

Mise En place Et Manipulation d'un Conteneur de Docker en Azure Cloud

"déploiement d'un projet Python Sous Docker"

Table des matières

- [1\)Définition de Conteneurisation](#)
- [2\) Comparaison des principaux logiciels de virtualisation de Container](#)
- [3\)Définition de docker](#)
- [4\)Définition d'un conteneur](#)
- [5\)Pourquoi utiliser Docker ?](#)
- [6\)Les concepts de Docker](#)
- [7\)Conteneurs et machines virtuelles](#)
- [8\)Avantages de l'utilisation des conteneurs par rapport aux machines virtuelles](#)
- [9\)Installer Docker Desktop](#)
- [11\) les Commandes d'utilisation quotidienne de Docker](#)
- [12\)Création de notre première application Docker Sous Azure Cloud](#)
- [13\)Conclusion](#)

1)La conteneurisation, qu'est-ce que c'est ?

La conteneurisation consiste à rassembler le code du logiciel et tous ses composants (bibliothèques, frameworks et autres dépendances) de manière à les isoler dans leur propre « conteneur ». Le logiciel ou l'application dans le conteneur peut ainsi être déplacé et exécuté de façon cohérente dans tous les environnements et sur toutes les infrastructures, indépendamment de leur système d'exploitation. Le conteneur fonctionne comme une sorte de bulle, ou comme un environnement de calcul qui enveloppe l'application et l'isole de son entourage. C'est en fait un environnement de calcul portable complet. Avec les conteneurs, vous n'avez plus besoin de coder pour une plateforme ou un système d'exploitation en particulier, une méthode qui complique le déplacement des applications étant donné que le code n'est pas toujours compatible avec le nouvel environnement. De plus, ces transferts génèrent souvent des bogues, des erreurs et des problèmes qui font perdre du temps, diminuent la productivité et engendrent une grande frustration.

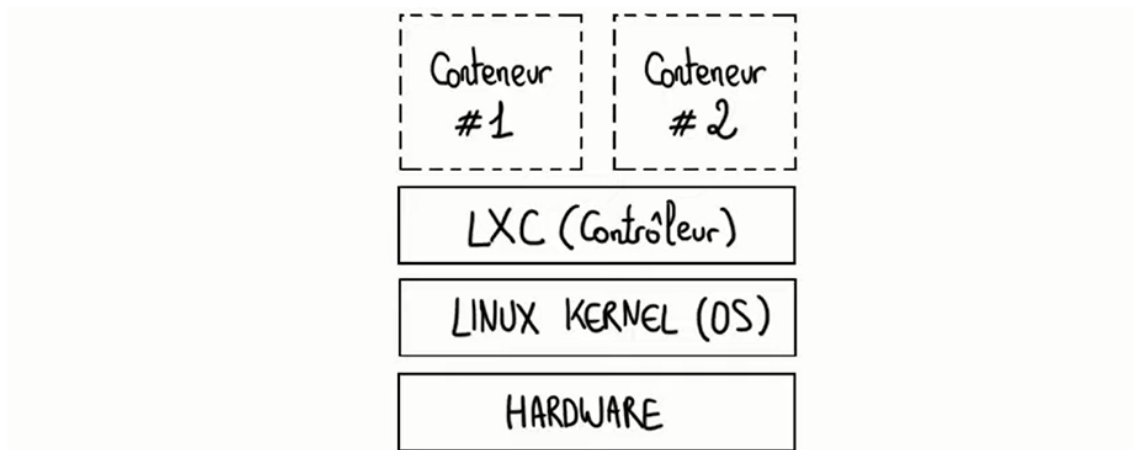
Les Avantages du conteneurisation

La légèreté ou portabilité des conteneurs découle de leur capacité à partager le noyau du système d'exploitation de la machine hôte. Ils n'ont pas besoin d'un système d'exploitation propre et les applications peuvent s'exécuter de la même manière sur toutes les infrastructures (systèmes clouds et même machines virtuelles) De même, avec les conteneurs nous pouvons utiliser les mêmes outils dans plusieurs environnements hôtes, ce qui simplifie sensiblement le développement et le déploiement des applications conteneurisées sur différents systèmes d'exploitation

Comparaison des principaux logiciels de virtualisation de Container

ETUDE DE CAS DE (DOCKER, ROCKET , RUNC , LXC , OPENSTACK)

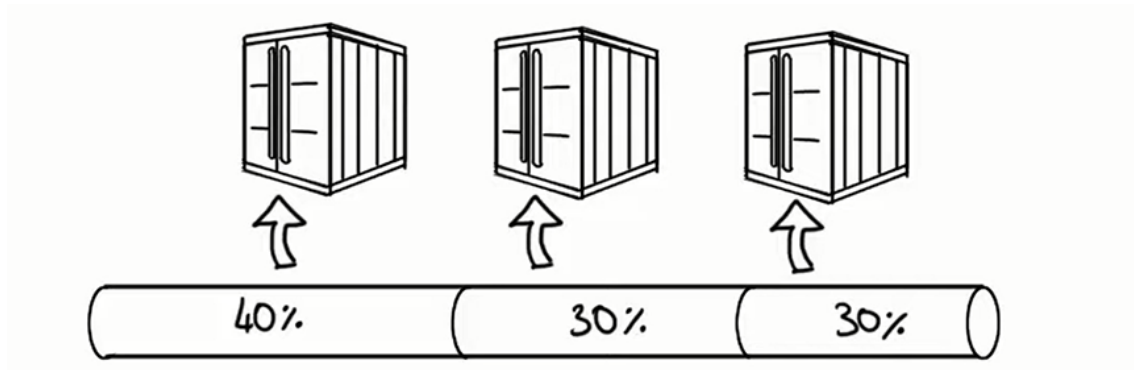
La virtualisation par conteneur se base sur la virtualisation LXC (Linux Containers) Il s'agit d'une méthode de cloisonnement au niveau de l'OS.



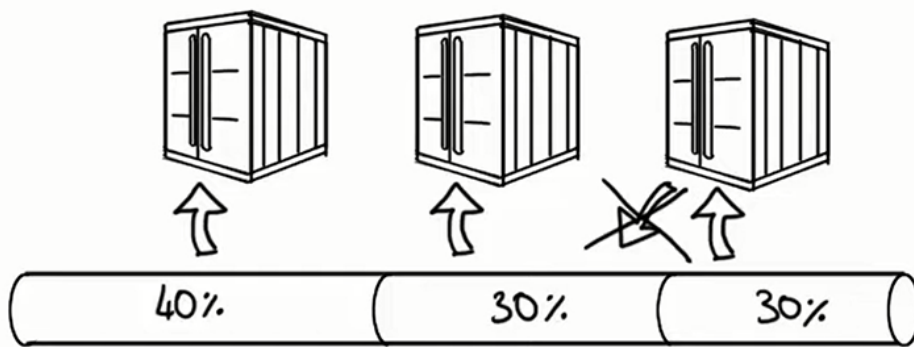
Le principe est de faire tourner des environnements Linux isolés les uns des autres dans des conteneurs partageant le même noyau, contrairement aux machines virtuelles traditionnelles un conteneur n'inclut pas d'OS puisqu'il s'appuie sur les fonctionnalités de l'OS de la machine hôte. Les conteneurs accèdent à l'OS hôte de manière totalement isolée.

Le conteneur virtualise l'environnement des exécutions comme le processeur, la mémoire vive ou le système de fichiers (et ne virtualise donc pas la machine) c'est pour cela que l'on parle de conteneurs (et non de VM).

LXC repose principalement sur deux fonctionnalités Linux :

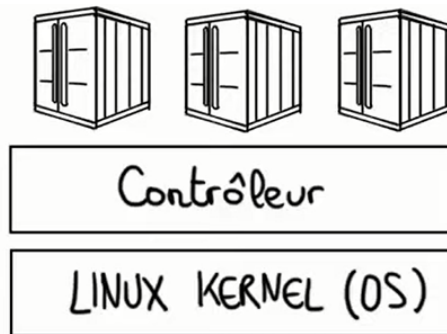


- la fonctionnalité des Cgroups (Control Groups) : permet de limiter et d'isoler l'utilisation des ressources



- La fonctionnalité de cloisonnement des espaces de nommage(namespace) permet d'empêcher qu'un groupe puisse voir les ressources des autres groupes

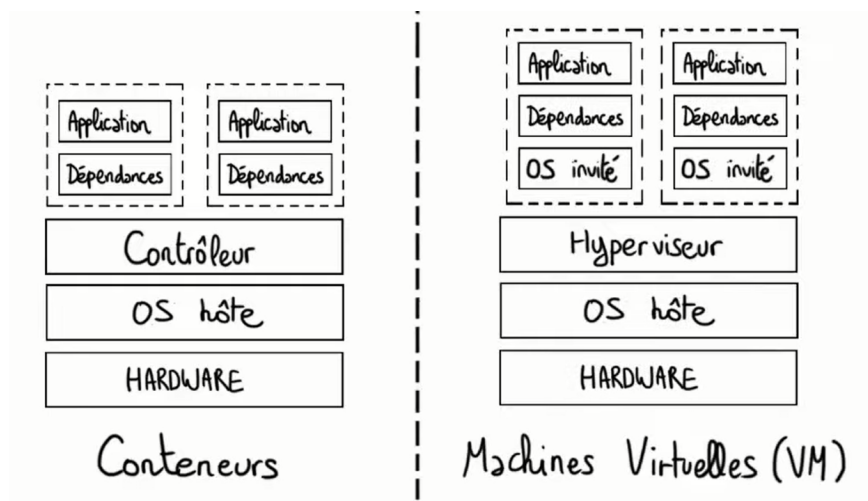
La virtualisation par conteneurs est aussi caractérisée par la couche intermédiaire du **contrôleur**.



Le contrôleur gère un ensemble de fonctionnalités pour les conteneurs:

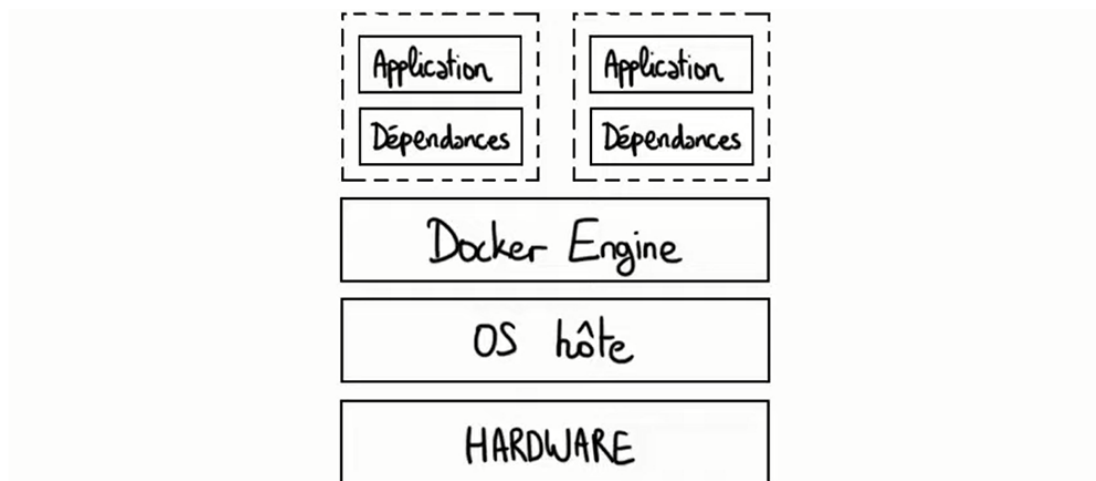
- les interactions des conteneurs avec l'OS
- la sécurité par la gestion de privilèges et de ressources
- La scalabilité (la duplication, la suppression des conteneurs)
- L'accessibilité des conteneurs à travers la gestion des API et CLI
- La portabilité ou la migration de conteneurs

la différence entre un conteneur une VM classique:



Une machine virtuelle (VM) recrée intégralement un OS complet, ses pilotes, ses fichiers, binaires ou bibliothèque ainsi que l'application elle-même. Le conteneur n'embarque pas d'OS, il est donc plus léger que la VM. Il est plus facile à migrer, télécharger et est plus rapide à sauvegarder et restaurer. La virtualisation par conteneur permet aussi aux serveurs d'héberger beaucoup plus de conteneurs que s'il s'agissait de VM.

