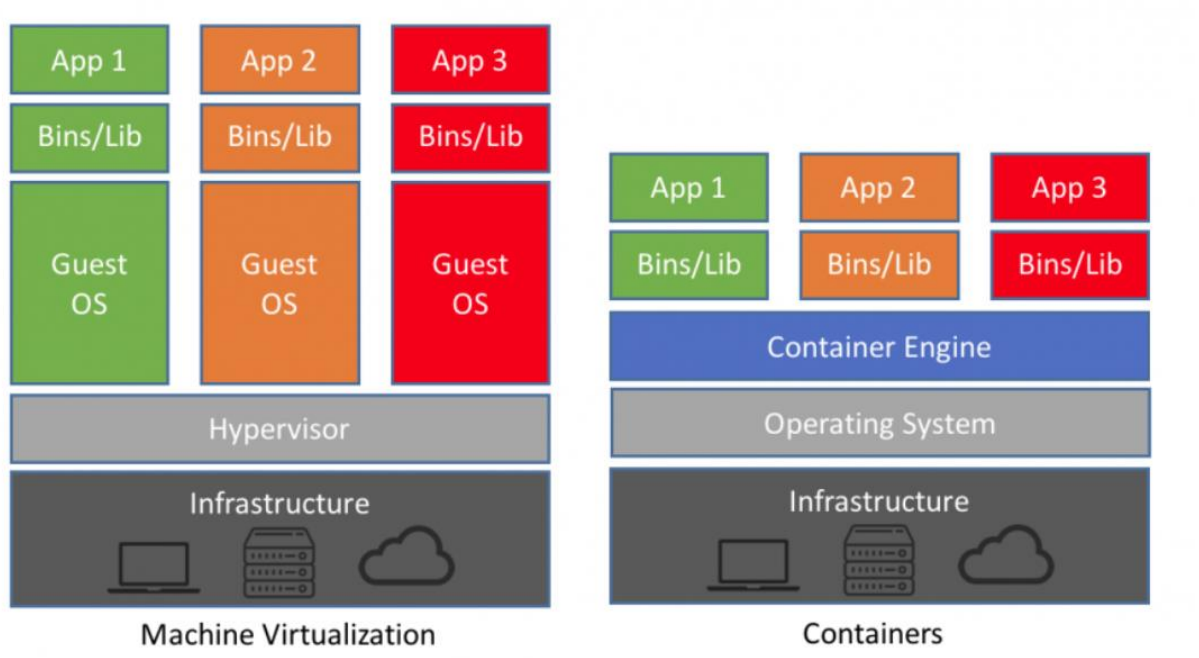
	Virtualisation et Cloud computing **Travaux pratique**	**SE : Windows 10 ou 11** **Enseignante TP : Sana BENZARTI** **Enseignante TP : Manel MILI**
	TP 4 : Installation et Initiation avec Docker	**Année universitaire : 2024-2025** **Niveau : LF3 INFO**

Prérequis :

- Un système Windows
- Ubuntu 22.04 ou Ubuntu 20.04
- Une connexion internet

Comprendre les Nouvelles notions :

- VM : une abstraction complète pour simuler des machines : un processeur, mémoire, appels systèmes, carte réseau, carte graphique, etc.
- Conteneur : un découpage dans Linux pour séparer des ressources (accès à des dossiers spécifiques sur le disque, accès réseau).



Un conteneur est **un groupe de *processus*** associé à un ensemble de permissions sur le système.

1. Avantages de l'utilisation de Docker :
   - Uniformisation face aux divers langages de programmation, configurations et briques logicielles
   - Installation sans accroc et automatisation beaucoup plus facile
   - Permet de simplifier l'intégration continue, la livraison continue et le déploiement continu
   - Rapproche le monde du développement des opérations (tout le monde utilise la même technologie)
   - Permet l'adoption plus large de la logique DevOps (notamment le concept *d'infrastructure as code*)

Partie 1: Installation du docker engine sous Windows: Utiliser Windows Subsystem for Linux (WSL 2)

