

Partie 1 :

- Ajoutez un alias ipa pour la commande ip -c -br a. Que fait cette commande ?

La commande permet d'afficher les interfaces réseaux présentes dans la machine.
Elle affiche un rapport si les interfaces sont "UP" ou "DOWN" avec leurs adresses IP

- Vérifiez que l'accès réseau est possible en lançant 3 ping consécutifs sur quad9.

La commande pour faire 3 ping consécutif est :

```
ping -c 3 9.9.9.9
```

Résultat :

```
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.  
64 bytes from 9.9.9.9: icmp_seq=2 ttl=63 time=29.8 ms  
64 bytes from 9.9.9.9: icmp_seq=3 ttl=63 time=30.4 ms
```

```
--- 9.9.9.9 ping statistics ---
```

```
3 packets transmitted, 2 received, 33.3333% packet loss, time 2007ms  
rtt min/avg/max/mdev = 29.764/30.057/30.351/0.293 ms
```

- On va accéder à la VM depuis la machine hôte. Quelle est l'adresse IP de cette VM ? Vérifiez en "pingant"

IP de la machine :

```
enp0s3      UP      10.0.2.15/24 metric 100 fe80::a00:27ff:fe04:2a39/64
```

On ping :

```
ping -c 3 10.0.2.15
```

Résultat :

```
PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data.  
64 bytes from 10.0.2.15: icmp_seq=1 ttl=64 time=0.012 ms  
64 bytes from 10.0.2.15: icmp_seq=2 ttl=64 time=0.022 ms  
64 bytes from 10.0.2.15: icmp_seq=3 ttl=64 time=0.022 ms
```

```
--- 10.0.2.15 ping statistics ---
```

```
3 packets transmitted, 3 received, 0% packet loss, time 2031ms  
rtt min/avg/max/mdev = 0.012/0.018/0.022/0.004 ms
```

- Vérifiez le statut du serveur ssh.

```
systemctl status ssh
```

```
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2024-11-13 12:25:06 UTC; 13min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
   Main PID: 704 (sshd)
     Tasks: 1 (limit: 2219)
    Memory: 6.7M
       CPU: 47ms
    CGroup: /system.slice/ssh.service
           └─704 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"
```

```
nov. 13 12:25:05 vmdocker systemd[1]: Starting OpenBSD Secure Shell server...
nov. 13 12:25:06 vmdocker sshd[704]: Server listening on 0.0.0.0 port 22.
nov. 13 12:25:06 vmdocker sshd[704]: Server listening on :: port 22.
nov. 13 12:25:06 vmdocker systemd[1]: Started OpenBSD Secure Shell server.
nov. 13 12:27:14 vmdocker sshd[971]: Accepted password for dockeruser from 10.0.2.2 port
54168 ssh2
nov. 13 12:27:14 vmdocker sshd[971]: pam_unix(sshd:session): session opened for user
dockeruser(uid=1000) by (uid=0)
```

- Ouvrez un terminal sur la machine hôte et établissez une connexion en ssh pour l'utilisateur

```
ssh dockeruser@10.0.2.15 -p 2222
```

- Installez l'outil jq, processeur de json en ligne de commande.

```
sudo apt update
sudo apt install -y jq
```

Partie 2 – Installation et démarrage de Docker

- Utilisez une commande docker pour afficher la version (client et serveur !)

```
sudo docker info
```

(la commande affiche beaucoup donc je n'ai pas mis le résultat, mais il y a bien l'affichage du client et du serveur)

- Comment voir les composants du daemon Docker qui tournent ?

```
systemctl status docker
```

résultat :

```
root      9938  0.0  3.6 1912912 72892 ?        Ssl 12:53   0:00 /usr/bin/dockerd -H fd://  
--containerd=/run/containerd/containerd.sock  
dockeru+ 10279  0.0  0.1  6480  2276 pts/0    S+   12:56   0:00 grep --color=auto dockerd
```

- Quels sont les services/socket utilisés par docker ? Quel utilisateur a démarré ces services ?

services : `systemctl list-units --type=service --all | grep docker`

socket : `systemctl list-units --type=socket --all | grep docker`

utilisateur : `systemctl status docker`

- Que faire pour arrêter docker ? Quel est le statut du socket ?

Arrêter Docker :

```
sudo systemctl stop docker
```

Vérifier le statut du socket Docker :

```
systemctl status docker.socket
```

- Que faire pour désactiver/réactiver le docker ?

Désactiver Docker :

```
sudo systemctl disable docker
```

Réactiver Docker :

```
sudo systemctl enable docker
```

- Vous constatez que les commandes docker s'utilisent en mode sudo. Pour permettre à l'utilisateur dockeruser d'exécuter les commandes docker sans passer par sudo, effectuez les étapes suivantes :

1. Créer un groupe nommé docker

```
sudo groupadd docker
```

2. Passer en root dans le répertoire home de l'utilisateur

`sudo -i`

3. Ajouter (append) l'utilisateur dockeruser au groupe docker

`sudo usermod -aG docker dockeruser`

Cela ajoute dockeruser au groupe docker sans supprimer ses autres appartenances à des groupes.

4. Redémarrer la machine virtuelle

`sudo reboot`

5. Réessayez d'afficher la version de Docker sans passer par **sudo**

`docker --version`

Docker version 27.3.1, build ce12230

Partie 3 – Premières manipulations de conteneurs et d'images

- Quel est le répertoire dans lequel Docker stocke ses objets.

dans `/var/lib/docker`

- Quelles sont les différentes catégories d'objets Docker qui peuvent être stockés ?

le répertoire docker stocke ses containers, ses images, ses volumes et les informations réseaux du docker dans des dossiers spécifiques :

`/var/lib/docker/containers`

`/var/lib/docker/images`

`/var/lib/docker/volumes`

`/var/lib/docker/network`

- Consultez le contenu du répertoire approprié qui contient les conteneurs : combien y en a-t-il pour l'instant ?

on utilise la commande : `sudo ls -l /var/lib/docker/containers`

sortie : total 0

on remarque qu'il y a 0 containers

- Utilisez une commande pour rechercher des images, essayez avec l'image hello-world

