

Docker



Plan du cours

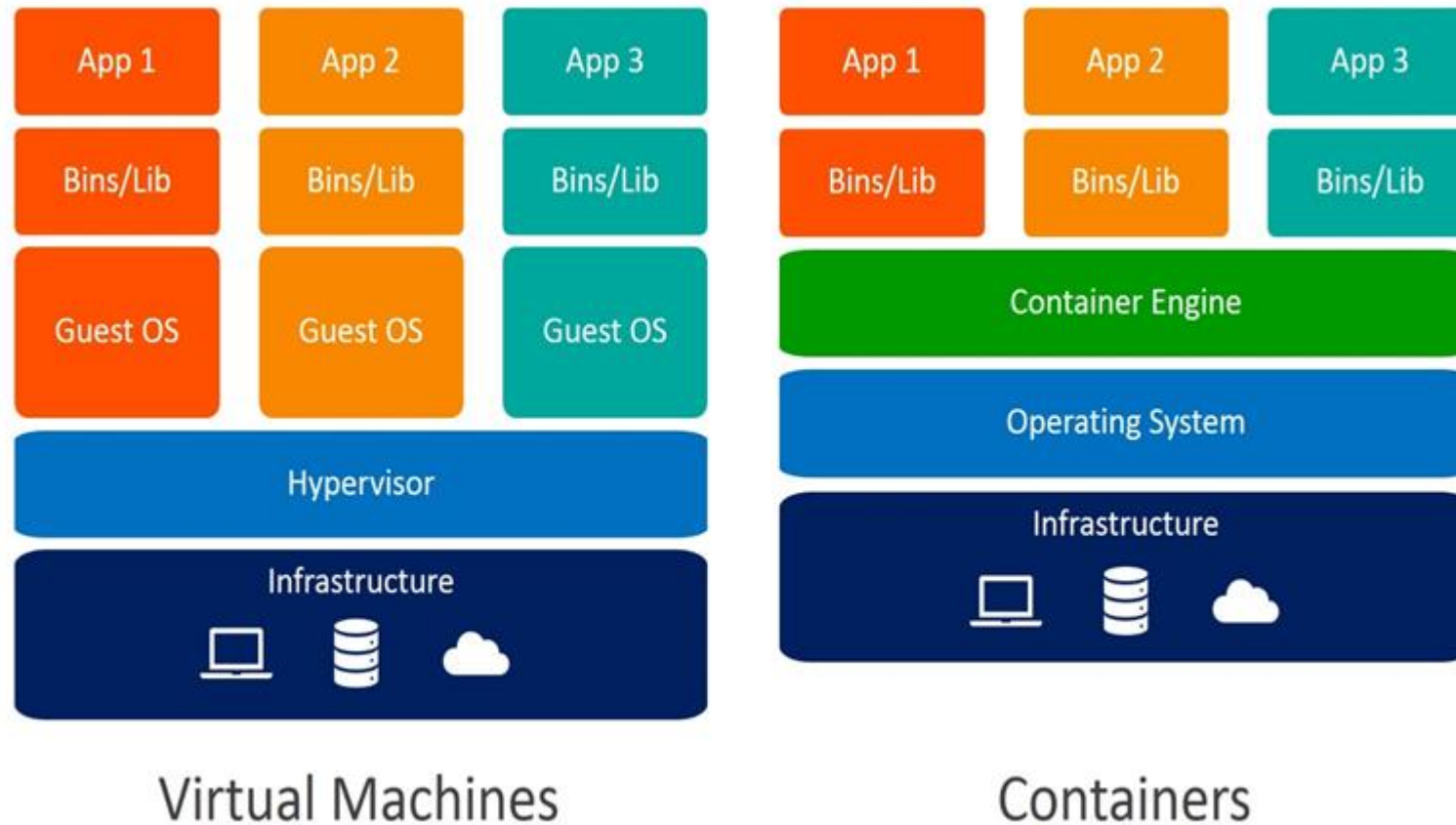
- Évolution de la machine de production
- Les VM
- Les Conteneurs
- VM vs Containers
- Docker
 - Définition
 - Architecture : CLI, Docker Host, Docker Demon, Docker Hub, ...
 - Container vs Image
 - Installation
 - Docker File
 - Commandes de base : build, run, pull, push

La virtualisation

- **La virtualisation:** La virtualisation est une technologie permettant de créer et d'exécuter une ou plusieurs représentations virtuelles d'un ordinateur et de ses différentes ressources sur une même machine physique.
- La virtualisation a eu le succès grâce au **cloud computing**: C'est un Data center ou une infrastructure offerte par un fournisseur dans laquelle la puissance de calcul et le stockage sont gérés par des serveurs distants auxquels les usagers se connectent via une liaison Internet sécurisée.
 - ❑ Elasticité rapide
 - ❑ Modèle Pay as You Go