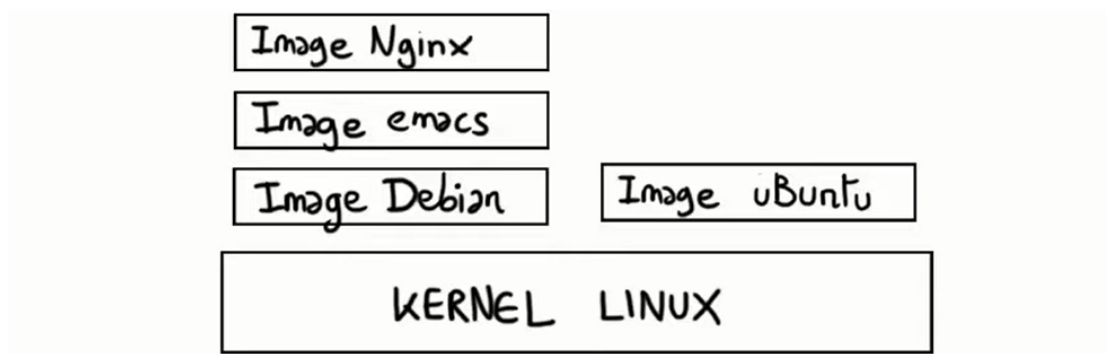
Docker Docker est une technologie de virtualisation par conteneurs reposant sur LXC, il permet de créer des conteneurs qui vont uniquement contenir des applications avec leurs dépendances et ils permettent d'embarquer des applications afin de les exécuter au sein de l'OS hôte mais de manière virtuellement isolée.



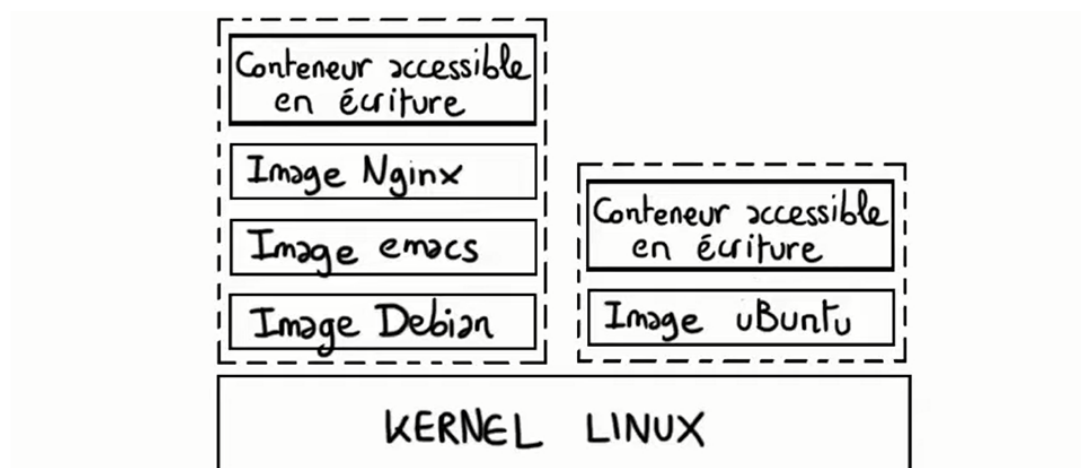
- **Docker Engine** ou moteur Docker fait tourner les conteneurs et joue le rôle de contrôleur

Docker fonctionne sur une architecture client-serveur, le **client Docker** communique avec le **docker daemon** qui fait tourner le Docker Engine. Le client Docker et le daemon Docker peuvent tourner sur la même machine comme sur des machines différentes.

Les images Docker



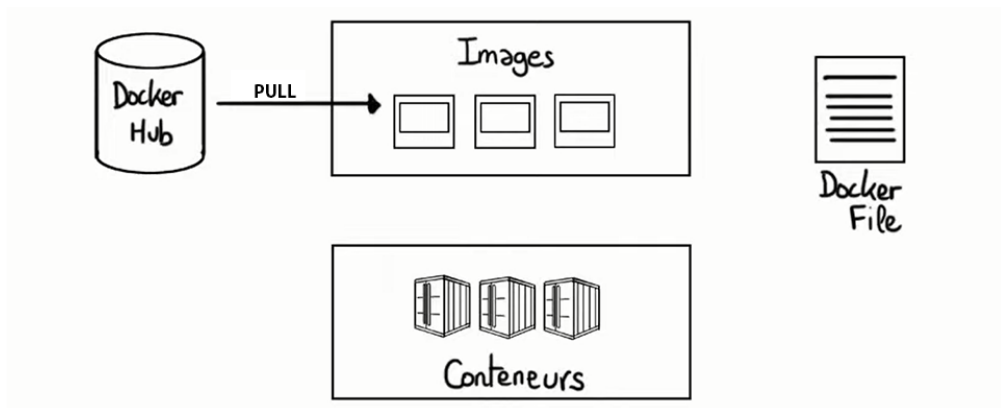
L'environnement d'exécution d'un conteneur peut nécessiter l'empilement de plusieurs images l'image Docker est un template prêt-à-l'emploi avec des instructions pour la création de conteneurs On trouve à la base les composants nécessaires à Docker qui sont fournis par le noyau Linux de l'hôte



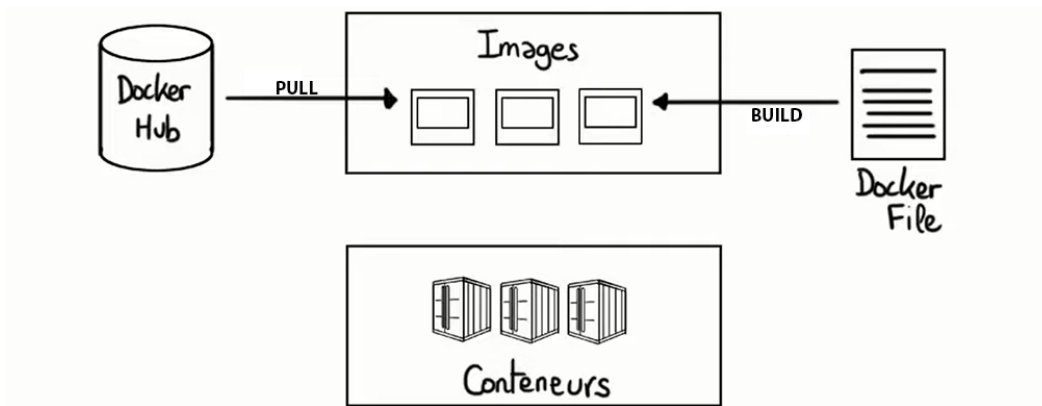
Note : une image ne peut être modifiée directement, elle est toujours en lecture seule La dernière couche qui est la partie du conteneur accessible en écriture et qui contiendra donc toutes les modifications apportées à l'application.

Une image peut être construite à partir d'un **Docker file** ou bien d'une image existante pour être récupérés sur un registre Docker, Ces registres sont accessibles depuis le **Docker Hub** qui est un dépôt public d'image, Pour comprendre comment tous ces éléments fonctionnent ensemble:

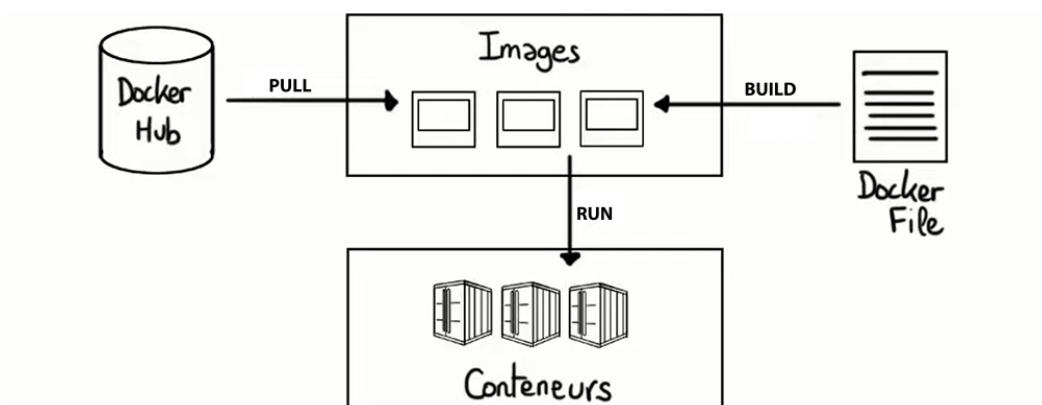
- Pour créer un conteneur nous avons la possibilité d'utiliser une image existante que nous reprenons du **registre Docker**



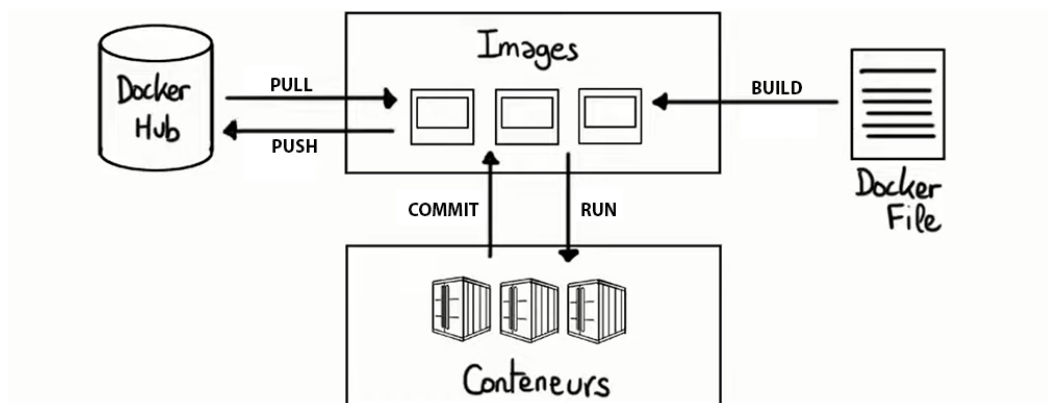
- On peut aussi créer l'image via le **Docker file**
- *Docker File** est un fichier qui permet de construire une image Docker étape par étape et ceci de façon automatisée.



- Le **Docker engine** exécute ensuite cette image pour créer un conteneur



- Une fois que ces nouveaux conteneurs est créé, en peut faire une nouvelle image que nous faisons stocker dans un registre



Rocket RKT (prononcé "rocket") est une CLI permettant d'exécuter des conteneurs d'applications sur une plate-forme Linux. Son objectif principal est d'être sécurisé, rapide et composable, et il est donc conçu comme tel et basé sur des normes. Il est développé avec le précepte "secure-by-default", et se compose d'une grande variété de fonctionnalités de sécurité essentielles. Il met en œuvre la spécification app, aide la spécification Container Networking Interface, et peut exécuter des images Docker et OCI.

- Rocket est un nouveau runtime conteneur qui représente une autre possibilité ou un autre choix que **Docker runtime**. Il est également conçu pour les environnements de serveur avec les exigences les plus strictes en matière de sécurité, de composabilité, de vitesse et de production.

Le logiciel se compose de deux éléments, chacun d'entre eux étant un outil en ligne de commande simple et autonome.

- **Actool**: C'est le premier composant qui administre la construction des conteneurs. Il gère même la validation et la découverte des conteneurs.
- **Rkt**: Il est nommé ainsi, car toutes les principales commandes UNIX utilisées dans Rocket ont trois lettres. Il aide à prendre en charge la récupération et l'exécution des images de conteneurs.

