

TP 2 – Docker – Conteneurs et images

Partie 2 – Installation et démarrage de Docker

Lors de l'installation de docker j'ai rencontré le problème suivant qui empêche docker d'être installé :

l'horloge du système (date/heure) était en retard de plusieurs jours par rapport à la date réelle.

```
dockeruser@vmdocker:~$ sudo apt-get update
Atteint :1 http://fr.archive.ubuntu.com/ubuntu jammy InRelease
Récception de :2 https://download.docker.com/linux/ubuntu jammy InRelease [48,5 kB]
Récception de :3 http://fr.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Récception de :4 http://fr.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Récception de :5 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Lecture des listes de paquets... Fait
E: Le fichier « Release » pour https://download.docker.com/linux/ubuntu/dists/jammy/InRelease n'est pas encore valable (invalidé pendant encore 3d 21h 15min 40s). Les mises à jour depuis ce dépôt ne s'effectueront pas.
E: Le fichier « Release » pour http://fr.archive.ubuntu.com/ubuntu/dists/jammy-updates/InRelease n'est pas encore valable (invalidé pendant encore 4d 13h 56min 26s). Les mises à jour depuis ce dépôt ne s'effectueront pas.
E: Le fichier « Release » pour http://fr.archive.ubuntu.com/ubuntu/dists/jammy-backports/InRelease n'est pas encore valable (invalidé pendant encore 4d 14h 0min 16s). Les mises à jour depuis ce dépôt ne s'effectueront pas.
E: Le fichier « Release » pour http://security.ubuntu.com/ubuntu/dists/jammy-security/InRelease n'est pas encore valable (invalidé pendant encore 4d 13h 53min 26s). Les mises à jour depuis ce dépôt ne s'effectueront pas.
```

la solution pour ce problème était juste de mettre à jour la date de système pour qu'elle soit la date d'aujourd'hui.

- On utilise la commande docker version pour afficher la version

```

dockeruser@vmdocker:~$ sudo docker version
Client: Docker Engine - Community
  Version:           28.5.1
  API version:      1.51
  Go version:       go1.24.8
  Git commit:       e180ab8
  Built:            Wed Oct  8 12:17:03 2025
  OS/Arch:          linux/amd64
  Context:          default

Server: Docker Engine - Community
  Engine:
    Version:          28.5.1
    API version:     1.51 (minimum version 1.24)
    Go version:      go1.24.8
    Git commit:      f8215cc
    Built:           Wed Oct  8 12:17:03 2025
    OS/Arch:         linux/amd64
    Experimental:   false
  containerd:
    Version:          v1.7.28
    GitCommit:        b98a3aace656320842a23f4a392a33f46af97866
  runc:
    Version:          1.3.0
    GitCommit:        v1.3.0-0-g4ca628d1
  docker-init:
    Version:          0.19.0
    GitCommit:        de40ad0

```

- pour voir les composants du daemon Docker qui tournent on utilise la commande :
systemctl status docker

```

dockeruser@vmdocker:~$ systemctl status docker
● docker.service - Docker Application Container Engine
  Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
  Active: active (running) since Thu 2025-10-09 16:38:48 UTC; 4min 38s ago
  TriggeredBy: ● docker.socket
    Docs: https://docs.docker.com
   Main PID: 7375 (dockerd)
     Tasks: 9
    Memory: 20.3M
      CPU: 663ms
     CGroup: /system.slice/docker.service
             └─7375 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

oct. 09 16:38:45 vmdocker dockerd[7375]: time="2025-10-09T16:38:45.704775314Z" level=info msg="detected 127.0.0.53 nameserver, >
oct. 09 16:38:45 vmdocker dockerd[7375]: time="2025-10-09T16:38:45.771206513Z" level=info msg="Creating a containerd client" ad>
oct. 09 16:38:46 vmdocker dockerd[7375]: time="2025-10-09T16:38:46.601113242Z" level=info msg="Loading containers: start.">
oct. 09 16:38:47 vmdocker dockerd[7375]: time="2025-10-09T16:38:47.838704316Z" level=info msg="Loading containers: done.">
oct. 09 16:38:48 vmdocker dockerd[7375]: time="2025-10-09T16:38:48.233437207Z" level=info msg="Docker daemon" commit=f8215cc co>
oct. 09 16:38:48 vmdocker dockerd[7375]: time="2025-10-09T16:38:48.233752379Z" level=info msg="Initializing buildkit">
oct. 09 16:38:48 vmdocker dockerd[7375]: time="2025-10-09T16:38:48.350276545Z" level=info msg="Completed buildkit initializat>
oct. 09 16:38:48 vmdocker dockerd[7375]: time="2025-10-09T16:38:48.360634144Z" level=info msg="Daemon has completed initializat>
oct. 09 16:38:48 vmdocker systemd[1]: Started Docker Application Container Engine.>
oct. 09 16:38:48 vmdocker dockerd[7375]: time="2025-10-09T16:38:48.363916731Z" level=info msg="API listen on /run/docker.sock"

```

- les services/socket utilisés par docker sont : docker.service , docker.socket
- c'est l'utilisateur root qui a démarré ces services puisque Docker doit accéder à des ressources système (réseau, montage, cgroups)

- pour arrêter docker on utilise la cmd : sudo systemctl stop docker

```
dockeruser@vmdocker:~$ sudo systemctl stop docker
[sudo] password for dockeruser:
Warning: Stopping docker.service, but it can still be activated by:
  docker.socket
```

- docker.service est arrêté mais docker.socket il se peut qu'il reste en etat de marche donc on lance la cmd : sudo systemctl status docker.socket pour voir son etat

```
dockeruser@vmdocker:~$ sudo systemctl status docker.socket
● docker.socket - Docker Socket for the API
   Loaded: loaded (/lib/systemd/system/docker.socket; enabled; vendor preset: enabled)
   Active: active (listening) since Thu 2025-10-09 16:38:45 UTC; 27min ago
     Triggers: ● docker.service
   Listen: /run/docker.sock (Stream)
     Tasks: 0 (limit: 2220)
    Memory: 0B
      CPU: 940us
     CGroup: /system.slice/docker.socket

oct. 09 16:38:45 vmdocker systemd[1]: Starting Docker Socket for the API...
oct. 09 16:38:45 vmdocker systemd[1]: Listening on Docker Socket for the API.
```