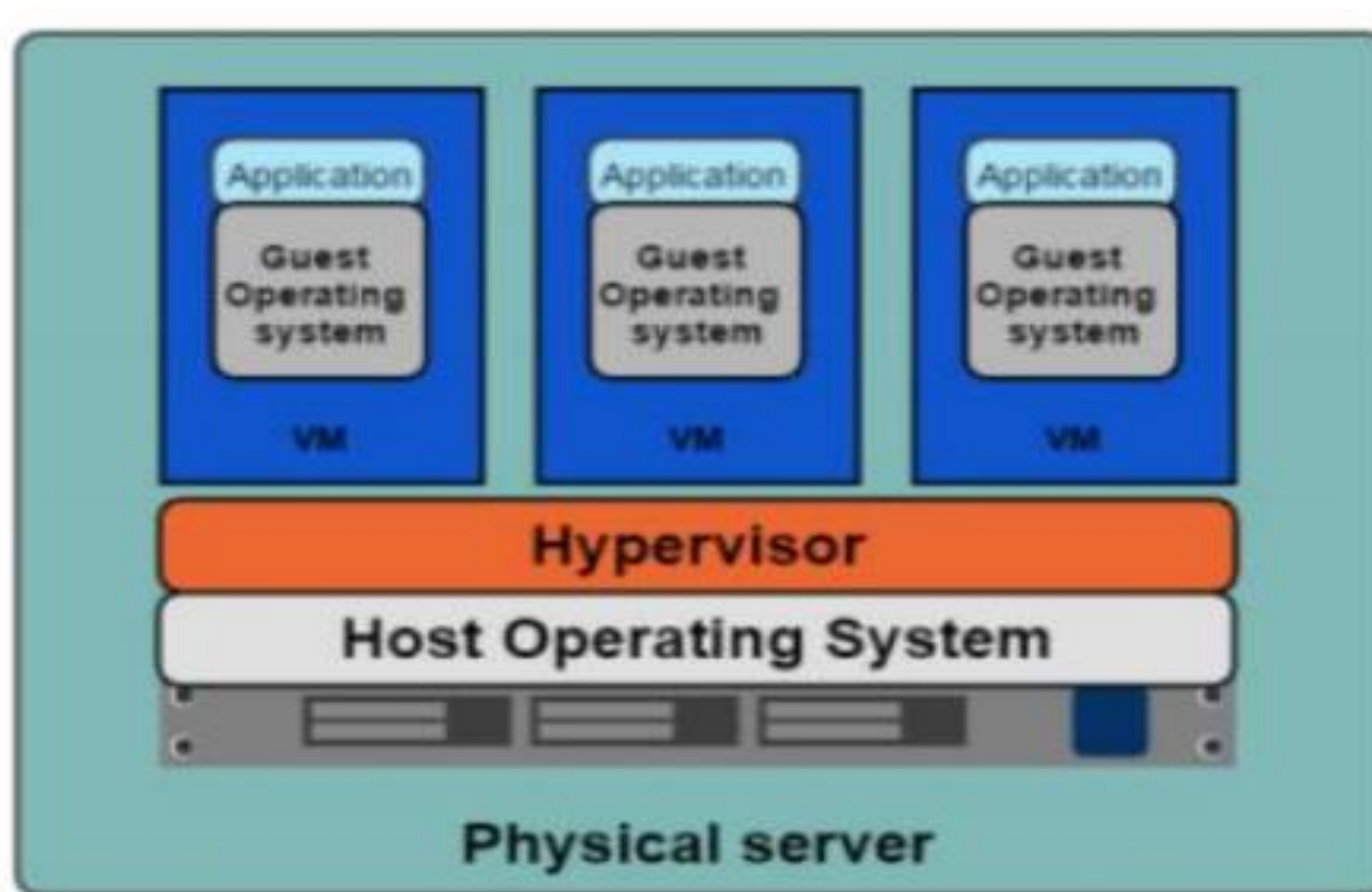
La virtualisation



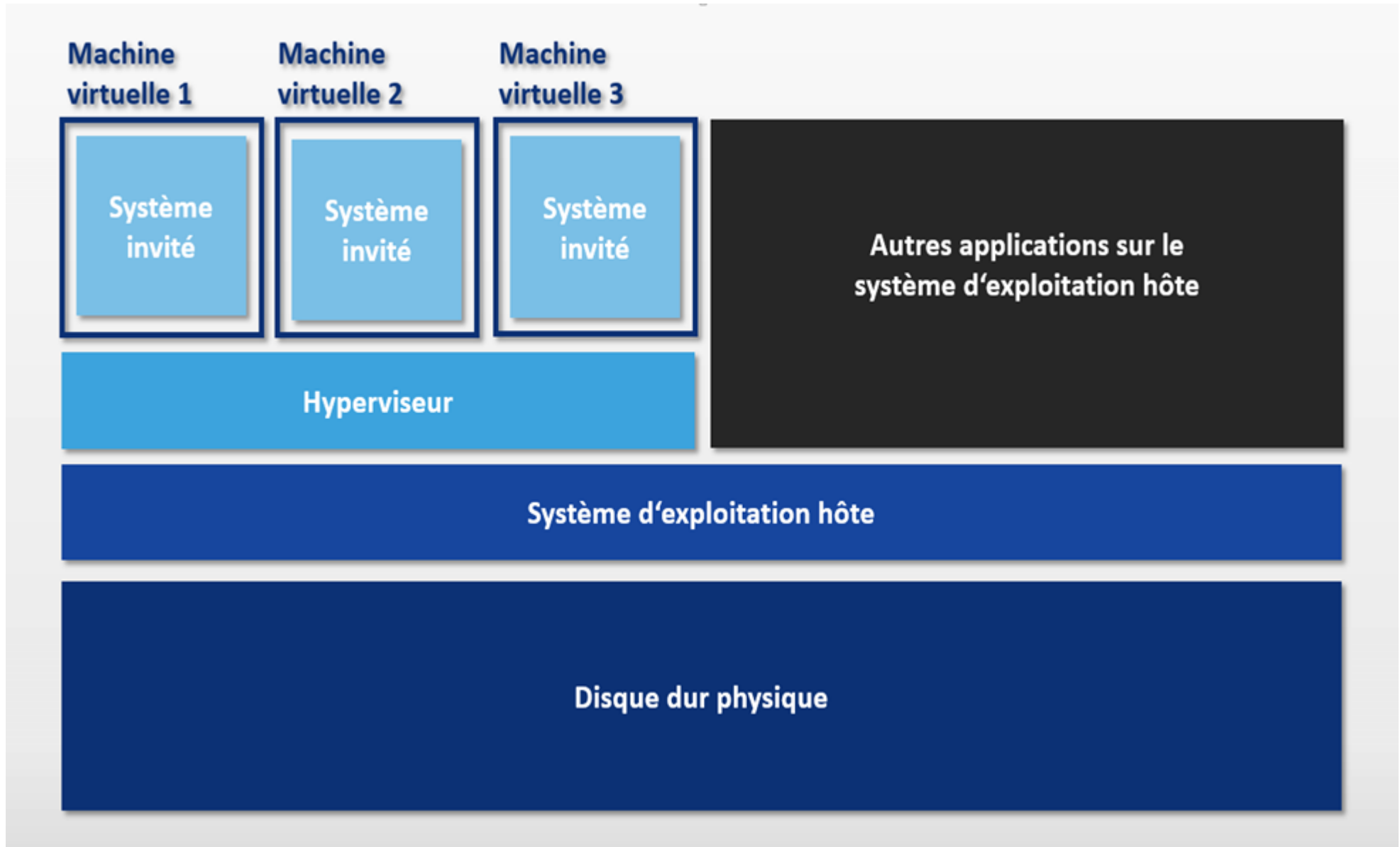
- On distingue deux types de virtualisation tels que:
 - ✓ La virtualisation lourde (VM)
 - ✓ La virtualisation légère (Conteneurs)

VM

La virtualisation Lourde ou à base de VM (utilisant un Hyperviseur): Elle permet de simuler une ou plusieurs machines physiques, et les exécuter sous forme de machines virtuelles (VM) sur un serveur. Ces VM intègrent un **OS** sur lequel des applications sont exécutées.



VM



VM

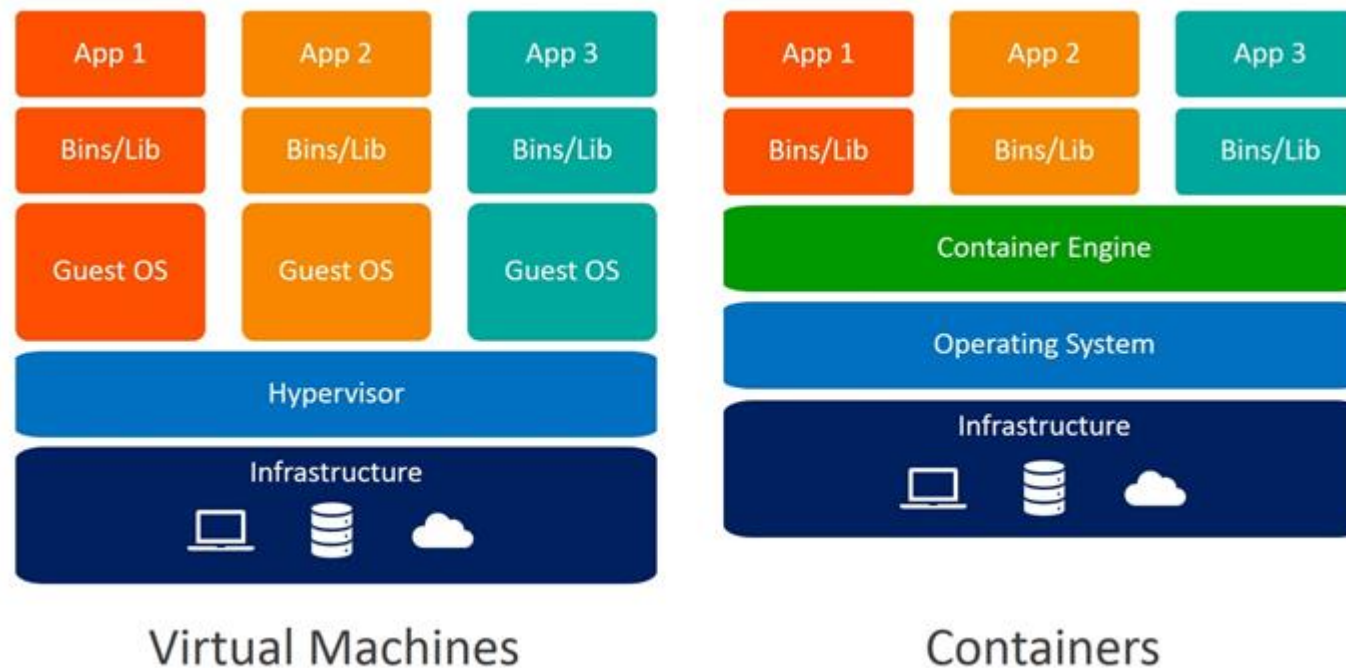
- **Avantages des VMs**
 - ✓ Une meilleure exploitation des ressources
 - ✓ Une machine physique est divisée en plusieurs machines virtuelles
- **Limites des VMs**
 - ✓ Chaque VM a besoin des ressources (CPU, Stockage (Disque), RAM, OS invité)
 - ✓ En augmentant les VMs, on demande plus de ressources
 - Chaque OS invité alloue ses propres ressources
 - Gaspillage
 - Portabilité d'applications n'est pas garantie

Conteneur

La virtualisation légère ou à base des conteneurs:

- Offre tous les services, scripts, API, librairies dont une application a besoin favorisant ainsi une juste utilisation des ressources.
- Repose sur la création de conteneurs isolés les uns des autres sur un **OS Commun**.
- Les conteneurs indépendants partagent un **OS commun** et un même **espace mémoire**.
- Un conteneur est une enveloppe (emballage) contenant toutes les ressources nécessaires pour faire fonctionner une application donnée (Environnement d'exécution comme JDK, livrable de l'application, dépendances nécessaires)

VM vs Conteneur



Virtual Machines

Containers

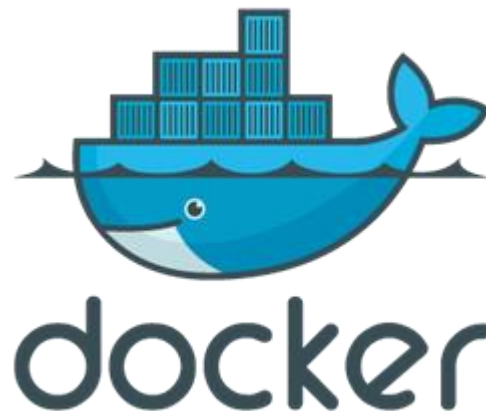
- Une machine virtuelle va recréer un serveur complet pour chaque application avec son propre système d'exploitation
- Le conteneur va isoler l'application tout en utilisant le système d'exploitation de son hôte. Ce même système d'exploitation peut être utilisé par d'autres conteneurs ayant des tâches totalement distinctes.

VM vs Conteneur

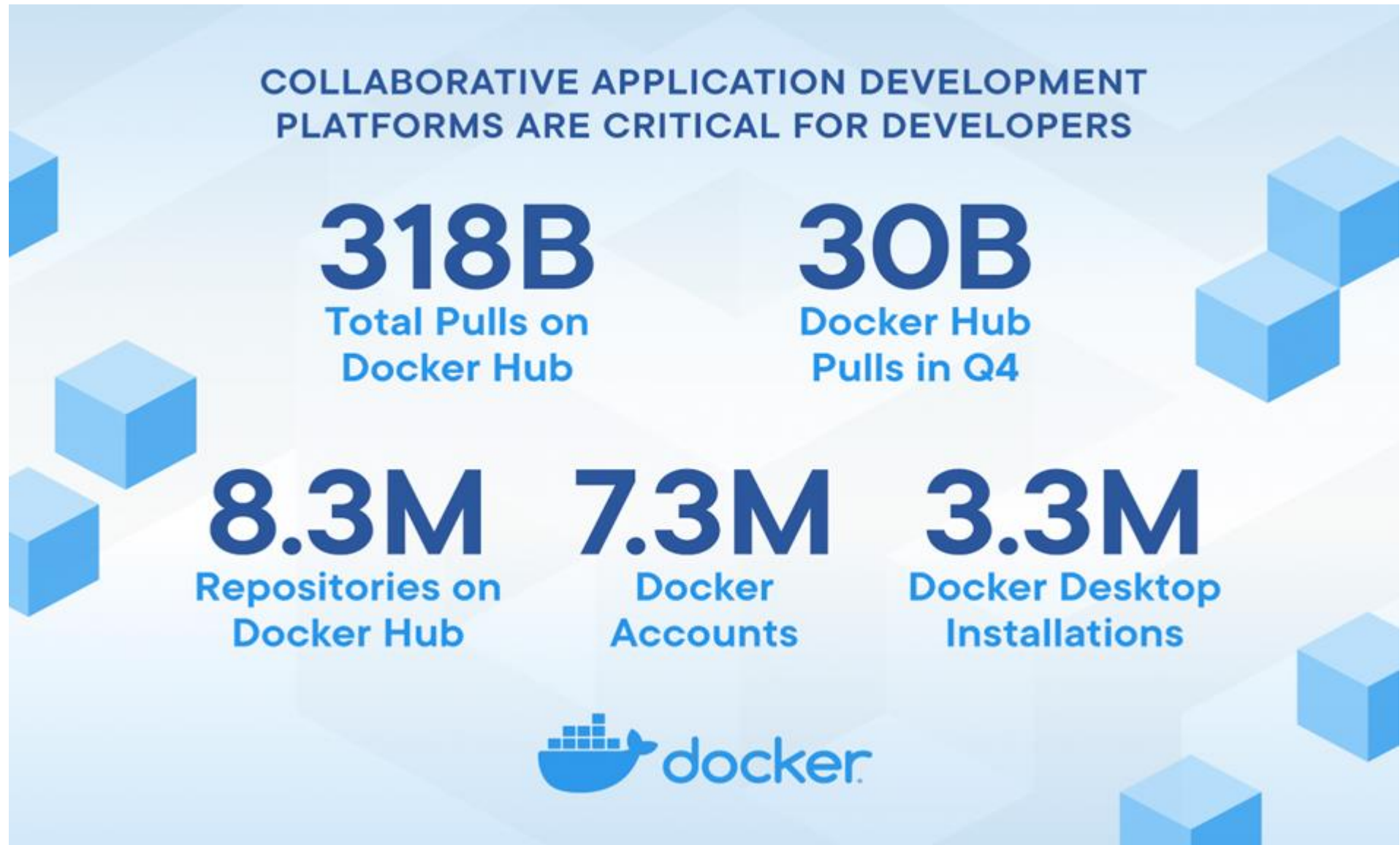
VM	Container
Une virtualisation lourde	Une virtualisation légère
À base de VM	À base des Conteneurs
Virtualisation des ressources hardware (CPU, RAM, disque, ...)	Virtualisation au niveau du système d'exploitation (de l' OS)
Machine invité (machine virtuelle)	Conteneur
Image ISO (OS Complet)	Librairies nécessaires
Hyperviseur sur la Machine hôte	Moteur de conteneur sur la machine hôte
Démarrage plus lent	Démarrage en quelques secondes
Entièrement isolé et donc plus sécurisé	Isolation au niveau du processus et donc moins sécurisée

Docker - Définition

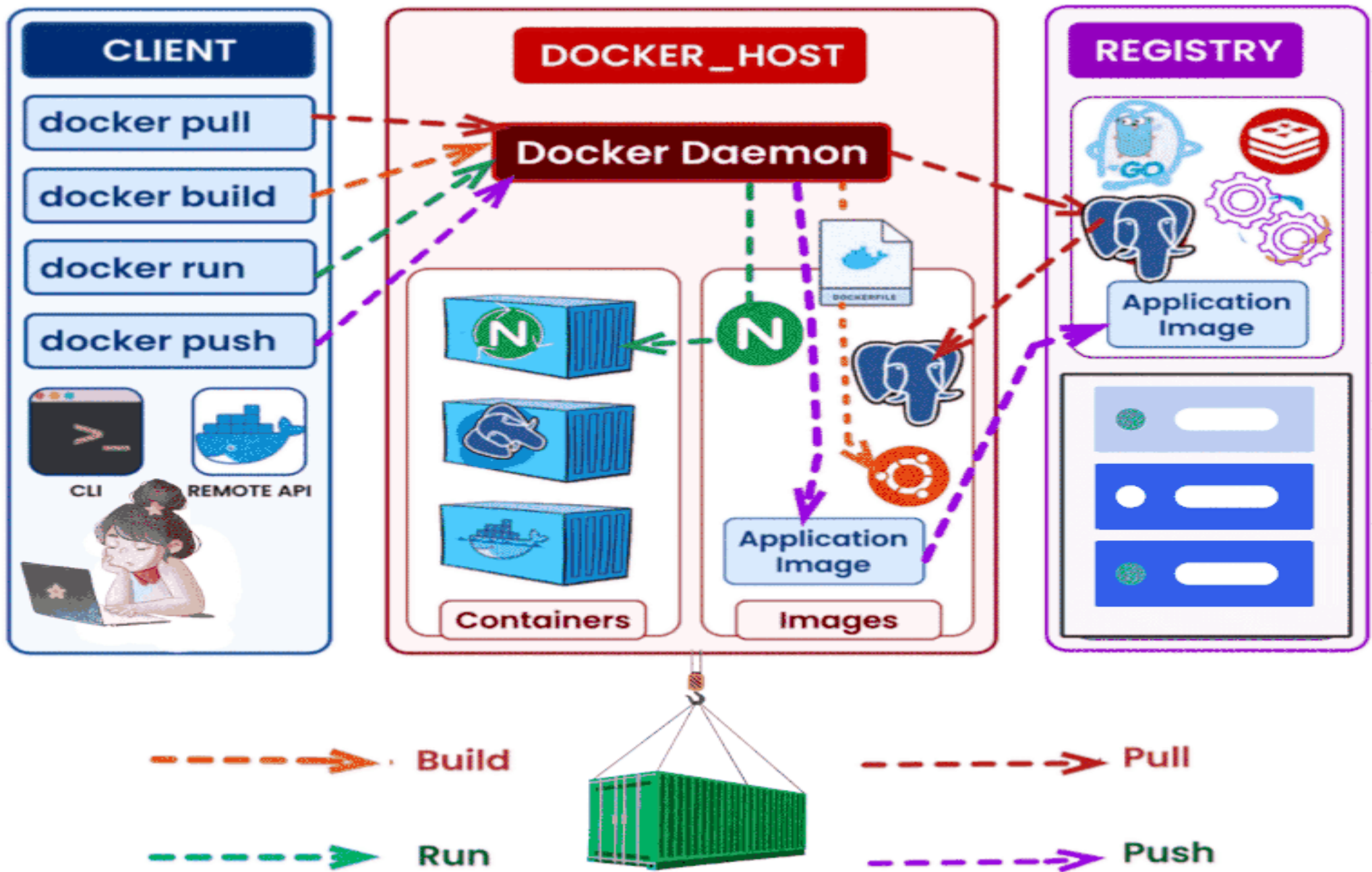
- Docker permet d'embarquer une application dans un ou plusieurs containers logiciels qui pourra s'exécuter sur n'importe quel serveur machine, qu'il soit physique ou virtuel.
- On distingue deux versions de docker:
 - Docker **E**ntreprise **E**dition : Docker EE payante
 - Docker **C**ommunity **E**dition: Docker CE gratuite

