

# Introduction à Docker



**Bureau E204**

# Plan du cours

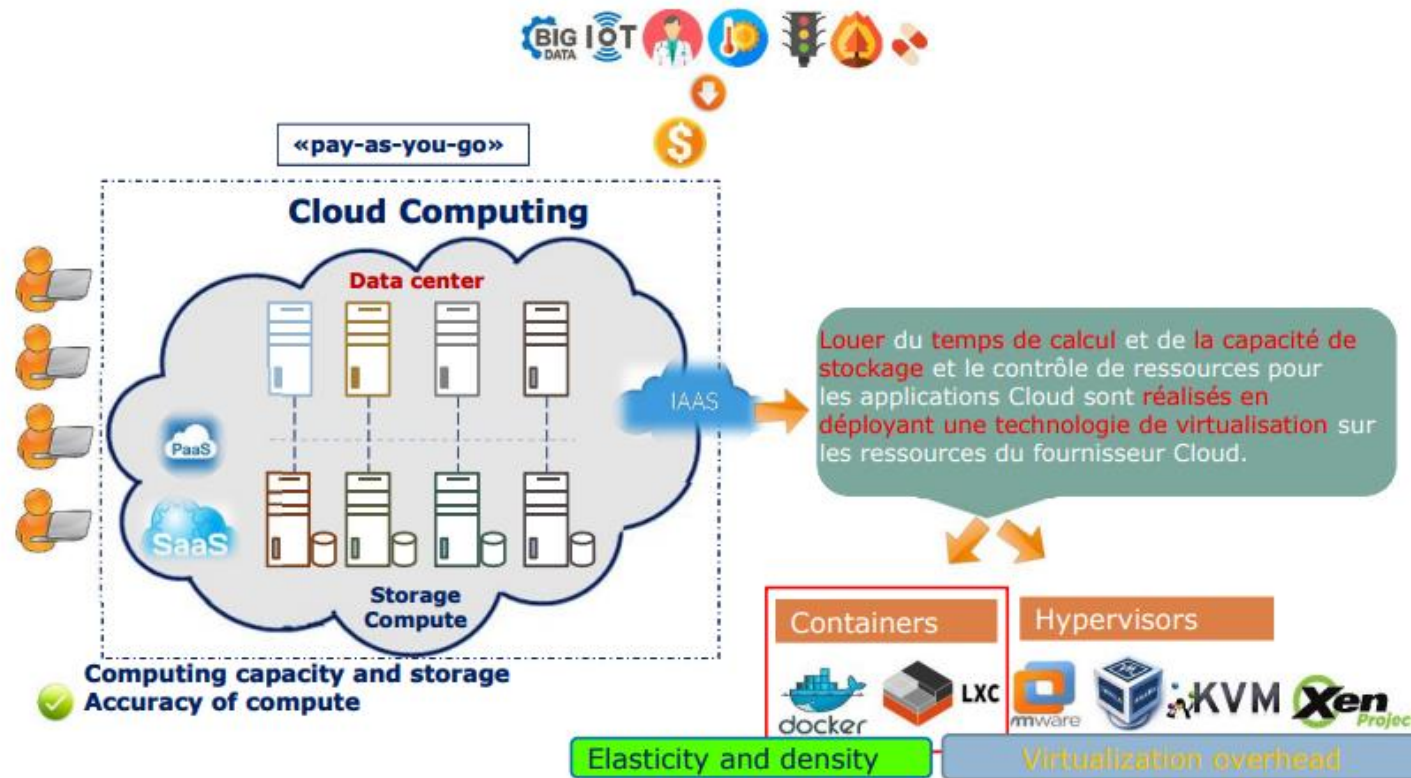
- La virtualisation
- La containerisation
- La virtualisation vs la containerisation
- Docker
  - Définition
  - Avantages et inconvénients
  - Composants
  - Installation
  - Manipulation
  - Docker Hub
  - Docker File
  - Commandes de base

# La virtualisation

- **La virtualisation:** La virtualisation est une technologie permettant de créer et d'exécuter une ou plusieurs représentations virtuelles d'un ordinateur ou 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

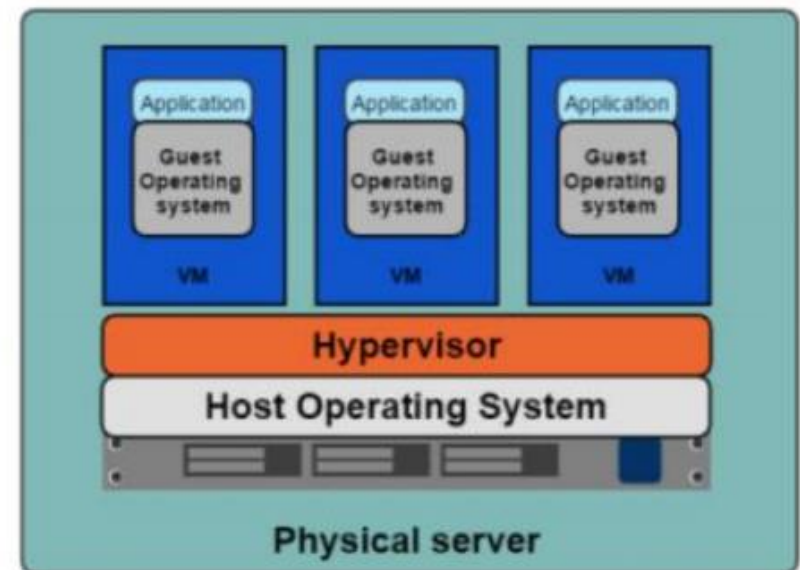


- On distingue plusieurs types de virtualisation tels que:
  - ✓ La virtualisation lourde
  - ✓ La virtualisation légère

# La virtualisation lourde

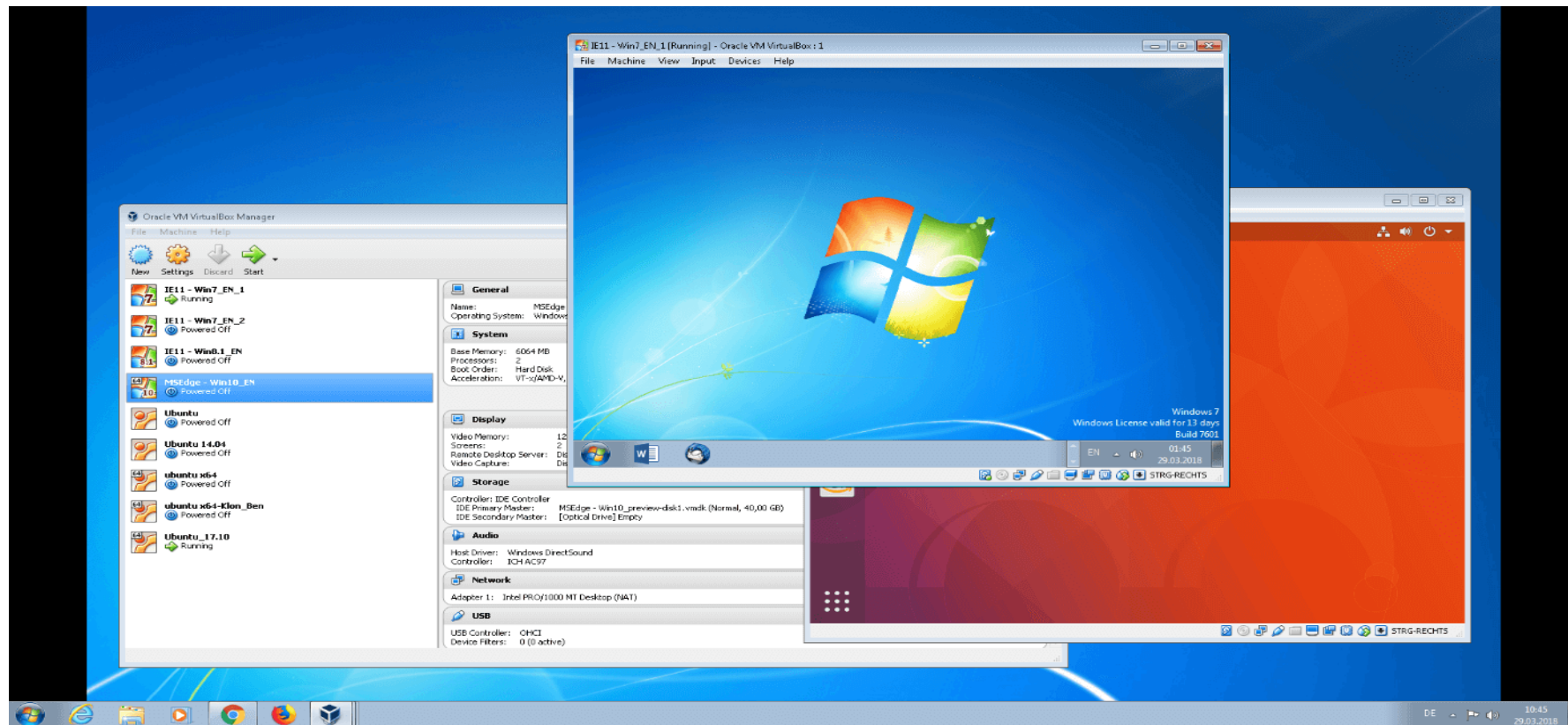
**La virtualisation Lourde ou à base d'hyperviseur:** Elle permet de simuler une ou plusieurs machines physiques, et les exécuter sous forme de machines virtuelles (VM) sur un serveur. Ces VM intègrent elles-mêmes un OS sur lequel des applications sont exécutées.