- pour désactiver docker on utilise la cmd : sudo systemctl disable docker
- et pour le réactiver on utilise : sudo systemctl enable docker

```
dockeruser@vmdocker:~$ sudo systemctl disable docker
Synchronizing state of docker.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable docker
dockeruser@vmdocker:~$ systemctl is-enabled docker
disabled
dockeruser@vmdocker:~$ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.
dockeruser@vmdocker:~$ systemctl is-enabled docker
enabled
```

- on constate que les commandes docker s'utilisent en mode sudo. Pour permettre à l'utilisateur dockeruser d'exécuter les commandes docker sans passer par sudo
- pour créer un groupe nommé docker on utilise : sudo groupadd docker
- ici on constate que le groupe docker existe déjà
- pour passer en root dans son répertoire home on utilise : sudo -i
- pour ajouter l'utilisateur dockeruser au groupe docker : sudo usermod -aG docker dockeruser

```
dockeruser@vmdocker:~$ sudo groupadd docker
groupadd: group 'docker' already exists
dockeruser@vmdocker:~$ sudo -i
root@vmdocker:~# cd /home/dockeruser
root@vmdocker:/home/dockeruser# sudo usermod -aG docker dockeruser
root@vmdocker:/home/dockeruser#
```

```
root@vmdocker:/home/dockeruser# sudo reboot
Connection to 192.168.0.24 closed by remote host.
Connection to 192.168.0.24 closed.
```

```
dockeruser@vmdocker:~$ docker version
Client: Docker Engine - Community
  Version:          28.5.1
  API version:     1.51
  Go version:      go1.24.8
  Git commit:      e180ab8
  Built:           Wed Oct  8 12:17:03 2025
  OS/Arch:         linux/amd64
  Context:         default

Server: Docker Engine - Community
  Engine:
    Version:          28.5.1
    API version:     1.51 (minimum version 1.24)
    Go version:      go1.24.8
    Git commit:      f8215cc
    Built:           Wed Oct  8 12:17:03 2025
```

- on redemarre la mv et maintenant le user dockeruser a le droit de lancer la cmd : docker version

Partie 3 – Premières manipulations de conteneurs et d’images

```
dockeruser@vmdocker:~$ docker info | grep "Docker Root Dir"
Docker Root Dir: /var/lib/docker
```

- Par défaut, les informations sur les conteneurs et les images sont dans le répertoire dans /var/lib/docker

```
dockeruser@vmdocker:~$ sudo -i
root@vmdocker:~# cd /var/lib/docker
root@vmdocker:/var/lib/docker# ls
buildkit containers engine-id image network overlay2 plugins runtimes swarm tmp volumes
root@vmdocker:/var/lib/docker#
```

- les différentes catégories d’objets Docker qui peuvent être stockés : conteneurs, images, volumes, réseaux ..

```
root@vmdocker:/var/lib/docker# cd containers
root@vmdocker:/var/lib/docker/containers# ls
root@vmdocker:/var/lib/docker/containers#
```

- -lorsque on Consulte le contenu du répertoire approprié qui contient les conteneurs : on trouve 0 conteneurs pour l'instant
- recherche des images, ici on essaye de rechercher l'image hello-world : docker search hello-world

```
root@vmdocker:/var/lib/docker/containers# docker search hello-world
NAME                           DESCRIPTION                                     STARS      OFFICIAL
hello-world                     Hello World! (an example of minimal Dockeriz... 2497      [OK]
rancher/hello-world             This container image is no longer maintained... 6
okteto/hello-world              0
atlassian/hello-world           1
goharbor/hello-world            0
tutum/hello-world               Image to test docker deployments. Has Apache... 91
dockercloud/hello-world         Hello World!
crccheck/hello-world            Hello World web server in under 2.5 MB       26
koudaiii/hello-world            0
ppc64le/hello-world             2
tsepotesting123/hello-world    0
prajwalendra/hello-world        0
kevindockercompany/hello-world  0
infrastructureascode/hello-world A tiny "Hello World" web server with a healt... 1
arm32v7/hello-world             Hello World! (an example of minimal Dockeriz... 3
cloudflare/hello-world          A simple example application which can be ru... 0
datawire/hello-world            Hello World! Simple Hello World implementati... 1
twistlocktest/hello-world       0
uniplaces/hello-world          0
wjimenez5271/hello-world       0
arm64v8/hello-world             Hello World! (an example of minimal Dockeriz... 3
danfengliu/hello-world          0
lbadger/hello-world              0
ansibleplaybookbundle/hello-world Simple containerized application that tests ... 0
swarna3005/hello-world          0
root@vmdocker:/var/lib/docker/containers#
```

- pour exécuter un conteneur qui correspond à l'image ayant le plus d'étoiles :
- on télécharge l'image : docker pull hello-world
- puis on lance un conteneur à partir de cette image : docker run hello-world

```
root@vmdocker:/var/lib/docker/containers# docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
17eec7bbc9d7: Pull complete
Digest: sha256:54e66cc1dd1fc1c3c58bd8017914dbed8701e2d8c74d9262e26bd9cc1642d31
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
```

```
root@vmdocker:/var/lib/docker/containers# docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
hello-world     latest   1b44b5a3e06a  2 months ago  10.1kB
root@vmdocker:/var/lib/docker/containers#
```

```
root@vmdocker:/var/lib/docker/containers# 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/
```

use

- maintenant lorsqu'on vérifie le répertoire des conteneurs on trouve un conteneur

```
root@vmdocker:/var/lib/docker/containers# cd /var/lib/docker/containers
root@vmdocker:/var/lib/docker/containers# ls
eb01f7c975eed6bb1092b78746d912251be81e51742e7c96f3c97f82b5d6f15f
root@vmdocker:/var/lib/docker/containers#
```

- de même pour les images

```
root@vmdocker:/var/lib/docker/containers# cd /var/lib/docker/image
root@vmdocker:/var/lib/docker/image# ls
overlay2
root@vmdocker:/var/lib/docker/image#
```

- le sha256 de l'image est :
sha256:54e66cc1dd1fcbb1c3c58bd8017914dbed8701e2d8c74d9262e26bd9cc1642d31
- on utilise la commande docker ps qui pour lister les conteneurs en exécution.ici on trouve qu'il n'y a pas de conteneurs en execution

```
root@vmdocker:/var/lib/docker/image# docker ps
CONTAINER ID        IMAGE               COMMAND       CREATED          STATUS          PORTS          NAMES
root@vmdocker:/var/lib/docker/image# docker ps -a
```

- docker ps est un raccourci pour la commande docker container ls
- la commande docker qui permet de lister les images est : docker images