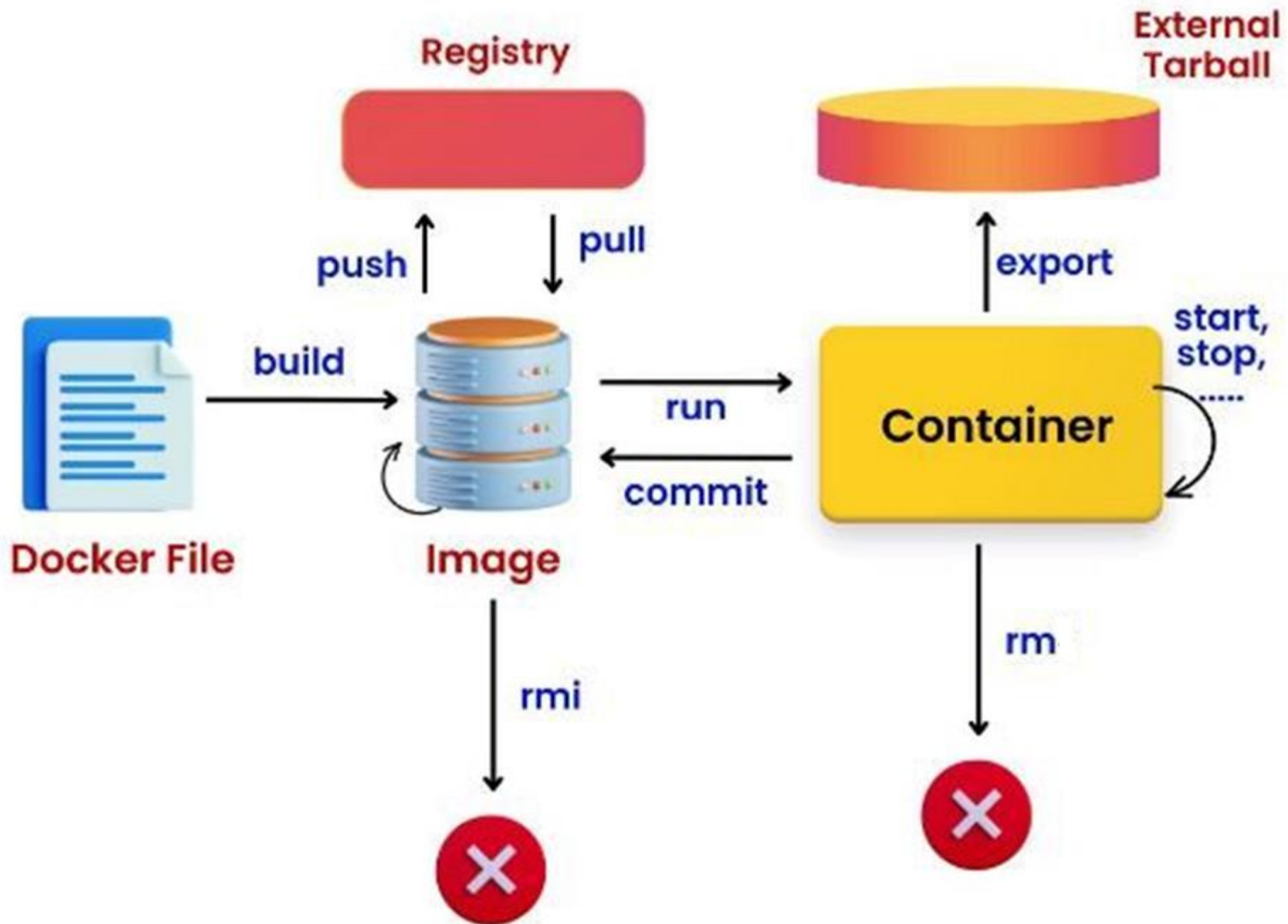


Docker - Définition



Docker - Architecture





Docker - Composants

Client Docker : Les utilisateurs de Docker peuvent interagir avec Docker via un client (CLI, Docker Desktop, ...). Lorsqu'une commande docker s'exécute, le client les envoie au démon docker, qui les exécute.

Docker Host (C'est notre VM Ubuntu) : Une machine physique ou virtuelle qui exécute un Docker Daemon (Le service installé sur le système d'exploitation hôte du serveur) et contient des images en cache ainsi que des conteneurs.

Docker – Composants

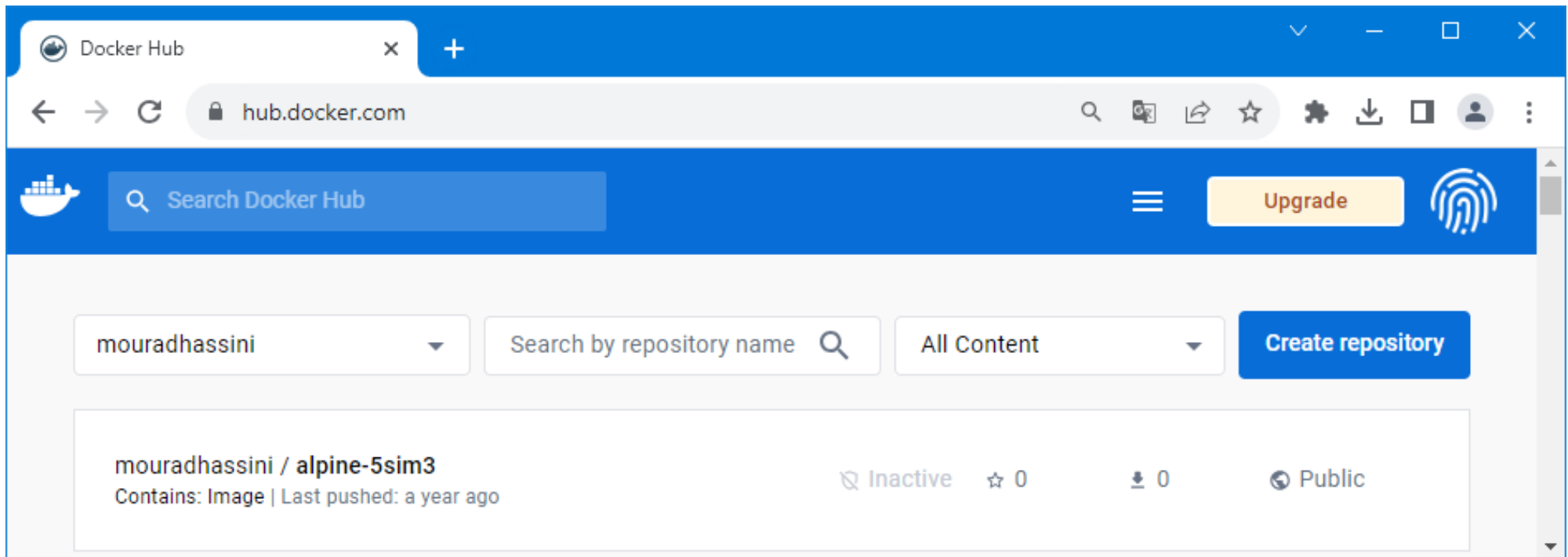
Images : Ils sont des modèles en lecture seule avec des instructions pour créer un conteneur Docker. L'image Docker peut être extraite d'un hub Docker et utilisée telle quelle, ou vous pouvez ajouter des instructions supplémentaires (via un Docker File) à l'image de base et créer une image Docker nouvelle et modifiée.