`docker search hello-world`

docker search hello-world

- Faites exécuter un conteneur qui correspond à l'image ayant le plus d'étoiles. Faites une copie d'écran qui enregistre les étapes réalisées par docker.

```
dockeruser@vmdocker:~$ docker search hello-world
NAME                DESCRIPTION                STARS    OFFICIAL
hello-world         Hello World! (an example of minimal Dockeriz... 2346     [OK]
rancher/hello-world This container image is no longer maintained... 6
pckteto/hello-world                                0
atlassian/hello-world                               0
rutum/hello-world      Image to test docker deployments. Has Apache... 90
dockercloud/hello-world Hello World!                20
trccheck/hello-world   Hello World web server in under 2.5 MB         25
koudaiiii/hello-world                                0
tsepotesting123/hello-world                         0
ppc64le/hello-world   Hello World! (an example of minimal Dockeriz... 2
infrastructureascode/hello-world A tiny "Hello World" web server with a healt... 1
kevindockercompany/hello-world                       0
prajwalendra/hello-world                             0
twistlocktest/hello-world                             0
datawire/hello-world   Hello World! Simple Hello World implementati... 1
arm32v7/hello-world    Hello World! (an example of minimal Dockeriz... 3
uniplaces/hello-world                                0
arm64v8/hello-world    Hello World! (an example of minimal Dockeriz... 3
vjimenez5271/hello-world                             0
abadger/hello-world    0
danfengliu/hello-world 0
ansibleplaybookbundle/hello-world Simple containerized application that tests ... 0
jensendw/hello-world   0
kousik93/hello-world   0
silver8642/hello-world 0
dockeruser@vmdocker:~$ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
Digest: sha256:d211f485f2dd1dee407a80973c8f129f00d54604d2c90732e8e320e5038a0348
Status: Image is up to date for hello-world:latest
docker.io/library/hello-world:latest
dockeruser@vmdocker:~$ 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.
   (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/
```

- Vérifiez maintenant le contenu du répertoire des conteneurs. Combien y a-t-il de conteneurs ?

docker ps -a

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
6e40176f277c	hello-world	"/hello"	2 minutes ago	Exited (0)	2 minutes ago

intelligent_gauss

docker ps -a -q | wc -l pour seulement compter le nombre de docker présent

on a seulement un seul docker au total

- Vérifiez aussi le contenu du répertoire des images

docker images

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

- Quel est le sha256 de l'image ?

docker images --digests

REPOSITORY	TAG	DIGEST	IMAGE ID	CREATED	SIZE
hello-world	latest	sha256:d211f485f2dd1dee407a80973c8f129f00d54604d2c90732e8e320e5038a0348	d2c94e258dcb	18 months ago	13.3kB

- Utilisez la commande docker (sans option) qui permet de lister les conteneurs en exécution.

docker ps

- Combien de conteneurs s'exécutent ?

Après vérification avec docker ps -q | wc -l aucun docker ne s'exécute.

- Quel est son alias ?

docker ps -a

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
6e40176f277c	hello-world	"/hello"	13 minutes ago	Exited (0)	13 minutes ago

intelligent_gauss

on se réfère à NAMES et on trouve l'alias intelligent_gauss

- Utilisez la commande docker qui permet de lister les images. Quel est l'identifiant de l'image utilisée ?

docker images

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

l'identifiant est d2c94e258dcb

- Réexécutez le conteneur et comparez à la précédente exécution : expliquez.

docker ps -a

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
eeb2f83c2c29	hello-world	"/hello"	40 seconds ago	Exited (0)	40 seconds ago
elated_liskov					
6e40176f277c	hello-world	"/hello"	18 minutes ago	Exited (0)	18 minutes ago
intelligent_gauss					

On voit qu'un deuxième container s'est créé en listant avec docker ps.
Il possède un alias différent qui est elated_liskov

Aussi il ne va pas télécharger l'image une seconde fois, il va directement se servir localement.

- Combien de conteneurs s'exécutent et combien sont stockés localement.

en utilisant docker ps on voit que aucun ne s'exécute
en utilisant docker ps -a il y a 2

- Essayez de supprimer l'image et expliquez. Ne la supprimez pas finalement.

docker rmi d2c94e258dcb

Error response from daemon: conflict: unable to delete d2c94e258dcb (must be forced) - image is being used by stopped container 6e40176f277c

Le container est actuellement utilisé, il ne peut le supprimer tant qu'il est en cours d'exécution.

- Utilisez une commande qui permet de lister tous les conteneurs et pas uniquement ceux en exécution et observez leur statut

docker ps -a

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
NAMES					

```

eeb2f83c2c29    hello-world    "/hello"    17 minutes ago    Exited (0) 16 minutes ago
elated_liskov
6e40176f277c    hello-world    "/hello"    35 minutes ago    Exited (0) 35 minutes ago
intelligent_gauss

```

On voit que les conteneurs ont été créés il y a plusieurs minutes et se sont arrêtés juste après leur exécution

- Quel est le nom de ces conteneurs ?

```

docker ps -a --format "{{.Names}}"
elated_liskov
intelligent_gauss

```

donc le conteneur eeb2f83c2c29 est elated_liskov et le conteneur 6e40176f277c est intelligent_gauss

- Utilisez une option pour afficher l'information de façon non tronquée.

```

docker ps -a --no-trunc

```

```

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
eeb2f83c2c29a07321d7c826364cf0709c88150e3b04eac095e3f754b37e46a1 hello-world
"/hello" 21 minutes ago Exited (0) 21 minutes ago elated_liskov
6e40176f277c6f81e5879d6bfca56dd8e318d0ce281695c9d8b878832b83202d hello-world
"/hello" 39 minutes ago Exited (0) 39 minutes ago intelligent_gauss

```

- Exécutez une nouvelle fois l'image, puis supprimez ce dernier conteneur en utilisant son nom.

```

docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
53c7f9a19eff hello-world "/hello" 5 seconds ago Exited (0) 4 seconds ago
nifty_brown
eeb2f83c2c29 hello-world "/hello" 23 minutes ago Exited (0) 23 minutes ago
elated_liskov
6e40176f277c hello-world "/hello" 41 minutes ago Exited (0) 41 minutes ago
intelligent_gauss

