : **docker ps -a**
- j'ai supprimé les conteneurs liés à l'image : **docker rm <id\_conteneur/nom>**
- finalement j'ai supprimé l'image : **docker rmi hello-world**

- En utilisant la commande **docker info**, je remarque qu'il y a trois containers et deux images.

- Pour n'afficher que les IDs des conteneurs, j'utilise la commande : **docker ps -aq**

- Une possibilité du bash pour exploiter les résultats de cette dernière commande afin de supprimer tous les conteneurs en une seule commande : **docker rm \$(docker ps -aq)**

- En utilisant la commande **docker rmi nginx**, je constate que l'image a bien été supprimée sans être forcée. Docker a permis la suppression car aucun container n'utilisait l'image suite à la suppression de tous les containers.

- Après l'utilisation de la commande **docker run --name hello1 hello-world**, je constate que le container a bien été créé avec le nom hello1, basé sur l'image hello-world.

- L'identifiant de l'image est :

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hello-world	latest	d2c94e258dcb	18 months ago	13.3kB

- J'ai créé le container en utilisant la commande **docker create --name hello1 hello-world**.

Après avoir utilisé la commande **docker ps -a**, je constate que son statut c'est **Exited (0)**. Le conteneur est terminé et a renvoyé un code de sortie (0) pour un arrêt normal.

- En utilisant la commande **docker start hello1**, j'obtiens:

```
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.
    (amd64)
 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://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

dockeruser@vmdocker:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND         CREATED        STATUS          PORTS   NAMES
b340de3c8c3a   hello-world    "/hello"       12 seconds ago Exited (0) 11 seconds ago
```

Son statut est toujours **Exited (0)**. Cela est dû au fait que l'image **hello-world** est conçue pour un usage très spécifique : elle affiche un simple message de bienvenue, puis le conteneur se termine immédiatement.

- L'option qui permettra d'attacher l'entrée et la sortie standard pour démarrer ce conteneur hello2 sont -i (interactif) et -t (terminal). J'utilise donc la commande: **docker run -it --name hello2 hello-world**

- **docker run -it --name hello1 hello-world**

- Pour exécuter un conteneur en mode sans affichage standard (detach), j'utilise la commande : **docker run -d --name hello3 hello-world**

Je constate qu'il est bien présent dans la liste des conteneurs.

- Pour exécuter un conteneur et le supprimer automatiquement après son exécution, j'utilise la commande : **docker run --rm --name hello4 hello-world**

Je constate qu'il n'est pas présent dans la liste des conteneurs.

## Partie 4 – DockerHub

- Sur ce registre public on trouve deux différents types d'images. Ils sont classifiées en : **images officielles** et **images communautaires**.

- J'ai trouvé l'image officielle d'Ubuntu en cherchant "ubuntu" sur DockerHub et en sélectionnant celle avec le badge "**Official Image**".

- Les différentes versions d'Ubuntu sont distinguées par des **tags** ou des noms de code.

Il y a des tags représentant des versions spécifiques, latest, rolling et des noms de code comme focal et bionic.

- La version la plus récente est actuellement **24.10**. La différence est que tag pointe généralement vers la version LTS (Long Term Support) recommandée mais n'est pas nécessairement la version la plus récente en termes de date de sortie. Leurs identifiants respectifs sont : **oracular** et **noble**.

- La version la plus récente n'a pas de vulnérabilités => **None found**

- Elles sont organisées de la manière suivante :

- Unspecified severity
- Low severity
- Medium severity
- High severity
- Critical severity

- La version **latest** est une version stable et largement utilisée, tandis que **noble** est une version plus récente ou optimisée. Utiliser **noble** peut offrir des avantages en termes de performance ou compatibilité avec des bibliothèques récentes.

- Elles diffèrent à partir du nombre de vulnérabilités.

- L'image Ubuntu téléchargé c'est la dernière (**latest**).