- l'identifiant de l'image listé est : 1b44b5a3e06a

```
root@vmdocker:/var/lib/docker/image# docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
hello-world    latest    1b44b5a3e06a  2 months ago  10.1kB
```

- lorsque on réexécute le conteneur on constate que
- Docker crée un nouveau conteneur à partir de l'image
- Chaque conteneur a un ID unique
- Le contenu affiché par le conteneur est le même puisque l'image n'a pas changé

```
root@vmdocker:/var/lib/docker/image# 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:
```

```
root@vmdocker:/var/lib/docker/image# docker ps -a
CONTAINER ID   IMAGE      COMMAND      CREATED      STATUS          PORTS      NAMES
e58b67a7aa6f   hello-world "/hello"   12 seconds ago   Exited (0) 9 seconds ago
eb01f7c975ee   hello-world "/hello"   16 minutes ago  Exited (0) 16 minutes ago
root@vmdocker:/var/lib/docker/image#
```

```
root@vmdocker:/var/lib/docker/image# docker ps
CONTAINER ID   IMAGE      COMMAND      CREATED      STATUS          PORTS      NAMES
root@vmdocker:/var/lib/docker/image#
```

- Ici on constate qu'il y a 0 conteneurs qui s'exécutent et 2 qui sont stockés localement puisque l'image hello-world permet juste d'afficher un message hello from docker ! et puis elle s'arrête .

```
root@vmdocker:/var/lib/docker/image# docker rmi hello-world
Error response from daemon: conflict: unable to remove repository reference "hello-world" (must force) - container eb01f7c975ee is using its referenced image 1b44b5a3e06a
root@vmdocker:/var/lib/docker/image#
```

- Lorsque on essaye de supprimer l'image hello-world on reçoit une erreur parce que l'image est utilisée par les conteneurs qu'on vient de lancer tout à l'heure

```
root@vmdocker:/var/lib/docker/image# docker ps -a
CONTAINER ID   IMAGE      COMMAND      CREATED      STATUS          PORTS      NAMES
e58b67a7aa6f   hello-world "/hello"   9 minutes ago  Exited (0) 9 minutes ago
eb01f7c975ee   hello-world "/hello"   25 minutes ago  Exited (0) 25 minutes ago
root@vmdocker:/var/lib/docker/image#
```

- On utilise la commande : docker ps -a
- Pour lister tous les conteneurs et pas uniquement ceux en exécution et observez leur statut.
- Les noms des conteneurs : funny_hodgkin et pensive_dhawan
- On utilise la cmd: docker ps -a --no-trunc pour afficher l'information de façon non tronquée

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
e58b67a7aa6f017680a0822f13efac43e5794c3c7036f31a9b6f5baef12b6522b	hello-world	"/hello"	11 minutes ago	Exited (0) 11 minutes ago		funny_hodgkin
eb01f7c975eed6bb1092b78746d912251be81e51742e7c96f3c97f82b5d6f15f	hello-world	"/hello"	27 minutes ago	Exited (0) 27 minutes ago		pensive_dhawan

- On Exécute une nouvelle fois l'image, puis on supprime le dernier conteneur en utilisant son nom en utilisant la cmd : docker rm adoring_jemison

```
root@vmdocker:/var/lib/docker/image# docker ps -a
CONTAINER ID   IMAGE      COMMAND   CREATED          STATUS
bba0f4ed695a   hello-world "/hello"  9 seconds ago   Exited (0) 7 seconds ago
e58b67a7aa6f   hello-world "/hello"  13 minutes ago  Exited (0) 13 minutes ago
eb01f7c975ee   hello-world "/hello"  29 minutes ago  Exited (0) 29 minutes ago
root@vmdocker:/var/lib/docker/image# docker rm adoring_jemison
adoring_jemison
root@vmdocker:/var/lib/docker/image# docker ps -a
CONTAINER ID   IMAGE      COMMAND   CREATED          STATUS
e58b67a7aa6f   hello-world "/hello"  14 minutes ago  Exited (0) 14 minutes ago
eb01f7c975ee   hello-world "/hello"  30 minutes ago  Exited (0) 30 minutes ago
root@vmdocker:/var/lib/docker/image#
```

- Maintenant on exécute une nouvelle fois l'image en utilisant son sha256 en donnant le nom 'mycont' au nouveau conteneur.
- Mais avant il faut avoir le sha256 de l'image en affichant les informations tronquée des images donc on commence par lancer la cmd : docker images --no-trunc

```
dockeruser@vmdocker:~$ docker images --no-trunc
REPOSITORY      TAG      IMAGE ID                               CREATED          SIZE
hello-world    latest   sha256:1b44b5a3e06a9aae883e7bf25e45c100be0bb81a0e01b32de604f3ac44711634  2 months ago  10.1kB
dockeruser@vmdocker:~$ docker run --name mycont sha256:1b44b5a3e06a9aae883e7bf25e45c100be0bb81a0e01b32de604f3ac44711634
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
```

- Si des conteneurs utilisent encore cette image, Docker refusera de la supprimer donc il faut tout d'abord supprimer tous les conteneurs qui utilise cette image
- Puis en supprime l'image tout simplement

```
dockeruser@vmdocker:~$ docker rm mycont vigilant_euler funny_hodgkin pensive_dhawan
mycont
vigilant_euler
funny_hodgkin
pensive_dhawan
dockeruser@vmdocker:~$ docker ps -a
CONTAINER ID   IMAGE      COMMAND   CREATED     STATUS      PORTS      NAMES
dockeruser@vmdocker:~$ docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
hello-world    latest    1b44b5a3e06a  2 months ago  10.1kB
dockeruser@vmdocker:~$ docker rmi hello-world
Untagged: hello-world:latest
Untagged: hello-world@sha256:54e66cc1dd1fcb1c3c58bd8017914dbed8701e2d8c74d9262e26bd9cc1642d31
Deleted: sha256:1b44b5a3e06a9aae883e7bf25e45c100be0bb81a0e01b32de604f3ac44711634
Deleted: sha256:53d204b3dc5ddbc129df4ce71996b8168711e211274c785de5e0d4eb68ec3851
dockeruser@vmdocker:~$
```