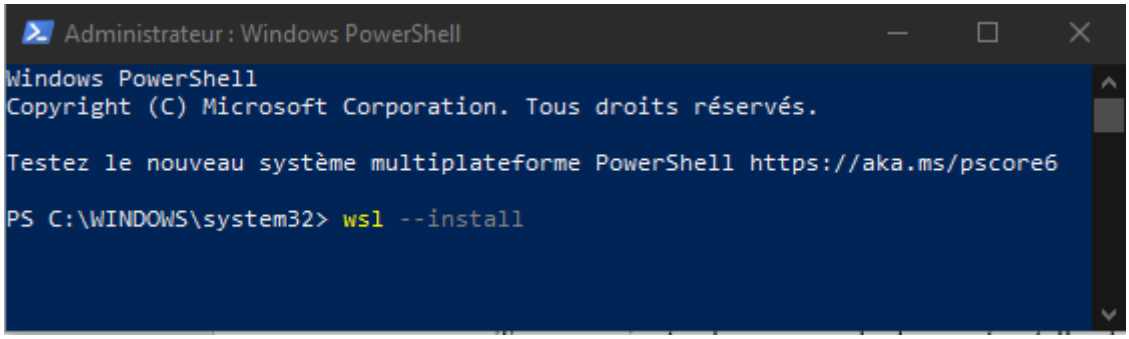
WSL est un sous-système qui permet d'exécuter un noyau Linux directement sur Windows sans avoir besoin d'une machine virtuelle.

Avec **WSL 2**, on peut exécuter un environnement Linux complet et utiliser des distributions populaires comme Ubuntu, Debian, etc.

Étape 1 : Activer WSL2

- Ouvrir **PowerShell** en tant qu'administrateur et exécute cette commande pour installer WSL :

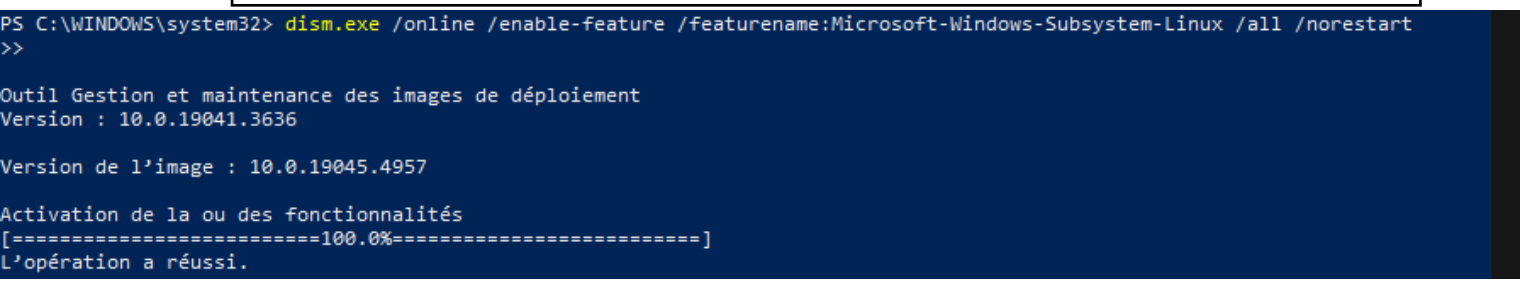
```
wsl -- install
```



Étape 2 : Activer WSL

- Une fois installé, exécuter cette commande pour activer WSL :

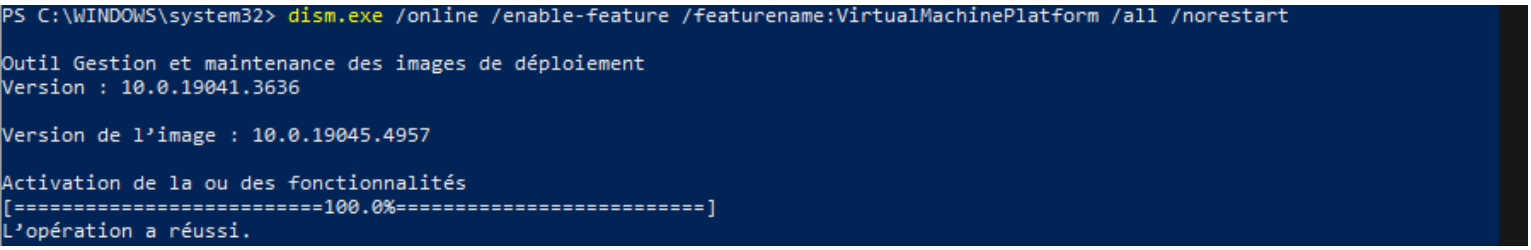
```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
```



