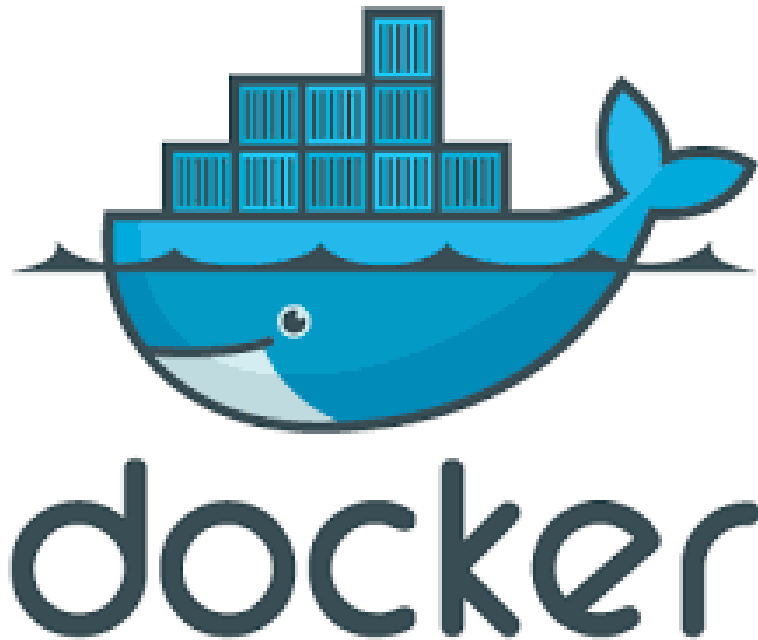


Docker



Introduction

Dans ce tutoriel, vous allez installer et utiliser Docker . Vous allez installer Docker lui-même, travailler avec des conteneurs et des images.

Conditions préalables

Pour suivre ce tutoriel, vous aurez besoin des éléments suivants :

- Un serveur Ubuntu

Les étapes:

▼ Étape 1 — Installation de Docker

- Tout d'abord, mettez à jour votre liste de packages existante :

```
sudo apt update
```

- Ensuite, installez quelques paquets pré-requis qui permettent à `apt` d'utiliser les paquets sur HTTPS :

```
sudo apt install apt-transport-https ca-certificates curl software-properties-comm  
on
```

- Ensuite, ajoutez la clé GPG du dépôt officiel de Docker à votre système :

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

- Ajoutez le référentiel Docker aux sources APT :

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu  
focal stable"
```

- Assurez-vous que vous êtes sur le point d'installer à partir du dépôt Docker et non du dépôt Ubuntu par défaut :

```
apt-cache policy docker-ce
```

Vous verrez un résultat comme celui-ci, bien que le numéro de version du Docker puisse être différent :

Output of `apt-cache policy docker-ce`

```
hilali@hilali-VirtualBox:~$ apt-cache policy docker-ce  
docker-ce:  
  Installé : 5:23.0.5-1~ubuntu.20.04~focal  
  Candidat : 5:23.0.5-1~ubuntu.20.04~focal  
Table de version :  
*** 5:23.0.5-1~ubuntu.20.04~focal 500  
    500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages  
    100 /var/lib/dpkg/status
```

Notez que le `docker-ce` n'est pas installé, mais que le candidat à l'installation provient du dépôt Docker pour Ubuntu 20.04 (`focal`).

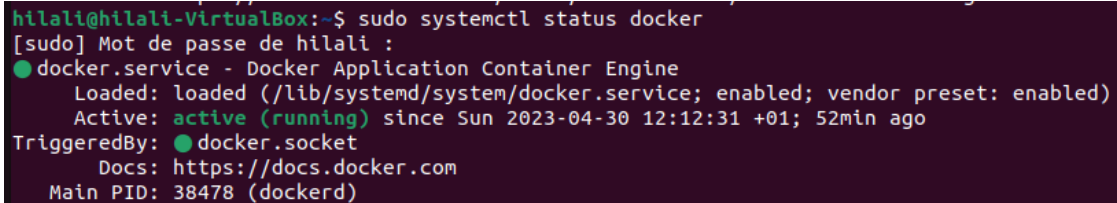
Enfin, installez Docker :

```
sudo apt install docker-ce
```

Le Docker devrait maintenant être installé, le démon démarré, et le processus autorisé à démarrer au boot. Vérifiez qu'il tourne :

