

# Docker



# docker

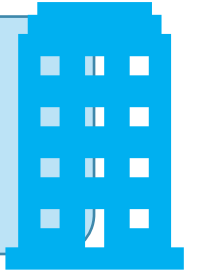
« Build, Ship and Run Any App, Anywhere »

# Introduction à Docker



Plateforme open-source qui automatise le déploiement d'applications dans des conteneurs logiciels.

En mars 2013, fondé en France par **Solomon Hykes**



Les conteneurs permettent à une application et à ses dépendances :

- 1 d'être empaquetées ensemble de manière portable
- 2 exécutées de manière cohérente sur n'importe quel environnement compatible.



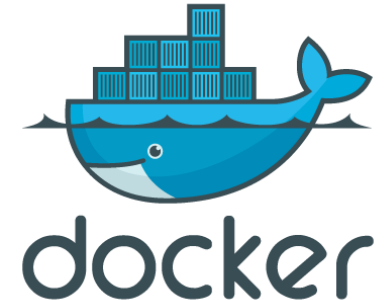
# Historique de la conteneurisation

La conteneurisation est loin d'être une technologie récente

chroot



solaris



1982

2000

2004

2008

2013

Chroot sur Unix

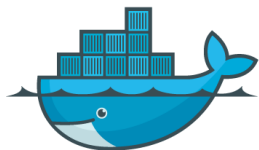
Jail sur BSD

conteneurs  
sur Solaris

LXC (Linux conteneurs)  
sur Linux

Docker

Célébrité  
de



docker



gérer facilement leurs conteneurs avec une  
interface en ligne de commande simple.

# Principes de base

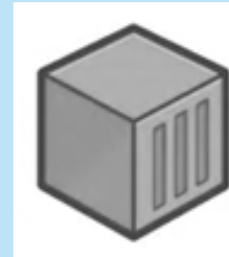
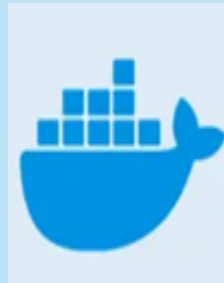
**Conteneur  
vs Machine  
virtuelle**