Conteneurs : Une fois que vous avez exécuté une image Docker, elle crée un conteneur Docker. Toutes les applications et leur environnement s'exécutent à l'intérieur de ce conteneur. Vous pouvez utiliser une interface (Docker Desktop, ...) ou la CLI Docker pour démarrer, arrêter, supprimer un conteneur Docker.

Le conteneur est une instance d'une image exécutée

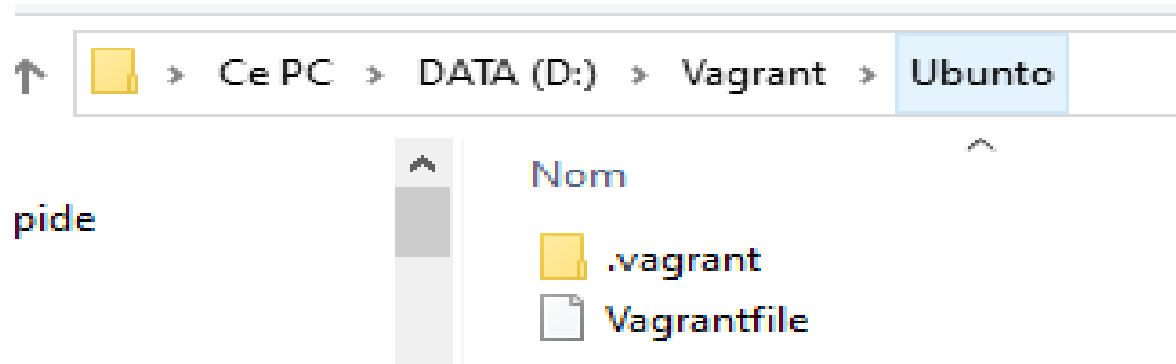
Docker - Composants

Registres Docker: C'est l'emplacement où les images Docker sont stockées. Il s'agit d'un registre docker public (Docker Hub) ou d'un registre docker privé.



Docker – Installation

- Si la VM n'est pas déjà démarrée, vérifier l'adresse IP de votre VM (Vagrantfile).
Puis sur votre machine Windows, se placer au niveau dossier de la VM et lancer
là, puis lancer un client ssh :



```
PS D:\Vagrant\Ubuntu> vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Checking if box 'bento/ubuntu-22.04' version '202309.08.0' is
up to date...
```

```
PS D:\Vagrant\Ubuntu> vagrant ssh
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-83-generic x86_64)
vagrant@vagrant:~$
```

Docker – Installation (7 Commandes)

sudo apt update

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

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

sudo add-apt-repository "deb [arch=amd64]

https://download.docker.com/linux/ubuntu \$(lsb_release -cs) stable"

apt-cache policy docker-ce

sudo apt install docker-ce -y

sudo systemctl status docker

Docker – Installation

- Docker est déjà lancé après l'installation, vérifier avec : **sudo systemctl status docker**

```
Sélection vagrant@vagrant: ~
vagrant@vagrant:~$ sudo 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 2023-09-24 16:59:00 UTC; 51s ago
     TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
    Main PID: 3822 (dockerd)
       Tasks: 8
      Memory: 26.7M
         CPU: 1.058s
      CGroup: /system.slice/docker.service
              └─3822 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

- **CTRL C** pour sortir de la commande de status.

Docker – Installation

- Donner les droits d'accès en lecture et écriture, mais pas exécution pour le user « vagrant », son groupe et tout autre utilisateur (6 = (R=1 / W=1 X=0)) :

sudo chmod 666 /var/run/docker.sock

- Pour s'assurer que tout est bon, vérifier la version de Docker et lancer l'image « hello-world » (qui sera récupérer de Docker Hub automatiquement :

docker -v

docker run hello-world

- Voir résultats page suivante :

Docker – Installation

```
Sélection vagrant@vagrant: ~  
vagrant@vagrant:~$ docker -v  
Docker version 24.0.6, build ed223bc  
vagrant@vagrant:~$ docker run hello-world  
docker: permission denied while trying to connect to the  
.sock: Post "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/cont  
: connect: permission denied.  
See 'docker run --help'  
vagrant@vagrant:~$ sudo chmod 666 /var/run/docker.sock  
vagrant@vagrant:~$ docker run hello-world  
Unable to find image 'hello-world:latest' locally  
latest: Pulling from library/hello-world  
719385e32844: Pull complete  
Digest: sha256:4f53e2564790c8e7856ec08e384732aa38dc43c52f  
Status: Downloaded newer image for hello-world:latest  
  
Hello from Docker!  
This message shows that your installation appears to be w  
  
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  
(amd64)  
3. The Docker daemon created a new container from that i  
executable that produces the output you are currently
```

Docker – Image

- Pour créer une image, nous pouvons utiliser les solutions suivantes :
 - Docker Hub (importer une image depuis le cloud Docker)
 - Utiliser un fichier Dockerfile pour créer une image à travers les commandes
 - A partir d'un registry privé (en entreprise)
 - A partir d'un conteneur (Image => conteneur => image)

Docker – Docker Hub

- DockerHub est le registre officiel de Docker
- C'est un répertoire **SaaS** (Software As A Service ou logiciel en tant que service) regroupant des applications containerisés (images) fournis par des développeurs et sont accessibles.
- Ces images peuvent également être fournies par Docker.
- Il est possible de télécharger ces images et de partager les vôtres.




Docker – Dockerfile

- Un Dockerfile est un document texte qui contient les commandes qu'on pourrait exécuter en ligne de commande pour créer sa propre image.
- Il est basé sur une image standard à la quelle on ajoute les éléments propres à l'application que l'on veut déployer. Créons le **Dockerfile** :


```
vagrant@vagrant: ~/docker
vagrant@vagrant:~$ mkdir docker
vagrant@vagrant:~$ cd docker
vagrant@vagrant:~/docker$ vi Dockerfile
vagrant@vagrant:~/docker$ cat Dockerfile
FROM alpine
RUN apk add openjdk11
EXPOSE 80
CMD "java"
vagrant@vagrant:~/docker$
```