```
sudo systemctl status docker
```

La sortie devrait être similaire à ce qui suit, montrant que le service est actif et en cours d'exécution :



```
hilali@hilali-VirtualBox:~$ sudo systemctl status docker
[sudo] Mot de passe de hilali :
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2023-04-30 12:12:31 +01; 52min ago
     TriggeredBy: ● docker.socket
   Docs: https://docs.docker.com
   Main PID: 38478 (dockerd)
```

L'installation de Docker vous donne maintenant non seulement le service Docker (démon) mais aussi l'utilitaire en ligne de commande `docker` == (ou le client Docker).

▼ Étape 2 — Exécution de la commande Docker sans sudo (facultatif)

Par défaut, la commande `docker` ne peut être exécutée que par l'utilisateur **root** ou par un utilisateur du groupe **docker**, qui est automatiquement créé lors du processus d'installation de Docker. Si vous essayez d'exécuter la commande `docker` sans la faire précéder de `sudo` ou sans être dans le groupe **docker**, vous obtiendrez une erreur .

- Si vous voulez éviter de taper `sudo` chaque fois que vous exécutez la commande `docker`, ajoutez votre nom d'utilisateur au groupe `docker` :

```
sudo usermod -aG docker ${USER}
```

- Pour appliquer la nouvelle appartenance au groupe, déconnectez-vous du serveur et reconnectez-vous, ou tapez ce qui suit :

```
su - ${USER}
```

- Vous serez invité à saisir le mot de passe utilisateur pour continuer.

Vérifiez que votre utilisateur est maintenant ajouté au groupe **docker** en tapant :

```
id -nG
```

```
hilali@hilali-VirtualBox:~$ id -nG
hilali adm cdrom sudo dip plugdev lpadmin lxd sambashare ubridge libvirt docker
```

Si vous devez ajouter un utilisateur au groupe `docker` pour lequel vous n'êtes pas connecté, déclarez ce nom d'utilisateur explicitement :

```
sudo usermod -aG dockerusername
```

La suite de cet article suppose que vous exécutez la commande `docker` en tant qu'utilisateur dans le groupe **docker**. Si vous choisissez de ne pas le faire, veuillez faire précéder les commandes de `sudo`.

Examinons maintenant la commande `docker`.

▼ Étape 3 — Utilisation de la commande Docker

L'utilisation de `docker` consiste à lui faire passer une chaîne d'options et de commandes suivie d'arguments. La syntaxe prend cette forme :

```
docker [option] [command] [arguments]
```

- Pour voir toutes les sous-commandes disponibles, tapez :

```
docker
```

À partir du docker 19, la liste complète des sous-commandes disponibles est incluse :

```
hilaighilali-VirtualBox:~$ docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Common Commands:
run      Create and run a new container from an image
exec     Execute a command in a running container
ps       List containers
build    Build an image from a Dockerfile
pull     Download an image from a registry
push     Upload an image to a registry
images   List images
login    Log in to a registry
logout   Log out from a registry
search   Search Docker Hub for images
version  Show the Docker version information
info     Display system-wide information

Management Commands:
builder  Manage builds
buildx*  Docker Buildx (Docker Inc., v0.10.4)
compose* Docker Compose (Docker Inc., v2.17.2)
container Manage containers
context  Manage contexts
image    Manage images
manifest Manage Docker image manifests and manifest lists
network  Manage networks
plugin   Manage plugins
system   Manage Docker
trust    Manage trust on Docker images
volume   Manage volumes

Swarm Commands:
```

- Pour voir les options disponibles pour une commande spécifique, tapez :

```
docker docker-subcommand --help
```

- Pour voir les informations sur Docker à l'échelle du système, utilisez :

```
docker info
```

Examinons certaines de ces commandes. Nous allons commencer par travailler avec des images.

▼ Étape 4 — comment créer un image docker

- Voici un exemple très simple de création d'une image Docker qui affiche "Bonjour, c'est l'image de RSSP" à partir d'un Dockerfile :
1. Créez un nouveau fichier nommé `Dockerfile` dans un nouveau répertoire vide.
 2. Ouvrez le fichier `Dockerfile` dans un éditeur de texte et ajoutez les lignes suivantes :

```
# Utilisez une image de base Ubuntu
FROM ubuntu

# Définissez l'étiquette du créateur
LABEL creator="RSSP"

# Exécutez la commande echo pour afficher le message
CMD echo "Bonjour, c'est l'image de RSSP"
```

1. Enregistrez le fichier `Dockerfile`.
2. Ouvrez un terminal dans le répertoire où se trouve le fichier `Dockerfile`.
3. Exécutez la commande suivante pour créer l'image Docker :