Contrairement à l'approche de Docker, Rocket n'implique pas de daemon extérieur et ces outils ne sont pas seulement une interface utilisateur pour converser avec un autre serveur. Chaque fois que vous demandez à rkt d'exécuter un conteneur, il le fait sans délai dans le cadre de son propre Arbre de processus et de son Cgroup. Il a été dit que le modèle de processus de Docker est fondamentalement défectueux car il permet à tout de s'exécuter via un démon central. Maintenant, Rocket tente d'accomplir et de construire des choses différentes de Docker dans de nombreux aspects. Voici les points clés qui ont été gardés à l'esprit pour construire un conteneur d'exécution sans faille.

- **Composition** : Tous les outils utilisés pour télécharger, installer et exécuter les conteneurs doivent être indépendants et composables.
- **Sécurité** : Pour fournir une meilleure sécurité, l'isolation doit être pluggable, avec audit d'image, identité d'application et primitives cryptographiques pour une confiance forte.
- **Distribution d'images** : La découverte d'images de conteneurs doit être simple, faciliter un espace de noms fédéré et distribué avec des protocoles alternatifs

enfichables tels que BitTorrent et des déploiements vers des environnements privés sans la nécessité d'un registre.

- **Ouvert** : Le format et le temps d'exécution doivent être bien spécifiés et développés par une communauté, ce qui permet aux implémentations indépendantes des outils d'être cohérentes.

Compraison de LXC/LXD et OpenStack

C'est quoi LXC et LXD?

LXC:

LXC est une solution de virtualisation open source basée sur la capacité du noyau Linux à faire fonctionner des environnements isolés. Les conteneurs partagent le même noyau mais l'utilisation des processeurs, mémoire vive, système de fichier... est isolée les uns des autres. Il s'agit donc d'une méthode de virtualisation légère puisque la machine elle-même n'est pas virtualisée. On parle alors de conteneur plutôt que de machine virtuelle.

LXD:

LXD est l'hyperviseur de LXC, donc LXC va nous permettre de créer rapidement des machines afin de pouvoir les configurer. Il s'agit donc d'un outil particulièrement utile à la fois pour le déploiement mais aussi les tests.

C'est quoi OpenStack?

OpenStack est un ensemble de logiciels open source destinés à la création d'un environnement Cloud et il permet de créer et gérer des clouds privés ou publics à partir de pools de ressources virtuelles. Les outils (ou « projets ») qui constituent la plateforme OpenStack assurent les principaux services de cloud computing, à savoir, le calcul, la mise en réseau, le stockage, la gestion des identités et la gestion des images. La dizaine de projets restants, disponibles en option, peuvent également être groupés pour créer des clouds uniques.

Dans le cadre de la virtualisation, les ressources (stockage, processeur, RAM, etc.) sont dissociées de divers programmes de fournisseur, séparées par un hyperviseur, puis distribuées selon les besoins. OpenStack s'appuie sur des interfaces de programmation d'application (API) pour repousser les limites de l'abstraction de ces ressources virtuelles en les répartissant dans des pools individuels, qui pilotent des outils de cloud computing standard avec lesquels les administrateurs et les utilisateurs interagissent directement.

RunC

Runc est en réalité la partie de Docker qui servait à créer un conteneur, Docker n'a fait que la "donner" à l'OCI. D'où le fait que l'on considère que Docker a gagné cette guerre contre CoreOS.

Runc est la container runtime la plus utilisée. Bien qu'elle soit basée sur les travaux de Docker, elle suit en réalité la spécification de l'OCI.

Runc est assez bas niveau et se charge uniquement de démarrer un conteneur, il ne prend pas en charge les activités de plus haut niveau comme gérer les images, les pull

et ne fournit pas d'API. Ce sont elles qui manipulent les namespaces et les cgroups sur lesquels se basent les conteneurs. Un développeur ou un sysadmin ne va que très peu interagir avec elles, ce sont donc les container runtimes de plus au niveau qui vont s'en charger.

3)Qu'est-ce qu'un "Docker" ?

Docker est un ensemble de produits "platform as a service" qui utilisent la virtualisation au niveau du système d'exploitation pour fournir des logiciels dans des paquets appelés conteneurs. Les conteneurs sont isolés les uns des autres et regroupent leurs propres logiciels, bibliothèques et fichiers de configuration ; ils peuvent communiquer entre eux par des canaux bien définis. Tous les conteneurs sont exécutés par un seul noyau de système d'exploitation et sont donc plus légers que les machines virtuelles.

2)Qu'est-ce qu'un "conteneur" ?

Le conteneur Docker est une unité standardisée qui peut être créée pour déployer une application ou un environnement particulier. Il peut s'agir d'un conteneur Ubuntu, d'un conteneur CentOS, etc. pour répondre à toutes les exigences du point de vue du système d'exploitation. Il peut également s'agir d'un conteneur orienté application comme un conteneur CakePHP ou un conteneur Tomcat-Ubuntu.

3)Pourquoi utiliser Docker ?

Docker nous fournit des conteneurs. Et la conteneurisation consiste en un environnement d'exécution complet, une application, toutes ses dépendances, bibliothèques, binaires et fichiers de configuration nécessaires à son exécution, regroupés en un seul paquet. Chaque application s'exécute séparément des autres. Docker résout le problème des dépendances en conservant les dépendances à l'intérieur des conteneurs. Il unit les développeurs contre les dépendances de leur projet.

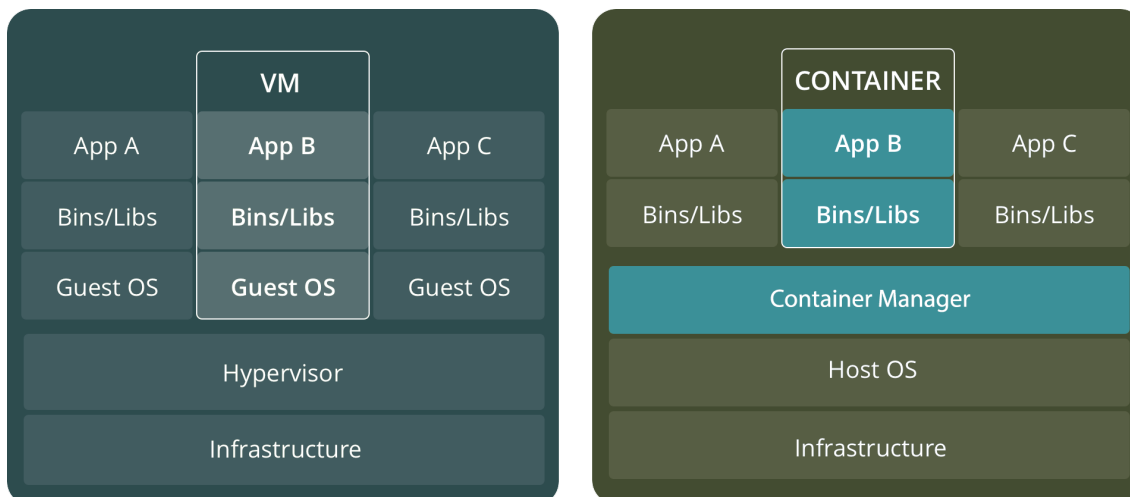
4)Les concepts de Docker :

Docker est une plateforme permettant aux développeurs et aux administrateurs système de créer, partager et exécuter des applications avec des conteneurs. L'utilisation de conteneurs pour déployer des applications est appelée conteneurisation.

La conteneurisation est de plus en plus populaire car les conteneurs sont:

- **Souple** : Même les applications les plus complexes peuvent être conteneurisées.
- **Léger** : Les conteneurs exploitent et partagent le noyau de l'hôte, ce qui les rend beaucoup plus efficaces en termes de ressources système que les machines virtuelles.
- **Portable** : Vous pouvez construire localement, déployer dans le nuage et exécuter partout.
- **Couplage lâche** : Les conteneurs sont très autonomes et encapsulés, ce qui vous permet de remplacer ou de mettre à niveau l'un d'entre eux sans perturber les autres.
- **Évolutif** : Vous pouvez augmenter et distribuer automatiquement les répliques de conteneurs dans un centre de données.
- **Sécurisé** : Les conteneurs appliquent des contraintes agressives et des isolations aux processus sans aucune configuration requise de la part de l'utilisateur.

5)Conteneurs et machines virtuelles :



6) Avantages de l'utilisation des conteneurs par rapport aux machines virtuelles :

- Contrairement aux VM qui s'exécutent sur un système d'exploitation invité, à l'aide d'un hyperviseur, les conteneurs Docker s'exécutent directement sur un serveur hôte (pour Linux), à l'aide d'un moteur Docker, ce qui les rend plus rapides et plus légers.
- Les conteneurs Docker peuvent être facilement intégrés par rapport aux VM.
- Avec un système entièrement virtualisé, vous obtenez une plus grande isolation. Cependant, il nécessite plus de ressources. Avec Docker, vous obtenez moins d'isolation. Cependant, comme il nécessite moins de ressources, vous pouvez exécuter des milliers de conteneurs sur un hôte.
- Le démarrage d'une machine virtuelle peut prendre au moins une minute, alors qu'un conteneur Docker démarre généralement en une fraction de seconde.
- Les conteneurs sont plus faciles à quitter qu'une machine virtuelle.
- Contrairement aux machines virtuelles, il n'est pas nécessaire de pré-allouer la RAM. Les conteneurs Docker utilisent donc moins de RAM que les machines virtuelles. Ainsi, seule la quantité de RAM nécessaire est utilisée.

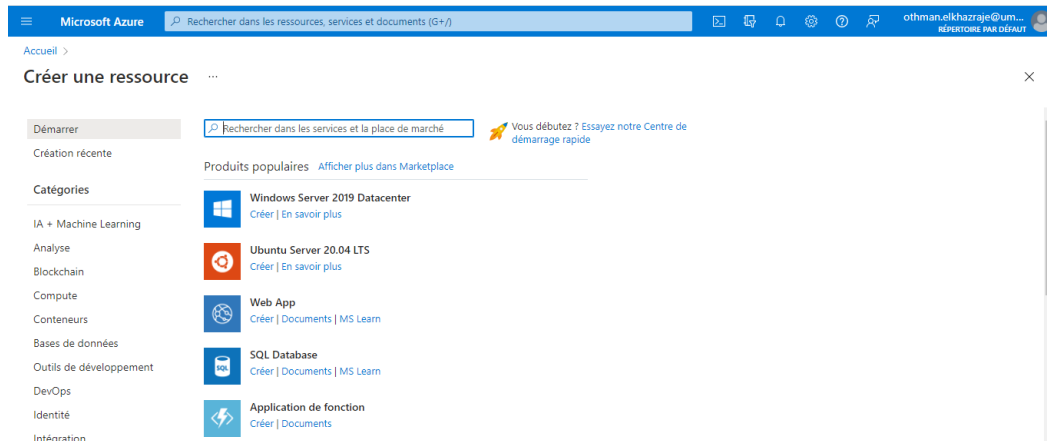
7) Installer Docker Desktop

- For OSX : <https://docs.docker.com/desktop/mac/install/>
- For Windows : <https://docs.docker.com/desktop/windows/install/>

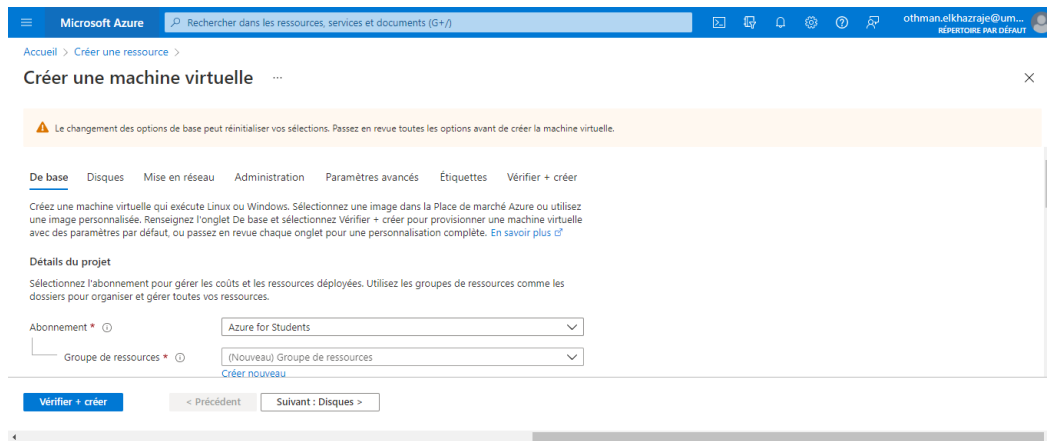
8) Installation du docker sur linux sous azure cloud : "Déploiement d'un Projet sous Docker"

Microsoft Azure est un service de cloud computing de Microsoft. Azure offre une gamme d'options de logiciel en tant que service (SaaS), de plateforme en tant que service (PaaS) et d'infrastructure en tant que service (IaaS) pour déployer des applications et des services sur une infrastructure de centre de données gérée par Microsoft.