```

on va supprimer nifty_brown avec docker rm -f

```

docker rm -f mon_conteneur

```

```

docker ps -a

```


CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
eeb2f83c2c29	hello-world	"/hello"	24 minutes ago	Exited (0)	24 minutes ago	elated_liskov
6e40176f277c	hello-world	"/hello"	42 minutes ago	Exited (0)	42 minutes ago	intelligent_gauss

- Exécutez une nouvelle fois l'image en utilisant son sha256 en donnant un nom au nouveau conteneur, puis constatez

```
docker run -d --name hello-world
sha256:d2c94e258dcb3c5ac2798d32e1249e42ef01cba4841c2234249495f87264ac5a
7163cea5f91f284aca0d46e6cd04d8407a8d3cec34423d4b8fbb0a39ba270c53
```

- Faites ce qu'il faut pour supprimer l'image (sans forcer).

```
dockeruser@vmdocker:~$ docker ps -a --filter "ancestor=sha256:d2c94e258dcb3c5ac2798d32e1249e42ef01cba4841c2234249495f87264ac5a"
CONTAINER ID   IMAGE          COMMAND        CREATED        STATUS        PORTS        NAMES
7163cea5f91f   d2c94e258dcb   "/hello"       4 minutes ago   Exited (0)    4 minutes ago   hello-world
eeb2f83c2c29   hello-world    "/hello"       45 minutes ago   Exited (0)    45 minutes ago   elated_liskov
6e40176f277c   hello-world    "/hello"       About an hour ago   Exited (0)    About an hour ago   intelligent_gauss
dockeruser@vmdocker:~$ docker rmi sha256:d2c94e258dcb3c5ac2798d32e1249e42ef01cba4841c2234249495f87264ac5a
Error response from daemon: conflict: unable to delete d2c94e258dcb (must be forced) - image is being used by stopped container 6e40176f277c
dockeruser@vmdocker:~$ docker rm 6e40176f277c
6e40176f277c
dockeruser@vmdocker:~$ docker ps -a --filter "ancestor=sha256:d2c94e258dcb3c5ac2798d32e1249e42ef01cba4841c2234249495f87264ac5a"
CONTAINER ID   IMAGE          COMMAND        CREATED        STATUS        PORTS        NAMES
7163cea5f91f   d2c94e258dcb   "/hello"       5 minutes ago   Exited (0)    5 minutes ago   hello-world
eeb2f83c2c29   hello-world    "/hello"       46 minutes ago   Exited (0)    46 minutes ago   elated_liskov
dockeruser@vmdocker:~$ docker rmi sha256:d2c94e258dcb3c5ac2798d32e1249e42ef01cba4841c2234249495f87264ac5a
Error response from daemon: conflict: unable to delete d2c94e258dcb (must be forced) - image is being used by stopped container eeb2f83c2c29
dockeruser@vmdocker:~$ docker ps -a --filter "ancestor=sha256:d2c94e258dcb3c5ac2798d32e1249e42ef01cba4841c2234249495f87264ac5a"
CONTAINER ID   IMAGE          COMMAND        CREATED        STATUS        PORTS        NAMES
7163cea5f91f   d2c94e258dcb   "/hello"       5 minutes ago   Exited (0)    5 minutes ago   hello-world
eeb2f83c2c29   hello-world    "/hello"       46 minutes ago   Exited (0)    46 minutes ago   elated_liskov
dockeruser@vmdocker:~$ docker rm 7163cea5f91f
7163cea5f91f
dockeruser@vmdocker:~$ docker rmi sha256:d2c94e258dcb3c5ac2798d32e1249e42ef01cba4841c2234249495f87264ac5a
Error response from daemon: conflict: unable to delete d2c94e258dcb (must be forced) - image is being used by stopped container eeb2f83c2c29
dockeruser@vmdocker:~$ docker ps -a --filter "ancestor=sha256:d2c94e258dcb3c5ac2798d32e1249e42ef01cba4841c2234249495f87264ac5a"
CONTAINER ID   IMAGE          COMMAND        CREATED        STATUS        PORTS        NAMES
eeb2f83c2c29   hello-world    "/hello"       46 minutes ago   Exited (0)    46 minutes ago   elated_liskov
dockeruser@vmdocker:~$ docker rm eeb2f83c2c29
eeb2f83c2c29
dockeruser@vmdocker:~$ docker rmi sha256:d2c94e258dcb3c5ac2798d32e1249e42ef01cba4841c2234249495f87264ac5a
Untagged: hello-world:latest
Untagged: hello-world@sha256:d211f485f2dd1dee407a80973c8f129f00d54604d2c90732e8e320e5038a0348
Deleted: sha256:d2c94e258dcb3c5ac2798d32e1249e42ef01cba4841c2234249495f87264ac5a
Deleted: sha256:ac2880ec8bb38d5c35b49d45a6ac4777544941199075dff8c4eb63e093aa81e
dockeruser@vmdocker:~$
```

Il a fallu supprimer les conteneurs utilisant l'image pour pouvoir supprimer l'image avec le sha256

- Utilisez une commande qui donne des infos globales sur le système et observez le nombre de conteneurs et d'images

```
docker system df
TYPE          TOTAL    ACTIVE    SIZE    RECLAIMABLE
Images        0        0         0B       0B
```

Containers	0	0	0B	0B
Local Volumes	0	0	0B	0B
Build Cache	0	0	0B	0B

Avec la suppression précédente il n'y a plus de conteneurs et d'images.

- Quelle commande (et options) docker pouvez-vous utiliser pour n'afficher que les IDs des conteneurs ?

on peut utiliser : `docker ps -q`

- Utilisez une possibilité du bash pour exploiter les résultats de cette dernière commande afin de supprimer tous les conteneurs en une seule commande.

```
if [ "$(docker ps -aq)" ]; then
  docker rm $(docker ps -aq)
else
  echo "Aucun conteneur à supprimer."
fi
```

On supprime s'il y a des conteneurs présents sinon on ne fait rien.