Étape 3 : Activer la machine virtuelle requise pour WSL 2:

- Ensuite, exécuter cette commande pour activer la fonctionnalité de virtualisation nécessaire pour WSL 2 :

```
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```



Étape 4 : Redémarrer votre PC

- Redémarrer votre ordinateur pour que les changements prennent effet.

Étape 5 : Installation d'une Distribution Linux

- Si WSL est bien activé, on peut installer une distribution Linux à partir du **Microsoft Store** :

  1. **Ouvrir le Microsoft Store**.
  2. **Chercher une distribution Linux** comme **Ubuntu 22.04 ou Ubuntu 20.04.**
  3. **Installer la distribution** choisie.

- Une fois installée, on pourra accéder à cette distribution via le menu Démarrer.

Étape 6 : Ouvrir Ubuntu sous WSL

- Cliquer sur le bouton **Démarrer** de Windows et taper **Ubuntu 22.04**, puis sélectionner l'application pour l'ouvrir.
- La première fois, essayer d'ajouter un **nom d'utilisateur** et un **mot de passe** :

```
sudo adduser manel
```

Pour **nom d'utilisateur** :

- Il doit être composé de lettres (a-z) ou de chiffres (0-9).
- Il peut contenir des tirets bas (_), mais il ne peut pas commencer par un chiffre ou un caractère spécial.
- Il doit généralement être en minuscules (bien que cela puisse varier en fonction des configurations du système).

- Se connecter avec le nouvel utilisateur

```
su – manel
```

- Ajouter un utilisateur au groupe **`sudo`**
  - Ouvrir maintenant PowerShell et taper ces commandes :

```
wsl -u root
usermod -aG sudo manel
```

L'option `-aG` signifie qu'on ajoute l'utilisateur au groupe sans le retirer des autres groupes auxquels il pourrait appartenir.

  - Vérifier que l'utilisateur a été ajouté

```
groups manel
```

- Déconnexion et reconnexion

```
Exit
wsl -u manel
```

- Vérifier la liste des distributions WSL installées

```
wsl -l -v
```

```
PS C:\WINDOWS\system32> wsl -l -v
  NAME            STATE           VERSION
* Ubuntu-22.04    Running         1
```

- Définir Ubuntu comme distribution par défaut

```
wsl --set-default Ubuntu-22.04
```

- Vérifier la version de WSL

```
wsl --set-version Ubuntu-22.04 2
```

Étape 7 : Mettre à jour votre système d'exploitation

Sous Ubuntu, mettre à jour votre système d'exploitation en tapant la commande suivante :