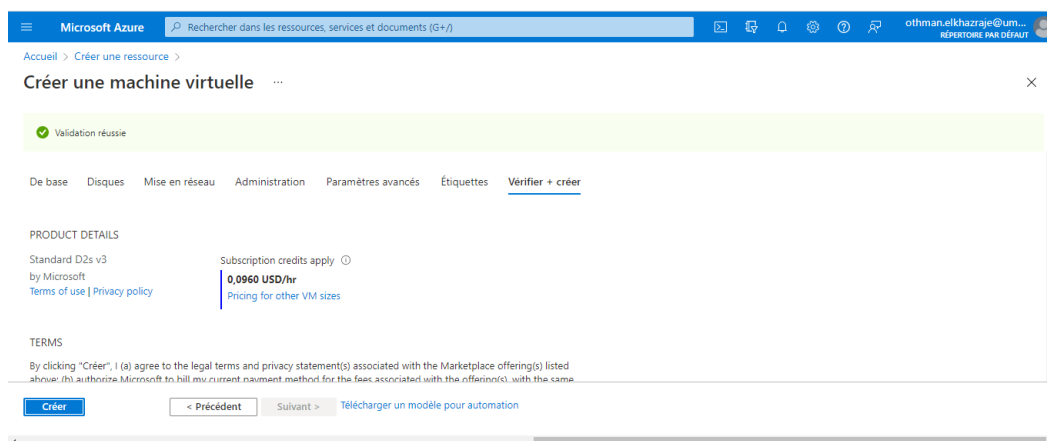
1. les étapes d'installation d'un VM ubuntu sous azure cloud :



l'interface graphique du Azure cloud pour quand peut créer une vm et les différents système d'exploitation on choisi ubuntu



cette interface contient les prérequis du Vm créer



le budget d'ubuntu dans Azure Cloud

Microsoft Azure | Rechercher dans les ressources, services et documents (G+/)

othman.elkhazraje@um...
RÉPERTOIRE PAR DÉFAUT

Accueil > CreateVm-canonical.0001-com-ubuntu-server-focal-2-20220110175846 | Vue d'ensemble

Rechercher (Ctrl+/) < Supprimer Annuler Redéployer Actualiser

Nous aimerions avoir votre avis ! →

*** Le déploiement est en cours

Nom du déploiement : CreateVm-canonical.0001-com-ubuntu-serv... Heure de début : 10/01/2022, 18:02:08
Abonnement : Azure for Students ID de corrélation : 719f4fd8-d404-4514-ae4f-92aa5ad32f09
Groupe de ressources : ips

^ Détails du déploiement (Télécharger)

Ressource	Type	Statut	Détails de l'opération
outhii	Microsoft.Compute/virtualMachines	Created	Détails de l'opération
outhii61	Microsoft.Network/networkInterfaces	Created	Détails de l'opération
ips-vnet	Microsoft.Network/virtualNetworks	OK	Détails de l'opération
outhii-nsg	Microsoft.Network/networkSecurityGro...	OK	Détails de l'opération

le deploiment du VM ubuntu sur le cloud

Microsoft Azure | Rechercher dans les ressources, services et documents (G+/)

othman.elkhazraje@um...
RÉPERTOIRE PAR DÉFAUT

Accueil > ips > outhii

outhii | Connexion

Machine virtuelle

Rechercher (Ctrl+/) < Pour améliorer la sécurité, activez l'accès juste-à-temps sur cette machine virtuelle. →

RDP SSH Bastion

Se connecter avec RDP

Pour vous connecter à votre machine virtuelle par le biais de RDP, sélectionnez une adresse IP, changez éventuellement le numéro de port et téléchargez le fichier RDP.

Adresse IP *

Adresse IP publique (20.36.215.178)

Numéro de port *

3389

Télécharger le fichier RDP

Impossible de se connecter ?

Installer et configurer le Bureau à distance pour se connecter à une machine virtuelle Linux

outhiurdp uthi_key.pem lic_openstack.txt Tout afficher

les outils disponible pour l'accès Réseau soit mode RDP, SSH, Bastion

Microsoft Azure | Rechercher dans les ressources, services et documents (G+/)

othman.elkhazraje@um...
RÉPERTOIRE PAR DÉFAUT

Accueil > ips > outhii

outhii | Connexion

Machine virtuelle

Rechercher (Ctrl+/) < Pour améliorer la sécurité, activez l'accès juste-à-temps sur cette machine virtuelle. →

RDP SSH Bastion

Se connecter avec RDP

Pour vous connecter à votre machine virtuelle par le biais de RDP, sélectionnez une adresse IP, changez éventuellement le numéro de port et téléchargez le fichier RDP.

Adresse IP *

Adresse IP publique (20.36.215.178)

Numéro de port *

3389

Télécharger le fichier RDP

Impossible de se connecter ?

Installer et configurer le Bureau à distance pour se connecter à une machine virtuelle Linux

outhiurdp uthi_key.pem lic_openstack.txt Tout afficher

```

C:\Users\asus>ssh -i uthi_key.pem azureuser@20.36.215.178
Microsoft Windows [version 10.0.19042.1415]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\asus>cd C:\Users\asus\Documents

C:\Users\asus\Documents>ssh -i uthi_key.pem azureuser@20.36.215.178
The authenticity of host '20.36.215.178 (20.36.215.178)' can't be established.
ECDSA key fingerprint is SHA256:W1neW1X13G1SmaD28B9x2bapXx2G1pS0xEdbg.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '20.36.215.178' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.11.0-1023-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Mon Jan 10 17:14:48 UTC 2022

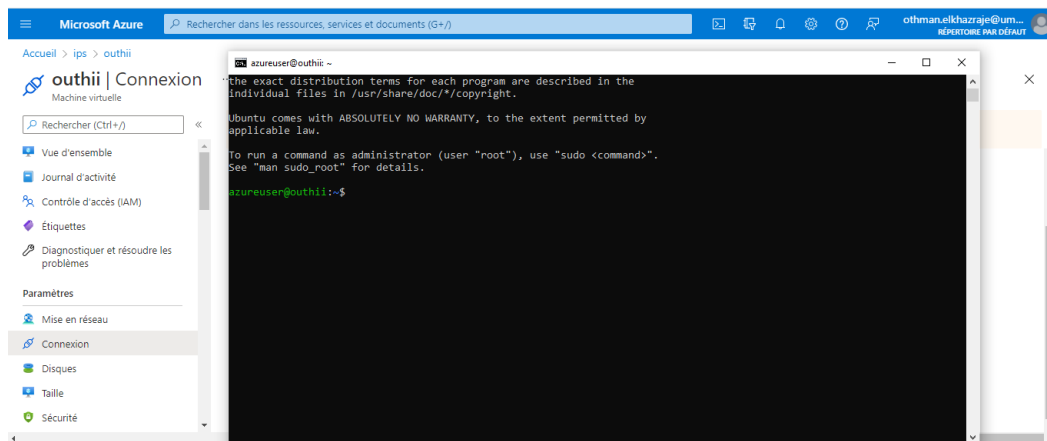
System load:  0.0          Processes:    136
Usage of /:   4.7% of 28.9GB Users logged in: 0
Memory usage: 3%          IPv4 address for eth0: 10.1.0.4
Swap usage:   0%

1 update can be applied immediately.
To see these additional updates run: apt list --upgradable

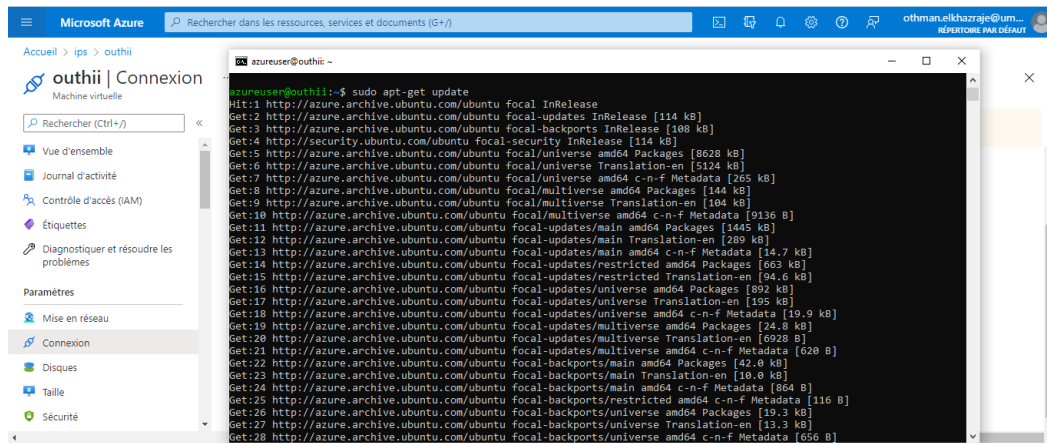
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the

```

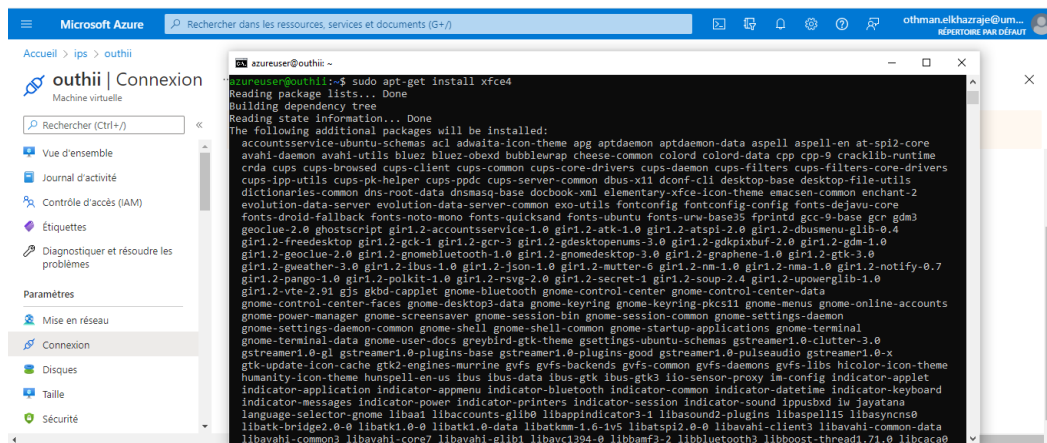
l'activation de Notre vm on a connecter en mode SSH a l'aide du clé secret