- Pour télécharger la version la plus récente : **docker pull ubuntu:24.10**

```
dockeruser@vmdocker:~$ docker images ubuntu
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        24.10     e40b6e31bd8c   2 weeks ago   80.1MB
ubuntu        latest    b1d9df8ab815   2 weeks ago   78.1MB
```

- La commande par défaut pour un conteneur Ubuntu est : **/bin/bash**

- La commande qui est lancée lorsqu'on exécute un conteneur à partir de l'image Ubuntu c'est : **docker inspect --format='{{.Config.Cmd}}' ubuntu**

```
dockeruser@vmdocker:~$ docker inspect --format='{{.Config.Cmd}}' ubuntu
[/bin/bash]
```

- Pour supprimer l'image j'ai utilisé : **docker rmi ubuntu:rolling**

## Partie 5 - Interagir avec un conteneur

-

```
dockeruser@vmdocker:~$ docker run ubuntu
dockeruser@vmdocker:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
0b3d893616a7   ubuntu   "/bin/bash"   10 seconds ago   Exited (0) 9 seconds ago           nifty_r
amanujan
dockeruser@vmdocker:~$ docker start -ai 0b3d893616a7
dockeruser@vmdocker:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
0b3d893616a7   ubuntu   "/bin/bash"   About a minute ago   Exited (0) 13 seconds ago           ni
fty_ramanujan
dockeruser@vmdocker:~$ docker rm 0b3d893616a7
0b3d893616a7
```

- Pour lancer un conteneur nommé **os\_ubuntu** en interactif et attaché à un terminal : **docker run -it --name os\_ubuntu ubuntu**

- La commande correspondant au processus de PID 1 de ce conteneur est : `ps -aux`

-

```
root@a289bc00ccf3:/# whoami
root
root@a289bc00ccf3:/# pwd
/
root@a289bc00ccf3:/# ls
bin    dev    home   lib64  mnt    proc   run    srv    tmp    var
boot  etc    lib    media  opt    root   sbin   sys    usr
root@a289bc00ccf3:/# hostname
a289bc00ccf3
```

- En plus de son statut qui indique qu'il est **en cours d'exécution**, je vérifie également le **nom** du conteneur, son **image**, et son **port d'attachement**.

-

```
root@97c5d458f348:/# cd /home
root@97c5d458f348:/home#
```

- Le conteneur a été arrêté et son statut indique "**EXITED(0)**" et depuis combien de secondes.

- Je me trouve dans le répertoire par défaut **/ (racine)**. Puisque aucun chemin de démarrage n'a été défini, le conteneur démarre dans le répertoire par défaut.

- PID container : **37468**

-

```
dockeruser@vmdocker:~$ docker inspect os_ubuntu | jq '[0].State.Pid'
37468
```

-

```
dockeruser@vmdocker:~$ ps -p 37468
  PID TTY          TIME CMD
 37468 pts/0    00:00:00 bash
```

- Le statut du conteneur est "**Up**" (**en cours d'exécution**) .