```
sudo apt-get update
```

Étape 8 : Installer des paquets supplémentaires

```
apt install apt-transport-https ca-certificates curl software-properties-common -y
```

1- apt-transport-https : Ce paquet permet à APT (Advanced Package Tool) d'utiliser le protocole HTTPS pour télécharger des paquets à partir de sources sécurisées.

2- ca-certificates : Ce paquet contient les certificats racine des autorités de certification, nécessaires pour vérifier l'authenticité des certificats SSL/TLS utilisés par les serveurs HTTPS.

3- curl : Curl est un outil en ligne de commande utilisé pour transférer des données à travers différents protocoles. Il est souvent utilisé pour récupérer des fichiers à partir d'URLs, ce qui peut être nécessaire lors de l'installation de certains logiciels.

4- software-properties-common : Ce paquet fournit des scripts et des outils communs pour gérer les sources de logiciels. Il inclut des utilitaires tels que add-apt-repository, qui permet d'ajouter facilement de nouveaux dépôts de logiciels à APT.

5- L'option -y à la fin de la commande indique à APT d'automatiquement répondre "oui" à toutes les questions de confirmation pendant le processus d'installation, ce qui permet une installation

automatique sans intervention de l'utilisateur.

Étape 9 : Ajouter une clé GPG

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

Étape 10 : Ajouter le dépôt Docker

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

1- add-apt-repository : C'est un outil qui permet d'ajouter des dépôts de logiciels à APT (Advanced Package Tool), le gestionnaire de paquets de Debian et d'Ubuntu.

2- deb : Indique qu'il s'agit d'un dépôt contenant des paquets binaires Debian.
3- [arch=amd64] : Spécifie l'architecture des paquets à télécharger (dans ce cas, pour les processeurs 64 bits).
4- https://download.docker.com/linux/ubuntu : L'URL du dépôt Docker pour Ubuntu.
5- focal : C'est le nom de code d'Ubuntu 20.04 LTS. Cela signifie que ce dépôt est destiné à cette version spécifique d'Ubuntu.
6- stable : Indique la branche ou la version du logiciel à utiliser à partir de ce dépôt. Dans ce cas, "stable" signifie que les paquets de la version stable de Docker seront disponibles dans ce dépôt.

Étape 11 : vérifiez s'il y a des paquets prêts à l'installation

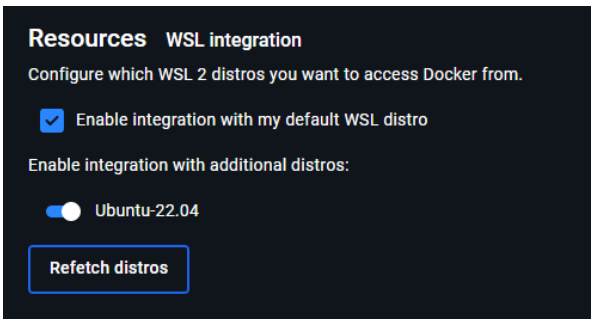
```
apt-cache policy docker-ce
```

Étape 12 : Installer Docker et configurer les paramètres :

- Installer Docker Desktop Installer
- Ouvrir Docker Desktop après l'installation.
- Cliquer sur **Settings** (Paramètres) dans le coin supérieur droit.
- Aller dans **General** (Général) et assurer-vous que l'option **Use the WSL 2 based engine** (Utiliser le moteur basé sur WSL 2) est activée.

Etape 13 : Intégrer Docker à la distribution WSL :

- Dans les **Settings** de Docker Desktop, aller à l'onglet **Resources** (Ressources), puis clique sur **WSL Integration**.
- Activer l'option **Enable integration with my default WSL distro** (Activer l'intégration avec ma distribution WSL par défaut).
- Sélectionne la distribution WSL sur laquelle tu travailles, probablement **Ubuntu** ou une autre distribution. Cocher la case à côté de cette distribution.



- Appliquer les paramètres et redémarrer Docker Desktop**.**

Étape 14 : Vérifier l'état de Docker

- Ouvrir une session WSL **:** Ouvrir ton terminal WSL **(par exemple, Ubuntu sur WSL**), et taper la commande suivante pour vérifier si Docker est opérationnel :

```
docker --version
```

```
manel@Manel-PC:~$ docker --version
Docker version 27.3.1, build ce12230
```

- Vérifier le statut de Docker dans Ubuntu sous WSL

```
systemctl status docker
```

```
manel@Manel-PC:~$ systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2024-10-13 20:52:31 WAT; 6min ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 560 (dockerd)
      Tasks: 10
     Memory: 110.0M
     CGroup: /system.slice/docker.service
             └─560 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Warning: some journal files were not opened due to insufficient permissions.
```

(Vous pouvez redémarrer Docker avec cette commande : sudo service docker start)

Étape 15 : Exécuter votre premier container

- Taper : `docker run hello-world`

```
manel@Manel-PC:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:d211f485f2dd1dee407a80973c8f129f00d54604d2c90732e8e320e5038a0348
Status: Downloaded newer image for hello-world:latest

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

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

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

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

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

Partie 2: Installation du docker engine sous Linux : Ubuntu 20.04 ou 22.04

Étape 1 : Mettre à jour votre système d'exploitation

Mettre à jour votre système d'exploitation en tapant la commande suivante :

```
sudo apt-get update
```

Étape 2 : Installer les packages nécessaires

```
sudo apt-get install ca-certificates curl
```

Cette commande installe les certificats nécessaires pour les connexions sécurisées et l'outil curl pour faciliter les transferts de données via HTTP, HTTPS, et d'autres protocoles.