- Pour avoir des infos globales sur le système on lance la cmd : docker info
- On observe donc :
- Le **nombre de conteneurs** : total, en cours d'exécution, arrêtés, et créés.
- Le **nombre d'images** stockées localement.
- Les informations sur le **serveur Docker**, le **cgroup**, le **storage driver**, la version de Docker

```
dockeruser@vmdocker:~$ docker info
Client: Docker Engine - Community
  Version: 28.5.1
  Context: default
  Debug Mode: false
  Plugins:
    buildx: Docker Buildx (Docker Inc.)
      Version: v0.29.1
      Path: /usr/libexec/docker/cli-plugins/docker-buildx
    compose: Docker Compose (Docker Inc.)
      Version: v2.40.0
      Path: /usr/libexec/docker/cli-plugins/docker-compose

Server:
  Containers: 0
    Running: 0
    Paused: 0
    Stopped: 0
  Images: 0
  Server Version: 28.5.1
  Storage Driver: overlay2
    Backing Filesystem: extfs
    Supports d_type: true
    Using metacopy: false
    Native Overlay Diff: true
    userxattr: false
  Logging Driver: json-file
  Cgroup Driver: systemd
  Cgroup Version: 2
  Plugins:
    Volume: local
    Network: bridge host ipvlan macvlan null overlay
```

- On utilise la commande : docker ps -aq pour n'afficher que les IDs des conteneurs

```
dockeruser@vmdocker:~$ sudo docker ps -aq
1cc609631425
89f1794dcfdc
3964efa9259b
```

- pour exploiter les résultats de la dernière commande afin de supprimer tous les conteneurs en une seule commande on utilise la cmd : sudo docker rm \$(docker ps -aq)

```
root@vmdocker:~# sudo docker rm $(docker ps -aq)
1cc609631425
89f1794dcfdc
3964efa9259b
```

- maintenant lorsqu'on essaye de supprimer l'image l'action se fait avec succès parce que à ce moment il n'a y a pas aucun conteneur qui utilise cette image puisqu'on a tous les supprimé

```
root@vmdocker:~# docker rmi hello-world
Untagged: hello-world:latest
Untagged: hello-world@sha256:54e66cc1dd1fcb1c3c58bd8017914dbed8701e2d8c74d9262e26bd9cc1642d31
Deleted: sha256:1b44b5a3e06a9aae883e7bf25e45c100be0bb81a0e01b32de604f3ac44711634
Deleted: sha256:53d204b3dc5ddbc129df4ce71996b8168711e211274c785de5e0d4eb68ec3851
root@vmdocker:~#
```

- maintenant si on Fait à nouveau exécuter un conteneur, qu'on le nomme hello1, pour la même image hello-world, on reçoit un message qui nous averti que l'image n'est pas trouvée et il lance le téléchargement de l'image automatiquement

```
dockeruser@vmdocker:~$ docker run --name hello1 hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
17eec7bbc9d7: Pull complete
Digest: sha256:54e66cc1dd1fcb1c3c58bd8017914dbed8701e2d8c74d9262e26bd9cc1642d31
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
```

- l'identifiant de l'image est : 1b44b5a3e06a

```
dockeruser@vmdocker:~$ docker images
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
hello-world    latest        1b44b5a3e06a   2 months ago   10.1kB
```

- on Crée maintenant un conteneur, qu'on le nomme hello2, mais sans le démarrer, on voit que son statut est created .

```
dockeruser@vmdocker:~$ docker create --name hello2 hello-world  
3948027aba89e2fbb0829ff60640958372388bf2e6818ac6158bf8a5062960a9
```

- On démarre un conteneur déjà créé en utilisant la cmd : docker start
- On constate qu'il affiche juste le nom de conteneur pour qu'il confirme qu'il est lancé avec succès

```
dockeruser@vmdocker:~$ docker start hello2  
hello2
```

- on utilise l'option -a qui permettra d'attacher l'entrée et la sortie standard pour démarrer ce conteneur hello2.

```
dockeruser@vmdocker:~$ docker start -a hello2  
  
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/
```

- maintenant on peut aussi l'appliquer sur le conteneur hello1 pour le relancer

```
dockeruser@vmdocker:~$ docker start -a hello1  
  
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.
```

- on lance maintenant un nouveau conteneur hello3 sans afficher sa sortie standard grâce à l'option -d
- donc ici en utilisant -d on indique à Docker d'exécuter le conteneur en arrière-plan, sans attacher le terminal.
- Et donc on obtient seulement l'id de conteneur lancé

```
dockeruser@vmdocker:~$ docker run -d --name hello3 hello-world
2887aa874734121257a413883810de3a41e1fbdc64121992e062abe8995abb34
```

- Pour créer un conteneur qui s'auto-supprime dès la fin de son exécution, on utilise l'option --rm

```
dockeruser@vmdocker:~$ docker run --rm --name hello4 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/
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
2887aa874734	hello-world	"/hello"	7 minutes ago	Exited (0) 7 minutes ago		hello3
55e060ab1f4c	1b44b5a3e06a	"/hello"	26 minutes ago	Exited (0) 26 minutes ago		mycont
8b51fcf4e181	1b44b5a3e06a	"/hello"	31 minutes ago	Exited (0) 31 minutes ago		heloooo
3948027aba89	hello-world	"/hello"	36 minutes ago	Exited (0) 11 minutes ago		hello2
054781c32a6e	hello-world	"/hello"	43 minutes ago	Exited (0) 9 minutes ago		hello1

- Finalement on supprimer l'image hello-world bien sur après avoir supprimer tous les conteneurs qui dépendent d'elle.