Les machines virtuelles intègrent un OS.

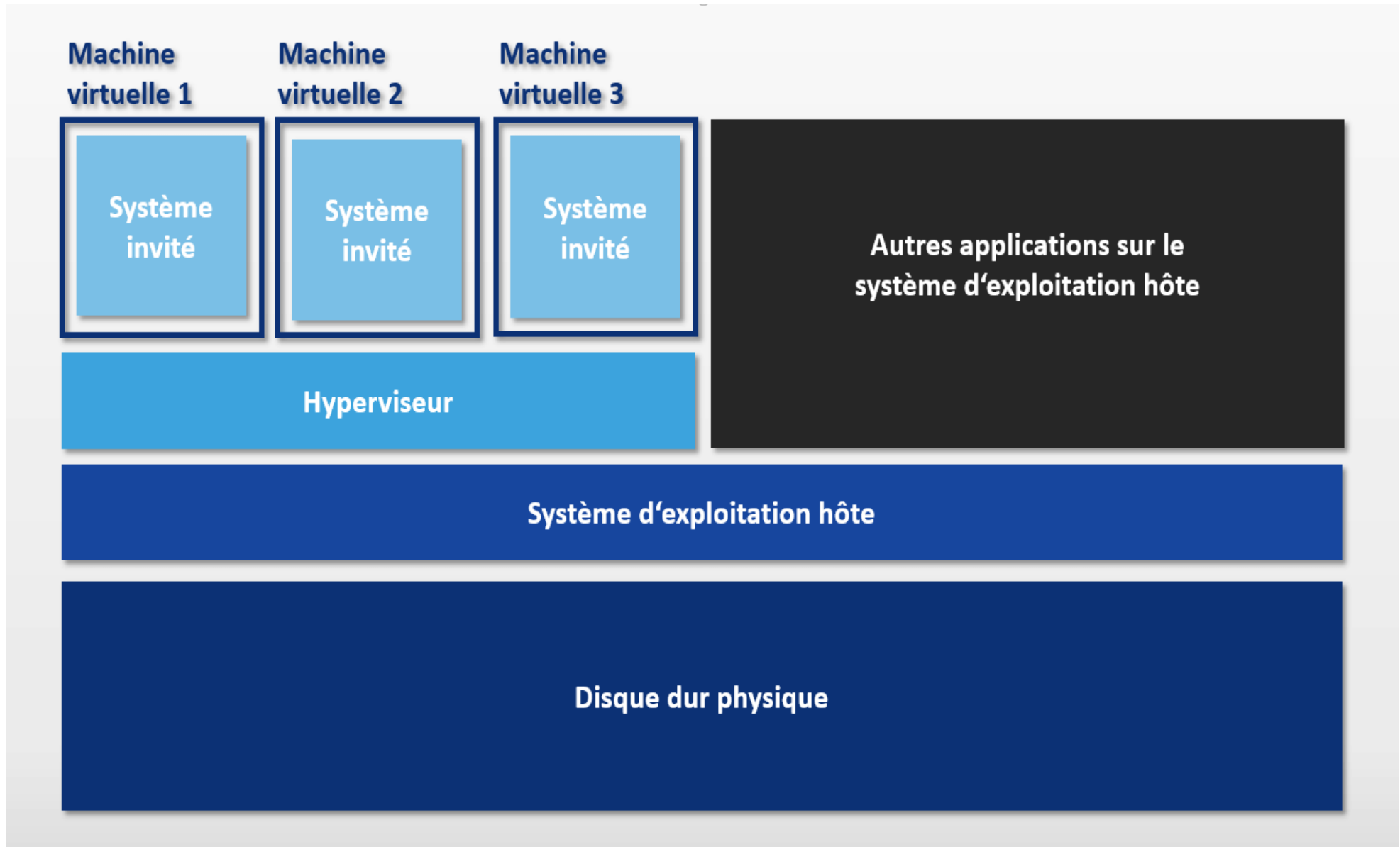


# La virtualisation lourde

Les machines virtuelles intègrent elles-mêmes un OS sur lequel des applications sont exécutées (exemple VMWare) .



# La virtualisation lourde



# La virtualisation lourde

- **Avantages des VMs**

- ✓ Une bonne exploitation des ressources
- ✓ Une machine physique est divisée en plusieurs machines virtuelles
- ✓ Plus facile de passer à l'échelle

- **Limites des VMs**

- ✓ Chaque VM a besoin des ressources (CPU, Stockage (Disque), RAM ,Un OS invité)
- ✓ En augmentant les VMs, on demande plus de ressources
  - Chaque OS invité alloue ses propres ressources
  - Gaspillage
  - Portabilité d'application n'est pas garantie



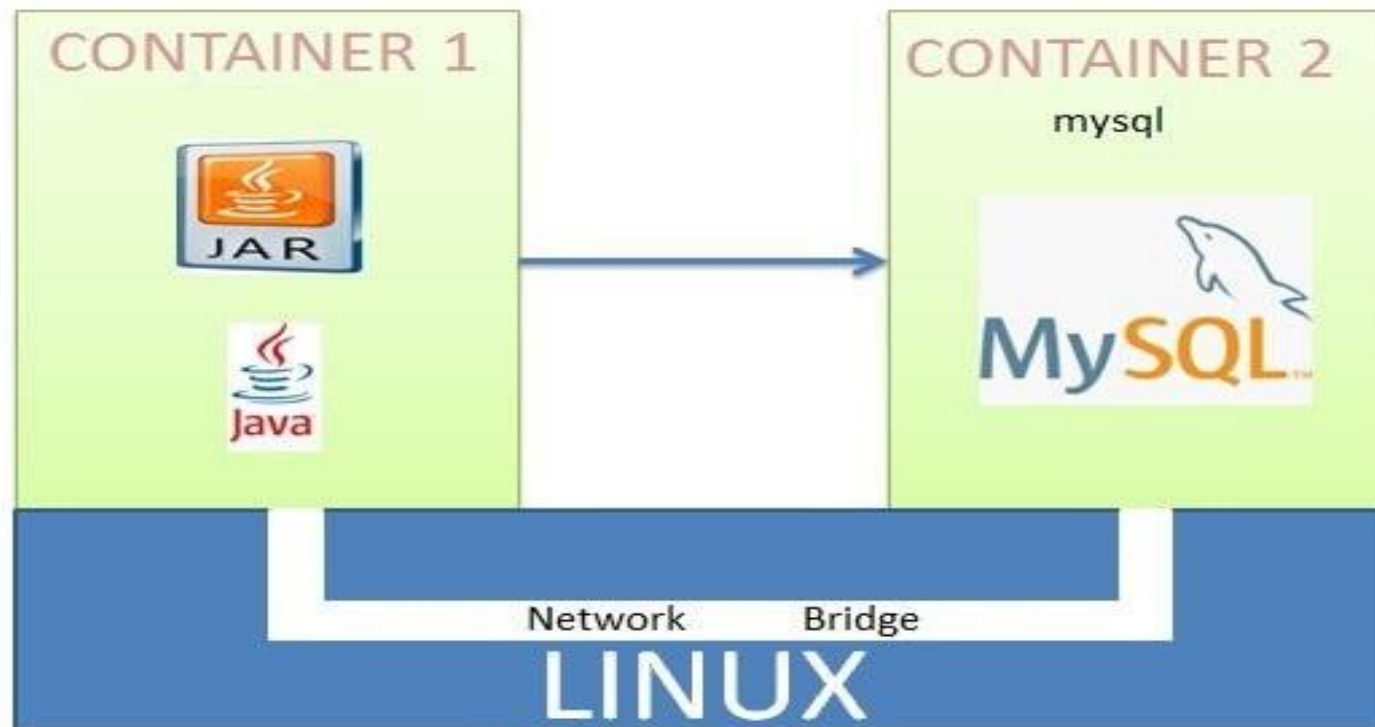
# La virtualisation légère: La containerisation

- ✓ La conteneurisation permet de packager tous les services, scripts, API, librairies dont une application a besoin favorisant ainsi une juste utilisation des ressources.
- ✓ La conteneurisation repose sur la création de conteneurs isolés les uns des autres sur un noyau commun.
- ✓ Les conteneurs indépendants partagent un noyau commun (donc un ou plusieurs systèmes d'exploitation) et un même espace mémoire.