Etape 3 :

```
sudo install -m 0755 -d /etc/apt/keyrings
```

Cette commande crée le répertoire /etc/apt/keyrings avec les permissions 0755, c'est-à-dire que le propriétaire peut lire, écrire et exécuter dans le répertoire, tandis que le groupe et les autres peuvent seulement lire et exécuter. Ce répertoire est souvent utilisé pour stocker les clés GPG nécessaires pour vérifier l'authenticité des paquets lors de l'installation de logiciels via apt.

Etape 4 :

```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
```

-fsSL :

- -f : Si une erreur HTTP est rencontrée, curl n'affiche rien (il échoue silencieusement).
- -s : Mode silencieux, ce qui signifie que curl n'affiche pas les messages de progression.
- -S : Affiche les messages d'erreur en mode silencieux, utile pour diagnostiquer les erreurs.
- -L : Suit les redirections si l'URL en contient.

Cette commande télécharge la clé GPG de Docker depuis le site officiel et l'enregistre dans le fichier /etc/apt/keyrings/docker.asc. Cette clé sera utilisée par apt pour vérifier la signature des paquets Docker, garantissant ainsi que les paquets proviennent bien de la source de confiance.

Etape 5 :

```
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

Cette commande rend le fichier docker.asc lisible par tous les utilisateurs. Cela est nécessaire pour que tous les utilisateurs du système puissent accéder à la clé GPG lors de la vérification des paquets Docker, ce qui est particulièrement important lorsque apt utilise cette clé pour vérifier les signatures des paquets.

Etape 6 :

```
# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

    - arch=$(dpkg --print-architecture) : Détecte l'architecture de la machine (par exemple, amd64 ou arm64), ce qui permet de télécharger les paquets appropriés.
    - signed-by=/etc/apt/keyrings/docker.asc : Indique que les paquets de ce dépôt doivent être vérifiés avec la clé GPG stockée dans /etc/apt/keyrings/docker.asc.
- https://download.docker.com/linux/ubuntu : URL du dépôt Docker.
- $(. /etc/os-release && echo "$VERSION_CODENAME") :
    - Exécute un sous-shell pour obtenir le nom de code de la version d'Ubuntu (comme focal pour Ubuntu 20.04, jammy pour 22.04).
    - /etc/os-release contient des informations sur la version de la distribution.
- stable : Spécifie la branche stable du dépôt Docker.
- | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null :
- | : Redirige la sortie de la commande echo vers la commande suivante.
- sudo tee /etc/apt/sources.list.d/docker.list : Écrit le contenu généré par echo dans le fichier /etc/apt/sources.list.d/docker.list avec les droits sudo.
    - L'utilisation de tee permet d'écrire le fichier tout en affichant la sortie, mais ici la sortie est redirigée vers /dev/null.
- > /dev/null : Empêche l'affichage de la sortie dans le terminal, pour une exécution plus silencieuse.

Etape 7 :

```
sudo apt-get update
```

Vous pouvez vérifier s’il y a des paquets prêts a l’installation en tapant la commande

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

Etape 8 :