```

dockeruser@vmdocker:~$ docker ps -a
CONTAINER ID   IMAGE      COMMAND     CREATED      STATUS          PORTS
2887aa874734   hello-world   "/hello"    8 minutes ago  Exited (0) 8 minutes ago
55e060ab1f4c   1b44b5a3e06a   "/hello"    27 minutes ago  Exited (0) 27 minutes ago
8b51fcf4e181   1b44b5a3e06a   "/hello"    32 minutes ago  Exited (0) 32 minutes ago
3948027aba89   hello-world   "/hello"    37 minutes ago  Exited (0) 13 minutes ago
054781c32a6e   hello-world   "/hello"    44 minutes ago  Exited (0) 10 minutes ago
dockeruser@vmdocker:~$ docker rm 2887aa874734 55e060ab1f4c 8b51fcf4e181 3948027aba89 054781c32a6e
2887aa874734
55e060ab1f4c
8b51fcf4e181
3948027aba89
054781c32a6e
dockeruser@vmdocker:~$ docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
hello-world      latest   1b44b5a3e06a   2 months ago  10.1kB
dockeruser@vmdocker:~$ docker rmi hello-world
Untagged: hello-world:latest
Untagged: hello-world@sha256:54e66cc1dd1fcb1c3c58bd8017914dbed8701e2d8c74d9262e26bd9cc1642d31
Deleted: sha256:1b44b5a3e06a9aae883e7bf25e45c100be0bb81a0e01b32de604f3ac44711634
Deleted: sha256:53d204b3dc5ddbc129df4ce71996b8168711e211274c785de5e0d4eb68ec3851

```

Partie 4 – DockerHub

- Sur <https://hub.docker.com/>, on trouve deux grandes catégories d’images :
 - Des images officielles : images validées et maintenues par Docker par exemple : ubuntu, nginx, mysql, postgres, tensorflow
 - Des Images de la communauté (Community Images) : images publiées par les utilisateurs ou les organisations .
- la classification se fait comme suit :
 - Par **éditeur** (Docker, communauté, entreprises)
 - Par **catégorie** (OS, base de données, langages, outils.)
 - Par **tags** (version spécifique d’une image)

Categories



- Networking
- Security
- Languages & frameworks
- Integration & delivery
- Message queues
- API management
- Internet of things
- Machine learning & AI
- Developer tools
- Data science
- Web servers
- Operating systems
- Content management system
- Databases & storage
- Monitoring & observability
- Web analytics

➤ voici l'image officielle de ubuntu

https://hub.docker.com/_/ubuntu

C'est l'image officielle maintenue par Canonical.



ubuntu



Docker Official Image

1B+

10K+

Ubuntu is a Debian-based Linux operating system based on free software.

OPERATING SYSTEMS

[Overview](#)

[Tags](#)

Quick reference

- Maintained by:
[Canonical](#)

- les différentes versions proposées

[22.04](#), [jammy-20251001](#), [jammy](#)
[24.04](#), [noble-20251001](#), [noble](#), [latest](#)
[25.04](#), [plucky-20251001](#), [plucky](#)
[25.10](#), [questing-20251007](#), [questing](#), [rolling](#)

Supported tags and respective Dockerfile links

- [22.04](#), [jammy-20251001](#), [jammy](#)
- [24.04](#), [noble-20251001](#), [noble](#), [latest](#)
- [25.04](#), [plucky-20251001](#), [plucky](#)
- [25.10](#), [questing-20251007](#), [questing](#), [rolling](#)

- Chaque tag correspond à une version spécifique de l'image, souvent liée à une version d'Ubuntu.

- La version la plus récente est **ubuntu:25.10**.

TAG 25.10			
Digest	OS/ARCH	Vulnerabilities	Compressed size ⓘ
af5be3d16518	linux/amd64	None found	32.86 MB
b3eabfa0a813	linux/arm/v7	None found	30.36 MB
1245e1b09134	linux/arm64/v8	None found	32.49 MB
+3 more...			

docker pull ubuntu:25.10 

- La différence avec la version latest est :

latest pointe vers **noble (24.04)**, une version stable.

25.10 est **plus récente**, mais **non encore LTS**, donc en phase de test.

- Pour amd64 :

ubuntu:25.10 a l'id af5be3d16518

ubuntu:latest (ou noble) a l'id d22e4fb38906

TAG 25.10			
Digest	OS/ARCH	Vulnerabilities	Compressed size ⓘ
af5be3d16518	linux/amd64	None found	32.86 MB

TAG latest			
Digest	OS/ARCH	Vulnerabilities	Compressed size ⓘ
d22e4fb38906	linux/amd64	0 0 2 5 0	28.35 MB

- Pour la version 25.10 il n'y a pas de vulnérabilité
- Pour la version latest il y a quelques vulnérabilités mineures (2 moyennes et 5 faibles)
- Les vulnérabilités sont classées par niveau de gravité :

Critique | Haute | Moyenne | Faible | Non spécifiée

- Le tag `noble` correspond spécifiquement à Ubuntu 24.04 LTS et garantit que tu restes toujours sur cette version précise, ce qui est important pour la stabilité en production.
 - Le tag `latest`, lui, pointe automatiquement vers la dernière version stable d'Ubuntu — aujourd'hui c'est aussi *noble*, mais plus tard, quand une nouvelle version sortira, `latest` pointerà vers elle.
 - Donc ici on a un tag **fixe** (`noble`) pour les environnements où la version ne doit pas changer, et un tag **évolutif** (`latest`) pour suivre automatiquement les nouvelles versions.
-
- L'intérêt du fait que la version `latest` et la version `noble` soient identiques est de permettre aux utilisateurs de choisir entre stabilité et souplesse.
 - En cliquant sur le tag de chacune des 2 je constate que les couches de `ubuntu:latest` et `ubuntu:noble` sont strictement identiques je pense que cela signifie que `latest` pointe actuellement vers la même version que `noble`, c'est-à-dire Ubuntu 24.04 LTS.



ubuntu:noble MULTI-PLATFORM

OPERATING SYSTEMS

INDEX DIGEST sha256:66460d557b25769b102175144d538d88219c077c678a49af4afca6fbfc1b5252

OS/ARCH

linux/amd64



COMPRESSED SIZE ⓘ

28.35 MB

LAST PUSHED

2 days by [doijanky](#)

TYPE

Image

VULNERABILITIES

0 0 2 5

Layers (6)

▼ ubuntu:noble

0 0 2 5 0

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:249778a1782b02a1c2bcf9f292f5778d81442a53c3de195... 29.72 MB

5 CMD ["/bin/bash"] 0 B

ubuntu:latest MULTI-PLATFORM
OPERATING SYSTEMS
INDEX DIGEST sha256:66460d557b25769b102175144d538d88219c077c678a49af4afca6fbfc1b5252 |
OS/ARCH
linux/amd64
COMPRESSED SIZE 28.35 MB LAST PUSHED 2 days by doijanky TYPE Image VULNERABILITIES 0 0 2 5

Layers (6)

ubuntu:latest		0 0 2 5 0
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:249778a1782b02a1c2bcf9f292f5778d81442a53c3de195...	29.72 MB !
5	CMD ["/bin/bash"]	0 B

- On visite le répertoire /var/lib/docker/image/overlay2/layerdb/sha256 et on liste son contenu.