{ ... }

**Docker File**

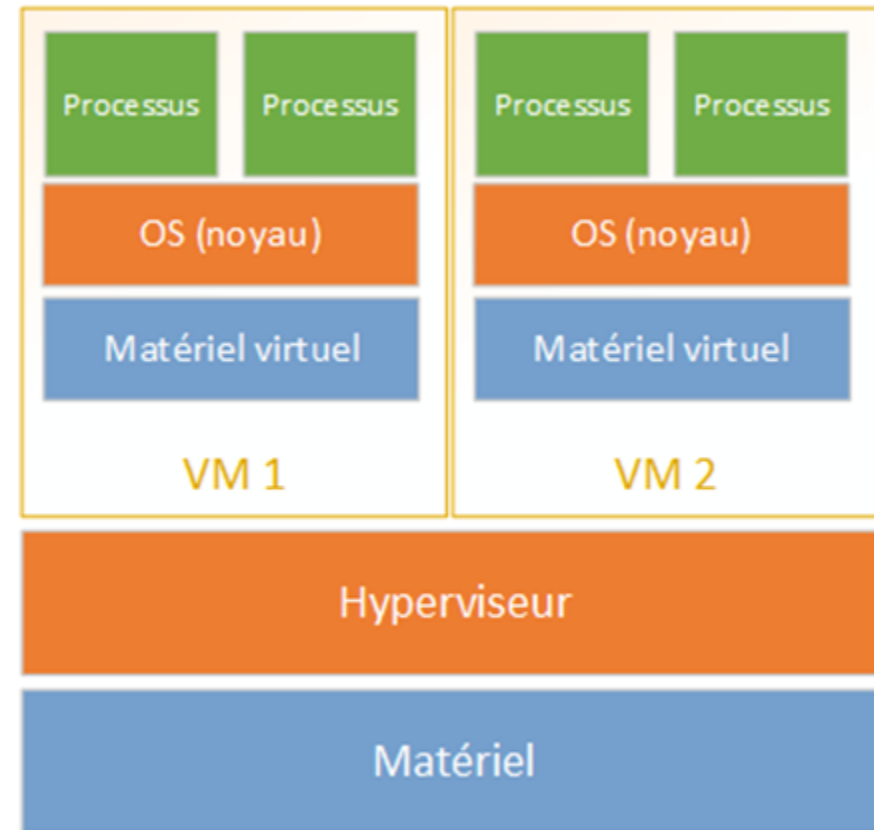
**Docker Hub**



**Docker Image**

# Virtualisation

**Machine virtuelle (VM) :** Une machine virtuelle est un environnement logiciel qui émule un ordinateur physique. Chaque machine virtuelle fonctionne sur une couche d'hyperviseur et exécute un système d'exploitation complet, avec ses propres ressources et noyau.



Les machines virtuelles

# Avantages de la Virtualisation

**1 Allocation efficace des ressources** en fonction des besoins spécifiques des applications.

**2 Réduction des dépenses** en éliminant ou en réduisant le besoin de matériel physique, ce qui diminue les coûts liés à l'équipement, à la maintenance, à l'alimentation et au refroidissement.

**3 Facilité de gestion des machines virtuelles**, permettant des actions telles que le déplacement, la sauvegarde et la duplication sans perturbation pour les utilisateurs.

**4 Amélioration de la facilité d'administration** grâce à la résilience des environnements virtuels face aux pannes matérielles, facilitant ainsi les opérations de sauvegarde et de reprise après une catastrophe.

# Inconvénients de la Virtualisation

## Diminution des performances

dues à l'accès aux ressources de manière virtuelle, résultant de la nécessité d'une couche d'abstraction matérielle qui interprète entre le matériel réel et celui simulé dans la machine virtuelle

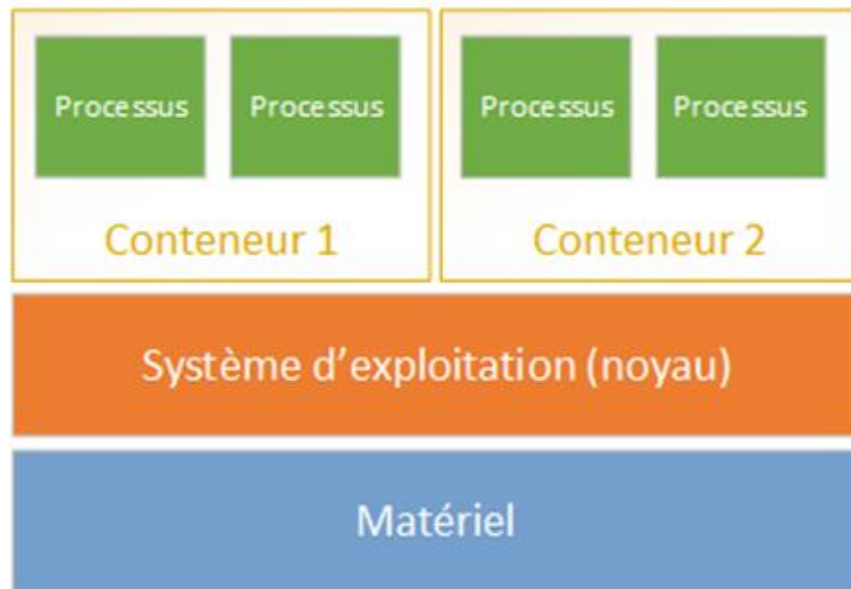
== Latence supplémentaire

## Surcharge des ressources système

L'exécution de plusieurs machines virtuelles sur un seul ordinateur physique, nécessite une surcharge en termes de ressources système pour l'exploitation de cette couche d'OS supplémentaire