```
docker build -t mon-image-rssp .
```

`-t` pour spécifier le nom de l'image

1. Attendez que la création de l'image soit terminée.
2. Exécutez la commande suivante pour exécuter un conteneur à partir de l'image que vous venez de créer :

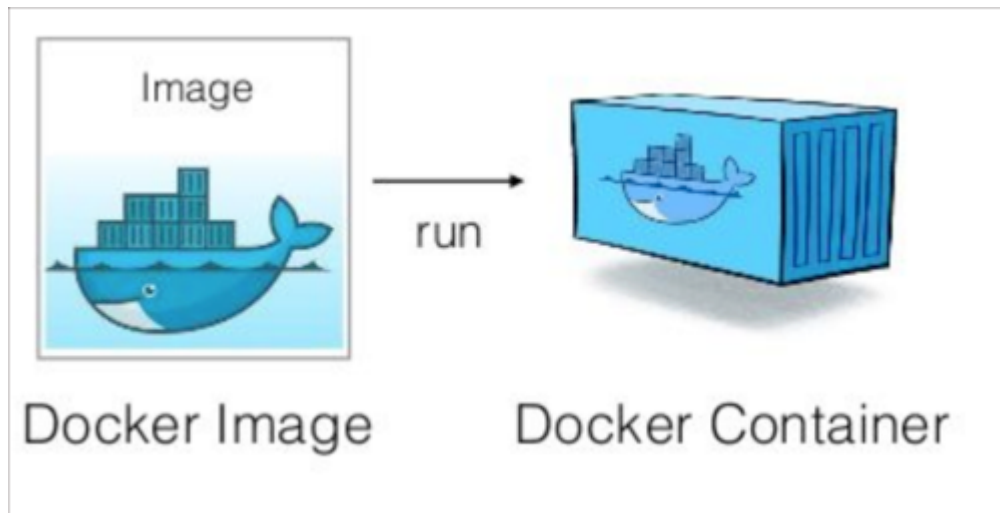
```
docker run mon-image-rssp
```

1. Vous devriez maintenant voir le message "Bonjour, c'est l'image de RSSP" s'afficher dans le terminal.
- Pour supprimer une image Docker, vous pouvez utiliser la commande `docker rmi`.

```
docker rmi myimage
```

Vous avez créé et exécuté avec succès une image Docker qui affiche un message simple. Bien sûr, cela ne représente qu'un exemple très simple, mais vous pouvez utiliser ce modèle pour créer des images plus complexes et utiles. Pour plus de détails vous pouvez visiter le site suivant: https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

▼ Étape 5 — Travailler avec des images Docker



- Les conteneurs Docker sont construits à partir d'images Docker. Par défaut, Docker tire ces images de Docker Hub,
Tout le monde peut héberger ses images Docker sur Docker Hub, de sorte que la plupart des applications et des distributions Linux dont vous aurez besoin y auront des images hébergées.
- Pour vérifier si vous pouvez accéder et télécharger des images de Docker Hub, tapez :

```
docker run hello-world
```

La sortie indiquera que Docker fonctionne correctement :

```
hilali@hilali-VirtualBox:~$ 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/
```

Au départ, Docker n'a pas pu trouver l'image `hello-world` localement, il a donc téléchargé l'image depuis Docker Hub, qui est le référentiel par défaut. Une fois l'image téléchargée, Docker a créé un conteneur à partir de l'image et l'application dans le conteneur s'est exécutée, affichant le message: "Hello from Docker!".

- Vous pouvez rechercher des images disponibles sur Docker Hub en utilisant la commande `docker` avec la sous-commande `search`. Par exemple, pour rechercher l'image Ubuntu, tapez :

```
docker search ubuntu
```

Le script va parcourir Docker Hub et retourner une liste de toutes les images dont le nom correspond à la chaîne de recherche. Dans ce cas, la sortie sera similaire à celle-ci :