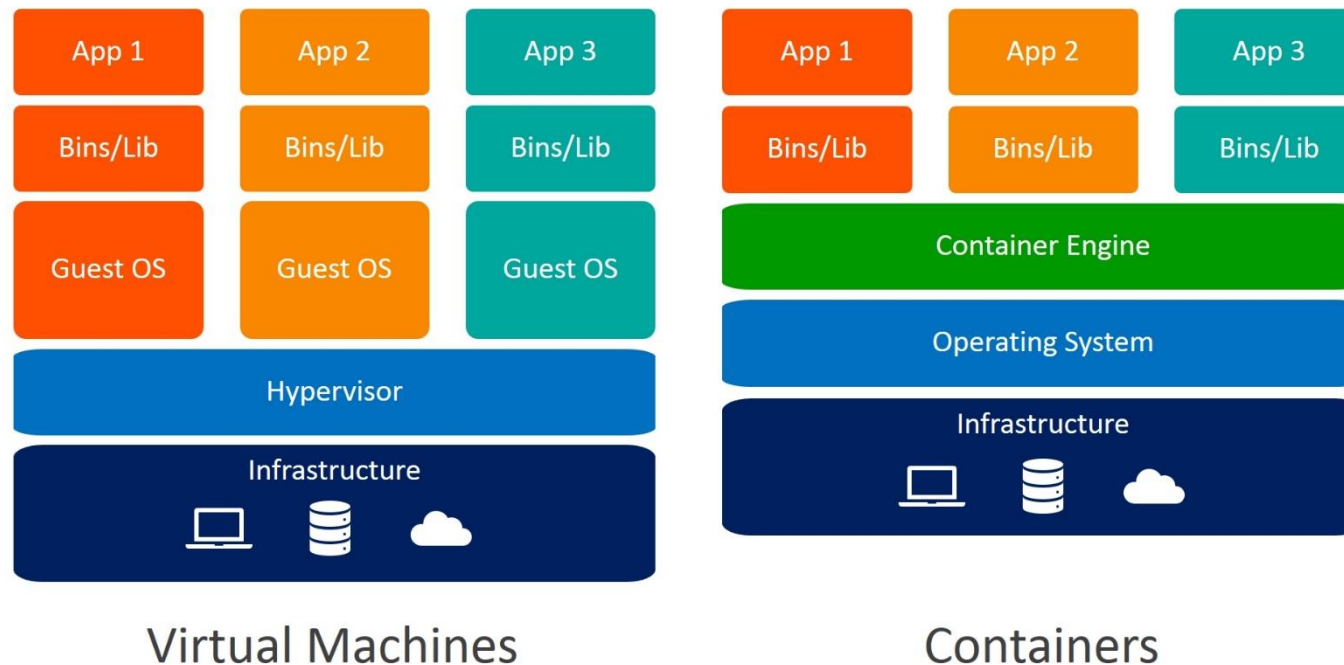


# La virtualisation légère: La containerisation

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)



# La virtualisation vs la containerisation

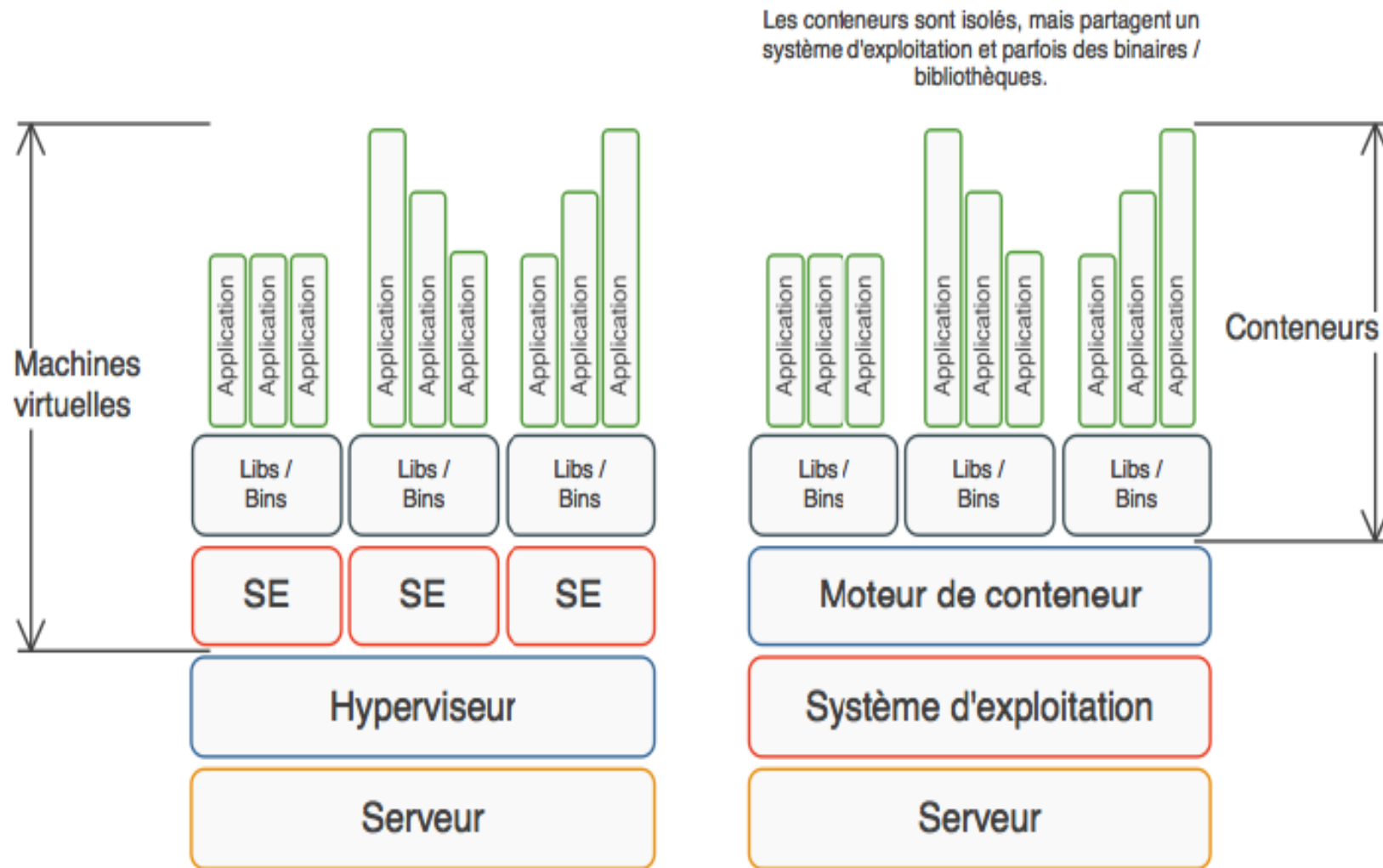


Virtual Machines

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.

# La virtualisation vs la containerisation

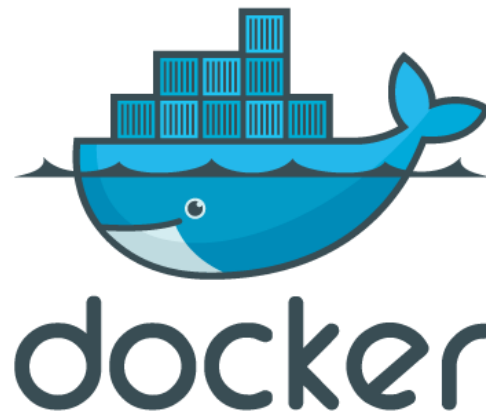


# La virtualisation vs Containerisation

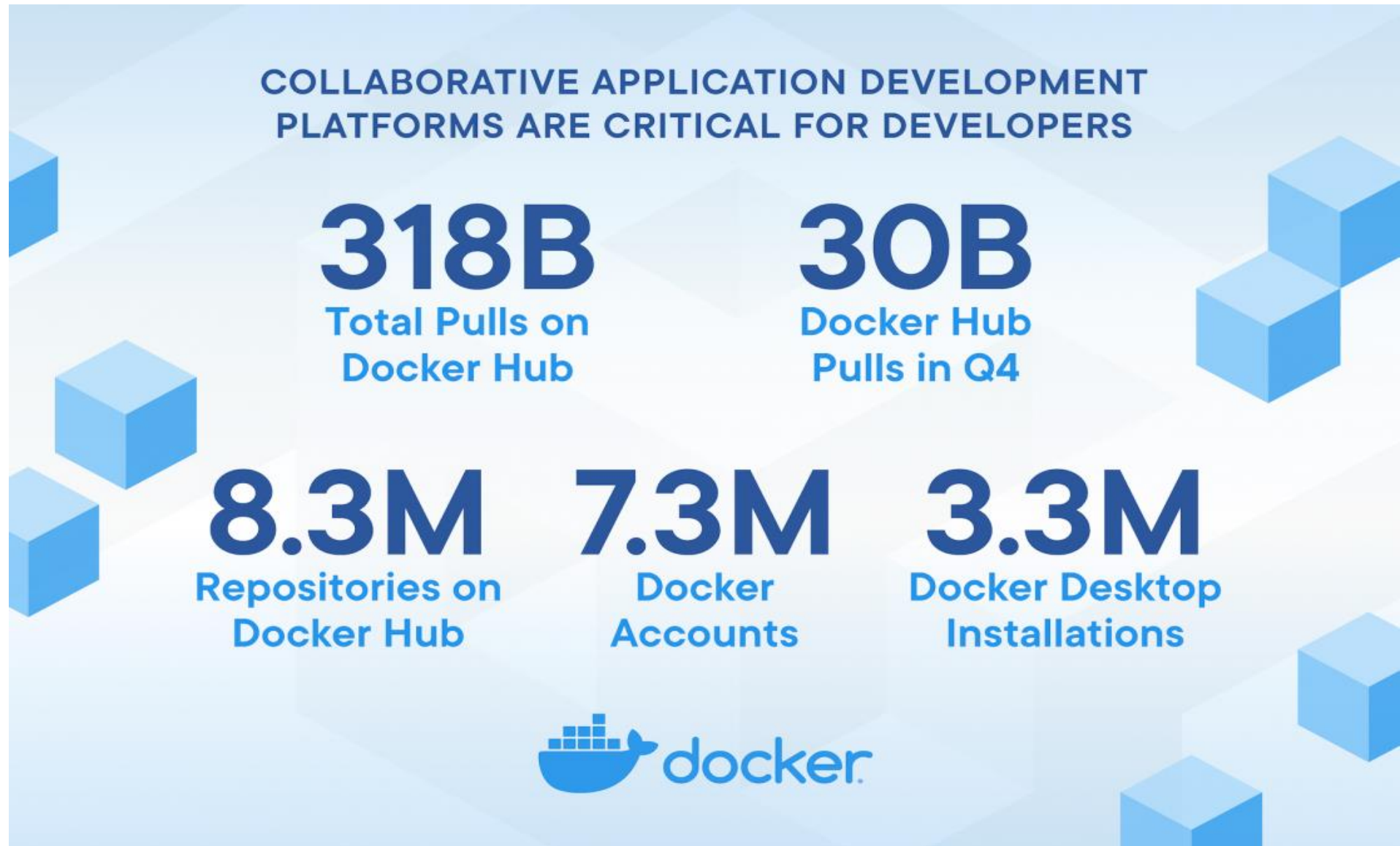
Virtualisation	Containerisation
Une virtualisation lourde	Une virtualisation légère
À base d'hyperviseur	À base des conteneurs
Virtualisation des ressources <b>hardware</b> (CPU, RAM, disque, ...)	Virtualisation au niveau du système d'exploitation
Machine invité (machine virtuelle)	Conteneur
Image ISO	Image Os ou image outil (OS + un outil installé)
Machine hôte	Moteur de conteneur
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 Entreprise Edition (Docker EE): Dockeree payante
  - Docker Community Edition (Docker CE): Dockeree gratuite