== Limitation des performances des applications

# Conteneur vs Machine Virtuelle

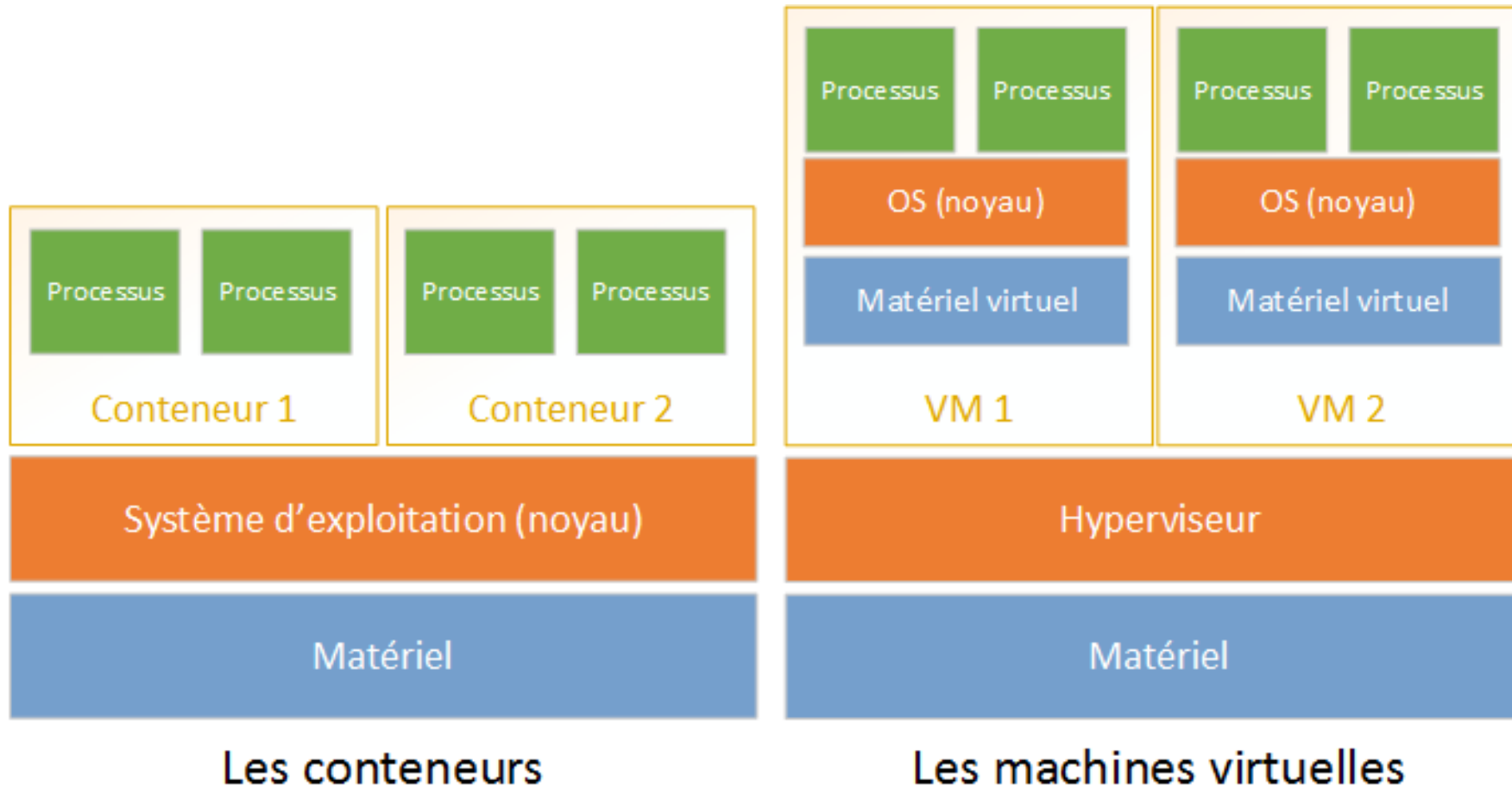


Les conteneurs

**Conteneur Docker :** Un conteneur Docker est une instance d'une image Docker, qui inclut l'application ainsi que toutes ses dépendances. Contrairement aux machines virtuelles, les conteneurs partagent le noyau du système hôte et utilisent une isolation des ressources plus légère, ce qui les rend plus rapides et plus efficaces.