- Le répertoire courant est toujours **root**. Le fichier a été créé.



```
dockeruser@vmdocker:~$ docker diff os_ubuntu
C /root
A /root/.bash_history
A /fichier_test.txt
```

```
dockeruser@vmdocker:~$ docker exec os_ubuntu hostname
97c5d458f348
```

```
dockeruser@vmdocker:~$ ps aux | grep bash
dockeru+ 36332 0.0 0.2 9184 5932 tty1 S+ 14:20 0:00 -bash
dockeru+ 37125 0.0 0.3 10388 6980 pts/0 Ss 15:31 0:00 -bash
dockeru+ 37350 0.0 0.2 8736 5532 pts/1 Ss 15:33 0:00 -bash
root 37468 0.0 0.1 4588 4016 pts/0 Ss+ 15:43 0:00 /bin/bash
dockeru+ 37573 0.0 1.3 1772560 26672 pts/0 Sl+ 16:02 0:00 docker exec -it os_ubuntu bash
root 37592 0.0 0.1 4588 3960 pts/1 Ss+ 16:02 0:00 bash
dockeru+ 37603 0.0 0.1 6480 2236 pts/1 S+ 16:02 0:00 grep --color=auto bash
```

```
dockeruser@vmdocker:~$ docker top os_ubuntu
UID          PID          PPID         C           STIME       TTY
TIME
root         37468        37445        0           15:43       pts/0
00:00:00     /bin/bash
root         37592        37445        0           16:02       pts/1
00:00:00     bash
```

```
root@97c5d458f348:/# exit
exit
dockeruser@vmdocker:~$
```

```
-
dockeruser@vmdocker:~$ docker run --name ll ubuntu ls -l
total 48
lrwxrwxrwx    1 root root    7 Apr 22  2024 bin -> usr/bin
drwxr-xr-x    2 root root 4096 Apr 22  2024 boot
drwxr-xr-x    5 root root  340 Dec  9 16:09 dev
drwxr-xr-x    1 root root 4096 Dec  9 16:09 etc
drwxr-xr-x    3 root root 4096 Nov 19 09:52 home
lrwxrwxrwx    1 root root    7 Apr 22  2024 lib -> usr/lib
lrwxrwxrwx    1 root root    9 Apr 22  2024 lib64 -> usr/lib64
drwxr-xr-x    2 root root 4096 Nov 19 09:46 media
drwxr-xr-x    2 root root 4096 Nov 19 09:46 mnt
drwxr-xr-x    2 root root 4096 Nov 19 09:46 opt
dr-xr-xr-x 191 root root    0 Dec  9 16:09 proc
drwx-----   2 root root 4096 Nov 19 09:52 root
drwxr-xr-x    4 root root 4096 Nov 19 09:52 run
lrwxrwxrwx    1 root root    8 Apr 22  2024 sbin -> usr/sbin
drwxr-xr-x    2 root root 4096 Nov 19 09:46 srv
dr-xr-xr-x   13 root root    0 Dec  9 16:09 sys
drwxrwxrwt    2 root root 4096 Nov 19 09:52 tmp
drwxr-xr-x   12 root root 4096 Nov 19 09:46 usr
drwxr-xr-x   11 root root 4096 Nov 19 09:52 var
```

```
-
dockeruser@vmdocker:~$ docker start ll
ll
```

```
-
dockeruser@vmdocker:~$ docker run --rm --name ps ubuntu ps aux
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  10.0   0.1   7888   3652 ?        Rs   16:12    0:00 ps aux
```

```
-
dockeruser@vmdocker:~$ docker run --name salut ubuntu echo Bonjour
Bonjour
dockeruser@vmdocker:~$ docker start salut
salut
```

```
-
dockeruser@vmdocker:~$ docker run -d --name infini ubuntu sh -c "while true; do sleep 3600; done"
186e909e51132f73e55757e52fb1cf61ccfc8d1542ec5aabaa4733d34aed970f
dockeruser@vmdocker:~$ docker rm -f infini
infini
```

[illegible]

```
dockeruser@vmdocker:~$ docker attach os_ubuntu
salut
salut
salut
salut
salut
salut
salut
^C
root@97c5d458f348:/#
```

```
root@97c5d458f348:/# exit 1
exit
dockeruser@vmdocker:~$
```

```
dockeruser@vmdocker:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
394388cb572a	ubuntu	"echo Bonjour"	8 minutes ago	Exited (0) 8 minutes ago		salut
d6e2cf0537aa	ubuntu	"ls -l"	12 minutes ago	Exited (0) 12 minutes ago		ll
97c5d458f348	ubuntu	"/bin/bash"	49 minutes ago	Exited (1) About a minute ago		os_ubuntu