```
root@vmdocker:~# cd /var/lib/docker/image/overlay2/layerdb/sha256
root@vmdocker:/var/lib/docker/image/overlay2/layerdb/sha256# sudo ls -l
total 4
drwx----- 2 root root 4096 oct. 9 14:31 e36a1e823faa951ea5bd76d8624b2802ce32ddd9326515c99cae311c72d48aea
root@vmdocker:/var/lib/docker/image/overlay2/layerdb/sha256#
```

- Ici je vois que par défaut la version latest est téléchargé

```
root@vmdocker:/var/lib/docker/image/overlay2/layerdb/sha256# docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
4b3ffd8ccb52: Pull complete
Digest: sha256:66460d557b25769b102175144d538d88219c077c678a49af4afca6fbfc1b5252
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```

- Maintenant on trouve que l'image ubuntu la plus récente selon dockerhub est celle qui a le tag rolling

```
dockeruser@vmdocker:~$ docker pull ubuntu:rolling
rolling: Pulling from library/ubuntu
9b965cd35928: Pull complete
Digest: sha256:9b61739164b58f2263067bd3ab31c7746ded4cade1f9d708e6f1b047b408a470
Status: Downloaded newer image for ubuntu:rolling
docker.io/library/ubuntu:rolling
dockeruser@vmdocker:~$
```

- pour n'afficher que les images référençant ubuntu on utilise la cmd : docker images ubuntu

```
dockeruser@vmdocker:~$ docker images ubuntu
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
ubuntu          rolling      38d4707cb77a   44 hours ago  90.1MB
ubuntu          latest       97bed23a3497   8 days ago   78.1MB
dockeruser@vmdocker:~$
```

- Lorsque on exécute un conteneur à partir de l'image Ubuntu, sans spécifier de commande, Docker lance par défaut celle qui est définie dans le Dockerfile de l'image.
- Pour le vérifier, on utilise la cmd : docker inspect ubuntu

```
"Config": {
    "Cmd": [
        "/bin/bash"
    ],
    "Entrypoint": null,
```

- On trouve que la commande par défaut exécutée est : /bin/bash

Partie 5 – Interagir avec un conteneur

- J'ai été dans une connexion SSH, la session SSH se termine et je deviens déconnecté.

```
dockeruser@vmdocker:~$ 
logout
Connection to 192.168.0.24 closed.
```

- Lorsque on lancez un conteneur à partir de l'image ubuntu avec la cmd docker run ubuntu on voit juste qu'il est execute avec success mais on ne voit pas une interface ou espace de commande pour intérragir avec le system ubutu qu'on a lance
- C'est pour ça on ajoute l'option -it dans la cmd pour q'elle devienne : docker run -it ubuntu

```
dockeruser@vmdocker:~$ docker run    ubuntu
dockeruser@vmdocker:~$
```

```
dockeruser@vmdocker:~$ docker run -it ubuntu
root@a651c93551fb:/#
```

- Pour lancer, à partir de l'image ubuntu, un conteneur nommé os_ubuntu en interactif et attaché à un terminal on utlise la cmd : docker run --name os_ubuntu -it ubuntu

```
exit
dockeruser@vmdocker:~$ docker run --name os_ubuntu -it ubuntu
root@9bdecf677258:/#
```

- On lance la commande ps -ef qui sert à afficher la liste de tous les processus en cours d'exécution dans le conteneur
- Ici on constate que la commande qui correspond au processus de PID 1 de ce conteneur est : /bin/bash

```
dockeruser@vmdocker:~$ docker run --name os_ubuntu -it ubuntu
root@9bdecf677258:/# ps -ef
UID          PID      PPID      C STIME      TTY          TIME CMD
root           1          0      0 16:46 pts/0      00:00:00 /bin/bash
root           9          1      0 16:47 pts/0      00:00:00 ps -ef
root@9bdecf677258:/#
```

- On exécute les commandes whoami, pwd, ls et hostname.on constate que l'ID du conteneur correspond à ce que renvoie hostname qui est dans notre cas : 9bdecf677258

```
root@9bdecf677258:/# whoami
root
root@9bdecf677258:/# pwd
/
root@9bdecf677258:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@9bdecf677258:/# hostname
9bdecf677258
root@9bdecf677258:/#
```

- Ici lorsque on Ouvre un deuxième terminal, et se connecte en ssh sur la VM et on observe le statut du conteneur en cours d'exécution. On constate que même si on ouvre un deuxième terminal SSH, on ne vera pas le terminal interactif du conteneur par défaut.

```
Last login: Thu Oct  9 16:31:24 2025 from 192.168.0.23
dockeruser@vmdocker:~$ docker ps
CONTAINER ID        IMAGE       COMMAND       CREATED          STATUS          PORTS     NAMES
9bdecf677258        ubuntu      "/bin/bash"   14 minutes ago   Up 14 minutes           os_ubuntu
dockeruser@vmdocker:~$
```

- On Revient dans le conteneur sur le premier terminal. Et on se déplace dans le répertoire home

```
root@9bdecf677258:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@9bdecf677258:/# cd home
root@9bdecf677258:/home#
```

- -maintenant on quittez ce conteneur avec la commande unix classique “exit” puis on observe le statut du conteneur en utilisant la cmd : docker ps -a

```
root@9bdecf677258:/# cd home
root@9bdecf677258:/home# exit
exit
dockeruser@vmdocker:~$ docker ps -a
CONTAINER ID        IMAGE       COMMAND       CREATED          STATUS          PORTS     NAMES
9bdecf677258        ubuntu      "/bin/bash"   27 minutes ago   Exited (0) 25 seconds ago           os_ubuntu

```

- On démarrez ce conteneur une nouvelle fois en interactif. On trouve qu'on se retrouve sur le répertoire / et non pas sur /home on constate donc que le shell ne se souvient pas du répertoire précédent où on était avant d'arrêter le conteneur

```
dockeruser@vmdocker:~$ docker start -i os_ubuntu
root@9bdecf677258:/# pwd
/
root@9bdecf677258:/#
```

- Dans le second terminal, on utilise la commande docker inspect os_ubuntu qui permet d'inspecter le conteneur.
- On trouve que son status est running et son pid est 4065

```

dockeruser@vmdocker:~$ docker inspect os_ubuntu
[
  {
    "Id": "9bdecf677258ed1c28cd622c7820168c00689904addca1563162d34bfc7c773c",
    "Created": "2025-10-09T16:46:28.228026098Z",
    "Path": "/bin/bash",
    "Args": [],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 4065,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2025-10-09T17:17:17.154925394Z",
      "FinishedAt": "2025-10-09T17:13:48.705617501Z"
    },
    "Image": "sha256:97bed23a34971024aa8d254abbe67b7168772340d1f494034773bc464e8dd5b6",
    "ResolvConfPath": "/var/lib/docker/containers/9bdecf677258ed1c28cd622c7820168c00689904addca1563162d
esolv.conf",
    "HostnamePath": "/var/lib/docker/containers/9bdecf677258ed1c28cd622c7820168c00689904addca1563162d34
tname",
  }
]