# Docker - Définition





# Docker - Avantages

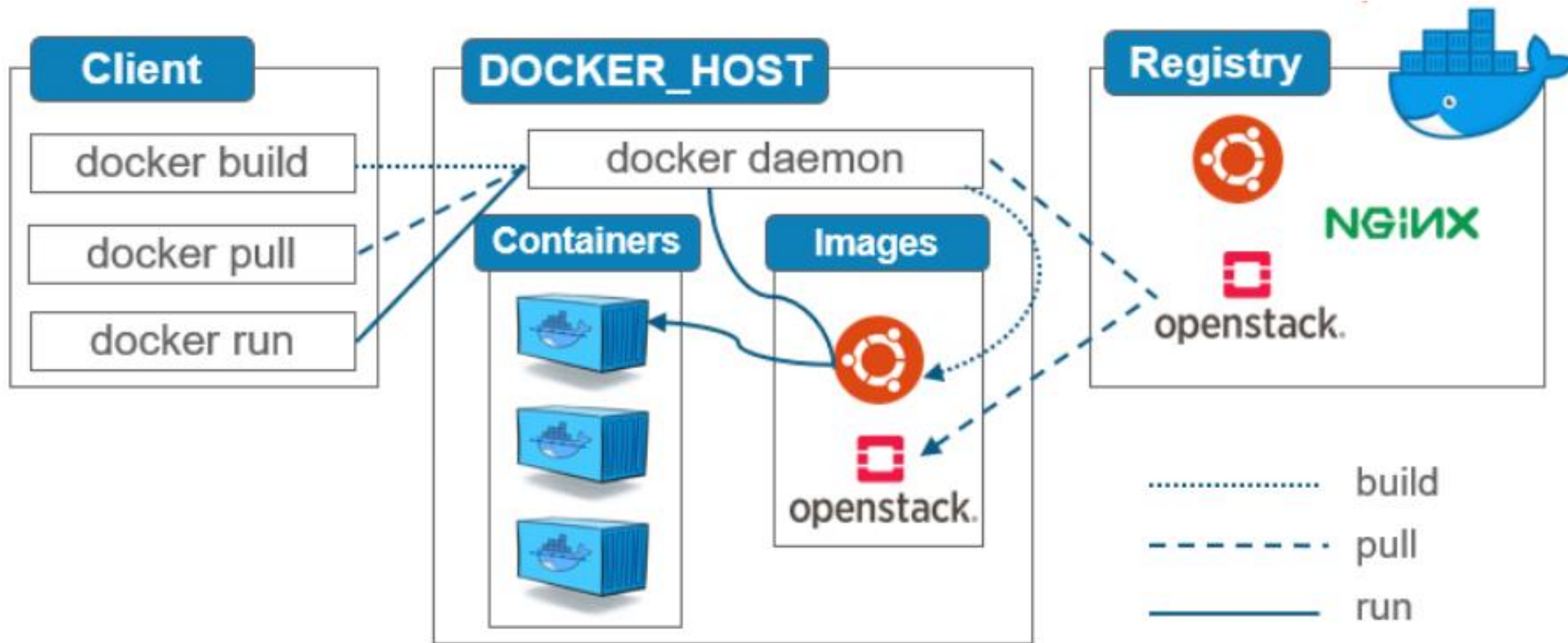
- **Flexible:** Même les applications les plus complexes peuvent être conteneurisées.
- **Léger:** Les conteneurs exploitent et partagent le noyau hôte, ce qui les rend beaucoup plus efficaces en termes de ressources système que les machines virtuelles.
- **Portable:** Moins de dépendances avec la machine hôte.
- **Faiblement couplé:** Les conteneurs sont hautement autonomes et encapsulés, ce qui vous permet de remplacer ou de mettre à niveau l'un sans en perturber les autres.
- **Évolutif:** Vous pouvez augmenter et distribuer automatiquement les répliques de conteneurs dans un centre de données.



# Docker - Inconvénients

- Docker présente quelques inconvénients parmi lesquelles nous pouvons citer :
  - ✓ La portabilité des containers entre les différents systèmes d'exploitation (Windows, linux, etc..) est très délicate.
  - ✓ La difficulté de gérer plusieurs conteneurs simultanément.
  - ✓ Possible failles de sécurité (conteneurs partagent le même système d'exploitation)

# Docker - Composants



# Docker - Composants

**Client Docker:** Les utilisateurs de Docker peuvent interagir avec Docker via un client. Lorsqu'une commande docker s'exécute, le client les envoie au démon docker, qui les exécute. L'API Docker est utilisée par les commandes Docker. Le client Docker peut communiquer avec plus d'un démon.

**Docker Host:** 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

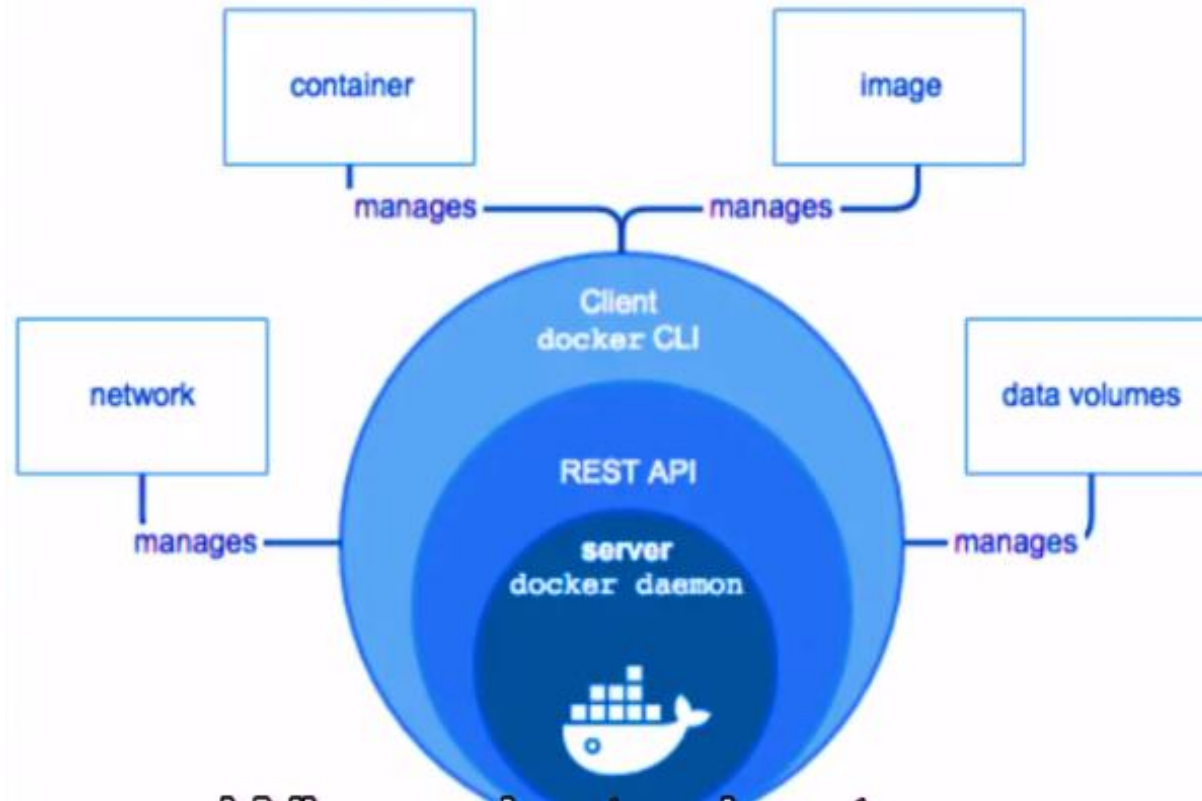
**Images:** Ils sont des modèles en lecture seule avec des instructions pour créer un conteneur Docker. L'image Docker peut être extraite d'un hub Docker et utilisée telle quelle, ou vous pouvez ajouter des instructions supplémentaires à l'image de base et créer une image Docker nouvelle et modifiée.