```
-
dockeruser@vmdocker:~$ docker inspect python:3.9 | jq '.[0].RootFS.Layers'
[
  "sha256:301c1bb42cc0bc6618fcdf036e8711f2aad66f76697f541e2014a69e1f456aa4",
  "sha256:0e82d78b3ea1b1db9fa3e1f18d6745e0c2380c25f2c7cec420257084e9cc44fe",
  "sha256:c81d4fdb67fcfd8ffbf9e9f93f440264d36a2d9e7c4e79b9ae5152c5ed2e3fd36",
  "sha256:0aeceb7c293df4fb677b2771713e9c6abeabf8b7f06bfb071310e6cc1a3aa084",
  "sha256:8f9a13bfb118975875edd547c5c0762eed442b686d86fa46832bf04337f75316",
  "sha256:24f0c2413cd7a5e1e06bbb497657405c0d81b86142567b8425dea83b3a1d635d",
  "sha256:fe5bbd4f8a4224acb21f695f361216e88e2db8bc531064ae2d482635d5f357ae"
]
```

```
-
dockeruser@vmdocker:~$ docker inspect python:3.9 | jq '.[0].RootFS.Layers[-1]'
"sha256:fe5bbd4f8a4224acb21f695f361216e88e2db8bc531064ae2d482635d5f357ae"
```

- Toutes les couches sont différentes. En fait, docker ne télécharge que les nouvelles.

```
dockeruser@vmdocker:~$ docker inspect python:3.14.0a1 | jq '.[0].RootFS.Layers'
[
  "sha256:24b5ce0f1e07d37a35460f50d058afc738619e431013d2e1727609bdf2d7fc",
  "sha256:b6ca42156b9f492afa27c366f20e4e864cef8dd8d0e0a100497764b05b39e6fc",
  "sha256:00547dd240c419fa2e1b33e66aba302e8dfa4bfe6401a972d94a03b1355cbc6c",
  "sha256:96d99c63b722657062d3f33cc230e33b191ea9855c050f44871e173709597e35",
  "sha256:9744b636d758d56bfeceb5e712ddfecbe662951562155cc3f93af8cfd538422c",
  "sha256:4068925b787de0e570d68e70bc04de2380276817a36a343c08aad3435c221113",
  "sha256:da64f9c6a005a83af29c8389262e6de4bb9b1b5ae96ceb6c567e01e7c7525cde"
]
```

- Il n'y a pas de couche partagée ou identique entre les deux versions.

```
dockeruser@vmdocker:~$ docker inspect python:3.14.0a1 | jq '.[0].RootFS.Layers'
[
  "sha256:24b5ce0f1e07d37a35460f50d058afc738619e431013d2e1727609bdf2d7fc",
  "sha256:b6ca42156b9f492afa27c366f20e4e864cef8dd8d0e0a100497764b05b39e6fc",
  "sha256:00547dd240c419fa2e1b33e66aba302e8dfa4bfe6401a972d94a03b1355cbc6c",
  "sha256:96d99c63b722657062d3f33cc230e33b191ea9855c050f44871e173709597e35",
  "sha256:9744b636d758d56bfeceb5e712ddfecbe662951562155cc3f93af8cfd538422c",
  "sha256:4068925b787de0e570d68e70bc04de2380276817a36a343c08aad3435c221113",
  "sha256:da64f9c6a005a83af29c8389262e6de4bb9b1b5ae96ceb6c567e01e7c7525cde"
]
dockeruser@vmdocker:~$ docker inspect python:3.9 | jq '.[0].RootFS.Layers'
[
  "sha256:301c1bb42cc0bc6618fcdf036e8711f2aad66f76697f541e2014a69e1f456aa4",
  "sha256:0e82d78b3ea1b1db9fa3e1f18d6745e0c2380c25f2c7cec420257084e9cc44fe",
  "sha256:c81d4fdb67fcfd8ffbf9e9f93f440264d36a2d9e7c4e79b9ae5152c5ed2e3fd36",
  "sha256:0aeceb7c293df4fb677b2771713e9c6abeabf8b7f06bfb071310e6cc1a3aa084",
  "sha256:8f9a13bfb118975875edd547c5c0762eed442b686d86fa46832bf04337f75316",
  "sha256:24f0c2413cd7a5e1e06bbb497657405c0d81b86142567b8425dea83b3a1d635d",
  "sha256:fe5bbd4f8a4224acb21f695f361216e88e2db8bc531064ae2d482635d5f357ae"
]
```