```
vagrant@vagrant: ~/docker
FROM alpine
RUN apk add openjdk11
EXPOSE 80
CMD "java"
~
~
~
-- INSERT --      4,11      All
```

Docker – Dockerfile



[Explore](#) [Repositories](#) [Organizations](#) [Help](#)




[Upgrade](#)  [sirinenalfar](#)

Filters

Products

- ☐ Images
- ☐ Extensions
- ☐ Plugins

Trusted Content

- ☐  Docker Official Image
- ☐  Verified Publisher
- ☐  Sponsored OSS

Operating Systems


- ☐ Linux
- ☐ Windows


Architectures

- ☐ ARM
- ☐ ARM 64
- ☐ IBM POWER

1 - 25 of 56 904 results for **alpine**.

Best Match



alpine  DOCKER OFFICIAL IMAGE


Updated a month ago


A minimal Docker image based on Alpine Linux with a complete package index and only 5 MB in size!

Linux riscv64 x86-64 ARM ARM 64 386 PowerPC 64 LE IBM Z

1B+ 9.2K

Downloads Stars



alpinelinux/alpine-gitlab-ci  SPONSORED OSS


By alpinelinux • Updated 2 days ago


Build Alpine Linux packages with Gitlab CI

Linux ppc64le riscv64 IBM Z 386 x86-64 arm arm64

100K+ 3

Downloads Stars



alpinelinux/docker-cli  SPONSORED OSS

By alpinelinux • Updated 2 days ago

Simple and lightweight Alpine Linux image with docker-cli

Linux 386 x86-64 arm arm64 ppc64le riscv64 IBM Z

500K+ 4

Downloads Stars

Docker – Dockerfile

- Les instructions de base que peut contenir un DockerFile sont les suivantes :
 - **FROM** permet de définir depuis quelle base votre image va être créée
 - **LABEL** permet de définir l'auteur de l'image
 - **RUN** permet de lancer une commande
 - **ADD** permet de copier un fichier depuis la machine hôte ou depuis une URL
 - **EXPOSE** permet d'exposer un port du container vers l'extérieur
 - **CMD** détermine la commande qui sera exécutée lorsque le container démarrera
 - **ENTRYPOINT** permet d'ajouter une commande qui sera exécutée par défaut
 - **WORKDIR** permet de définir le dossier de travail pour toutes les autres commandes
 - **ENV** permet de définir des variables d'environnements
 - **VOLUMES** permet de créer un point de montage qui permettra de persister les données

Docker – Dockerfile

- Pour construire l'image docker défini dans le fichier Dockerfile cidessus :
docker build -t « Nom de l'image à créé » « Path vers le fichier »
- Vous pouvez spécifier . **(point)** si le fichier docker est dans le répertoire courant.
- Si vous n'utilisez pas le nom de Dockerfile par défaut, l'option -f est requise
- Dans le nom de l'image à créer, il faut préciser **votre login à dockerhub** pour pouvoir envoyer ton image sur DockerHub (**mouradhassini**/alpine).
- Voir commande **(page suivante)** : à partir de l'image officielle de alpine (sur DockerHub), on va créer notre propre image alpine:1.0.0 « customisée » avec java 11).
- Mettons-nous dans le dossier contenant le Dockerfile:

Docker – build

```

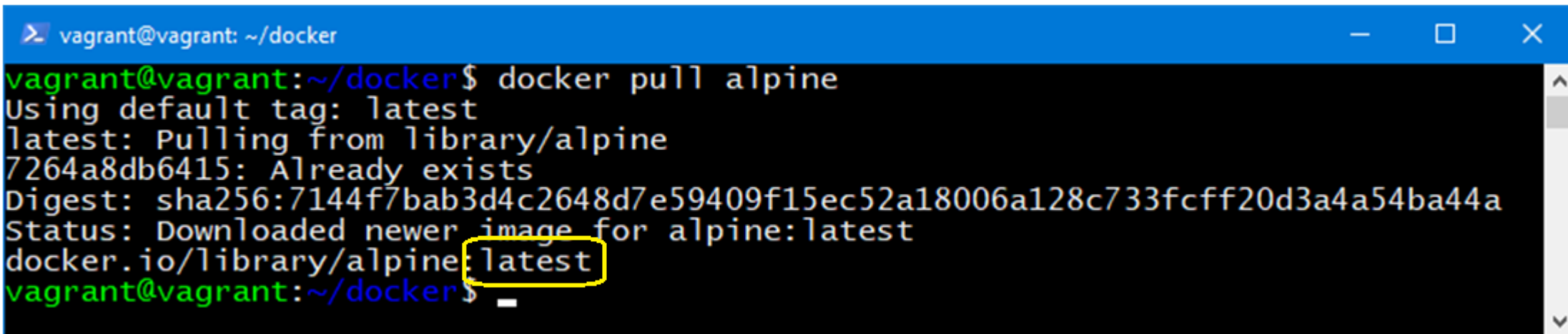
> vagrant@vagrant: ~/docker
vagrant@vagrant:~$ mkdir docker
vagrant@vagrant:~$ cd docker
vagrant@vagrant:~/docker$ vi Dockerfile
vagrant@vagrant:~/docker$ vagrant@vagrant:~/docker$ cat Dockerfile
FROM alpine
RUN apk add openjdk11
EXPOSE 80
CMD "java"
vagrant@vagrant:~/docker$ docker build -t mouradhassini/alpine:1.0.0 .
[+] Building 70.3s (6/6) FINISHED
=> [internal] load .dockerignore                                docker:default 0.1s
=> => transferring context: 2B                                  0.0s
=> [internal] load build definition from Dockerfile            0.1s
=> => transferring dockerfile: 92B                              0.0s
=> [internal] load metadata for docker.io/library/alpine:latest 9.9s
=> [1/2] FROM docker.io/library/alpine@sha256:7144f7bab3d4c2648d7e59409f15ec52a18006a128c733fcff20d3a4a54ba44a 2.0s
=> => resolve docker.io/library/alpine@sha256:7144f7bab3d4c2648d7e59409f15ec52a18006a128c733fcff20d3a4a54ba44a 0.1s
=> => sha256:c5c5fda71656f28e49ac9c5416b3643eaa6a108a8093151d6d1afc9463be8e33 528B / 528B 0.0s
=> => sha256:7e01a0d0a1dcd9e539f8e9bbd80106d59efbdf97293b3d38f5d7a34501526cdb 1.47kB / 1.47kB 0.0s
=> => sha256:7264a8db6415046d36d16ba98b79778e18acce6ffa71850405994cffa9be7de 3.40MB / 3.40MB 1.5s
=> => sha256:7144f7bab3d4c2648d7e59409f15ec52a18006a128c733fcff20d3a4a54ba44a 1.64kB / 1.64kB 0.0s
=> => extracting sha256:7264a8db6415046d36d16ba98b79778e18acce6ffa71850405994cffa9be7de 0.3s
=> [2/2] RUN apk add openjdk11                                  56.4s
=> exporting to image                                           1.8s
=> => exporting layers                                           1.8s
=> => writing image sha256:6cab9f55bce9fba3615ed857cca458a3545aad5a80ec47652b4e5b08cf814f43 0.0s
=> => naming to mouradhassini/alpine:1.0.0                      0.0s
vagrant@vagrant:~/docker$ docker images
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
mouradhassini/alpine 1.0.0       6cab9f55bce9  About a minute ago  273MB
hello-world         latest      9c7a54a9a43c  4 months ago   13.3kB
vagrant@vagrant:~/docker$
```

Docker – pull

- Extraire une image d'un registre:

docker pull «nom_image»:«version»

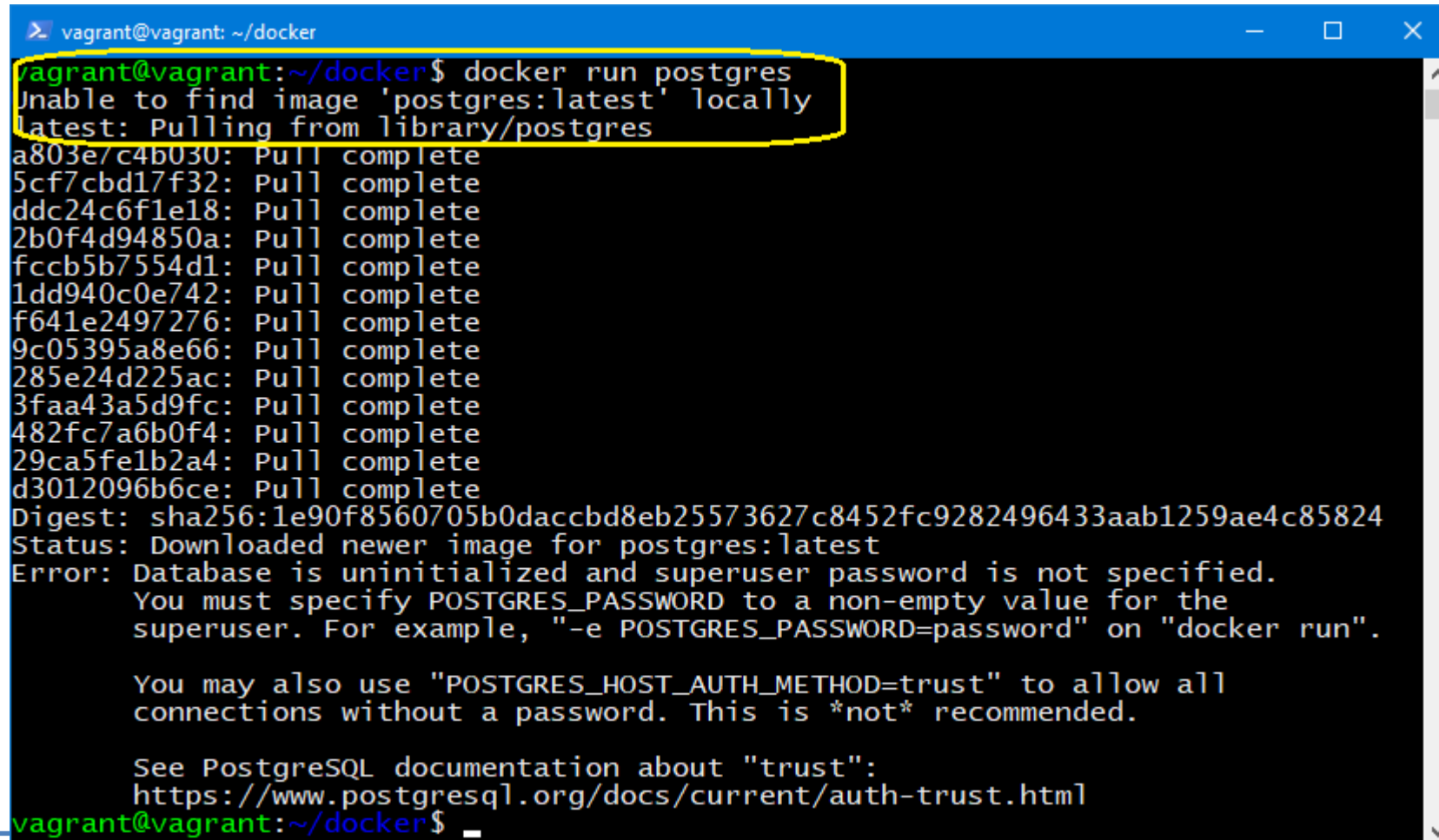
Si on ne spécifie pas la version, docker télécharge la dernière version