**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 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

- Avant de faire l'installation, il faut vérifier que la machine est compatible.
- Pour installer docker sur la machine, il suffit de poursuivre les commandes suivantes: (Si vous utilis  un compte 'super user', ex cutez la commande sans **sudo**)

✓ sudo yum update

```
[root@localhost vagrant]# sudo yum update
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
* base: centos.crazyfrogs.org
* extras: centos.crazyfrogs.org
* updates: centos.crazyfrogs.org
```

✓ sudo yum install -y yum-utils device-mapper-persistent-data lvm2 (**T l charger les d pendances n cessaires   l'installation de Docker**)

```
[root@localhost vagrant]# sudo yum install -y yum-utils device-mapper-persistent-data lvm2
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
* base: it2.mirror.vhosting-it.com
* extras: centos.mirror.ate.info
* updates: centos.mirror.ate.info
Package yum-utils-1.1.31-54.el7_8.noarch already installed and latest version
Resolving Dependencies
--> Running transaction check
```

# Docker – Installation

- ✓ `sudo yum-config-manager --add-repo`

<https://download.docker.com/linux/centos/docker-ce.repo> (**Ajouter le dépôt stable de Docker CE à votre système**)

```
[root@localhost vagrant]# sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
Loaded plugins: fastestmirror
adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
grabbing file https://download.docker.com/linux/centos/docker-ce.repo to /etc/yum.repos.d/docker-ce.repo
```

- ✓ `sudo yum install docker`

```
[root@localhost vagrant]# sudo yum install docker
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
```

- ✓ `sudo systemctl start docker`

```
[root@localhost vagrant]# sudo systemctl start docker
```

- ✓ `sudo systemctl enable docker` (**Pour lancer Docker comme un service lors du démarrage de la machine**)

```
[root@localhost vagrant]# sudo systemctl enable docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service
```

# Docker – Installation

- ✓ `sudo systemctl status docker`

```
[root@localhost vagrant]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Wed 2022-07-20 08:28:04 UTC; 13h ago
     Docs: https://docs.docker.com
  Main PID: 2414 (dockerd)
    Tasks: 9
   Memory: 150.8M
    CGroup: /system.slice/docker.service
            └─2414 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

- ✓ `sudo chmod 666 /var/run/docker.sock` (Donner les droits d'accès en lecture et en écriture)

```
[root@localhost vagrant]# sudo chmod 666 /var/run/docker.sock
```

- ✓ Pour s'assurer que tout est bon :

- ✓ Vérifier la version de docker : `docker -v`

```
[root@localhost vagrant]# docker -v
Docker version 20.10.17, build 100c701
```

- ✓ Lancer l'image «Hello-World» : `docker run hello-world`

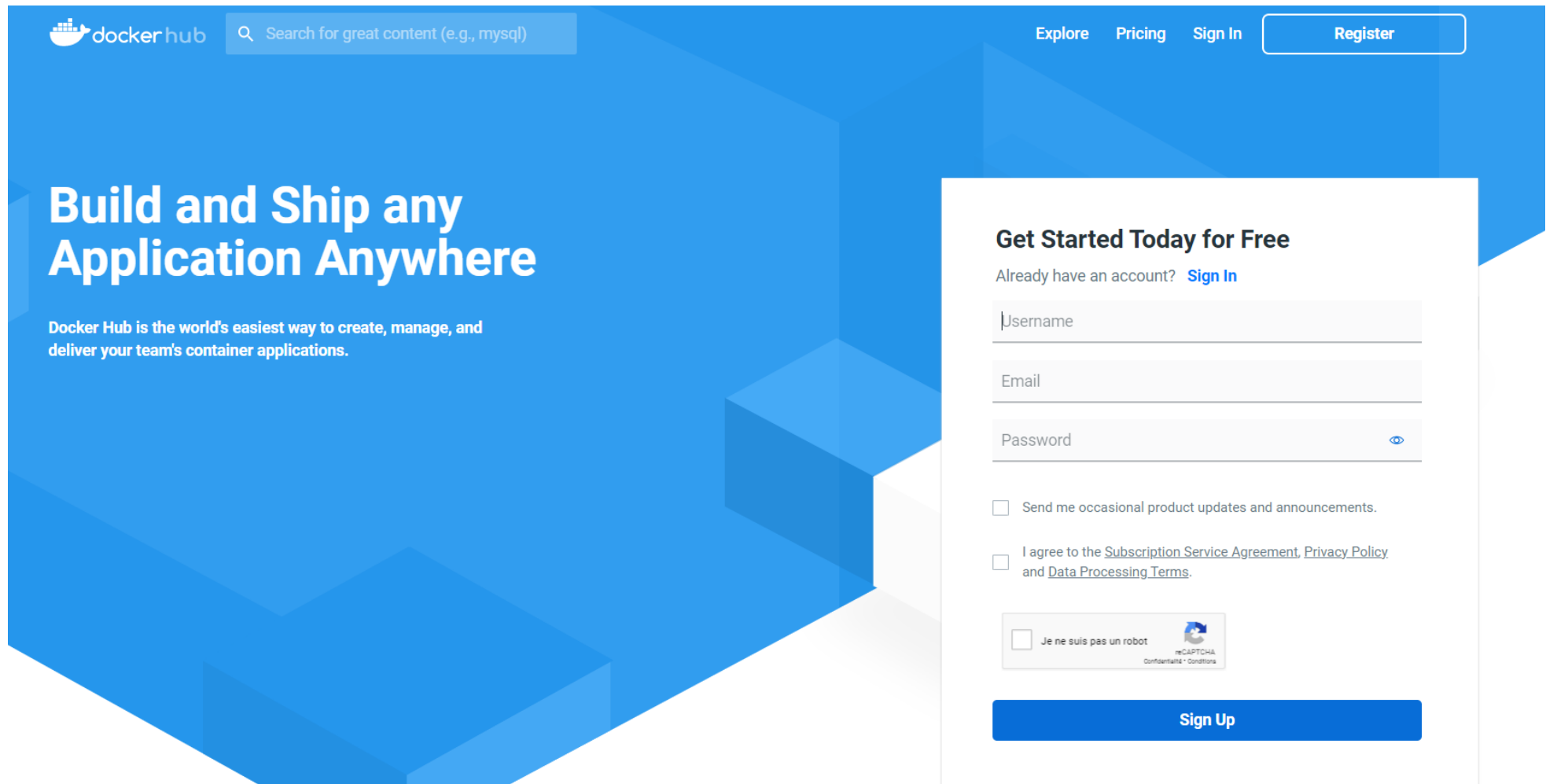
```
[root@localhost vagrant]# docker run hello-world