```
-
dockeruser@vmdocker:~$ docker run -it --name os_ubuntu ubuntu
root@36c04c733b8c:/# echo "partie 6" > /home/test.txt
root@36c04c733b8c:/# dockeruser@vmdocker:~$ |
```

```
dockeruser@vmdocker:~$ docker export os_ubuntu -o os_ubuntu.tar
dockeruser@vmdocker:~$ |
```

- Elle est taggée "latest"

```
dockeruser@vmdocker:~$ cat os_ubuntu.tar | docker import - image_ubuntu_with_file
sha256:52eb9f9a4932b11dec86653409700dcf0a3abad153461f04ba4ebf2d068ce50d
dockeruser@vmdocker:~$ docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
image_ubuntu_with_file latest          52eb9f9a4932    10 seconds ago  78.1MB
python               3.9             f327fe247a06    5 days ago     999MB
ubuntu               24.10           e40b6e31bd8c    2 weeks ago    80.1MB
ubuntu               latest          b1d9df8ab815    2 weeks ago    78.1MB
python               3.14.0a1        80ad471000e7    7 weeks ago    1.02GB
dockeruser@vmdocker:~$ docker history image_ubuntu_with_file
IMAGE               CREATED             CREATED BY          SIZE              COMMENT
52eb9f9a4932        19 seconds ago     /bin/sh -c #(nop)  78.1MB            Imported from -
```

- Effectivement le document créé précédemment existe toujours.

```
dockeruser@vmdocker:~$ docker commit os_ubuntu image2
sha256:33fb230ad8c417883a887b1dc76f9471e440a6faedaa83363f71d53956cb63ad
dockeruser@vmdocker:~$ docker history image2
IMAGE               CREATED             CREATED BY          SIZE              COMMENT
33fb230ad8c4        6 seconds ago      /bin/bash           9B
b1d9df8ab815        2 weeks ago        /bin/sh -c #(nop)  0B                CMD ["/bin/bash"]
<missing>           2 weeks ago        /bin/sh -c #(nop)  78.1MB            ADD file:bcebbf0fddcba5b86...
<missing>           2 weeks ago        /bin/sh -c #(nop)  0B                LABEL org.opencontainers....
<missing>           2 weeks ago        /bin/sh -c #(nop)  0B                LABEL org.opencontainers....
<missing>           2 weeks ago        /bin/sh -c #(nop)  0B                ARG LAUNCHPAD_BUILD_ARCH
<missing>           2 weeks ago        /bin/sh -c #(nop)  0B                ARG RELEASE
dockeruser@vmdocker:~$ docker run -it image2
root@1f97265002bf:/# |
```

- Suppression des conteneurs et des images créés.

```
dockeruser@vmdocker:~$ docker rm -f $(docker ps -aq)
1f97265002bf
36c04c733b8c
dockeruser@vmdocker:~$ docker rmi image2 image_ubuntu_with_file
Untagged: image2:latest
Deleted: sha256:33fb230ad8c417883a887b1dc76f9471e440a6faedaa83363f71d53956cb63ad
Deleted: sha256:d947ee34b6128d9f5a17b583a9e24b799dc974f321d9865d7c48f9447edd32e2
Untagged: image_ubuntu_with_file:latest
Deleted: sha256:52eb9f9a4932b11dec86653409700dcf0a3abad153461f04ba4ebf2d068ce50d
Deleted: sha256:d3c08ea279facedbd51ff6bb8f89bcce8e335ef5f5eeadc076f3b4b240d021bf
dockeruser@vmdocker:~$ |
```

## Partie 7 – Quelques informations générales

- Pour consulter la consommation des conteneurs en exécution, j'utilise la commande :  
**docker stats** (**--no-stream** => tag optionnelle)

- Pour consulter la consommation disque des différents objets Docker, j'utilise la commande:  
**docker system df** (**--verbose** => tag optionnelle)