```

- On se basant des exemples trouvés ici : <https://blog.madrzejewski.com/jq-traiter-parser-json-shell-cli/>
- On a pu identifié la cmd : docker inspect os_ubuntu | jq '.[0].State.Pid' qui permet d'afficher directement le pid

```

dockeruser@vmdocker:~$ docker inspect os_ubuntu | jq '.[0].State.Pid'
4065
dockeruser@vmdocker:~$

```

- Dans le même terminal, on retrouve le processus dont le PID est celui qu'on vient d'identifier on utilise la cmd ps -fp 4065
- -f pour affiche ren format complet (UID, PID, PPID, CPU, CMD.).
- -p 4065 filtre pour n'afficher que le processus ayant ce PID.
- On constate que PID correspond bien au processus principal du conteneur
- Ce processus tourne sur l'hôte, mais fait partie du conteneur Docker

```

dockeruser@vmdocker:~$ ps -fp 4065
UID          PID      PPID   C STIME TTY          TIME CMD
root        4065      4038   0 17:17 pts/0    00:00:00 /bin/bash
dockeruser@vmdocker:~$ 

```

- Maintenant on on Revient dans le conteneur (dans le premier terminal) et on se déplace à nouveau dans home.

```
root@9bdecf677258:/# cd home  
root@9bdecf677258:/home#
```

- On Quittez maintenant ce conteneur en utilisant la combinaison de touches Ctrl-q. on constate que maintenant le statut du conteneur est encore actif

```
root@9bdecf677258:/home# docker user@vmdocker:~$ docker ps  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
9bdecf677258 ubuntu "/bin/bash" About an hour ago Up 29 minutes os_ubuntu  
docker user@vmdocker:~$
```

- On utilise la commande docker : docker diff os_ubuntu pour voir les différences dans le système de fichiers du conteneur.

```
docker user@vmdocker:~$ docker diff os_ubuntu  
C /root  
A /root/.bash_history  
docker user@vmdocker:~$
```

- On lance la cmd docker exec os_ubuntu hostname pour exécuter la commande Unix hostname dans ce conteneur (sans le passer en foreground).

```
docker user@vmdocker:~$ docker exec os_ubuntu hostname  
9bdecf677258  
docker user@vmdocker:~$
```

- pour exécuter la commande Unix bash en interactif dans ce conteneur on utilise la cmd docker exec -it os_ubuntu bash
- Il y a 2 shells bash actifs dans le conteneur :
 - Le shell principal du conteneur (PID 1)
 - Le shell ouvert avec docker exec (PID 21)

Le troisième (grep) est temporaire, juste pour la recherche

```
docker user@vmdocker:~$ docker exec -it os_ubuntu bash  
root@9bdecf677258:/# ps -ef | grep bash  
root 1 0 17:17 pts/0 00:00:00 /bin/bash  
root 21 0 17:56 pts/1 00:00:00 bash  
root 30 21 0 17:56 pts/1 00:00:00 grep --color=auto bash  
root@9bdecf677258:/#
```

- On quitte à nouveau ce conteneur sans l'arrêter, puis on utilise la commande docker top os_ubuntu qui affiche les processus du conteneur.

```

dockeruser@vmdocker:~$ docker top os_ubuntu
UID          PID    PPID      C      STIME      TTY      TIME      CMD
root        4065    4038      0   17:17 pts/0    00:00:00 /bin/bash
root       16136    4038      0   17:56 pts/1    00:00:00 bash
dockeruser@vmdocker:~$
```

- Maintenant on revient dans le conteneur en tapant la cmd docker attach os_ubuntu et
- Et on l'arrête en le quittant en tapant exit

```

dockeruser@vmdocker:~$ docker attach os_ubuntu
root@9bdecf677258:/home# exit
exit
dockeruser@vmdocker:~$
```

- On lance 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 Oct  9 18:06 dev
drwxr-xr-x  1 root root 4096 Oct  9 18:06 etc
drwxr-xr-x  3 root root 4096 Oct  1 02:10 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 Oct  1 02:03 media
drwxr-xr-x  2 root root 4096 Oct  1 02:03 mnt
drwxr-xr-x  2 root root 4096 Oct  1 02:03 opt
dr-xr-xr-x 191 root root  0 Oct  9 18:06 proc
drwxr----- 2 root root 4096 Oct  1 02:09 root
drwxr-xr-x  4 root root 4096 Oct  1 02:10 run
lrwxrwxrwx  1 root root  8 Apr 22 2024 sbin -> usr/sbin
drwxr-xr-x  2 root root 4096 Oct  1 02:03 srv
dr-xr-xr-x 13 root root  0 Oct  9 18:06 sys
drwxrwxrwt  2 root root 4096 Oct  1 02:09 tmp
drwxr-xr-x 12 root root 4096 Oct  1 02:03 usr
drwxr-xr-x 11 root root 4096 Oct  1 02:09 var
dockeruser@vmdocker:~$
```

- On peut le redémarrer pour obtenir le même affichage

```

dockeruser@vmdocker:~$ docker start -a ll
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 Oct   9 18:07 dev
drwxr-xr-x  1 root root 4096 Oct   9 18:06 etc
drwxr-xr-x  3 root root 4096 Oct   1 02:10 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 Oct   1 02:03 media
drwxr-xr-x  2 root root 4096 Oct   1 02:03 mnt
drwxr-xr-x  2 root root 4096 Oct   1 02:03 opt
dr-xr-xr-x 193 root root    0 Oct   9 18:07 proc
drwxr----- 2 root root 4096 Oct   1 02:09 root
drwxr-xr-x  4 root root 4096 Oct   1 02:10 run
lrwxrwxrwx  1 root root    8 Apr 22  2024 sbin -> usr/sbin
drwxr-xr-x  2 root root 4096 Oct   1 02:03 srv
dr-xr-xr-x 13 root root    0 Oct   9 18:06 sys
drwxrwxrwt  2 root root 4096 Oct   1 02:09 tmp
drwxr-xr-x 12 root root 4096 Oct   1 02:03 usr
drwxr-xr-x 11 root root 4096 Oct   1 02:09 var
dockeruser@vmdocker:~$
```

- on lance un nouveau conteneur nommé ps avec la commande ps aux, mais en faisant en sorte que ce conteneur disparaisse après son exécution.
- On constate Ici, comme on exécute ps aux directement, ce processus devient le PID 1.
- Comme on a utilisé --rm, le conteneur **disparaît automatiquement** après la fin de la commande.