Hello from Docker!
```



# Docker – Installation

- ✓ Créer un compte « Docker Hub »: <https://hub.docker.com/>



The screenshot shows the Docker Hub website with a blue background and geometric shapes. The main heading is "Build and Ship any Application Anywhere". Below it, a subheading states: "Docker Hub is the world's easiest way to create, manage, and deliver your team's container applications." The top navigation bar includes the Docker Hub logo, a search bar, and links for "Explore", "Pricing", "Sign In", and "Register". A registration modal is open on the right side, titled "Get Started Today for Free". It contains a "Sign In" link for existing users, input fields for "Username", "Email", and "Password", and checkboxes for "Send me occasional product updates and announcements.", "I agree to the Subscription Service Agreement, Privacy Policy and Data Processing Terms.", and a reCAPTCHA checkbox labeled "Je ne suis pas un robot". A blue "Sign Up" button is at the bottom of the modal.

**Build and Ship any Application Anywhere**

Docker Hub is the world's easiest way to create, manage, and deliver your team's container applications.

**Get Started Today for Free**

Already have an account? [Sign In](#)

Username

Email

Password

☐ Send me occasional product updates and announcements.

☐ I agree to the [Subscription Service Agreement](#), [Privacy Policy](#) and [Data Processing Terms](#).

☐ Je ne suis pas un robot

**Sign Up**

# 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/opérationnels et accessibles.
- Ces images peuvent également être fournies par Docker.
- Il est possible de télécharger ces images et de partager les vôtres.




# Docker – Manipulation


- Pour importer 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 local (sans accéder au cloud)
  - A partir d'un conteneur ( Image → Conteneur → Image)

# Docker – Docker Hub

<https://hub.docker.com/>



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

[Upgrade](#)  [sirinenaifar](#)

Filters

Products

☐ Images

☐ Extensions

☐ Plugins

Trusted Content

☐  Docker Official Image

☐  Verified Publisher

☐  Sponsored OSS

Operating Systems

☐ Linux

☐ Windows

Architectures

☐ ARM

1 - 25 of 9 672 526 available results.



ubuntu

 DOCKER OFFICIAL IMAGE

Updated 19 days ago

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

Linux

PowerPC 64 LE

riscv64

IBM Z

386

x86-64

ARM

ARM 64

1B+

10K+

Downloads

Stars



redis

 DOCKER OFFICIAL IMAGE

Updated 3 days ago

Redis is an open source key-value store that functions as a data structure server.

Windows

Linux

ARM 64

386

mips64le

PowerPC 64 LE

IBM Z

x86-64

ARM

1B+

10K+

Downloads

Stars



alpine

 DOCKER OFFICIAL IMAGE

Updated 2 months ago

1B+

9.2K

Downloads

Stars

Suggested

# Docker – Docker Hub

<https://hub.docker.com/>

ghassenhammouda

▼

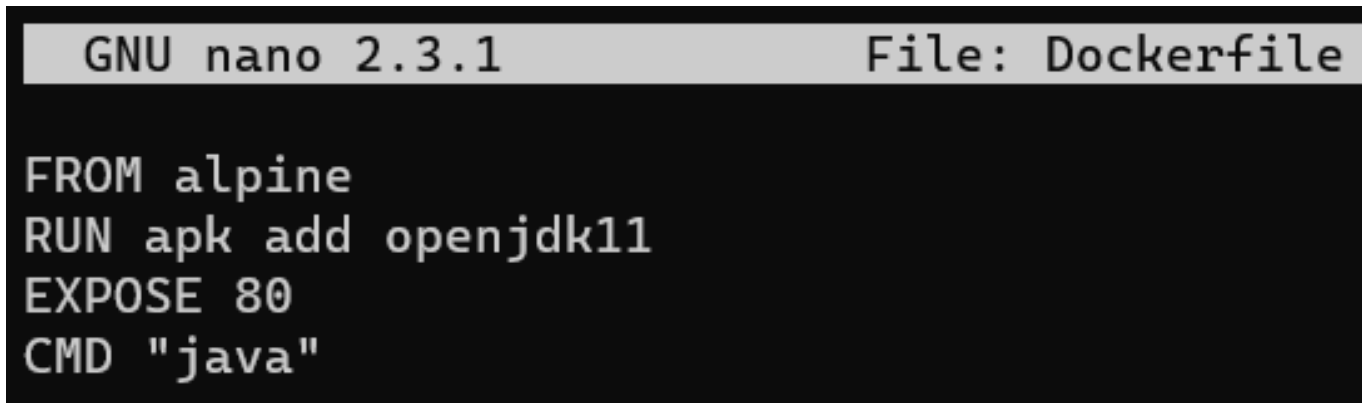
🔍 Search by repository name

Create repository

ghassenhammouda / <b>restaurant</b> Last pushed: 6 days ago	🛡️ Not Scanned	☆ 0	↓ 6	🌐 Public
ghassenhammouda / <b>helloworld</b> Last pushed: a year ago	🛡️ Not Scanned	☆ 0	↓ 0	🌐 Public
ghassenhammouda / <b>project-spring-data-jpa</b> Last pushed: a year ago	🛡️ Not Scanned	☆ 0	↓ 13	🌐 Public


# Docker – Docker File

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

A screenshot of a terminal window with a dark background. The title bar at the top shows "GNU nano 2.3.1" on the left and "File: Dockerfile" on the right. The main area of the terminal displays the following text:


```
FROM alpine
RUN apk add openjdk11
EXPOSE 80
CMD "java"
```

# Docker – Docker File

 dockerhub

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

[Upgrade](#)

 sirinenaifar

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 – Docker File

- 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 – Commandes de base

- Extraire une image ou un référentiel d'un registre:

`docker pull « nom de l'image » : « version »`

→ Si on ne spécifie rien, docker téléchargera la dernière version

```
[root@adsl-172-10-0-35 docker]# docker pull alpine
Using default tag: latest
Trying to pull repository docker.io/library/alpine ...
latest: Pulling from docker.io/library/alpine
Digest: sha256:bc41182d7ef5fffc53a40b044e725193bc10142a1243f395ee852a8d9730fc2ad
Status: Image is up to date for docker.io/alpine:latest
```

```
[root@adsl-172-10-0-35 docker]# docker pull nginx:1.15.12
Trying to pull repository docker.io/library/nginx ...
1.15.12: Pulling from docker.io/library/nginx
743f2d6c1f65: Pulling fs layer
6bfc4ec4420a: Pulling fs layer
688a776db95f: Pulling fs layer
```

# Docker – Commandes de base

- Pour construire une image docker définie dans un fichier Dockerfile , vous devez exécuter la commande:  
`docker build -t « username dockerHub » / « Nom de l'image à créé » « Path vers le fichier »`
- Vous pouvez également spécifier .(point) lorsque vous utilisez le fichier docker à partir du répertoire actuel, mais s'il se trouve dans un autre répertoire, vous devez spécifier le chemin complet.
- De plus, si vous n'utilisez pas le nom de dockerfile par défaut, l'option -f est requise

# Docker – Commandes de base

```
Login Succeeded
[root@localhost docker]# docker build -t sirinenaifar/alpine:1.0.0 .
Sending build context to Docker daemon 2.048 kB
Step 1/4 : FROM alpine
Trying to pull repository docker.io/library/alpine ...
latest: Pulling from docker.io/library/alpine
213ec9aee27d: Pull complete
Digest: sha256:bc41182d7ef5fffc53a40b044e725193bc10142a1243f395ee852a8d9730fc2ad
Status: Downloaded newer image for docker.io/alpine:latest
----> 9c6f07244728
Step 2/4 : RUN apk add openjdk11
----> Running in 7fd26d1641c7

fetch https://dl-cdn.alpinelinux.org/alpine/v3.16/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.16/community/x86_64/APKINDEX.tar.gz
(1/30) Installing java-common (0.5-r0)
(2/30) Installing libffi (3.4.2-r1)
(3/30) Installing p11-kit (0.24.1-r0)
(4/30) Installing libtasn1 (4.18.0-r0)
(5/30) Installing p11-kit-trust (0.24.1-r0)
(6/30) Installing ca-certificates (20220614-r0)
(7/30) Installing java-cacerts (1.0-r1)
(8/30) Installing openjdk11-jre-headless (11.0.16.1_p1-r0)
```

# Docker – Commandes de base

```
(28/30) Installing openjdk11-demos (11.0.16.1_p1-r0)
(29/30) Installing openjdk11-doc (11.0.16.1_p1-r0)
(30/30) Installing openjdk11 (11.0.16.1_p1-r0)
Executing busybox-1.35.0-r17.trigger
Executing java-common-0.5-r0.trigger
Executing ca-certificates-20220614-r0.trigger
OK: 263 MiB in 44 packages
---> 8b6fe0f16608
Removing intermediate container 7fd26d1641c7
Step 3/4 : EXPOSE 80
---> Running in 941a0a0f9b1b
---> 2b64f0ea948a
Removing intermediate container 941a0a0f9b1b
Step 4/4 : CMD "java"
---> Running in b07a6103ec11
---> 698f1a073864
Removing intermediate container b07a6103ec11
Successfully built 698f1a073864
```

# Docker – Commandes de base

```
[root@localhost docker]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
sirinenafar/alpine	1.0.0	698f1a073864	18 minutes ago	275 MB
docker.io/alpine	latest	9c6f07244728	6 weeks ago	5.54 MB

- L'image «sirinenafar/alpine » est local (Accessible et utilisé par le créateur seulement). Pour rendre cette image accessible pour tous les utilisateurs de l'outil Docker, il faut la déposer dans le Docker Hub
  - ✓ La première chose à faire est de s'authentifier à partir du terminal: `docker login --username=userName`

```
[root@adsl-172-10-0-35 docker]# docker login --username sirinenafar
Password:
Login Succeeded
```

# Docker – Docker File

- ✓ Envoyer une image que vous avez créé sur dockerhub:  
docker push « nom de l'image »

```
[root@localhost vagrant]# docker push sirinenaifar/alpine:1.0.0
The push refers to a repository [docker.io/sirinenaifar/alpine]
47dfe7a8c4ee: Pushing [=====>] 56.93 MB/269.1 MB
994393dc58e7: Layer already exists
```

The screenshot shows the Docker Hub web interface. At the top is a blue navigation bar with the Docker Hub logo, a search bar, and links for Explore, Repositories, Organizations, and Help. A user profile for 'sirinenaifar' is visible on the right. Below the navigation bar, there's a search bar with 'sirinenaifar' entered. A red rectangle highlights the repository 'sirinenaifar / alpine'. This repository is marked as 'Not Scanned', has 0 stars, 0 downloads, and is 'Public'. It was last pushed 2 minutes ago. Below it, another repository 'sirinenaifar / springbootimage' is visible, marked as 'Not Scanned', with 0 stars, 70 downloads, and is 'Public', last pushed a year ago. On the right side of the interface, there's a promotional graphic for creating an organization.