# Conteneur vs Machine Virtuelle



## Avantages conteneurisation vs virtualisation traditionnelle

### Espace nécessaire

Les machines virtuelles intègrent elles-mêmes un OS pouvant aller jusqu'à des Giga-octets. Le conteneur appelle directement l'OS pour réaliser ses **appels système** et exécuter ses applications. Il est beaucoup moins gourmand en ressources

### Facilité de déploiement

On peut déplacer les conteneurs d'un environnement à l'autre très rapidement (simple partage des fichiers de config).

On peut déplacer une VM de serveurs en serveurs mais la couche d'OS rendra le déploiement beaucoup plus lent, sans oublier le processus d'émulation des ressources physiques, qui lui-même demandera un certain temps d'exécution.

# Avantages de la conteneurisation

- 1 **Flexible:** même les applications les plus complexes peuvent être conteneurisées.
- 2 **Léger:** les conteneurs exploitent et partagent le noyau hôte.
- 3 **Interchangeable:** vous pouvez déployer des mises à jour à la volée
- 4 **Portable:** vous pouvez créer localement, déployer sur le cloud et exécuter n'importe où votre application.
- 5 **Évolutif:** vous pouvez augmenter et distribuer automatiquement les répliques (les clones) de conteneur.
- 6 **Empilable:** Vous pouvez empiler des services verticalement et à la volée.

# En savoir plus sur les conteneurs

Le noyau Linux offre quelques fonctionnalités comme :

## namespaces

ce qu'un processus peut voir  
Les namespaces isolent les ressources partagées.

## cgroups

ce qu'un processus peut utiliser en terme de ressources  
Ils permettent d'**isoler les processus Linux** les uns des autres.

Lorsque vous utilisez ces fonctionnalités, on appelle cela **des conteneurs**.

## Exemple simple

Si jamais on souhaite créer un conteneur contenant la distribution Ubuntu.

Les fonctionnalités d'isolation proposées par le noyau Linux, vont permettre de prétendre d'avoir quelque chose qui ressemble à une machine virtuelle avec l'OS Ubuntu,

En réalité ce n'est pas du tout une machine virtuelle mais un **processus isolé** s'exécutant dans le même noyau Linux .

# Les namespaces : limiter les vues

Les namespaces donnent à chaque processus sa propre vue unique du système, limitant ainsi leur accès aux ressources système sans que le processus en cours ne soit au courant des limitations.

## Namespace PID

fournit un ensemble indépendant d'identifiants de processus (PID). Il s'agit d'une structure hiérarchique dans laquelle le namespace parent peut afficher tous les PID des namespaces enfants.

Lorsqu'un nouveau namespace est créé, le premier processus obtient le PID 1 et constitue une sorte de processus init de ce namespace. Cela signifie également que si on tue ce processus PID 1 alors on mettra immédiatement fin à tous les processus de son namespace PID et à tous ses descendants.

# Les namespaces : limiter les vues

## Namespace IPC

empêche la communication avec les autres processus.

## Namespace NET

crée une pile réseau entièrement nouvelle, y compris : un ensemble privé d'adresses IP, sa propre table de routage, liste de socket, table de suivi des connexions, pare-feu et autres ressources liées au réseau.

## Namespace MOUNT

monte un système de fichier propre au processus qui est différent du système de fichier de la machine hôte. Vous pouvez ainsi monter et démonter des systèmes de fichiers sans que cela n'affecte le système de fichiers hôte.

## Namespace UTS

associe un nom d'hôte et de domaine au processus pour avoir son propre hostname.

# Les namespaces : limiter les vues

## Namespace USER

Le **namespace USER** : fournit à la fois l'isolation des privilèges et la séparation des identifications d'utilisateurs entre plusieurs ensembles. Il permet par exemple de donner un accès root dans le conteneur sans qu'il soit root sur la machine hôte.



# Les Cgroups : limiter les ressources

Comment limiter la quantité de mémoire ou de processeur utilisée par un processus ?