```
vagrant@vagrant: ~/docker
vagrant@vagrant:~/docker$ docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
7264a8db6415: Already exists
Digest: sha256:7144f7bab3d4c2648d7e59409f15ec52a18006a128c733fcff20d3a4a54ba44a
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
vagrant@vagrant:~/docker$
```

Docker – run

- Créer un conteneur: **docker run « nom de l'image »** . Si docker ne trouve pas l'image, il la télécharge de Dockerhub et après il crée le conteneur
- Exemple déjà fait page 23



```
vagrant@vagrant: ~/docker
vagrant@vagrant:~/docker$ docker run postgres
Unable to find image 'postgres:latest' locally
latest: Pulling from library/postgres
a803e/c4b030: Pull complete
5cf7cbd17f32: Pull complete
ddc24c6f1e18: Pull complete
2b0f4d94850a: Pull complete
fccb5b7554d1: Pull complete
1dd940c0e742: Pull complete
f641e2497276: Pull complete
9c05395a8e66: Pull complete
285e24d225ac: Pull complete
3faa43a5d9fc: Pull complete
482fc7a6b0f4: Pull complete
29ca5fe1b2a4: Pull complete
d3012096b6ce: Pull complete
Digest: sha256:1e90f8560705b0daccbd8eb25573627c8452fc9282496433aab1259ae4c85824
Status: Downloaded newer image for postgres:latest
Error: Database is uninitialized and superuser password is not specified.
You must specify POSTGRES_PASSWORD to a non-empty value for the
superuser. For example, "-e POSTGRES_PASSWORD=password" on "docker run".

You may also use "POSTGRES_HOST_AUTH_METHOD=trust" to allow all
connections without a password. This is *not* recommended.

See PostgreSQL documentation about "trust":
https://www.postgresql.org/docs/current/auth-trust.html
vagrant@vagrant:~/docker$
```

Docker – images

- Récupérer la liste des images:

```
vagrant@vagrant: ~/docker
vagrant@vagrant:~/docker$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
mouradhassini/alpine 1.0.0              6cab9f55bce9       12 minutes ago     273MB
postgres             latest             ec7f99c50d3c       4 days ago         418MB
alpine               latest             7e01a0d0a1dc       6 weeks ago        7.34MB
hello-world          latest             9c7a54a9a43c       4 months ago       13.3kB
vagrant@vagrant:~/docker$
```


Docker – ps

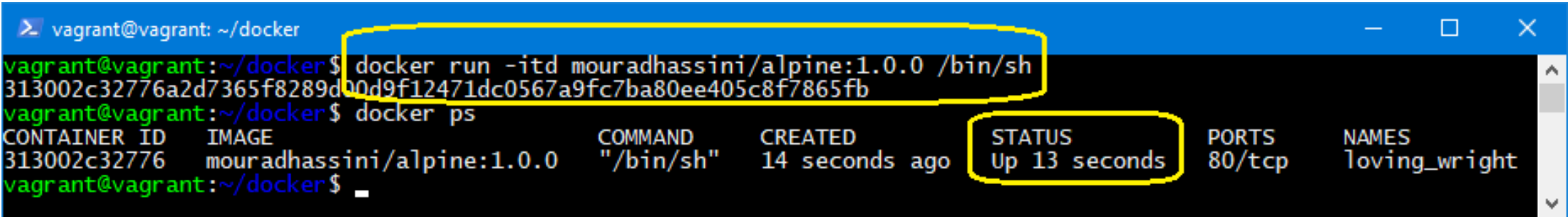
- Récupérer la liste des conteneurs: **docker ps -a**
(l'option -a ou --all veut dire **all**)
- Récupérer la liste des conteneurs **actifs seulement**: **docker ps**

```
vagrant@vagrant: ~/docker
vagrant@vagrant:~/docker$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
9f903569b63c   postgres      "docker-entrypoint.s..." 7 minutes ago  Exited (1)    7 minutes ago  jolly_murdock
cf063f7757d8   hello-world    "/hello"                 38 minutes ago Exited (0)    37 minutes ago affectionate_haibt
vagrant@vagrant:~/docker$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
vagrant@vagrant:~/docker$
```

- Pourquoi on ne voit pas de conteneur alpine:1.0.0?

Docker – -d

- Lancer un conteneur et le laisser tourner en **background** (avec l'option **-d** : **docker run -itd**) pour pouvoir y accéder plus tard :
- La commande ci-dessous, lance un conteneur et lance dedans un client ssh (invite de commande) :



A terminal window titled 'vagrant@vagrant: ~/docker' showing the execution of Docker commands. The first command, 'docker run -itd mouradhassini/alpine:1.0.0 /bin/sh', is highlighted with a yellow box. The second command, 'docker ps', is also highlighted. Below the command, the output of 'docker ps' is shown as a table with columns: CONTAINER ID, IMAGE, COMMAND, CREATED, STATUS, PORTS, and NAMES. The first row of data is highlighted with a yellow box.

```
vagrant@vagrant: ~/docker
vagrant@vagrant:~/docker$ docker run -itd mouradhassini/alpine:1.0.0 /bin/sh
313002c32776a2d7365f8289d00d9f12471dc0567a9fc7ba80ee405c8f7865fb
vagrant@vagrant:~/docker$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
313002c32776	mouradhassini/alpine:1.0.0	"/bin/sh"	14 seconds ago	Up 13 seconds	80/tcp	loving_wright

```
vagrant@vagrant:~/docker$
```

Docker – exec

- Pour accéder au conteneur déjà lancé (avec **docker exec**).
- Vous pouvez manipuler le conteneur :

```
vagrant@vagrant: ~/docker
vagrant@vagrant:~/docker$ docker run -itd mouradhassini/alpine:1.0.0 /bin/sh
798e0b8ecf12bcb9158c5892476acd29b3b3306a626fd15b49de4718a73535eb
vagrant@vagrant:~/docker$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS        NAMES
798e0b8ecf12   mouradhassini/alpine:1.0.0         "/bin/sh"              25 seconds ago Up 24 seconds 80/tcp        objective_dhawan
vagrant@vagrant:~/docker$ docker exec -it 798e0 /bin/sh
/# pwd
/# ls
bin  dev  etc  home  lib  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
/# _
```