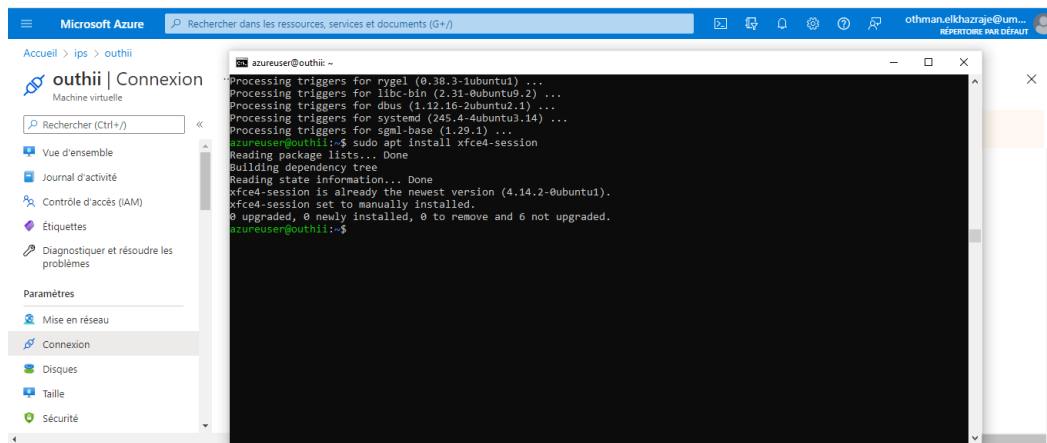
l'entrée au VM ubuntu en mode SSH



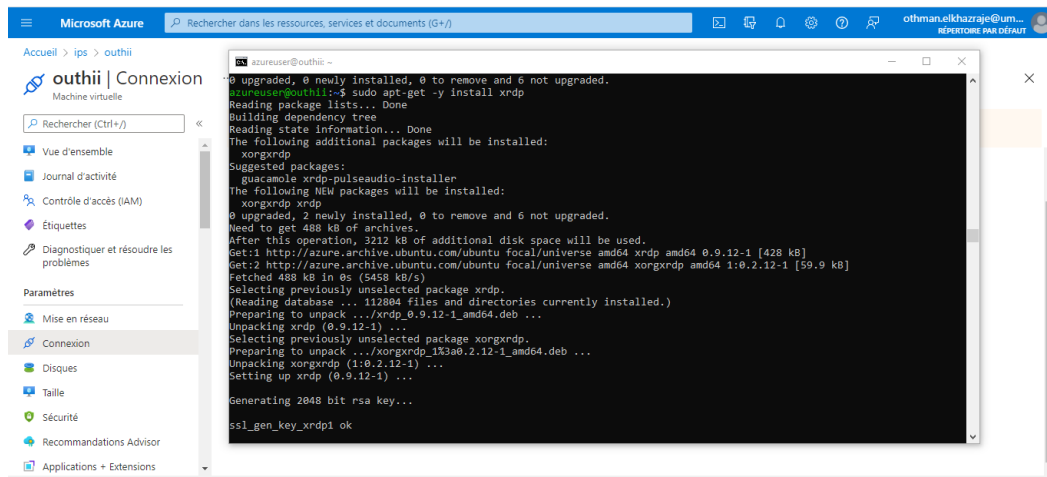
Mise a jour du sytème ubuntu



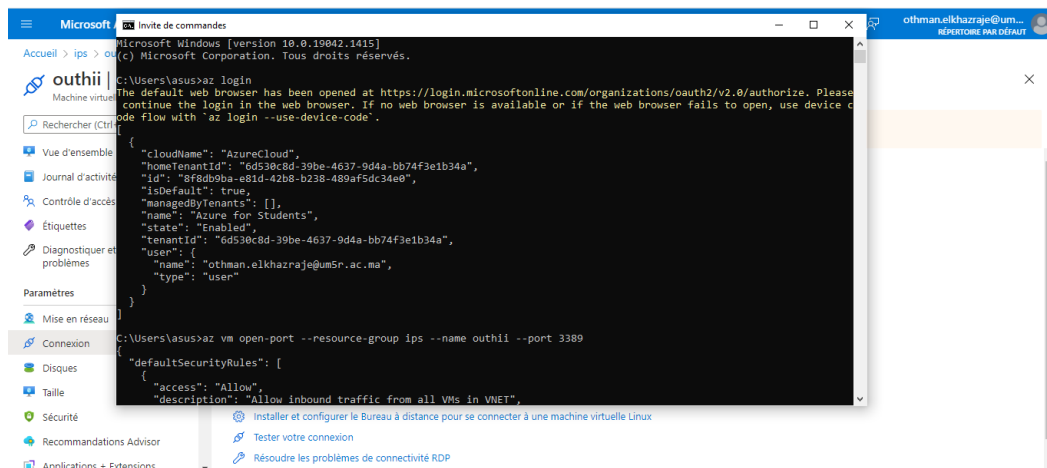
on installe Xfce est un environnement de bureau léger pour les systèmes d'exploitation de type UNIX. Il vise à être rapide, peu gourmand en ressources système, tout en étant visuellement attrayant et convivial.



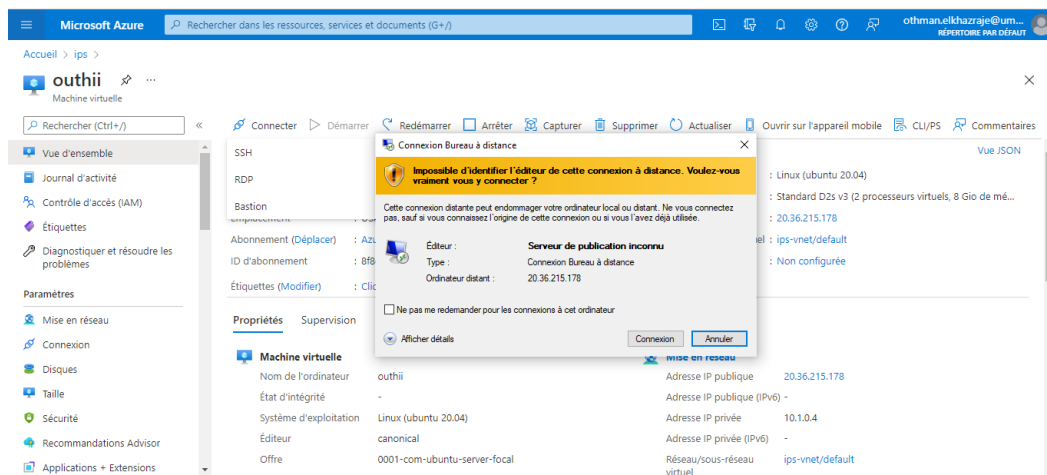
l'installation du session du xfce



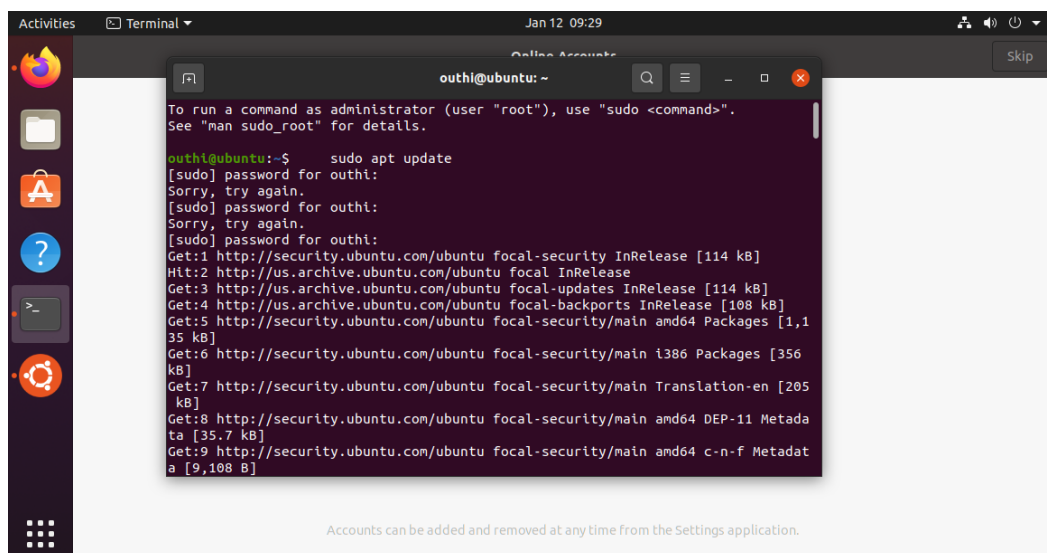
l'installation du xrdp est un serveur RDP (Remote Desktop Protocol) open source



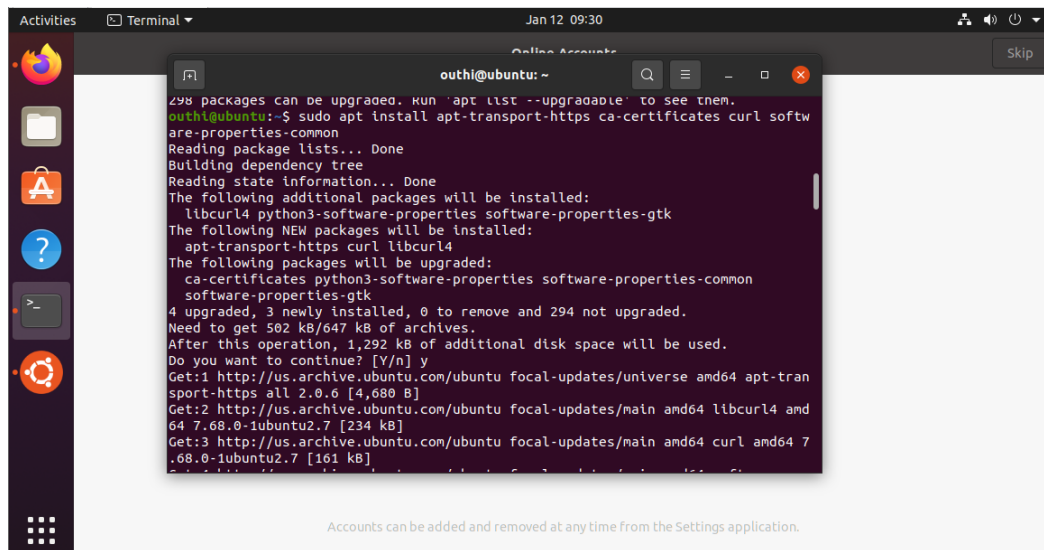
On a déjà installé Azure CLI et en démarre notre session avec la commande `az login` pour entrer les coordonnées