- **cgroup cpuset** : assigne des processeurs individuels et des nœuds de mémoire à des groupes de contrôle
- **cgroup cpu** : planifie un accès aux ressources du processeur
- **cgroup cpuacct** : génère des rapports sur les ressources du processeur utilisées
- **cgroup devices** : autorise ou refuse l'accès aux périphériques
- **cgroup net\_prio** : permet de définir la priorité du trafic réseau
- **cgroup memory** : définit la limite d'utilisation de la mémoire
- **cgroup blkio** : limite de la vitesse E/S (lecture/écriture) sur périphériques de type bloc (ex : disque dur)
- **cgroup pid** : limite le nombre de processus

# Docker et ses origines sous Linux

Pour résumer, la technologie Docker possède de nombreuses fonctionnalités de nos jours, mais beaucoup d'entre elles **reposent sur les fonctionnalités de base du noyau Linux.**

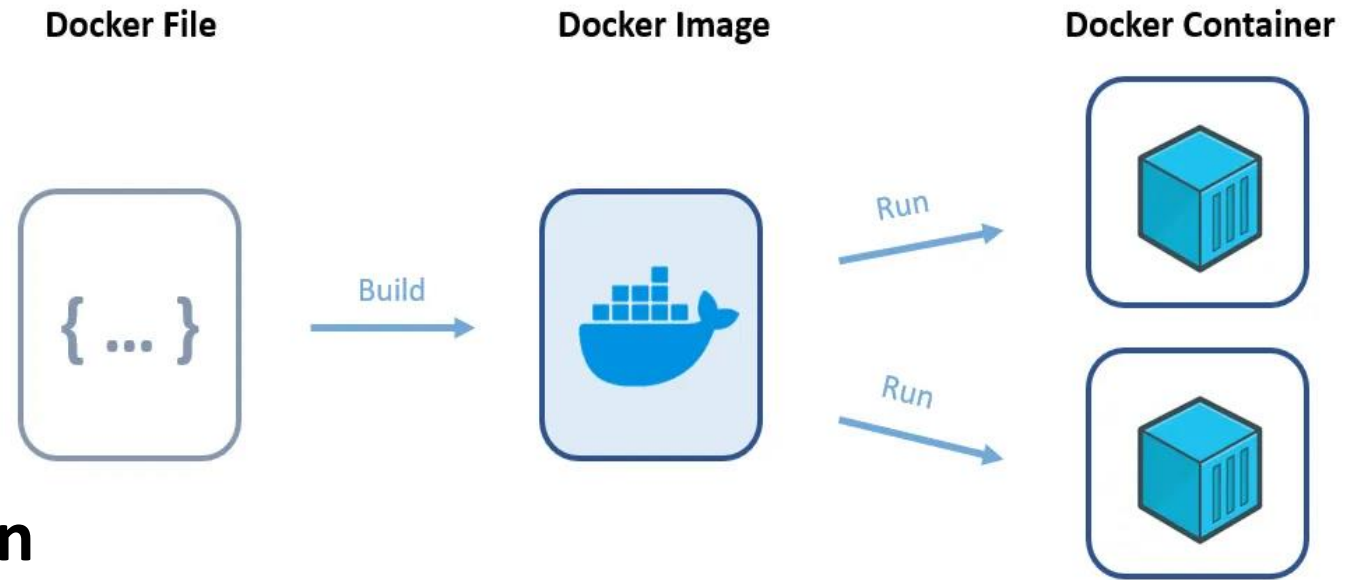
De façon plus précise, **les conteneurs contiennent généralement un ou plusieurs programme(s) de manière à les maintenir isolées du système hôte sur lequel elles s'exécutent.** Ils permettent à un développeur de conditionner une application avec toutes ses dépendances, et de l'expédier dans un package unique.