Docker – rmi

- Supprimer une image: **docker image rm «nom image » --force**
- ou **docker rmi « nom image » --force**

```
Sélection vagrant@vagrant: ~/docker
vagrant@vagrant:~/docker$ docker images
REPOSITORY          TAG          IMAGE ID      CREATED        SIZE
mouradmardini/alpine 1.0.0       6cab9f55bce9  28 minutes ago 273MB
postgres            latest      ec7f99c50d3c  4 days ago    418MB
alpine              latest      7e01a0d0a1dc  6 weeks ago    7.34MB
hello-world         latest      9c7a54a9a43c  4 months ago   13.3kB
vagrant@vagrant:~/docker$ docker rmi hello-world --force
Untagged: hello-world:latest
Untagged: hello-world@sha256:4f53e2564790c8e7856ec08e384732aa38dc43c52f02952483e3f003afbf23db
Deleted: sha256:9c7a54a9a43cca047013b82af109fe963fde787f63f9e016fdc3384500c2823d
vagrant@vagrant:~/docker$ docker images
REPOSITORY          TAG          IMAGE ID      CREATED        SIZE
mouradmardini/alpine 1.0.0       6cab9f55bce9  29 minutes ago 273MB
postgres            latest      ec7f99c50d3c  4 days ago    418MB
alpine              latest      7e01a0d0a1dc  6 weeks ago    7.34MB
vagrant@vagrant:~/docker$
```

Docker – rm

- Supprimer un conteneur : **docker rm « Id du conteneur »**

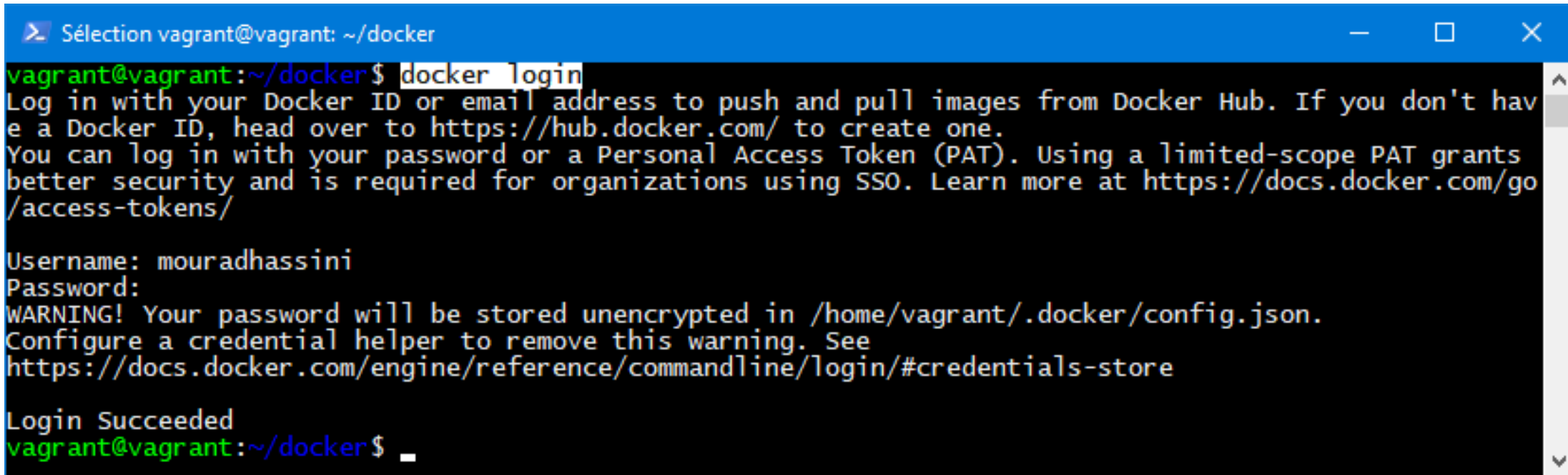
```
Sélection vagrant@vagrant: ~/docker
vagrant@vagrant:~/docker$ docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS              PORTS          NAMES
29fe6a8e2a02   mouradhassini/alpine:1.0.0         "/bin/sh"               9 seconds ago  Up 7 seconds       80/tcp         zen_solomon
9f903569b63c   postgres                            "docker-entrypoint.s..." 26 minutes ago Exited (1) 26 minutes ago          jolly_murdock
cf063f7757d8   9c7a54a9a43c                       "/hello"                56 minutes ago Exited (0) 56 minutes ago          affectionate_haibt

vagrant@vagrant:~/docker$ docker remove 29fe6
Error response from daemon: You cannot remove a running container 29fe6a8e2a02045d32214799e992538cc3ebca8cd8fb1cfbc2306514cf392ab2. Stop the container before attempting removal or force remove
vagrant@vagrant:~/docker$ docker stop 29fe6
29fe6
vagrant@vagrant:~/docker$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS      PORTS          NAMES
vagrant@vagrant:~/docker$ docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS              PORTS          NAMES
29fe6a8e2a02   mouradhassini/alpine:1.0.0         "/bin/sh"               About a minute ago Exited (137) 19 seconds ago          zen_solomon
9f903569b63c   postgres                            "docker-entrypoint.s..." 27 minutes ago Exited (1) 27 minutes ago          jolly_murdock
cf063f7757d8   9c7a54a9a43c                       "/hello"                58 minutes ago Exited (0) 57 minutes ago          affectionate_haibt

vagrant@vagrant:~/docker$ docker remove 29fe6
29fe6
vagrant@vagrant:~/docker$ docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS              PORTS          NAMES
9f903569b63c   postgres                            "docker-entrypoint.s..." 28 minutes ago Exited (1) 27 minutes ago          jolly_murdock
cf063f7757d8   9c7a54a9a43c                       "/hello"                58 minutes ago Exited (0) 58 minutes ago          affectionate_haibt
vagrant@vagrant:~/docker$
```

Docker – login

- Créer un compte sur DockerHub <https://hub.docker.com> (Si ce n'est déjà fait), et s'y connecter (**docker login**) :

A terminal window titled 'Sélection vagrant@vagrant: ~/docker' with standard window controls. The terminal shows the command 'docker login' being executed. It displays instructions for logging in with a Docker ID or email, followed by a warning about unencrypted passwords and a successful login message.

```
Sélection vagrant@vagrant: ~/docker
vagrant@vagrant:~/docker$ docker login
Log in with your Docker ID or email address to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com/ to create one.
You can log in with your password or a Personal Access Token (PAT). Using a limited-scope PAT grants better security and is required for organizations using SSO. Learn more at https://docs.docker.com/go/access-tokens/

Username: mouradhassini
Password:
WARNING! Your password will be stored unencrypted in /home/vagrant/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
vagrant@vagrant:~/docker$ _
```

Docker – push

- Envoyer une image que vous avez créée sur dockerhub (**docker push**) :

```
vagrant@vagrant: ~/docker
vagrant@vagrant:~/docker$ docker images
REPOSITORY          TAG          IMAGE ID          CREATED           SIZE
mouradhassini/alpine 1.0.0        6cab9f55bce9      46 minutes ago   273MB
postgres             latest       ec7f99c50d3c      4 days ago       418MB
alpine               latest       7e01a0d0a1dc      6 weeks ago      7.34MB
vagrant@vagrant:~/docker$ docker push mouradhassini/alpine
Using default tag: latest
The push refers to repository [docker.io/mouradhassini/alpine]
tag does not exist: mouradhassini/alpine:latest
vagrant@vagrant:~/docker$ docker push mouradhassini/alpine:1.0.0
The push refers to repository [docker.io/mouradhassini/alpine]
892937171735: Pushed
4693057ce236: Mounted from library/alpine

1.0.0: digest: sha256:91db88c2c261cd038f6a477d681bb6c36a4353dc970857513b0eb0d8bf908420 size: 741
vagrant@vagrant:~/docker$
```

Docker – dockerhub

The screenshot shows the Docker Hub web interface. The browser's address bar displays 'hub.docker.com'. The navigation bar includes a search bar, links for 'Explore', 'Repositories', 'Organizations', and 'Help', an 'Upgrade' button, and a user profile dropdown menu labeled 'mouradhassini'. Below the navigation bar, there is a search bar with 'mouradhassini' entered and a 'Create repository' button. The main content area lists repositories for the user 'mouradhassini'. The first repository, 'mouradhassini / alpine', is highlighted with a red box. It shows 'Last pushed: a minute ago', 'Not Scanned', 0 stars, 0 downloads, and is 'Public'. The second repository, 'mouradhassini / repo1', shows 'Last pushed: 8 months ago', 'Not Scanned', 0 stars, 14 downloads, and is 'Public'.

Repository	Last pushed	Not Scanned	Stars	Downloads	Visibility
mouradhassini / alpine	a minute ago	Not Scanned	0	0	Public
mouradhassini / repo1	8 months ago	Not Scanned	0	14	Public

Docker