- Supprimez maintenant l'image et constatez.

Si l'image est utilisée alors une erreur surviendra
sinon la suppression s'effectue sans problème

- Faites à nouveau exécuter un conteneur, que vous nommerez hello1, pour la même image hello-world, que se passe-t-il ?

```
dockeruser@vmdocker:~$ docker run --name hello1 hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:305243c734571da2d100c8c8b3c3167a098cab6049c9a5b066b6021a60fcb966
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
```

l'image hello world n'existant plus, le docker va installer la version la plus récente de la librairie.

- Quel est l'identifiant de l'image ? Constatez.

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hello-world	latest	d2c94e258dcb	19 months ago	13.3kB

le conteneur hello1 a à l'image d2c94e258dcb ce qui correspond à hello-world.

- Créez un conteneur, que vous nommerez hello2, mais sans le démarrer, puis observez son statut.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
5ad63915f36b	hello-world	"/hello"	About a minute ago	Created		hello2

Le statut du conteneur est "CREATED" pour signifier qu'il a juste été créé mais pas démarré.

- Démarrez-le ensuite en utilisant son nom et constatez son statut ensuite. Quel est l'affichage ?

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

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
5ad63915f36b	hello-world	"/hello"	11 minutes ago	Exited (0)	6 seconds ago	hello2
e027f181089c	hello-world	"/hello"	42 minutes ago	Exited (0)	41 minutes ago	hello1

On remarque qu'après avoir inséré la commande, le conteneur hello2 a été exécuté.

- Utilisez maintenant l'option qui permettra d'attacher l'entrée et la sortie standard pour démarrer ce conteneur hello2.

on utilisera l'option attach qui est -a

```
docker start -a hello2
```

- Utilisez la même option pour démarrer le conteneur hello1.

```
docker start -a hello1
```

- Exécutez maintenant un nouveau conteneur nommé hello3, mais en faisant en sorte qu'il n'affiche pas d'information sur la sortie standard (en background donc). Constatez dans la liste (totale) des conteneurs.

```
docker run -d --name hello3 hello-world
```

Exécutez maintenant un nouveau conteneur nommé hello4, mais en faisant en sorte que ce conteneur ait totalement disparu après son exécution. Constatez dans la liste (totale) des conteneurs.

```
docker run --rm --name hello4 hello-world
```

on affiche avec docker ps -a

```
docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
PORTS	NAMES			
2ed17fb4587e	hello-world	"/hello"	About a minute ago	Exited (0) About a minute ago
hello3				
5ad63915f36b	hello-world	"/hello"	19 minutes ago	Exited (0) 5 minutes ago
hello2				
e027f181089c	hello-world	"/hello"	50 minutes ago	Exited (0) About a minute ago
hello1				

on remarque que hello4 s'est bien supprimé après exécution

- Vous pouvez supprimer l'image hello-world avant de passer à la partie suivante.

```
dockeruser@vmdocker:~$ docker rm hello1 hello2 hello3
hello1
hello2
hello3
```

```
dockeruser@vmdocker:~$ docker rmi hello-world
Untagged: hello-world:latest
Untagged:
hello-world@sha256:305243c734571da2d100c8c8b3c3167a098cab6049c9a5b066b6021a60fcb966
Deleted:
sha256:d2c94e258dcb3c5ac2798d32e1249e42ef01cba4841c2234249495f87264ac5a
Deleted:
sha256:ac28800ec8bb38d5c35b49d45a6ac4777544941199075dff8c4eb63e093aa81e
dockeruser@vmdocker:~$
```

Partie 4 – DockerHub

- Combien de différents types d'images trouve-t-on sur ce registre public, comment se fait la classification ?

On trouve 3 types d'images, les images officielles, les images vérifiées et les sponsored.

La classification s'effectue de manière à catégoriser les images par rapport au domaine recherché, l'OS utilisé (ubuntu/windows) et le type d'architecture de la machine.

- Retrouvez l'image officielle de l'OS Ubuntu.

En recherchant Ubuntu et en cochant la case docker Official Image on trouve l'image officielle de ubuntu.

- Consultez les différentes versions proposées, comment les distingue-t-on ?

- [20.04.focal-20241011.focal](#)
- [22.04.jammy-20240911.1.jammy](#)
- [24.04.noble-20241118.1.noble.latest](#)
- [24.10.oracular-20241120.oracular.rolling](#)
- [25.04.plucky-20241124.plucky.devel](#)

- Quelle est la version la plus récente ? Quelle différence a-t-elle avec la version latest ? Quels sont leurs identifiants respectifs ?

La version la plus récente est la version 25.04

La version latest est la version la plus stable actuelle ce qui assure qu'on utilise une version possédant le moins d'erreurs pouvant être provoquée durant leurs utilisations.



la version latest possède 2 medium errors et 5 low errors

Les identifiants sont les noms à côté de la version :

- [focal-20241011.focal](#)
- [jammy-20240911.1.jammy](#)
- [noble-20241118.1.noble.latest](#)
- [oracular-20241120.oracular.rolling](#)
- [plucky-20241124.plucky.devel](#)

- Quelles sont les vulnérabilités de la version la plus récente ? Et celles de la version latest ?

Aucune vulnérabilité a été détecté dans la version la plus récente :

TAG			
25.04			
Last pushed 6 days ago by dojanky			
docker pull ubuntu:25.04 Copy			
Digest	OS/ARCH	 Vulnerabilities	Compressed size 
82386eb29995	linux/amd64	None found	29.77 MB
0af5fcfe1770	linux/arm/v7	None found	26.31 MB
465c3e2c5136	linux/arm64/v8	None found	28.96 MB
732c21ecc586	linux/ppc64le	None found	33.52 MB
804d0a3f1d53	linux/riscv64	None found	30.36 MB
235b0ccb9cef	linux/s390x	None found	29.43 MB



- Observez comment ces vulnérabilités sont classées.

Elles sont classés en fonction de l'architecture

- Comparez la version latest avec la version noble : quel pourrait être l'intérêt de cette situation ?

La version noble possède plusieurs vulnérabilités dans chacune de ses couches

- En cliquant sur le tag de chacune des 2, consultez les couches de ces images, et observez à partir de quand elles diffèrent.

Les deux fichiers diffèrent de la version ADD file car la version noble a une erreur sur la couche add file.

ubuntu:25.04

TAG			
25.04			
Last pushed 6 days ago by dojanky			
		docker pull ubuntu:25.04 Copy	
Digest	OS/ARCH	Vulnerabilities	Compressed size
82386eb29995	linux/amd64	None found	29.77 MB
0af5fcfe1770	linux/arm/v7	None found	26.31 MB
465c3e2c5136	linux/arm64/v8	None found	28.96 MB
732c21ecc586	linux/ppc64le	None found	33.52 MB
804d0a3f1d53	linux/riscv64	None found	30.36 MB
235b0ccb9ccf	linux/s390x	None found	29.43 MB

ubuntu:24.04

Image hierarchy

ALL		ubuntu:24.04	
-----	--	--------------	--

Layers (6)			
0	ARG RELEASE	0 B	
1	ARG LAUNCHPAD_BUILD_ARCH	0 B	
2	LABEL org.opencontainers.image.ref.name=ubuntu	0 B	
3	LABEL org.opencontainers.image.version=24.04	0 B	
4	ADD file:765dfd09ec2ac4870c8b3efd6ef4a994f99695c574d546d7a9a0e69bbb970b03 in /	28.89 MB	
5	CMD ["/bin/bash"]	0 B	

- Placez-vous dans le répertoire `/var/lib/docker/image/overlay2/layerdb/sha256` et listez son contenu vide.

Le répertoire étant vide, une sortie vide est retournée précisant que rien n'est contenu dans le répertoire.

- Téléchargez l'image ubuntu : par défaut, quelle est celle qui est téléchargée