On connecte en mode RDP

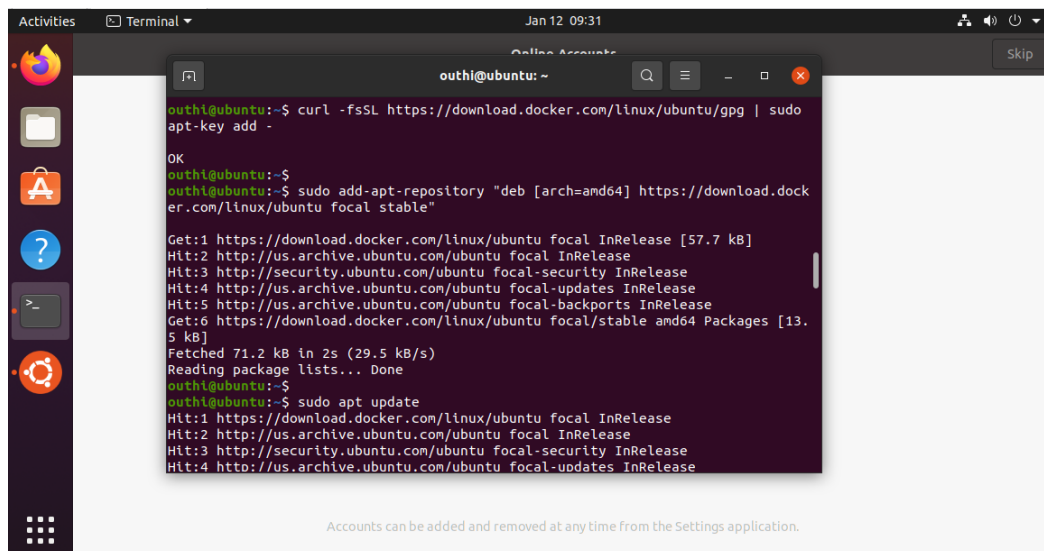


Mise a jour du Systeme



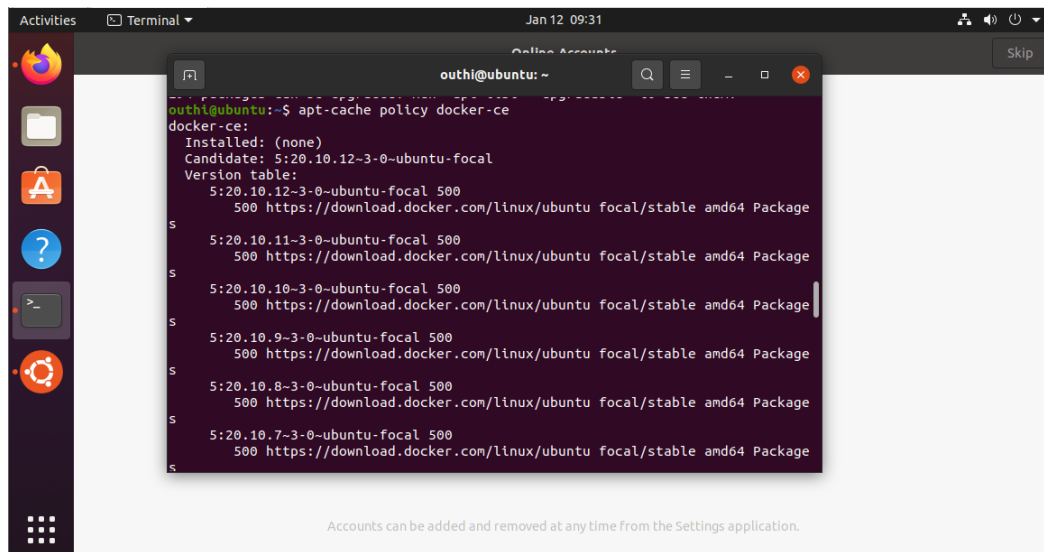
```
outhi@ubuntu: ~  
298 packages can be upgraded. Run 'apt list --upgradable' to see them.  
outhi@ubuntu:~$ sudo apt install apt-transport-https ca-certificates curl software-properties-common  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  libcurl4 python3-software-properties software-properties-gtk  
The following NEW packages will be installed:  
  apt-transport-https curl libcurl4  
The following packages will be upgraded:  
  ca-certificates python3-software-properties software-properties-common  
  software-properties-gtk  
4 upgraded, 3 newly installed, 0 to remove and 294 not upgraded.  
Need to get 502 kB/647 kB of archives.  
After this operation, 1,292 kB of additional disk space will be used.  
Do you want to continue? [Y/n] y  
Get:1 http://us.archive.ubuntu.com/ubuntu focal-updates/universe amd64 apt-transport-https all 2.0.6 [4,680 B]  
Get:2 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 libcurl4 amd64 7.68.0-1ubuntu2.7 [234 kB]  
Get:3 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 curl amd64 7.68.0-1ubuntu2.7 [161 kB]  
Get:4 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 ca-certificates all 20230110~ubuntu0.23.04.1 [151 kB]  
Get:5 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 python3-software-properties all 0.99.23-1ubuntu0.23.04.1 [14.5 kB]  
Get:6 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 software-properties-gtk amd64 0.99.23-1ubuntu0.23.04.1 [14.5 kB]  
Fetched 502 kB in 1s (44.5 kB/s)  
debconf: delaying package configuration, since apt-utils is not installed  
+-----+  
| 298 packages can be upgraded. Run 'apt list --upgradable' to see them. |  
+-----+  
Accounts can be added and removed at any time from the Settings application.
```

Installation de quelques paquets pré-requis qui permettent à apt d'utiliser les paquets sur HTTPS

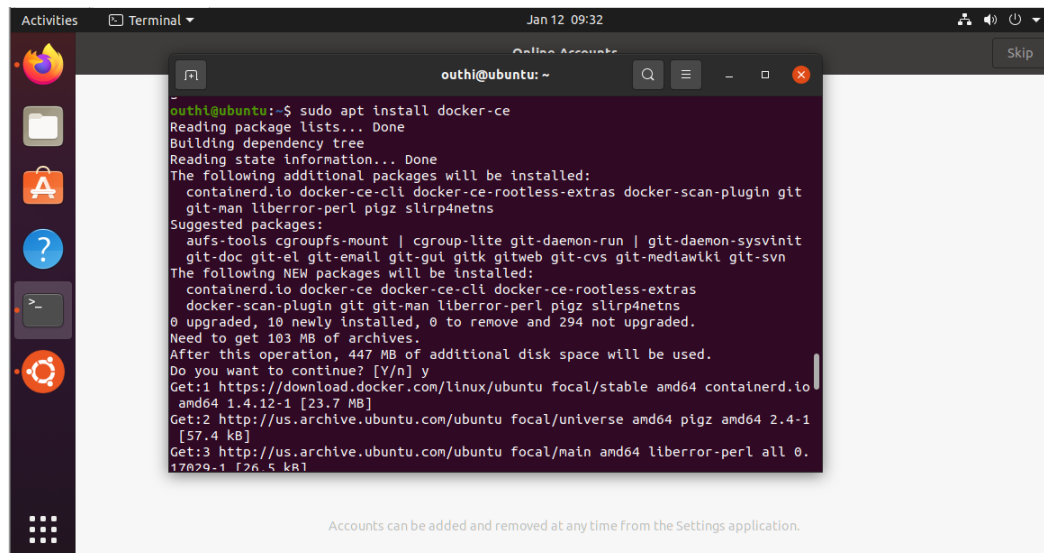


```
outhi@ubuntu: ~  
outhi@ubuntu:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -  
OK  
outhi@ubuntu:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"  
Get:1 https://download.docker.com/linux/ubuntu focal InRelease [57.7 kB]  
Hit:2 http://us.archive.ubuntu.com/ubuntu focal InRelease  
Hit:3 http://security.ubuntu.com/ubuntu focal-security InRelease  
Hit:4 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease  
Hit:5 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease  
Get:6 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages [13.5 kB]  
Fetched 71.2 kB in 2s (29.5 kB/s)  
Reading package lists... Done  
outhi@ubuntu:~$ sudo apt update  
Hit:1 https://download.docker.com/linux/ubuntu focal InRelease  
Hit:2 http://us.archive.ubuntu.com/ubuntu focal InRelease  
Hit:3 http://security.ubuntu.com/ubuntu focal-security InRelease  
Hit:4 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease
```

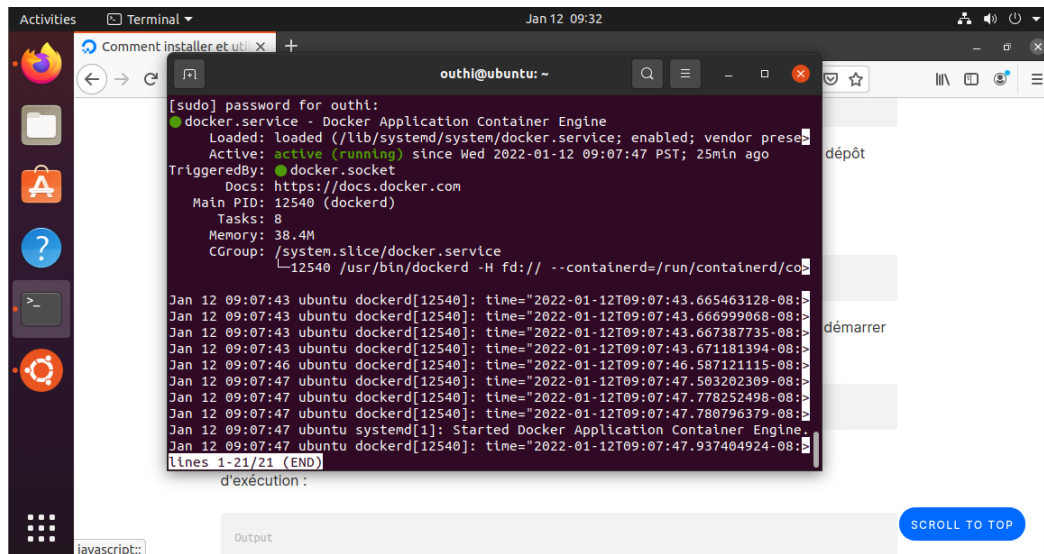
Ensuite, on a ajouté la clé GPG du dépôt officiel de Docker à notre système, et ajouté le référentiel Docker aux sources APT, et on met à jour la base de données des paquets avec les paquets Docker à partir du référentiel qui vient d'être ajouté



le numéro de version du Docker



Installation du Docker

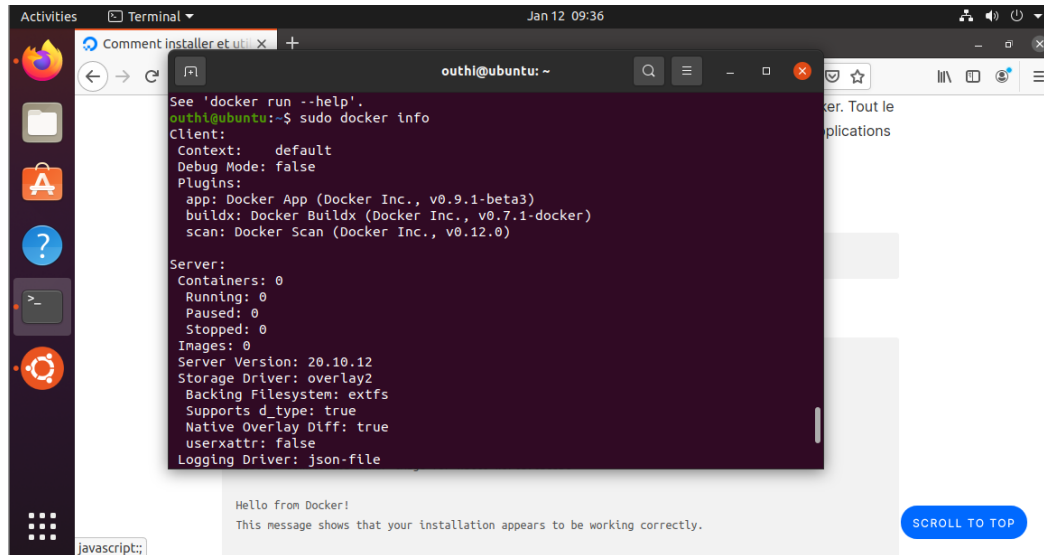


A terminal window on an Ubuntu system showing the status of the Docker service. The user has run `sudo systemctl status docker`. The output shows that the service is active and running. Below the status, the user has run `sudo journalctl -u docker.service` to view the logs. The logs show the Docker daemon starting successfully at 09:07:47.

```
[sudo] password for outhi:
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor prese
   Active: active (running) since Wed 2022-01-12 09:07:47 PST; 25min ago
   TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
    Main PID: 12540 (dockerd)
      Tasks: 8
     Memory: 38.4M
    CGroup: /system.slice/docker.service
           └─12540 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/co

Jan 12 09:07:43 ubuntu dockerd[12540]: time="2022-01-12T09:07:43.665463128-08:
Jan 12 09:07:43 ubuntu dockerd[12540]: time="2022-01-12T09:07:43.666999068-08:
Jan 12 09:07:43 ubuntu dockerd[12540]: time="2022-01-12T09:07:43.667387735-08:
Jan 12 09:07:43 ubuntu dockerd[12540]: time="2022-01-12T09:07:43.671181394-08:
Jan 12 09:07:46 ubuntu dockerd[12540]: time="2022-01-12T09:07:46.587121115-08:
Jan 12 09:07:47 ubuntu dockerd[12540]: time="2022-01-12T09:07:47.503202309-08:
Jan 12 09:07:47 ubuntu dockerd[12540]: time="2022-01-12T09:07:47.778252498-08:
Jan 12 09:07:47 ubuntu dockerd[12540]: time="2022-01-12T09:07:47.780796379-08:
Jan 12 09:07:47 ubuntu systemd[1]: Started Docker Application Container Engine.
Jan 12 09:07:47 ubuntu dockerd[12540]: time="2022-01-12T09:07:47.937404924-08:
lines 1-21/21 (END)
d'exécution :
```