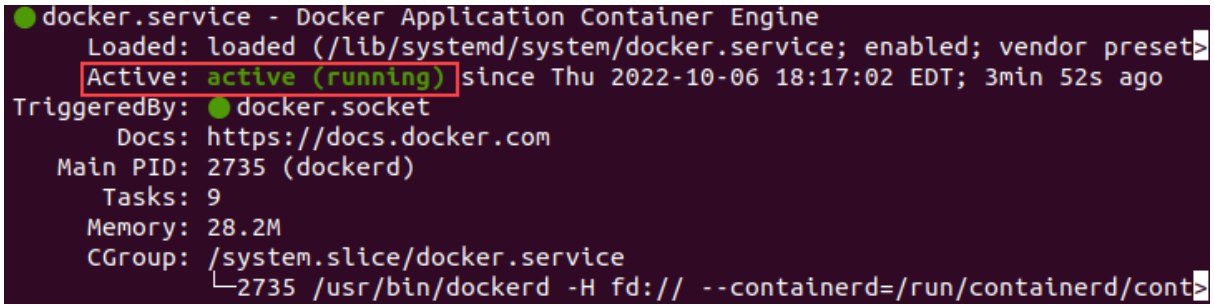
```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

Les paquets à installer sont :

- docker-ce (Docker Community Edition) : Le moteur Docker principal, qui permet de créer, déployer et exécuter des conteneurs.
- docker-ce-cli : L'interface en ligne de commande pour Docker, qui permet de gérer les conteneurs, les images et les réseaux Docker.
- containerd.io : Le runtime de conteneur utilisé par Docker pour exécuter et gérer les conteneurs. C'est une couche essentielle qui gère le cycle de vie des conteneurs.
- docker-buildx-plugin : Un plugin qui améliore la commande docker build en ajoutant des fonctionnalités avancées pour la construction d'images Docker, comme le support multi-plateforme.
- docker-compose-plugin : Permet d'utiliser docker compose, un outil pour définir et exécuter des applications multi-conteneurs à partir de fichiers YAML.

Étape 9 : Vérifier l'état de Docker

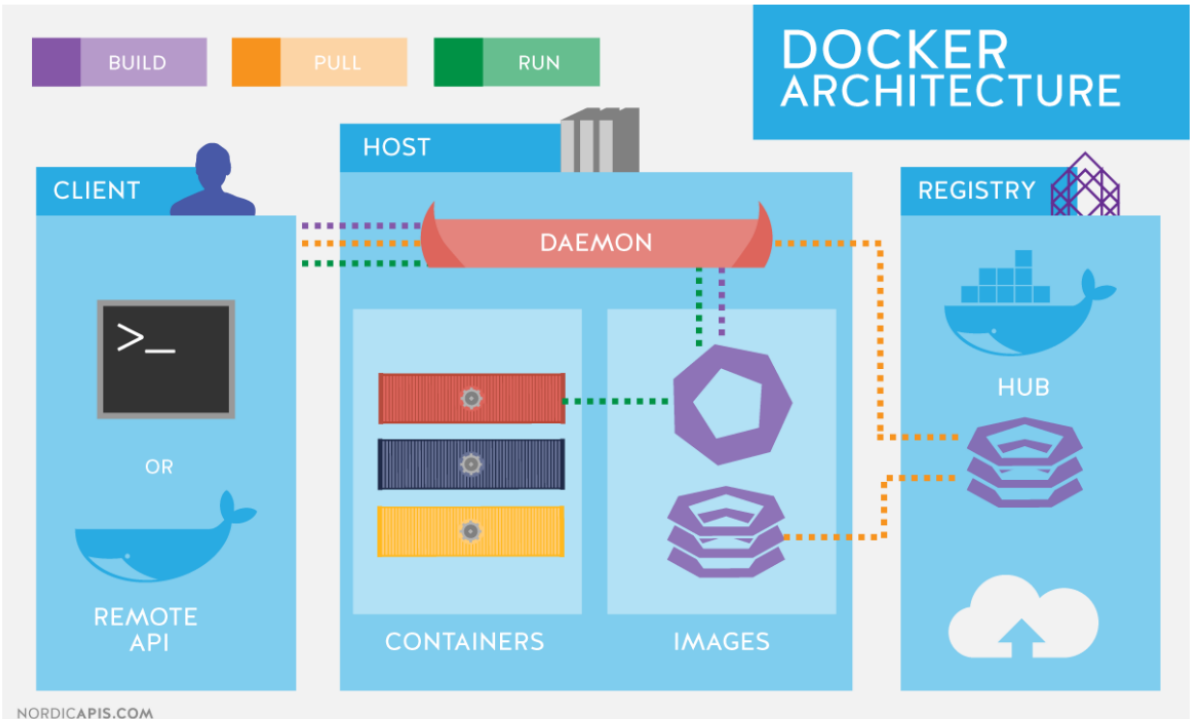
```
sudo systemctl status docker
```



Exécuter votre premier container en tapant : **docker run hello-world**

```
root@sonson-VirtualBox:/home/sonson# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:cc15c5b292d8525effc0f89cb299f1804f3a725c8d05e158653a563f15e4f685
Status: Downloaded newer image for hello-world:latest

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

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

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

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

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

Partie 3 : Création d'une application dans Docker



1. Créer un dossier portant votre prénom et contenant 2 fichiers (Dockerfile et fichier main.py).
   - Dockerfile
   - main.py