```
dockeruser@vmdocker:~$ docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
de44b265507a: Pull complete
Digest:
sha256:80dd3c3b9c6cecb9f1667e9290b3bc61b78c2678c02cbdae5f0fea92cc6734ab
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```

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

La version latest a été téléchargée par défaut, ce qui n'est pas surprenant vu que c'est la version la plus stable.

- Utilisez un filtre pour n'afficher que les images référençant ubuntu.

```
dockeruser@vmdocker:~$ docker images --filter=reference='ubuntu*'
```

```
REPOSITORY TAG IMAGE ID CREATED SIZE
ubuntu latest b1d9df8ab815 2 weeks ago 78.1MB
```

- Identifiez la commande qui est lancée lorsqu'on exécute un conteneur à partir de l'image ubuntu.

```
dockeruser@vmdocker:~$ docker run -it ubuntu
root@18f2ec4ee7ca:/# ls
bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr
var
```

La commande `docker run` permet de lancer le conteneur et de simuler ubuntu.

Partie 5 – Interagir avec un conteneur

- Lancez un conteneur à partir de l'image ubuntu et constatez son statut. Essayez de redémarrer ce même conteneur en interactif. Enfin supprimez-le.

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

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
1e4e03b8601e	ubuntu	"/bin/bash"	13 seconds ago	Exited (0)	12 seconds ago

distracted_hypatia

```
dockeruser@vmdocker:~$ docker start -i 1e4e03b8601e
root@1e4e03b8601e:/#
exit
```

```
dockeruser@vmdocker:~$ docker rm 1e4e03b8601e
1e4e03b8601e
dockeruser@vmdocker:~$
```

- Lancez, à partir de l'image ubuntu, un conteneur nommé os_ubuntu en interactif et attaché à un terminal.

```
dockeruser@vmdocker:~$ docker run -it --name os_ubuntu ubuntu
root@aa553e0d6672:/#
exit
```

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

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
aa553e0d6672	ubuntu	"/bin/bash"	17 seconds ago	Exited (0)	5 seconds ago

os_ubuntu

- Quelle est la commande qui correspond au processus de PID 1 de ce conteneur ?

Lorsqu'on exécute la commande `docker run -it ubuntu`, le processus `bash` est lancé par défaut comme processus PID 1 dans le conteneur

```
root@aa553e0d6672:/# ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.1	4588	3780	pts/0	Ss	12:01	0:00	/bin/bash

- Dans cet OS, exécutez les commandes `whoami`, `pwd`, `ls` et `hostname`. Notez son ID.

```
root@aa553e0d6672:/# whoami
root
```

```
root@aa553e0d6672:/# pwd
/
```



```
root@aa553e0d6672:/# ls
bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr
var
```

```
root@aa553e0d6672:/# hostname
aa553e0d6672
```

- Ouvrez un deuxième terminal, connectez-vous en ssh sur la VM et observez le statut du conteneur en cours d'exécution. Que constatez-vous (à part le statut) ?

```
dockeruser@vmdocker:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS
NAMES
aa553e0d6672   ubuntu   "/bin/bash"             About an hour ago Up 9 seconds     os_ubuntu
```

On constate que le conteneur est toujours en cours d'utilisation depuis 9 secondes. Son statut changera en fonction de son état.

- Revenez dans le conteneur sur le premier terminal. Déplacez-vous dans le répertoire home.

```
cd home
```

- Quittez ce conteneur avec la commande unix classique puis observez le statut du conteneur.

```
exit
```

```
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS
NAMES
aa553e0d6672   ubuntu   "/bin/bash"             About an hour ago Exited (0) 5 seconds ago
os_ubuntu
```

Le conteneur n'est plus en exécution.

- Démarrez ce conteneur en interactif. Dans quel répertoire vous trouvez-vous ? Pourquoi ?

Après le démarrage en interactif on arrive dans le répertoire racine /
c'est le comportement standard car c'est le répertoire de travail initial lorsque l'on utilise un shell comme bash dans une image de base Ubuntu

- Dans le second terminal, utilisez une commande qui permet d'inspecter le conteneur. Constatez qu'il est en cours d'exécution. Retrouvez son Pid.