# Docker – Commandes de base

- Récupérer la liste des images:

`docker images`

```
[root@adsl-172-10-0-35 docker]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED
SIZE			
docker.io/alpine	latest	9c6f07244728	3 weeks ago
5.54 MB			
docker.io/hello-world	latest	feb5d9fea6a5	11 months ago
o 13.3 kB			
docker.io/centos	8	5d0da3dc9764	11 months ago
o 231 MB			
docker.io/centos	latest	5d0da3dc9764	11 months ago
o 231 MB			
docker.io/nginx	1.15.12	53f3fd8007f7	3 years ago
109 MB			

# Docker – Commandes de base

- Pour savoir plus de détails sur une image Docker, exécutez la commande :

`docker image inspect « nom de l'image »`

```
[root@adsl-172-10-0-35 docker]# docker image inspect 5d0da3dc9764
[
  {
    "Id": "sha256:5d0da3dc976460b72c77d94c8a1ad043720b0416bfc16c52c45d4847e53fadb6",
    "RepoTags": [
      "docker.io/centos:8"
    ],
    "RepoDigests": [],
    "Parent": "",
    "Comment": "",
    "Created": "2021-09-15T18:20:05.184694267Z",
    "Container": "9bf8a9e2ddff4c0d76a587c40239679f29c863a967f23abf7a5babb6c2121bf1",
    "ContainerConfig": {
      "Hostname": "9bf8a9e2ddff",
      "Domainname": "",
      "User": "",
```



# Docker – Commandes de base

- Créer un conteneur:

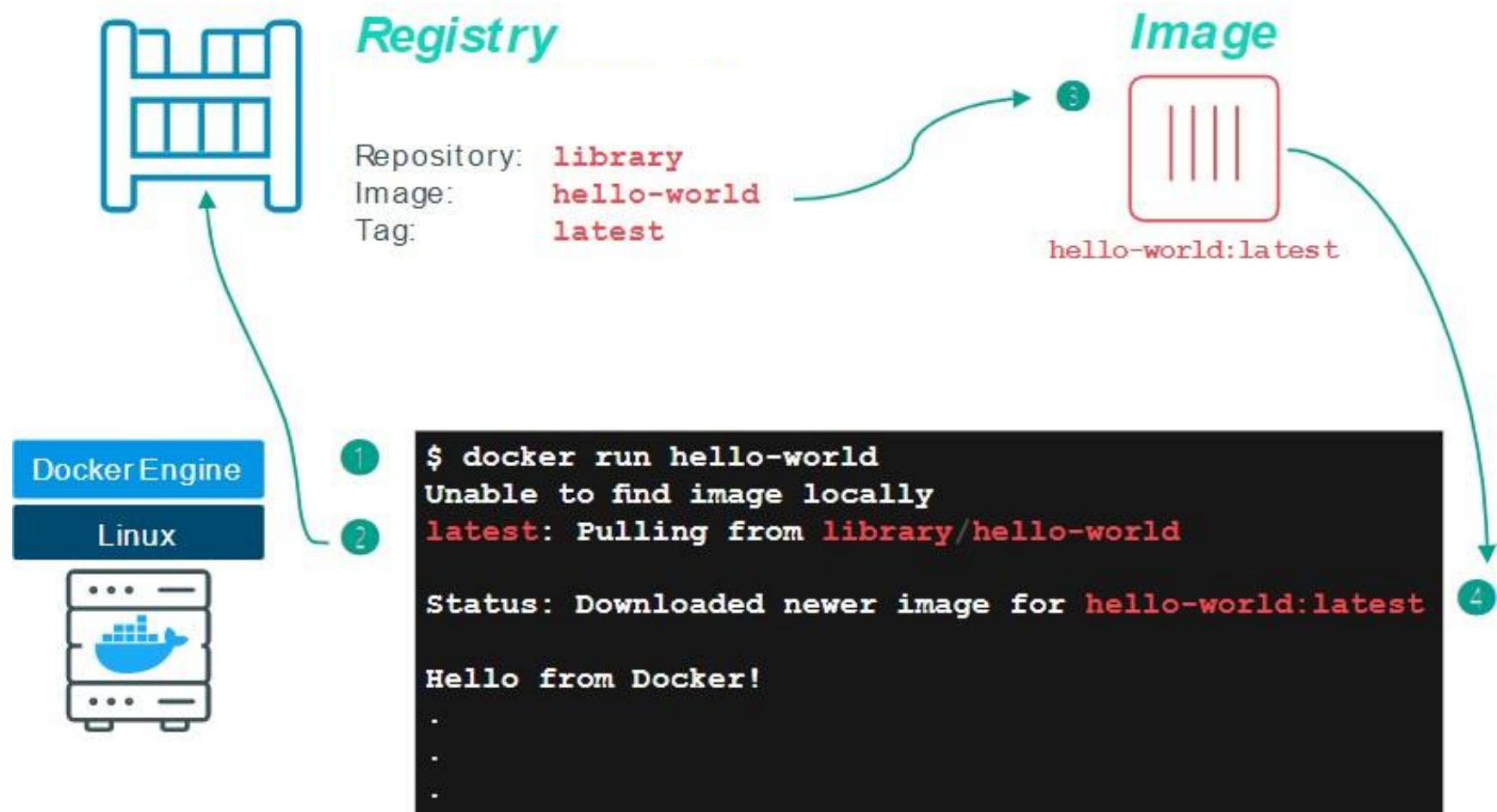
`docker run « nom de l'image »`

➔ Si docker ne trouve pas l'image, il la télécharge et après il crée le conteneur

```
[root@adsl-172-10-0-35 docker]# docker run centos
Unable to find image 'centos:latest' locally
Trying to pull repository docker.io/library/centos ...
latest: Pulling from docker.io/library/centos
Digest: sha256:a27fd8080b517143cbbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177
Status: Downloaded newer image for docker.io/centos:latest
```

# Docker – Commandes de base

## Hello World: What Happened?



# Docker – Commandes de base

Pour faire un tag d'une image docker, il faut utiliser la commande suivante :

**docker tag** **id\_image** **docker\_hub** repository\_name/version

```
PS D:\devOps projects> docker tag feb5d9fea6a5 helloworld:version1.0
PS D:\devOps projects> █
```

Nous pouvons faire le tag directement sur le nom de l'image :

**docker tag** **nom\_image** **docker\_hub** repository\_name/version

```
PS D:\devOps projects> docker tag hello-world helloworld:version1.0
PS D:\devOps projects> █
```

# Docker – Commandes de base

- Lorsque vous appelez run, le client Docker recherche l'image (alpine dans ce cas), crée le conteneur, puis exécute une commande dans ce conteneur.

```
[root@localhost vagrant]# docker run alpine ls -l
total 8
drwxr-xr-x    2 root    root          4096 Aug  9 08:47 bin
drwxr-xr-x    5 root    root          340 Sep 25 19:15 dev
drwxr-xr-x    1 root    root           66 Sep 25 19:15 etc
drwxr-xr-x    2 root    root           6 Aug  9 08:47 home
drwxr-xr-x    7 root    root        247 Aug  9 08:47 lib
drwxr-xr-x    5 root    root         44 Aug  9 08:47 media
drwxr-xr-x    2 root    root           6 Aug  9 08:47 mnt
drwxr-xr-x    2 root    root           6 Aug  9 08:47 opt
dr-xr-xr-x  153 root    root           0 Sep 25 19:15 proc
drwx-----   2 root    root           6 Aug  9 08:47 root
drwxr-xr-x    1 root    root          21 Sep 25 19:15 run
drwxr-xr-x    2 root    root       4096 Aug  9 08:47/sbin
drwxr-xr-x    2 root    root           6 Aug  9 08:47 srv
dr-xr-xr-x   13 root    root           0 Sep 25 18:42 sys
drwxrwxrwt    2 root    root           6 Aug  9 08:47 tmp
drwxr-xr-x    7 root    root          66 Aug  9 08:47 usr
drwxr-xr-x   12 root    root        137 Aug  9 08:47 var
```

# Docker – Commandes de base

- Lorsque vous exécutez la dernière commande, vous avez fourni une commande (`ls -l`), donc Docker a exécuté cette commande dans le conteneur pour lequel vous avez vu la liste des répertoires. Une fois la commande `ls` terminée, le conteneur s'est arrêté.