Le service du docker est actif et en cours d'exécution



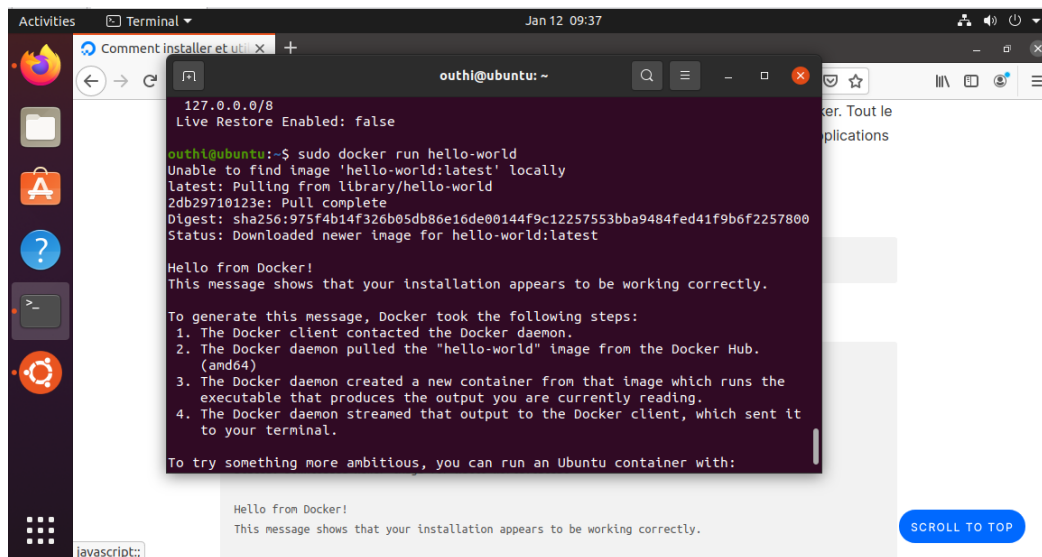
A terminal window on an Ubuntu system showing the output of the `sudo docker info` command. The output provides detailed information about the Docker client and server configuration, including the version (20.10.12), storage driver (overlay2), and various settings.

```
See 'docker run --help'.
outhi@ubuntu:~$ sudo docker info
Client:
 Context: default
 Debug Mode: false
 Plugins:
  app: Docker App (Docker Inc., v0.9.1-beta3)
  buildx: Docker Buildx (Docker Inc., v0.7.1-docker)
  scan: Docker Scan (Docker Inc., v0.12.0)

Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 0
 Server Version: 20.10.12
 Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Native Overlay Diff: true
 userxattr: false
 Logging Driver: json-file

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

les informations sur Docker à l'échelle du système



```
127.0.0.0/8
Live Restore Enabled: false

outhi@ubuntu:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:975f4b14f326b05db86e16de00144f9c12257553bba9484fed41f9b6f2257800
Status: Downloaded newer image for hello-world:latest

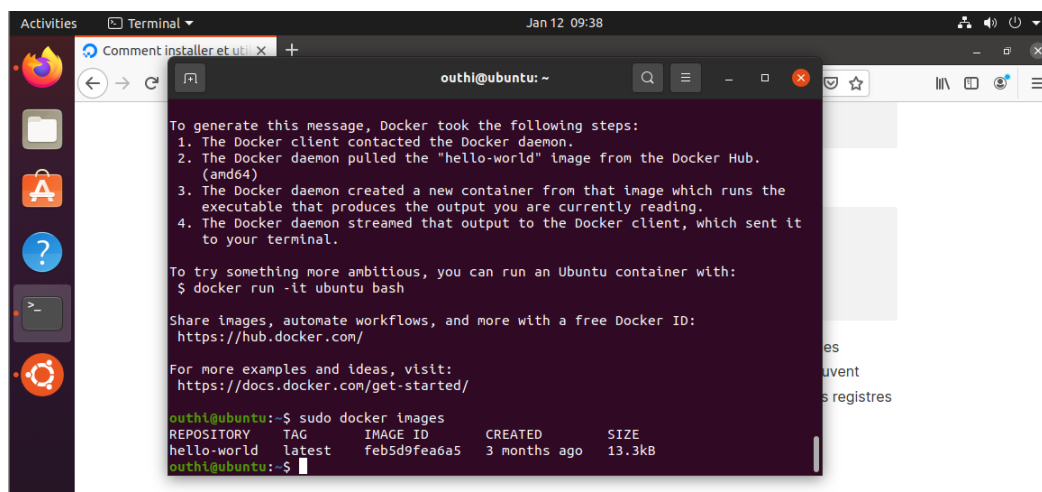
Hello from Docker!
This message shows that your installation appears to be working correctly.

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:

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

Les conteneurs Docker sont construits à partir d'images Docker. Par défaut, Docker tire ces images de Docker Hub, un registre Docker géré par Docker, l'entreprise à l'origine du projet Docker. 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`



```
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/

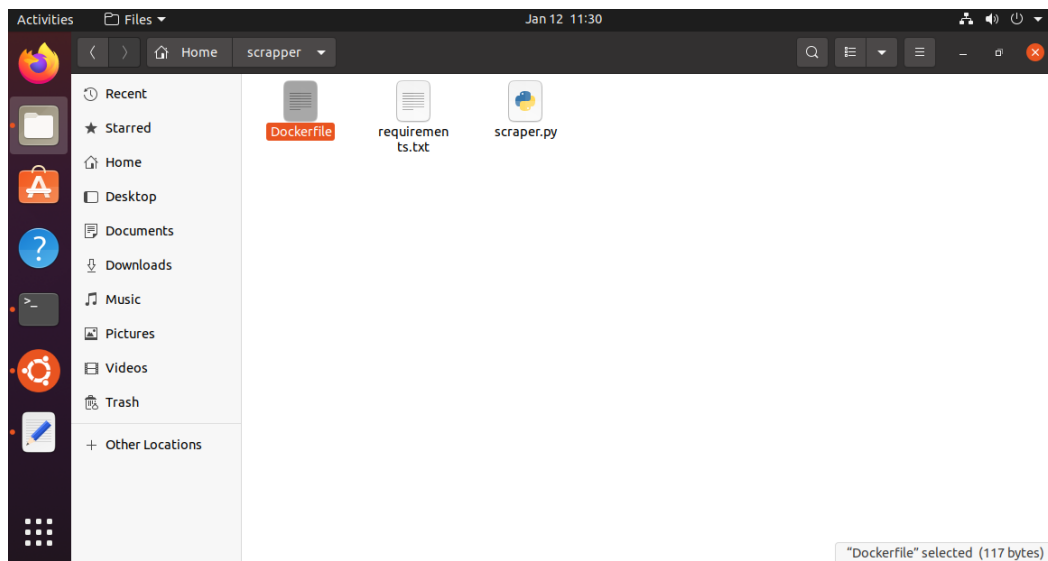
For more examples and ideas, visit:
https://docs.docker.com/get-started/

outhi@ubuntu:~$ sudo docker images
REPOSITORY    TAG       IMAGE ID      CREATED        SIZE
hello-world    latest    feb5d9fea6a5  3 months ago  13.3kB
outhi@ubuntu:~$
```

Pour voir les images qui ont été téléchargées sur notre ordinateur

8) Deploiement d'un Script Python Sur Docker

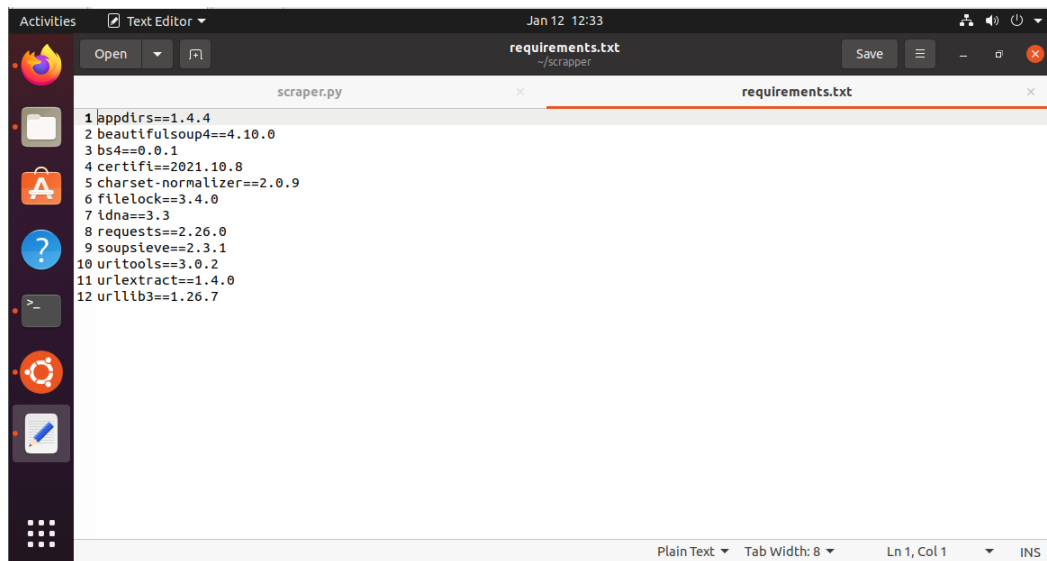
Dans cette partie on va déployer un projet python sur un conteneur docker, Ce projet il permet de scraper les différents images d'un site commerciales notre exemple c'est sur jumia et il le mettre localement Pour quand puisse déployer notre projet on a créer un Fichier DockerFile et un fichier requirement et le fichier scraper.py qu'il permet de faire le scrapping des image Premièrement on a créer un dossier qui contient les trois fichier



Deuxièmement en affiche le contenu du fichier DockerFile ,contient la version du python qui est utiliser dans le projet ainsi le Workdir ca veut dire le répertoire d'exécution du conteneur Run il permet d'exécuter les différents package python qui ont utilisé dans scraper.py et CMD c'est l'exécution du projet



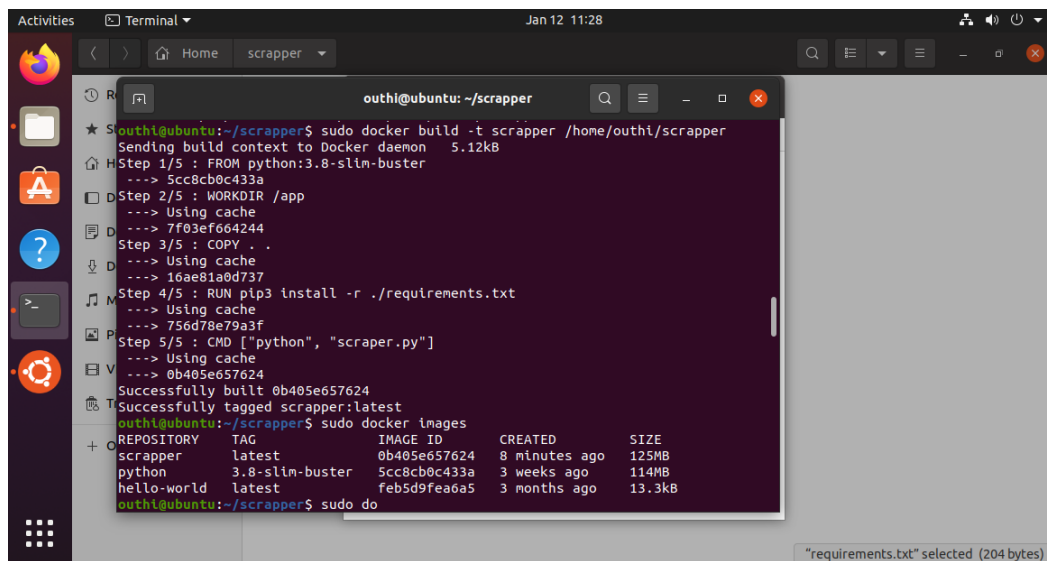
troisièmement le contenu du fichier requirement il est utiliser pour importer les différents package utiliser dans notre script



The screenshot shows a text editor window with two tabs: 'scraper.py' and 'requirements.txt'. The 'requirements.txt' tab is active and contains a list of 12 dependencies. The 'scraper.py' tab is also visible but empty.

```
1 appdirs==1.4.4
2 beautifulsoup4==4.10.0
3 bs4==0.0.1
4 certifi==2021.10.8
5 charset-normalizer==2.0.9
6 filelock==3.4.0
7 idna==3.3
8 requests==2.26.0
9 soupsieve==2.3.1
10 uritools==3.0.2
11 urlextract==1.4.0
12 urllib3==1.26.7
```