```
dockeruser@vmdocker:~$ docker inspect os_ubuntu
[
```

```
{
  "Id": "aa553e0d6672187fa668173f4c6c961d754a5757c9718fc1259cc0476d9b1024",
  "Created": "2024-12-09T11:04:40.042714889Z",
  "Path": "/bin/bash",
  "Args": [],
  "State": {
    "Status": "running",
    "Running": true,
    "Paused": false,
    "Restarting": false,
    "OOMKilled": false,
    "Dead": false,
    "Pid": 2134,
    "ExitCode": 0,
    "Error": "",
    "StartedAt": "2024-12-09T12:13:30.191226745Z",
    "FinishedAt": "2024-12-09T12:11:35.178571223Z"
  },
}
```

On voit que le conteneur est en route avec le flag running true et que le PID équivaut à 2134

Dans ce même terminal, retrouvez le processus dont le PID est celui que vous venez d'identifier et constatez.

```
ps -p 2134 -l
F S  UID    PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY      TIME CMD
4 S   0    2134   2113  0  80   0 - 1147 -   pts/0  00:00:00 bash
```

Le processus 2134 est le shell bash exécuté dans le conteneur Ubuntu, gérant les commandes interactives avec le conteneur.

- Revenez dans le conteneur (dans le premier terminal) et déplacez-vous à nouveau dans home.

```
dockeruser@vmdocker:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS
NAMES
aa553e0d6672   ubuntu   "/bin/bash"             About an hour ago   Up 16 minutes
os_ubuntu
```

Le conteneur est toujours exécuté.

```
dockeruser@vmdocker:~$ docker attach os_ubuntu
```

```
root@aa553e0d6672:/home#
```

- Attachez le terminal au conteneur. Quel est le répertoire courant ? Ajoutez-y un fichier avec un contenu quelconque.

Le répertoire courant est le home

```
root@aa553e0d6672:/home# echo "fichier test" > fichier_test.txt
root@aa553e0d6672:/home# ls
fichier_test.txt  ubuntu
```

- Quittez à nouveau ce conteneur sans l'arrêter, puis utilisez une commande docker pour voir les différences dans le système de fichiers du conteneur.

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

- Utilisez une commande docker pour exécuter la commande Unix hostname dans ce conteneur (sans le passer en foreground).

```
dockeruser@vmdocker:~$ docker exec -it os_ubuntu ls
bin  dev  home  lib64  mnt  proc  run  srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
```

- Utilisez une commande docker pour exécuter la commande Unix bash en interactif dans ce conteneur. Vérifiez le nombre de processus bash qui tournent dans le conteneur.

```
dockeruser@vmdocker:~$ docker exec -it os_ubuntu bash
```

```
root@aa553e0d6672:/# ps aux | grep bash
root      1  0.0  0.1  4588  3732 pts/0  Ss+  12:56   0:00 /bin/bash
root     17  0.0  0.1  4588  3828 pts/1   Ss   12:59   0:00 bash
root     26  0.0  0.0  3528  1616 pts/1   S+   12:59   0:00 grep --color=auto bash
```

- Quittez à nouveau ce conteneur sans l'arrêter, puis utilisez une commande docker qui affiche les processus du conteneur.

```
dockeruser@vmdocker:~$ docker top os_ubuntu
UID          PID         PPID        C    STIME  TTY   TIME      CMD
root         1531        1511        0    12:56  pts/0  00:00:00  /bin/bash
```

```
root          1609          1511      0      12:59      ?      00:00:00      bash
```

- Revenez dans le conteneur et arrêtez-le en le quittant.

```
dockeruser@vmdocker:~$ docker attach os_ubuntu
root@aa553e0d6672:/#
exit
```

- Lancez maintenant un nouveau conteneur nommé ll à partir de l'image ubuntu dont la commande est maintenant ls -l.

```
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 13:08 dev
drwxr-xr-x  1 root root 4096 Dec  9 13:08 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 187 root root   0 Dec  9 13:08 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 13:08 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
```

- Redémarrez ce conteneur pour obtenir le même affichage.

```
dockeruser@vmdocker:~$ docker run -it --name ll ubuntu
root@0b85442d91dc:/# dockeruser@vmdocker:~$ docker start ll
ll
dockeruser@vmdocker:~$ docker attach ll
root@0b85442d91dc:/#
```

- Lancez un nouveau conteneur nommé ps avec la commande ps aux, mais en faisant en sorte que ce conteneur disparaisse après son exécution. Constatez le PID et constatez le statut.

```
dockeruser@vmdocker:~$ docker run --rm --name ps ubuntu ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	15.3	0.1	7888	3584	?	Rs	13:36	0:00	ps aux

On a le PID 1, cela signifie que le processus était le premier et unique à s'être exécuté dans ce conteneur et il s'est terminé immédiatement après avoir affiché les informations

Le statut du conteneur est Exited car le conteneur a été supprimé après l'exécution de la commande grâce à l'option `--rm`.

- Lancez un nouveau conteneur nommé salut avec la commande `echo Bonjour`, puis relancez ce conteneur.

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

```
dockeruser@vmdocker:~$ docker run --rm -d --name infinite-container ubuntu sh -c "while true; do sleep 3600; done"
f132ea114778236aee2e48a9f61a32de17ddb9cb237dcf03128bf13ffeba3e19
dockeruser@vmdocker:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
f132ea114778	ubuntu	"sh -c 'while true; ...'"	21 seconds ago	Up 20 seconds		infinite-container
e266217fa9cc	ubuntu	"echo Bonjour"	10 minutes ago	Exited (0) 10 minutes ago		salut
0b85442d91dc	ubuntu	"/bin/bash"	15 minutes ago	Exited (0) 12 minutes ago		ll
5419fae20662	ubuntu	"bash"	29 minutes ago	Exited (137) 22 minutes ago		gallant_stonebraker
aa553e0d6672	ubuntu	"/bin/bash"	3 hours ago	Exited (130) 9 minutes ago		os_ubuntu