# Image Docker

**Une image est un package** qui inclut tout ce qui est nécessaire à l'exécution d'une application, à savoir :

- **Le code**
- **L'exécution**
- **Les variables d'environnement**
- **Les bibliothèques**
- **Les fichiers de configuration**

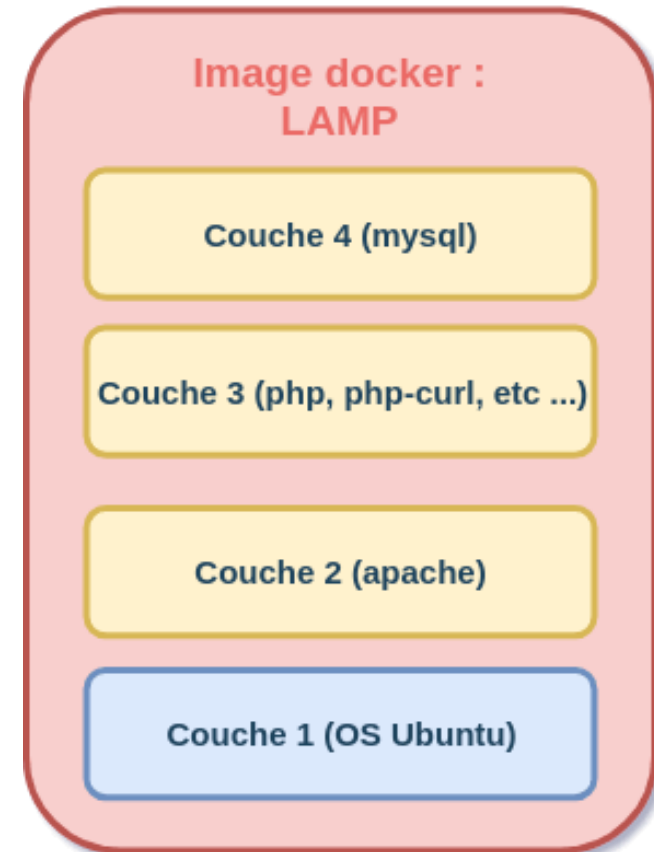
Lorsqu'une image est **instanciée**, son nom est un conteneur, un conteneur est donc une image en cours d'exécution.



# Exemple d'Image Docker

On souhaite déployer une application web dans un serveur LAMP (Linux Apache MySQL PHP) au moyen de Docker. Pour créer notre stack, nous aurons besoin de :

- **Une couche OS** pour exécuter notre Apache, MySQL et Php
- **Une couche Apache** pour démarrer notre serveur web et pourquoi pas la config qui va avec (.htaccess, apache2.conf, site-available/, etc ... )
- **Une couche php** qui contiendra un interpréteur Php mais aussi les bibliothèques qui vont avec (exemple : php-curl)
- **Une couche Mysql** qui contiendra notre système de gestion de bases de données Mysql



# DockerFile

Un Dockerfile est un fichier texte qui contient les instructions pour construire une image Docker.

Les instructions Dockerfile incluent des actions telles que l'installation de dépendances, la copie de fichiers, la définition de variables d'environnement, et l'exécution de commandes pour configurer l'environnement de l'application.

Exemple de Dockerfile pour une application Python simple

```
# Utilise une image de base avec Python
FROM python:3.9

# Définit le répertoire de travail dans le conteneur
WORKDIR /app

# Copie le fichier des dépendances (requirements.txt) dans le répertoire de travail
COPY requirements.txt .

# Installe les dépendances de l'application
RUN pip install --no-cache-dir -r requirements.txt

# Copie le reste des fichiers de l'application dans le répertoire de travail
COPY . .

# Commande pour démarrer l'application lorsque le conteneur démarre
CMD ["python", "app.py"]
```

# Docker Registry

Un Registry est une application côté serveur qui permet de stocker et de distribuer des images Docker, le Docker Hub Registry est le registre officiel de Docker.

Il en existe d'autres parmi lesquels



- Elastic Container Services (ECS)
- Elastic Kubernetes Services (EKS)
- AWS Lambdas

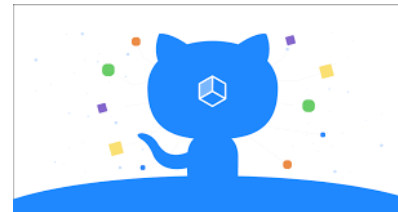
**Amazon ECR**



**Harbor**



**Azure Container  
Registry**



**GitHub Container Registry**



**Google Container  
Registry**

# Docker Hub

Docker Hub est un registre cloud public où les utilisateurs Docker peuvent stocker, partager et rechercher des images Docker.