Et on entre sans notre répertoire puis en va déployer notre projet sur une conteneur, le premier etape c'est de créer une image ey puis on afficher et finalement en va le runner



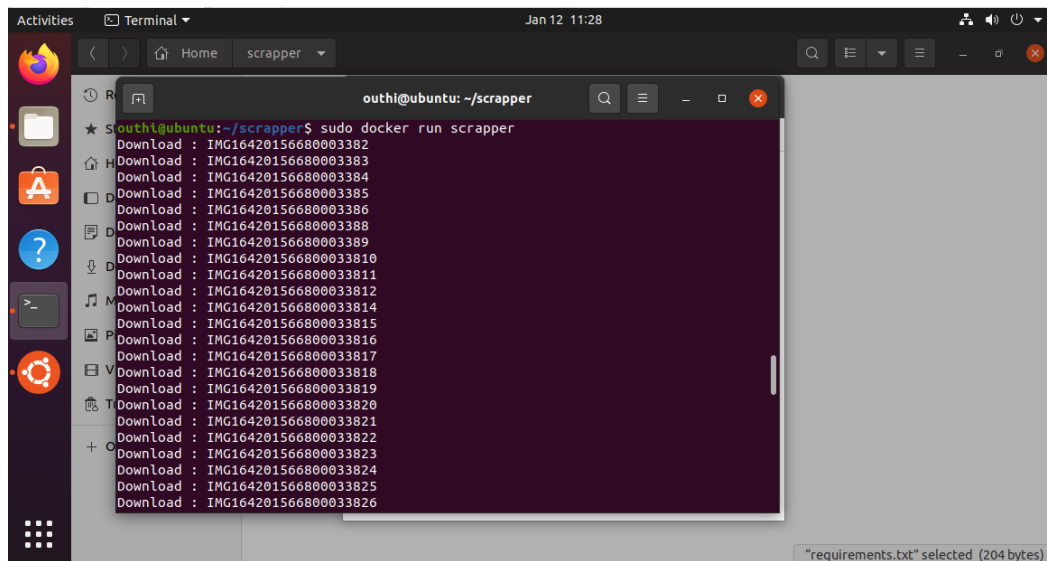
The screenshot shows a terminal window with the following commands and output:

```
outhi@ubuntu: ~/scrapper
outhi@ubuntu:~/scrapper$ sudo docker build -t scrapper /home/outhi/scrapper
Sending build context to Docker daemon 5.12kB
Step 1/5 : FROM python:3.8-slim-buster
--> 5cc8cb0c433a
Step 2/5 : WORKDIR /app
--> Using cache
--> 7f03ef664244
Step 3/5 : COPY . .
--> Using cache
--> 16ae81a0d737
Step 4/5 : RUN pip3 install -r ./requirements.txt
--> Using cache
--> 756d78e79a3f
Step 5/5 : CMD ["python", "scraper.py"]
--> Using cache
--> 0b405e657624
Successfully built 0b405e657624
Successfully tagged scrapper:latest
outhi@ubuntu:~/scrapper$ sudo docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
scrapper	latest	0b405e657624	8 minutes ago	125MB
python	3.8-slim-buster	5cc8cb0c433a	3 weeks ago	114MB
hello-world	latest	feb5d9fea6a5	3 months ago	13.3kB

```
outhi@ubuntu:~/scrapper$ sudo do
```

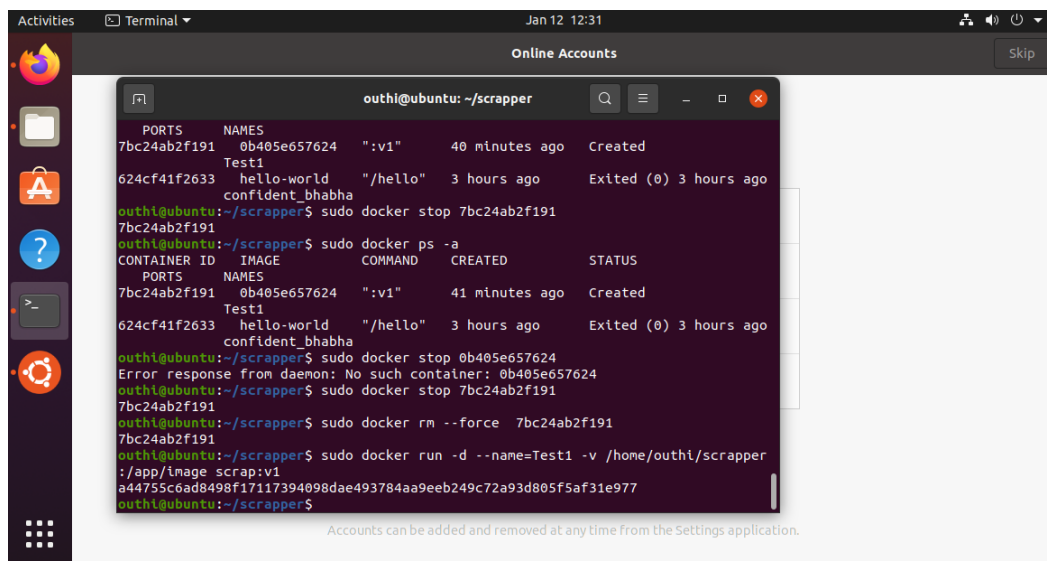
Notre iamge et bien créer d'abord on va le runner



A terminal window titled "outhi@ubuntu: ~/scrapper" showing the output of the command `sudo docker run scrapper`. The output consists of a series of "Download" messages, each followed by an image ID (e.g., `IMG16420156680003382`). The window is part of a desktop environment with a sidebar on the left and a top bar showing the date and time as "Jan 12 11:28".

```
outhi@ubuntu: ~/scrapper
outhi@ubuntu:~/scrapper$ sudo docker run scrapper
Download : IMG16420156680003382
Download : IMG16420156680003383
Download : IMG16420156680003384
Download : IMG16420156680003385
Download : IMG16420156680003386
Download : IMG16420156680003388
Download : IMG16420156680003389
Download : IMG164201566800033810
Download : IMG164201566800033811
Download : IMG164201566800033812
Download : IMG164201566800033814
Download : IMG164201566800033815
Download : IMG164201566800033816
Download : IMG164201566800033817
Download : IMG164201566800033818
Download : IMG164201566800033819
Download : IMG164201566800033820
Download : IMG164201566800033821
Download : IMG164201566800033822
Download : IMG164201566800033823
Download : IMG164201566800033824
Download : IMG164201566800033825
Download : IMG164201566800033826
```

En tombant sur une erreur au niveau du Docker, Docker permet de faire un chemin spécifique pour avoir stocker les données donc, on a créé un chemin localement pour pouvoir stocker les images d'une façon quand on peut l'accéder en utilisant la commande suivante :

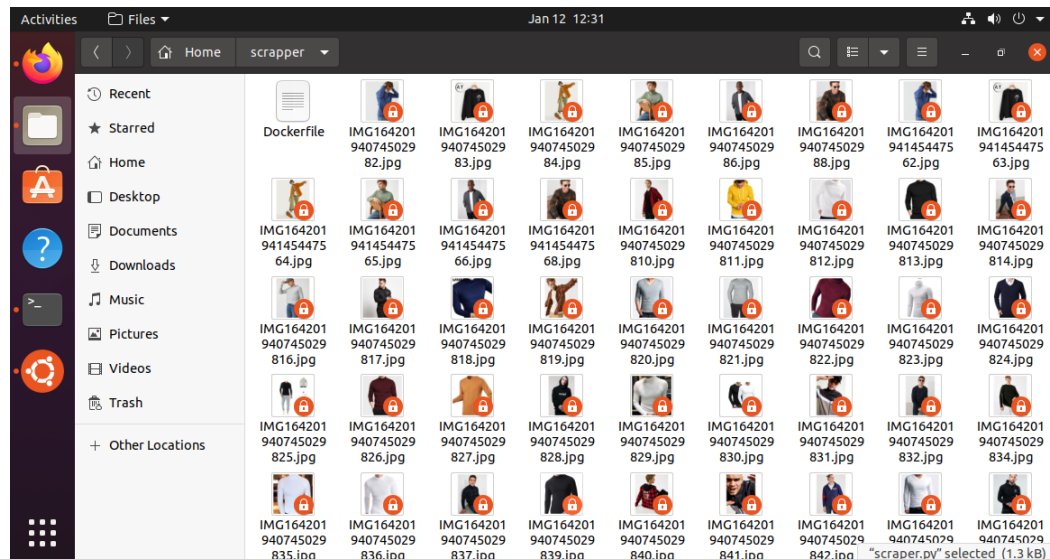


A terminal window titled "outhi@ubuntu: ~/scrapper" showing a series of Docker commands and their outputs. The commands include `docker ps -a`, `docker stop`, and `docker rm`. The output shows details about two containers: one created 40 minutes ago and another that exited 3 hours ago. The window is part of a desktop environment with a sidebar on the left and a top bar showing the date and time as "Jan 12 12:31".

```
outhi@ubuntu: ~/scrapper
outhi@ubuntu:~/scrapper$ sudo docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS
7bc24ab2f191   0b405e657624   ":v1"                   40 minutes ago Created
624cf41f2633   hello-world    "/hello"                3 hours ago   Exited (0) 3 hours ago
confident_bhabha

outhi@ubuntu:~/scrapper$ sudo docker stop 7bc24ab2f191
7bc24ab2f191
outhi@ubuntu:~/scrapper$ sudo docker stop 0b405e657624
Error response from daemon: No such container: 0b405e657624
outhi@ubuntu:~/scrapper$ sudo docker stop 7bc24ab2f191
7bc24ab2f191
outhi@ubuntu:~/scrapper$ sudo docker rm --force 7bc24ab2f191
7bc24ab2f191
outhi@ubuntu:~/scrapper$ sudo docker run -d --name=Test1 -v /home/outhi/scrapper
:/app/image scrap:v1
a44755c6ad8498f7117394098dae493784aa9eeb249c72a93d805f5af31e977
outhi@ubuntu:~/scrapper$
```

Et voici En arrive a centraliser nos données localement



8) Commandes d'utilisation quotidienne de Docker:

1. `docker -version` Cette commande est utilisée pour obtenir la version actuellement installée de docker.
2. `docker pull` Cette commande est utilisée pour extraire des images du dépôt de docker (hub.docker.com).
3. `docker run -it -d` Cette commande est utilisée pour créer un conteneur à partir d'une image.
4. `docker ps` Cette commande est utilisée pour lister les conteneurs en cours d'exécution.
5. `docker ps -a` Cette commande est utilisée pour montrer tous les conteneurs en cours d'exécution et sortis.
6. `docker exec -it bash` Cette commande est utilisée pour accéder au conteneur en cours d'exécution.
7. `docker stop` Cette commande arrête un conteneur en cours d'exécution.
8. `docker kill` Cette commande tue le conteneur en arrêtant immédiatement son exécution. La différence entre "docker kill" et "docker stop" est que "docker stop" donne au conteneur le temps de s'arrêter de manière gracieuse. Dans les situations où l'arrêt du conteneur prend trop de temps, on peut choisir de le tuer.
9. `docker commit <username/>image name` Cette commande crée une nouvelle image d'un conteneur édité sur le système local.
10. `docker login` Cette commande est utilisée pour se connecter au référentiel docker hub.
11. `docker push <username/>image name` Cette commande est utilisée pour pousser une image vers le dépôt docker hub.
12. `docker images` Cette commande liste toutes les images docker stockées localement.
13. `docker rm` Cette commande est utilisée pour supprimer un conteneur arrêté.
14. `docker rmi` Cette commande est utilisée pour supprimer une image.
15. `docker build` Cette commande est utilisée pour construire une image à partir d'un fichier docker spécifié.

9) Conclusion :