```
dockeruser@vmdocker:~$ docker stop infinite-container
infinite-container
dockeruser@vmdocker:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
e266217fa9cc	ubuntu	"echo Bonjour"	11 minutes ago	Exited (0) 11 minutes ago		salut
0b85442d91dc	ubuntu	"/bin/bash"	16 minutes ago	Exited (0) 13 minutes ago		ll
5419fae20662	ubuntu	"bash"	30 minutes ago	Exited (137) 23 minutes ago		gallant_stonebraker
aa553e0d6672	ubuntu	"/bin/bash"	3 hours ago	Exited (130) 9 minutes ago		os_ubuntu

- Démarrez le conteneur `os_ubuntu` en interactif. Dans ce shell, lancez une commande qui affiche salut toutes les 3 secondes.

```
root@aa553e0d6672:/# while true; do echo salut; sleep 3; done
salut
salut
salut
salut
salut
salut
salut
salut
salut
salut
salut
```

- Dans le deuxième terminal connecté en ssh à la VM, utilisez une commande `docker` pour vous attacher au conteneur qui tourne dans le premier terminal puis

constatez. Interrompez le processus qui effectue le salut, lancez une commande basique (par exemple ls) et constatez dans l'autre terminal.

premier terminal :

```
root@aa553e0d6672:/# while true; do echo salut; sleep 3; done
salut
salut
salut
salut
salut
salut
salut
salut
salut
salut
salut
^C
root@aa553e0d6672:/# ls
bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr v
ar
root@aa553e0d6672:/# _
```

second terminal :

```
dockeruser@vmdocker:~$ docker attach os_ubuntu
salut
salut
salut
salut
salut
^C
root@aa553e0d6672:/# ls
bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
root@aa553e0d6672:/#
```

La commande ls a été exécutée dans le second terminal, vu qu'ils sont relié l'exécution d'un terminal s'effectuera dans l'autre.

- Quittez ce conteneur en utilisant la commande bash exit avec un code retour non nul. Constatez le statut de ce conteneur.

La session a été arrêtée dans les deux terminaux.

- Utilisez une commande docker pour voir le log du conteneur os_ubuntu.

Toutes les actions faites dans os_ubuntu a été affichées lors de l'exécution de la commande.

- Inspectez le conteneur à la recherche du fichier contenant son journal (log). Vous pouvez le consulter avec jq.

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

```
"/var/lib/docker/containers/aa553e0d6672187fa668173f4c6c961d754a5757c9718fc1259cc0476d9b1024/aa553e0d6672187fa668173f4c6c961d754a5757c9718fc1259cc0476d9b1024-j  
son.log"
```

- Utilisez une commande docker pour supprimer tous les conteneurs arrêtés.

```
dockeruser@vmdocker:~$ docker container prune  
WARNING! This will remove all stopped containers.  
Are you sure you want to continue? [y/N] y  
Deleted Containers:  
e266217fa9cc073a6f0760fe5e57c2308fea8d78bfbccacedfe85c1264c9efb0  
0b85442d91dc17eec7db1fc8f8f6f3cf07086dd3d82204bdb217b4726a9567ff  
5419fae20662746a7f6284f40b7cc625252df6d258cb96485090a277e800c71a  
aa553e0d6672187fa668173f4c6c961d754a5757c9718fc1259cc0476d9b1024
```

```
Total reclaimed space: 365B  
dockeruser@vmdocker:~$
```

Partie 6 – Inspection et manipulation d'images

- Combien de couches ont été téléchargées ?

7 couches ont été téléchargées :

1. `fdf894e782a2`
2. `5bd71677db44`
3. `551df7f94f9c`
4. `ce82e98d553d`
5. `2620fd693e71`
6. `3b175598111c`
7. `03d057de1320`

- Observez les couches de cette image (il existe aussi une option non tronquée)

```
dockeruser@vmdocker:~$ docker image inspect python:3.9 --format='{{json  
.RootFS.Layers}}' | jq  
[  
  "sha256:301c1bb42cc0bc6618fcf036e8711f2aad66f76697f541e2014a69e1f456aa4",  
  "sha256:0e82d78b3ea1b1db9fa3e1f18d6745e0c2380c25f2c7cec420257084e9cc44fe",  
  "sha256:c81d4fdb67fcfd8ffbf9f93f440264d36a2d9e7c4e79b9ae5152c5ed2e3fd36",  
  "sha256:0aeceb7c293df4fb677b2771713e9c6abeabf8b7f06bfb071310e6cc1a3aa084",  
  "sha256:8f9a13bfb118975875edd547c5c0762eed442b686d86fa46832bf04337f75316",  
  "sha256:24f0c2413cd7a5e1e06bbb497657405c0d81b86142567b8425dea83b3a1d635d",  
  "sha256:fe5bbd4f8a4224acb21f695f361216e88e2db8bc531064ae2d482635d5f357ae"  
]
```

- Quelle est la dernière couche ?

C'est la dernière ligne donc :

```
"sha256:fe5bbd4f8a4224acb21f695f361216e88e2db8bc531064ae2d482635d5f357ae"
```

- Puis téléchargez le python : 3.14.0a1. Combien de couches ont été téléchargées ? Expliquez.

7 couches ont été téléchargées

```
dockeruser@vmdocker:~$ docker pull python:3.14.0a1
3.14.0a1: Pulling from library/python
b2b31b28ee3c: Pull complete
c3cc7b6f0473: Pull complete
2112e5e7c3ff: Pull complete
af247aac0764: Pull complete
46bd469f680f: Pull complete
db2f885547b6: Pull complete
33fcfb54d100: Pull complete
Digest:
sha256:3a852c145c357b55d0fdbf624207bb81c5a546f17f622e5c208cc0b36edaaa0e
Status: Downloaded newer image for python:3.14.0a1
docker.io/library/python:3.14.0a1
```