```
[root@localhost vagrant]# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
6d85b72310a7	alpine	"ls -l"	54 seconds ago	Exited (0) 48 seconds ago

```
[root@localhost vagrant]# docker run alpine echo "Hello from Alpine"
```

```
Hello from Alpine
```

```
[root@localhost vagrant]# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
41b12ed7a19e	alpine	"echo 'Hello from ...'"	5 seconds ago	Exited (0) 4 seconds ago
6d85b72310a7	alpine	"ls -l"	3 minutes ago	Exited (0) 3 minutes ago

# Docker – Commandes de base

- Pour obtenir un shell interactif où vous pouvez taper quelques commandes, Docker a une option pour cela.
  - ✓ -i pour le mode interactif
  - ✓ -t pour avoir un pseudo TTY

```
[root@localhost vagrant]# docker run -it alpine
/ # echo "hello"
hello
/ # date
Sun Sep 25 19:22:01 UTC 2022
/ # ls -l
total 8
drwxr-xr-x  2 root    root          4096 Aug  9 08:47 bin
drwxr-xr-x  5 root    root          360 Sep 25 19:21 dev
drwxr-xr-x  1 root    root           66 Sep 25 19:21 etc
drwxr-xr-x  2 root    root           6 Aug  9 08:47 home
drwxr-xr-x  7 root    root        247 Aug  9 08:47 lib
drwxr-xr-x  5 root    root          44 Aug  9 08:47 media
drwxr-xr-x  2 root    root           6 Aug  9 08:47 mnt
drwxr-xr-x  2 root    root           6 Aug  9 08:47 opt
dr-xr-xr-x 154 root    root           0 Sep 25 19:21 proc
drwx----- 1 root    root          26 Sep 25 19:21 root
drwxr-xr-x  1 root    root          21 Sep 25 19:21 run
drwxr-xr-x  2 root    root        4096 Aug  9 08:47 sbin
drwxr-xr-x  2 root    root           6 Aug  9 08:47 srv
dr-xr-xr-x 13 root    root           0 Sep 25 18:42 sys
drwxrwxrwt  2 root    root           6 Aug  9 08:47 tmp
drwxr-xr-x  7 root    root           6 Aug  9 08:47 usr
drwxr-xr-x 12 root    root        137 Aug  9 08:47 var
```

# Docker – Commandes de base

- Pour stopper un conteneur, tapez exit.

```
[root@localhost vagrant]# docker run -it alpine
/ # echo "hello"
hello
/ # date
Sun Sep 25 19:22:01 UTC 2022
/ # ls -l
total 8
drwxr-xr-x  2 root    root      4096 Aug  9 08:47 bin
drwxr-xr-x  5 root    root      360 Sep 25 19:21 dev
drwxr-xr-x  1 root    root        66 Sep 25 19:21 etc
drwxr-xr-x  2 root    root        6 Aug  9 08:47 home
drwxr-xr-x  7 root    root     247 Aug  9 08:47 lib
drwxr-xr-x  5 root    root      44 Aug  9 08:47 media
drwxr-xr-x  2 root    root        6 Aug  9 08:47 mnt
drwxr-xr-x  2 root    root        6 Aug  9 08:47 opt
dr-xr-xr-x 154 root    root        0 Sep 25 19:21 proc
drwx----- 1 root    root      26 Sep 25 19:21 root
drwxr-xr-x  1 root    root      21 Sep 25 19:21 run
drwxr-xr-x  2 root    root     4096 Aug  9 08:47 sbi
drwxr-xr-x  2 root    root        6 Aug  9 08:47 srv
dr-xr-xr-x 13 root    root        0 Sep 25 18:42 sys
drwxrwxrwt  2 root    root        6 Aug  9 08:47 tmp
drwxr-xr-x  7 root    root      66 Aug  9 08:47 usr
drwxr-xr-x 12 root    root     137 Aug  9 08:47 var
/ # exit
```

```
[root@localhost vagrant]# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
d64af7009ce2	alpine	"/bin/sh"	3 minutes ago	Exited (0) 2 minutes ago		boring_banach
41b12ed7a19e	alpine	"echo 'Hello from ...'"	5 minutes ago	Exited (0) 5 minutes ago		zen_babbage
6d85b72310a7	alpine	"ls -l"	9 minutes ago	Exited (0) 9 minutes ago		zen_feynman

# Docker – Commandes de base

- Une fois que vous vous êtes attaché à un conteneur Docker via une console CMD, le fait de taper **exit** sur la console permet de se détacher du conteneur et de l'arrêter.

```
[root@localhost vagrant]# docker run -it alpine
/ # echo "hello"
hello
/ # date
Sun Sep 25 19:22:01 UTC 2022
/ # ls -l
total 8
drwxr-xr-x  2 root    root          4096 Aug  9 08:47 bin
drwxr-xr-x  5 root    root          360 Sep 25 19:21 dev
drwxr-xr-x  1 root    root           66 Sep 25 19:21 etc
drwxr-xr-x  2 root    root           6 Aug  9 08:47 home
drwxr-xr-x  7 root    root         247 Aug  9 08:47 lib
drwxr-xr-x  5 root    root          44 Aug  9 08:47 media
drwxr-xr-x  2 root    root           6 Aug  9 08:47 mnt
drwxr-xr-x  2 root    root           6 Aug  9 08:47 opt
dr-xr-xr-x 154 root    root           0 Sep 25 19:21 proc
drwx----- 1 root    root          26 Sep 25 19:21 root
drwxr-xr-x  1 root    root          21 Sep 25 19:21 run
drwxr-xr-x  2 root    root        4096 Aug  9 08:47 sbin
drwxr-xr-x  2 root    root           6 Aug  9 08:47 srv
dr-xr-xr-x 13 root    root           0 Sep 25 18:42 sys
drwxrwxrwt  2 root    root           6 Aug  9 08:47 tmp
drwxr-xr-x  7 root    root          66 Aug  9 08:47 usr
drwxr-xr-x 12 root    root         137 Aug  9 08:47 var
/ # exit
```

```
[root@localhost vagrant]# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
d64af7009ce2	alpine	"/bin/sh"	3 minutes ago	Exited (0) 2 minutes ago		boring_banach
41b12ed7a19e	alpine	"echo 'Hello from ...'"	5 minutes ago	Exited (0) 5 minutes ago		zen_babbage
6d85b72310a7	alpine	"ls -l"	9 minutes ago	Exited (0) 9 minutes ago		zen_feynman



# Docker – Commandes de base

- Pour se détacher du conteneur sans l'arrêter, appuyez sur CTRL+P suivi de CTRL+Q.

```
/ # [root@localhost vagrant]# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
491bc43338b1	alpine	"/bin/sh"	47 seconds ago	Up 44 seconds	
d64af7009ce2	alpine	"/bin/sh"	5 minutes ago	Exited (0) 5 minutes ago	
41b12ed7a19e	alpine	"echo 'Hello from ...'"	8 minutes ago	Exited (0) 8 minutes ago	
6d85b72310a7	alpine	"ls -l"	11 minutes ago	Exited (0) 11 minutes ago	

- Pour retourner dedans (Le conteneur doit être lancé en parallèle):

Docker attach « container id »

```
[root@localhost vagrant]# docker attach 491bc43338b1  
/ #
```

- Nous pouvons envoyer une commande à exécuter dans le conteneur en utilisant la commande exec, comme suit:

```
/ # [root@localhost vagrant]# docker exec 491bc43338b1 echo "Hi from here"  
Hi from here
```

# Docker – Commandes de base

- Récupérer la liste des conteneurs:

docker container ls ou bien docker ps -a

```
[root@adsl-172-10-0-35 docker]# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
a15af1503a9d	centos	"/bin/bash"	41 minutes ago	Exited (0) 41 minutes ago		hopeful_galileo
4f97a1e0e6fa	9c6f07244728	"/bin/sh -c 'yum i...'"	About an hour ago	Exited (127) About an hour ago		reverent_wilson
d556c59a5432	9c6f07244728	"/bin/sh -c 'yum i...'"	2 hours ago	Exited (127) 2 hours ago		jovial_tesla
ab90c4bfc1a6	5d0da3dc9764	"/bin/sh -c 'yum i...'"	2 hours ago	Exited (1) 2 hours ago		serene_keller
3e96099e1f5d	5d0da3dc9764	"/bin/sh -c 'yum u...'"	2 hours ago	Exited (1) 2 hours ago		flamboyant_turing
a1ef5bace358	5d0da3dc9764	"/bin/sh -c 'dnf u...'"	2 hours ago	Exited (1) 2 hours ago		dreamy_dubinsky
dfcc9ee52765	5d0da3dc9764	"/bin/sh -c 'dnf u...'"	2 hours ago	Exited (1) 2 hours ago		gifted_hamilton
971590f59e86	5d0da3dc9764	"/bin/sh -c 'yum u...'"	2 hours ago	Exited (1) 2 hours ago		mystifying_knuth
eadec32d97ef	5d0da3dc9764	"/bin/sh -c 'yum u...'"	2 hours ago	Exited (1) 2 hours ago		sleepy_einstein
b29d66b74ba7	5d0da3dc9764	"/bin/sh -c 'sudo ...'"	2 hours ago	Exited (127) 2 hours ago		jovial_shirleye87a
c97c75e1	5d0da3dc9764	"/bin/sh -c 'dnf u...'"	3 hours ago	Exited (1) 3 hours ago		fervent_bassi

- Récupérer la liste des conteneurs actifs :

docker ps

# Docker – Commandes de base

- Supprimer une image:

docker image rm «nom de l'image » - - force  
ou bien docker rmi « nom de l'image »

```
[root@adsl-172-10-0-35 docker]# docker image rm hello-world --force
Untagged: hello-world:latest
Untagged: docker.io/hello-world@sha256:7d246653d0511db2a6b2e0436cfd0e52ac8c066000264b3ce63331ac66dca625
Deleted: sha256:feb5d9fea6a5e9606aa995e879d862b825965ba48de054caab5ef356dc6b3412
```

- Supprimer un conteneur:

docker rm « Id du conteneur »

```
[root@adsl-172-10-0-35 docker]# docker rm 6b3337199055
6b3337199055
```

# Introduction à Docker

Si vous avez des questions, n'hésitez pas à nous contacter :

**Département Informatique**  
**UP ASI**

Bureau E204

# Introduction à Docker