```
hilali@hilali-VirtualBox:~$ docker search ubuntu
```

NAME	DESCRIPTION	STARS	OFFICIAL
automated			
ubuntu	Ubuntu is a Debian-based Linux operating sys...	15897	[OK]
websphere-liberty	WebSphere Liberty multi-architecture images ...	293	[OK]
open-liberty	Open Liberty multi-architecture images based...	59	[OK]
neurodebian	NeuroDebian provides neuroscience research s...	100	[OK]
ubuntu-debootstrap	DEPRECATED; use "ubuntu" instead	51	[OK]
ubuntu-upstart	DEPRECATED, as is Upstart (find other proces...	114	[OK]
ubuntu/nginx	Nginx, a high-performance reverse proxy & we...	86	

- Une fois que vous avez identifié l'image que vous souhaitez utiliser, vous pouvez la télécharger sur votre ordinateur à l'aide de la sous-commande `pull`.

Exécutez la commande suivante pour télécharger l'image officielle d' `ubuntu` sur votre ordinateur

```
docker pull ubuntu
```

- Une fois qu'une image a été téléchargée, vous pouvez alors lancer un conteneur en utilisant l'image téléchargée avec la sous-commande `run`. Comme vous l'avez vu avec l'exemple `hello-world`, si une image n'a pas été téléchargée lorsque `docker` est exécuté avec la sous-commande `run`, le client Docker téléchargera d'abord l'image, puis lancera un conteneur en l'utilisant.

Pour voir les images qui ont été téléchargées sur votre ordinateur, tapez :

```
docker images
```

La sortie ressemblera à ce qui suit :

```
hilali@hilali-VirtualBox:~$ docker images
REPOSITORY    TAG       IMAGE ID      CREATED        SIZE
nginx          latest    6efc10a0510f  2 weeks ago   142MB
ubuntu         latest    08d22c0ceb15  7 weeks ago   77.8MB
hello-world    latest    feb5d9fea6a5  19 months ago 13.3kB
```

Comme vous le verrez plus loin dans ce tutoriel, les images que vous utilisez pour gérer les conteneurs peuvent être modifiées et utilisées pour générer de nouvelles images, qui peuvent ensuite être téléchargées (*poussées* est le terme technique) vers Docker Hub ou d'autres registres Docker.

▼ Étape 6 — Exécution d'un conteneur Docker

Le conteneur `hello-world` que vous avez exécuté à l'étape précédente est un exemple de conteneur qui fonctionne et qui quitte après avoir émis un message de test. Les conteneurs peuvent être beaucoup plus utiles que cela, et ils peuvent être interactifs. Après tout, ils sont similaires aux machines virtuelles, mais ils sont plus économes en ressources.

- À titre d'exemple, exécutons un conteneur en utilisant la dernière image d'Ubuntu. La combinaison des commutateurs `-i` et `-t` vous donne un accès interactif au shell dans le conteneur :

```
docker run -it ubuntu
```

Votre invite de commande devrait changer pour refléter le fait que vous travaillez maintenant à l'intérieur du conteneur et devrait prendre cette forme :

```
hilali@hilali-VirtualBox:~$ docker run -it ubuntu
root@10f112630d86:/#
```

Notez l'identifiant du conteneur dans l'invite de commande. Dans cet exemple, il s'agit de `d9b100f2f636`. Vous aurez besoin de cet ID de conteneur plus tard pour identifier le conteneur lorsque vous voudrez le supprimer.

- Launch Ubuntu in the background:

```
docker run -it -d ubuntu
```

- Vous pouvez maintenant exécuter n'importe quelle commande à l'intérieur du conteneur. Mettons par exemple à jour la base de données des paquets à l'intérieur du conteneur. Vous ne devez pas préfixer une commande avec `sudo`, car vous opérez à l'intérieur du conteneur en tant qu'utilisateur **root** :

```
apt update
```

- Ensuite, installez n'importe quelle application dans le conteneur. Installons Node.js :

```
apt install nodejs
```

- Ceci installe Node.js dans le conteneur à partir du dépôt officiel d'Ubuntu. Une fois l'installation terminée, vérifiez que Node.js est installé :

```
node -v
```

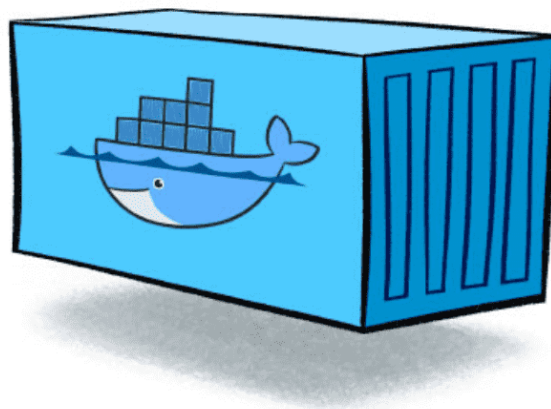
Vous verrez le numéro de version affiché dans votre terminal :