```

dockeruser@vmdocker:~$ docker run --name ps --rm ubuntu ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root        1  8.5  0.1  7888  3668 ?          Rs   18:09   0:00 ps aux
dockeruser@vmdocker:~$ docker ps
CONTAINER ID   IMAGE   COMMAND   CREATED   STATUS    PORTS   NAMES
dockeruser@vmdocker:~$
```

- On lance un nouveau conteneur nommé salut avec la commande echo Bonjour, puis le relance.
- Pour voir à nouveau la sortie, il faudrait utiliser docker logs.

```
CONTAINER ID        IMAGE       COMMAND      CREATED      STATUS      PORTS      NAMES
dockeruser@vmdocker:~$ docker run --name salut ubuntu echo Bonjour
Bonjour
dockeruser@vmdocker:~$ docker start salut
salut
dockeruser@vmdocker:~$ docker logs salut
Bonjour
Bonjour
```

- On lancez un nouveau conteneur avec une commande infinie (sh -c "while true ; sleep 3600 ; done"), puis on le supprime apès l'avoir stopper.

```
dockeruser@vmdocker:~$ docker run -d --name inf ubuntu sh -c "while true; do sleep 3600; done"
c92fd7c0ceb7f53b4250299d2b5266b41cf9479c005957369ef22dc639157678
dockeruser@vmdocker:~$ docker ps
CONTAINER ID   IMAGE      COMMAND      CREATED      STATUS      PORTS      NAMES
c92fd7c0ceb7   ubuntu     "sh -c 'while true; ...'"   8 seconds ago   Up 5 seconds   inf
```

```
dockeruser@vmdocker:~$ docker stop inf
inf
dockeruser@vmdocker:~$ docker rm inf
inf
dockeruser@vmdocker:~$ docker ps
CONTAINER ID   IMAGE      COMMAND      CREATED      STATUS      PORTS      NAMES
dockeruser@vmdocker:~$
```

- On Démarrer le conteneur os_ubuntu en interactif. Dans ce shell, on lance la commande " while true; do echo salut; sleep 3; done " qui affiche salut toutes les 3 secondes

```
dockeruser@vmdocker:~$ docker start -ai os_ubuntu
root@9bdecf677258:/# while true; do echo salut; sleep 3; done
salut
salut
salut
```

- Dans le deuxième terminal connecté en ssh à la VM, on utilise la commande docker attach os_ubuntu pour s'attacher au conteneur qui tourne dans le premier terminal.
- On voit le même résultat qui fait le boucle qu'on a déjà lancé
- Maintenant on appuie sur Ctrl + C pour stopper la boucle infinie.
- On lance la cmd ls
- On constate le même résultat dans les deux terminaux

- On quitte ce conteneur en utilisant la commande bash exit avec un code retour non nul. Ici on a écrit 1
 - On voit Exited (1) qui indique que le conteneur s'est arrêté avec un code de sortie 1.

```
root@9bdecf677258:/# exit 1
exit
dockeruser@vmdocker:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
dockeruser@vmdocker:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
df2c2e09ab06        ubuntu              "echo Bonjour"    14 minutes ago   Exited (0) 14 minutes ago   salut
496fa2960fc5        ubuntu              "ls -l"           23 minutes ago   Exited (0) 22 minutes ago   ll
9bdecf677258        ubuntu              "/bin/bash"       2 hours ago      Exited (1) 12 seconds ago   os_ubuntu
a651c93551fb        ubuntu              "/bin/bash"       2 hours ago      Exited (127) 2 hours ago   goofy_driscoll
dc067703543a        ubuntu              "/bin/bash"       2 hours ago      Exited (0) 2 hours ago   determined_chaum
268323fe47b5        ubuntu              "/bin/bash"       2 hours ago      Exited (0) 2 hours ago   testubuntu
```

- On utilise la cmd docker logs os_ubuntu pour voir le log du conteneur os_ubuntu

```
dockeruser@vmdocker:~$ docker logs os_ubuntu
root@9bdecf677258:/# ps -ef
UID      PID  PPID  C STIME TTY          TIME CMD
root        1      0  0 16:46 pts/0    00:00:00 /bin/bash
root        9      1  0 16:47 pts/0    00:00:00 ps -ef
root@9bdecf677258:/# whoami
root
root@9bdecf677258:/# pwd
/
root@9bdecf677258:/# hostname
9bdecf677258
root@9bdecf677258:/# ps -ef
UID      PID  PPID  C STIME TTY          TIME CMD
root        1      0  0 16:46 pts/0    00:00:00 /bin/bash
root       12      1  0 16:53 pts/0    00:00:00 ps -ef
root@9bdecf677258:/# whoami
root
root@9bdecf677258:/# pwd
/
root@9bdecf677258:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@9bdecf677258:/# hostname
9bdecf677258
root@9bdecf677258:/# cd /root
root@9bdecf677258:~/#
root@9bdecf677258:~/# pwd
/root
root@9bdecf677258:~/# ls
root@9bdecf677258:~/# cd home
bash: cd: home: No such file or directory
root@9bdecf677258:~/# cd /
root@9bdecf677258:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@9bdecf677258:/# cd home
root@9bdecf677258:/home# exit
exit
```

- on utilise la cmd docker container prune 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:
df2c2e09ab06f99025979e7950449aff8763ebecb23559f923bf915db5323319
496fa2960cf5b4706bc1d21c5f4175e48506a41e5a4baa773a7622dfa9ff4deb
9bdecf677258ed1c28cd622c7820168c00689904addca1563162d34bfc7c773c
a651c93551fb1ee777586816f44844380f8c2c6a08c9897893a3bf386ee0b5ad
dc067703543a44cddb7c7dd299e8a802d91593b2f92a24e3ca52c206c94c2014
268323fe47b51c7eff334fdb51b565804d4a2448b6d901529a95a019074bf750
f230ae57c5be460a72a67894fb64a2435afa378f2b6056d10af2881c5fd2c10c

Total reclaimed space: 222B
dockeruser@vmdocker:~$
```