2. Modifier **main.py** avec le code ci-dessous.

```
#!/usr/bin/env python3

print("Je suis étudiant(e) ISIMM et c'est mon premier test python avec docker")
```

3. Modifier **Dockerfile** avec les commandes ci-dessous.

```
FROM python:latest
COPY main.py /
CMD [ "python", "./main.py" ]
```

4. Créer une image Docker.
Une fois que vous avez créé et édité le fichier main.py et le Dockerfile, créez votre image pour contenir votre application.

```
$ docker build -t python-test .
```

5. Exécuter l'image Docker
Une fois l'image créée, votre code est prêt à être lancé.

```
$ docker run python-test
```

6. Modifier le fichier main.py en ajoutant la ligne suivante :

```
A=22+4
print(A)
```

7. Transférer une image vers Docker Hub
- Créer un compte sur Docker Hub.
- Cliquer sur le bouton « repository », mettre le nom du fichier.

8. Tester les commandes suivantes :

```
docker images
docker ps -a
```

9. La syntaxe pour baliser l'image est :

```
docker tag  python-test <your dockerhub username>/python-test:latest

par exemple :
 docker tag  python-test <your dockerhub username>/python-test:latest
```

10. Transférer l'image vers le référentiel Docker Hub via la commande push

```
$ docker push <your dockerhub username>/python-test
```

11. Récupérer et exécuter l'image à partir de Docker Hub

- Supprimer toutes les versions de l'image python-test du système local. Utiliser l'ID de l'image pour la suppression.

```
$ docker rmi -f  ID_image
```

- Exécuter l'image :

```
$ docker run <your dockerhub username>/python-test
```

Qu'est-ce que vous remarquez ? Interprétez.

Partie 3 : Transférer une image apache vers dockerhub

Créer un dossier « apache » sous le répertoire qui porte votre prénom et écrire les commandes suivantes:

```
FROM debian:latest

RUN apt-get -yqq update && apt-get install -yqq apache2
WORKDIR /var/www/html

ENV APACHE_RUN_USER www-data
ENV APACHE_RUN_GROUP www-data
ENV APACHE_LOG_DIR /var/log/apache2
ENV APACHE_PID_FILE /var/run/apache2.pid
ENV APACHE_RUN_DIR /var/run/apache2
ENV APACHE_LOCK_DIR /var/lock/apache2

RUN mkdir -p $APACHE_RUN_DIR $APACHE_LOCK_DIR $APACHE_LOG_DIR
ENTRYPOINT [ "/usr/sbin/apache2" ]
CMD ["-D", "FOREGROUND"]
EXPOSE 80
```

Créer votre image :

```
$ docker build -t your dockerhub username/apache .
```

3. Démarrer le container :

```
$ docker run -d -p 80 --name=apache your dockerhub username/apache
```

4. Afficher le port :

```
$ docker port apache 80
```

5. Tester le serveur web en ouvrant votre navigateur : localhost :numero_port

6. Modifier le numéro de port 1111. Vous devez arrêter et supprimer le container apache puis le redémarrer.

7. Sous le répertoire **apache**, créer un répertoire nommé **website**. Mettre un fichier **index.html** sous le répertoire website comportant les lignes suivantes :

```
<html>
Bienvenue sur ma page web. Je suis etudiant(e) ISIMM.
</html>
```

8. Démarrer votre container en montant le répertoire apache :

```
$ docker rm -f apache

$ docker run -d -p 1111:80 –v /votre_chemin/apache/website:/var/www/html --name=apache your dockerhub username/apache
```

9. Faire monter votre image locale vers docker hub avec la commande push. (docker logout, docker build, docker tag, docker login, docker push).

10. Accéder à la page web via l'adresse IP. Chercher l'adresse IP du container « your dockerhub username/apache » en tapant :

```
$ docker inspect <container id> | grep "IPAddress"
```

```
root@sonson-VirtualBox:/home/sonson/Bureau/apache# docker inspect 25a128159ba2
| grep "IPAddress"
            "SecondaryIPAddresses": null,
            "IPAddress": "172.17.0.2",
                    "IPAddress": "172.17.0.2",
```