```
root@10f112630d86:/# node -v
v12.22.9
```

Les modifications que vous apportez à l'intérieur du conteneur ne s'appliquent qu'à ce conteneur.

Pour quitter le conteneur, tapez `exit` à l'invite.

▼ Étape 7 — Gestion des conteneurs Docker



- près avoir utilisé Docker pendant un certain temps, vous aurez de nombreux conteneurs actifs (en cours d'exécution) et inactifs sur votre ordinateur. Pour voir les **actifs**, utilisez :

```
docker ps
```

Vous verrez une sortie similaire à celle-ci :

```
hilali@hilali-VirtualBox:~$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
--------------	-------	---------	---------	--------	-------	-------

- Dans ce tutoriel, vous avez lancé deux conteneurs ; un à partir de l'image `hello-world` et un autre à partir de l'image `ubuntu`. Les deux conteneurs ne sont plus actifs, mais ils existent toujours sur votre système.

Pour voir tous les conteneurs, actifs et inactifs, exécutez `docker ps` avec le commutateur `-a` :

```
docker ps -a
```

Vous verrez une sortie semblable à celle-ci :

```
hilali@hilali-VirtualBox:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
833d9b8f0786	ubuntu	"/bin/bash"	About a minute ago	Exited (127) 51 seconds ago		bold_kepler
10f112630d86	ubuntu	"/bin/bash"	6 minutes ago	Exited (0) About a minute ago		heuristic_leakey
a0ef511e8333	hello-world	"/hello"	13 minutes ago	Exited (0) 13 minutes ago		boring_grothendieck
4fad87124a72	ubuntu	"/bin/bash"	3 hours ago	Exited (0) 3 hours ago		magical_grothendieck
c95c9c811dd4	ubuntu	"/bin/bash"	3 hours ago	Exited (137) 3 hours ago		musling_shockley

-

```
docker ps -l
```

```
hilali@hilali-VirtualBox:~$ docker ps -l
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
833d9b8f0786   ubuntu   "/bin/bash"   2 minutes ago   Exited (127) 2 minutes ago   bold_kepler
```

- Pour démarrer un conteneur arrêté, utilisez `docker start`, suivi de l’ID du conteneur ou de son nom. Démarrons le conteneur basé sur Ubuntu avec l’ID de

`833d9b8f0786` :

```
docker start 833d9b8f0786
```

Le conteneur démarrera, et vous pouvez utiliser `docker ps` pour voir son statut

```
hilali@hilali-VirtualBox:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
833d9b8f0786   ubuntu   "/bin/bash"   3 minutes ago   Up 19 seconds   bold_kepler
```

- Pour arrêter un conteneur en cours d’exécution, utilisez `docker stop`, suivi de l’ID ou du nom du conteneur. Cette fois, nous utiliserons le nom que Docker a attribué au conteneur, qui est `bold_kepler` :

```
docker stop bold_kepler
```

- Une fois que vous avez décidé que vous n’avez plus besoin d’un conteneur, retirez-le avec la commande `docker rm`, en utilisant à nouveau l’ID ou le nom du conteneur. Utilisez la commande `docker ps -a` pour trouver l’ID ou le nom du conteneur associé à l’image `hello-world` et supprimez-le.

```
docker rm a0ef511e8333
```

- La commande pour supprimer tous les conteneurs Docker est la suivante:

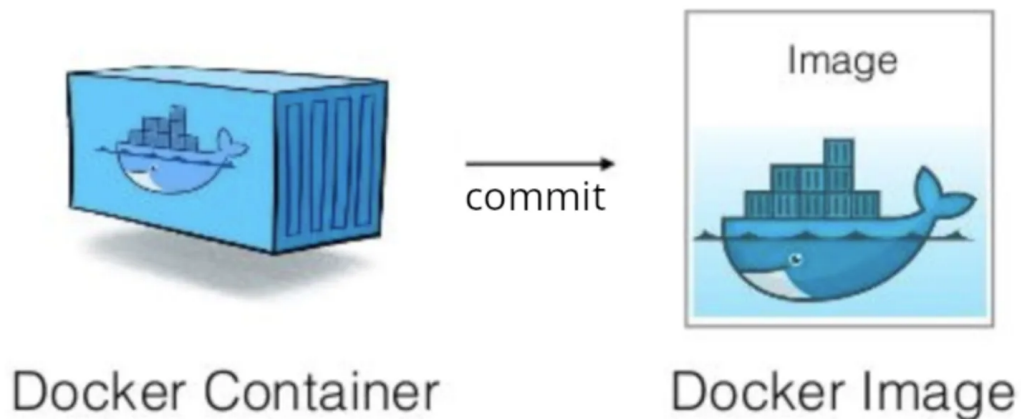
```
docker rm -f $(docker ps -aq)
```

- Si vous souhaitez supprimer uniquement les conteneurs arrêtés, vous pouvez utiliser la commande suivante:

```
docker container prune
```

- Vous pouvez démarrer un nouveau conteneur et lui donner un nom en utilisant le commutateur `--name`. Vous pouvez également utiliser le commutateur `--rm` pour créer un conteneur qui se supprime de lui-même lorsqu'il est arrêté. Voir la commande `docker run help` pour plus d'informations sur ces options et d'autres.

▼ Étape 8 — Transformation d'un conteneur en une image Docker



Après avoir installé Node.js dans le conteneur Ubuntu, vous avez maintenant un conteneur qui s'exécute à partir d'une image.

- Docker permet de créer, modifier et supprimer des fichiers dans un conteneur, mais ces modifications ne s'appliquent qu'à ce conteneur. Pour enregistrer l'état d'un conteneur en tant que nouvelle image Docker, il faut utiliser la commande `docker commit`. Cela permet de réutiliser le conteneur comme base pour de nouvelles images.

```
docker commit -m "What you did to the image" -a "Author Name" container_id repository/new_image_name
```

Le commutateur **-m** est destiné au message de validation qui vous aide, ainsi que les autres, à connaître les modifications que vous avez apportées, tandis que **-a** est utilisé pour spécifier l'auteur.

Le `container_id` est celui que vous avez noté plus tôt dans le tutoriel lorsque vous avez lancé la session interactive de Docker.

À moins de créer des référentiels supplémentaires sur Docker Hub, le `référentiel` est généralement votre nom d'utilisateur Docker Hub.

Par exemple, pour l'utilisateur **sammy**, avec l'ID de conteneur `d9b100f2f636`, la commande serait :

```
docker commit -m "added Node.js" -a "hilali" d9b100f2f636 hilali/ubuntu-nodejs
```

Lorsque vous *validez* une image, la nouvelle image est enregistrée localement sur votre ordinateur. Plus loin dans ce tutoriel, vous apprendrez comment pousser une image vers un registre Docker comme Docker Hub pour que d'autres puissent y accéder.

- L'énumération des images Docker affichera à nouveau la nouvelle image, ainsi que l'ancienne image dont elle est issue :

```
docker images
```

Vous verrez une sortie de ce type :

```
hilali@hilali-VirtualBox:~$ docker images
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
hilali/ubuntu-nodejs latest      66dcac1a9002  36 seconds ago 205MB
nginx                latest      6efc10a0510f  2 weeks ago   142MB
ubuntu               latest      08d22c0ceb15  7 weeks ago   77.8MB
hello-world          latest      feb5d9fea6a5  19 months ago 13.3kB
```

Dans cet exemple, `ubuntu-nodejs` est la nouvelle image, qui a été dérivée de l'image `ubuntu`

existante à partir de Docker Hub. La différence de taille reflète les modifications apportées. Et dans cet exemple, le changement est que NodeJS a été installé. Donc la prochaine fois que vous aurez besoin d'exécuter un conteneur en utilisant Ubuntu avec NodeJS pré-installé, vous pourrez simplement utiliser la nouvelle image.

```
hilali@hilali-VirtualBox:~$ docker run -it hilali/ubuntu-nodejs
root@d5e112225c68:/# node -v
v12.22.9
```

- Vous pouvez également construire des images à partir d'un `Dockerfile`, qui vous permet d'automatiser l'installation de logiciels dans une nouvelle image. Cependant, cela n'entre pas dans le cadre de ce tutoriel.



Partageons maintenant la nouvelle image avec d'autres personnes afin qu'elles puissent créer des conteneurs à partir de celle-ci.

- Tu est besoin d'un compte sur [Docker Hub](#) si vous souhaitez créer vos propres images et les pousser vers Docker Hub.