- Utilisez l'inspection d'une image et l'outil jq pour afficher les couches des 2 images python, observez les couches communes.

```
docker image inspect python:3.9 | jq '[0].RootFS.Layers'
docker image inspect python:3.14.0a1 | jq '[0].RootFS.Layers'
[
  "sha256:301c1bb42cc0bc6618fcf036e8711f2aad66f76697f541e2014a69e1f456aa4",
  "sha256:0e82d78b3ea1b1db9fa3e1f18d6745e0c2380c25f2c7cec420257084e9cc44fe",
  "sha256:c81d4fdb67fcfd8ffbe9f93f440264d36a2d9e7c4e79b9ae5152c5ed2e3fd36",
  "sha256:0aeceb7c293df4fb677b2771713e9c6abeabf8b7f06bfb071310e6cc1a3aa084",
  "sha256:8f9a13bfb118975875edd547c5c0762eed442b686d86fa46832bf04337f75316",
  "sha256:24f0c2413cd7a5e1e06bbb497657405c0d81b86142567b8425dea83b3a1d635d",
  "sha256:fe5bbd4f8a4224acb21f695f361216e88e2db8bc531064ae2d482635d5f357ae"
]
[
  "sha256:24b5ce0f1e07d37a35460f50d058afc738619e431013d2e1727609bdff2d7fc",
```



```

"sha256:b6ca42156b9f492afa27c366f20e4e864cef8dd8d0e0a100497764b05b39e6fc",
"sha256:00547dd240c419fa2e1b33e66aba302e8dfa4bfe6401a972d94a03b1355cbc6c",
"sha256:96d99c63b722657062d3f33cc230e33b191ea9855c050f44871e173709597e35",
"sha256:9744b636d758d56bfeceb5e712ddfecbe662951562155cc3f93af8cfd538422c",
"sha256:4068925b787de0e570d68e70bc04de2380276817a36a343c08aad3435c221113",
"sha256:da64f9c6a005a83af29c8389262e6de4bb9b1b5ae96ceb6c567e01e7c7525cde"
]

```

Il n'y a pas de couches communes entre les deux versions

- Recréez un nouveau conteneur `os_ubuntu` à partir de l'image `ubuntu` et ajoutez dans le répertoire `home` un nouveau fichier, avec un contenu quelconque. Quittez-le sans l'arrêter. Utilisez une commande `docker` pour exporter le système de fichiers dans une archive `tar`.

```
dockeruser@vmdocker:~$ docker run -it --name os_ubuntu ubuntu
```

```
root@d3843fe28f75:/# cd home
```

```
root@d3843fe28f75:/home# echo "test fichier " > testfile.txt
```

```
root@d3843fe28f75:/homedocker export os_ubuntu > os_ubuntu_fs.tarubuntu >
os_ubuntu_fs.tar
```

```
dockeruser@vmdocker:~$
```

```
dockeruser@vmdocker:~$ tar -tf os_ubuntu_fs.tar | grep home/testfile.txt
home/testfile.txt
```

```
dockeruser@vmdocker:~$
```

- Créez une image `image_ubuntu_with_file` en important l'archive. Comment est-elle taggée ? Observez ses couches (son historique).

```
dockeruser@vmdocker:~$ docker import os_ubuntu_fs.tar image_ubuntu_file
sha256:718a579d86bbe52aa1d36ae311e47f69ee6b4a898f3594338cce22faf5ea16e5
```

```
dockeruser@vmdocker:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
image_ubuntu_file	latest	718a579d86bb	4 seconds ago	78.1MB
python	3.9	f327fe247a06	5 days ago	999MB
ubuntu	latest	b1d9df8ab815	2 weeks ago	78.1MB
python	3.14.0a1	80ad471000e7	7 weeks ago	1.02GB

```
dockeruser@vmdocker:~$
```

Par défaut, l'image est taguée avec un identifiant SHA256

historique :

```
dockeruser@vmdocker:~$ docker history image_ubuntu_file
IMAGE          CREATED          CREATED BY          SIZE      COMMENT
718a579d86bb   2 minutes ago              78.1MB   Imported from -
dockeruser@vmdocker:~$
```

- Créez un conteneur à partir de cette image. Si vous avez un bash, vous pouvez vérifier que le fichier créé précédemment existe.

```
dockeruser@vmdocker:~$ docker run -it image_ubuntu_file bash
root@f657be7607d4:/# cd home
root@f657be7607d4:/home# ls
testfile.txt ubuntu
root@f657be7607d4:/home#
```

fichier créé précédemment existe. - A partir du conteneur os_ubuntu, committez maintenant une image image2 et consultez ses couches. Utilisez maintenant cette image pour créer un conteneur.

```
dockeruser@vmdocker:~$ docker commit os_ubuntu image2
sha256:fdd4623033303e7ffffd23017efa56b8f3b6ea72d4d23b7ae8e8443834829d18d
dockeruser@vmdocker:~$ docker history image2
IMAGE          CREATED          CREATED BY          SIZE      COMMENT
fdd462303330   10 seconds ago   /bin/bash           14B
01d9df8ab815   2 weeks ago      /bin/sh -c #(nop)   0B
<missing>       2 weeks ago      /bin/sh -c #(nop)   78.1MB
<missing>       2 weeks ago      /bin/sh -c #(nop)   0B
<missing>       2 weeks ago      /bin/sh -c #(nop)   0B
<missing>       2 weeks ago      /bin/sh -c #(nop)   0B
<missing>       2 weeks ago      /bin/sh -c #(nop)   0B
dockeruser@vmdocker:~$ docker run -it image2 bash
root@c9162cccfc1e:/# cd home
root@c9162cccfc1e:/home# ls
testfile.txt ubuntu
root@c9162cccfc1e:/home#
```

Partie 7 – Quelques informations générales

- Utilisez une commande docker pour consulter la consommation des conteneurs en exécution.

```
dockeruser@vmdocker:~$ docker stats
```

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O
d3843fe28f75	os_ubuntu	0.00%	1.152MiB / 1.918GiB	0.06%	1.23kB / 0B
8.19kB	1				1.04MB /

- Utilisez une commande docker pour voir la consommation disque des différents objets docker. Essayez aussi la version détaillée.

```
dockeruser@vmdocker:~$ docker system df
```

TYPE	TOTAL	ACTIVE	SIZE	RECLAIMABLE
Images	5	3	2.175GB	2.096GB (96%)
Containers	3	1	36B	22B (61%)
Local Volumes	0	0	0B	0B
Build Cache	0	0	0B	0B