Il contient une vaste collection d'images prêtes à l'emploi, ce qui simplifie le déploiement d'applications avec Docker.



Il est possible d'avoir des registres privés et des registres publics pour y déposer ses images

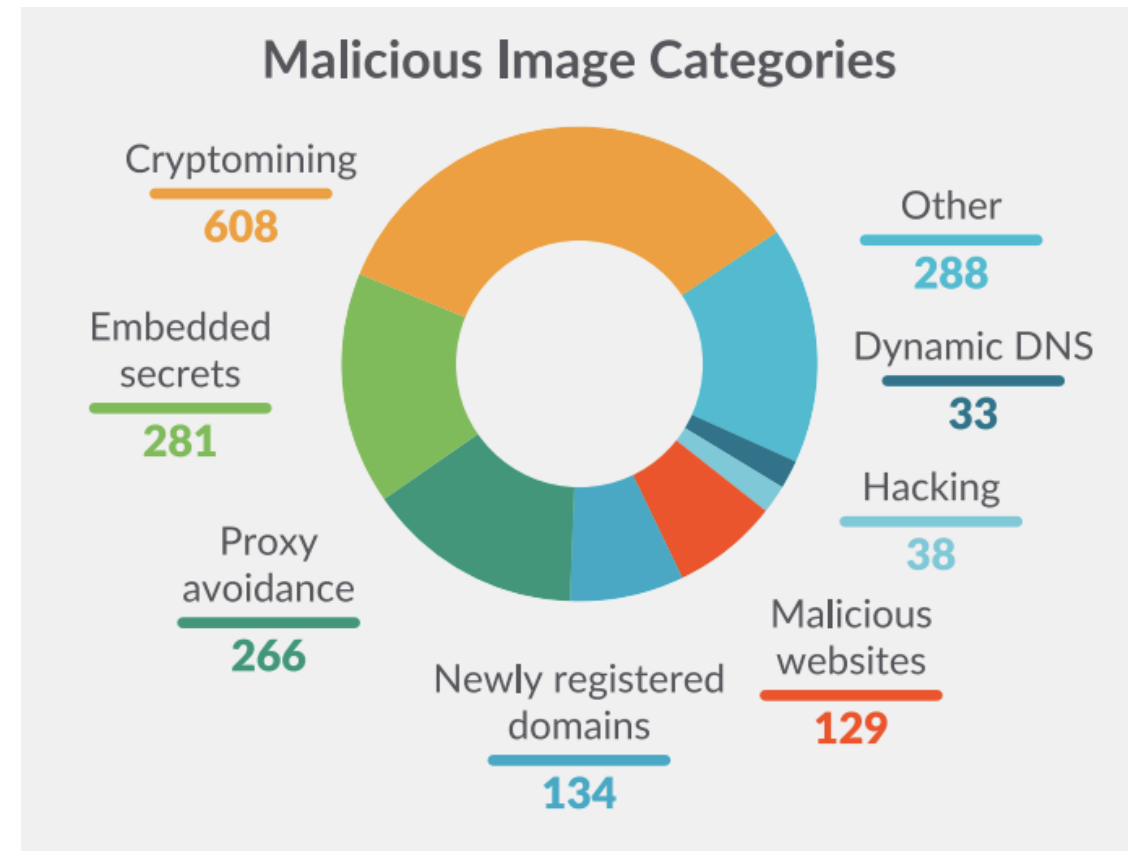


# Sécurité des images de Docker Hub



**Attention aux images  
qu'on télécharge !**

1. vérifier au préalable le code source de l'image (DockerFile)
2. Privilégier les images certifiées



<https://sysdig.com/blog/analysis-of-supply-chain-attacks-through-public-docker-images/>

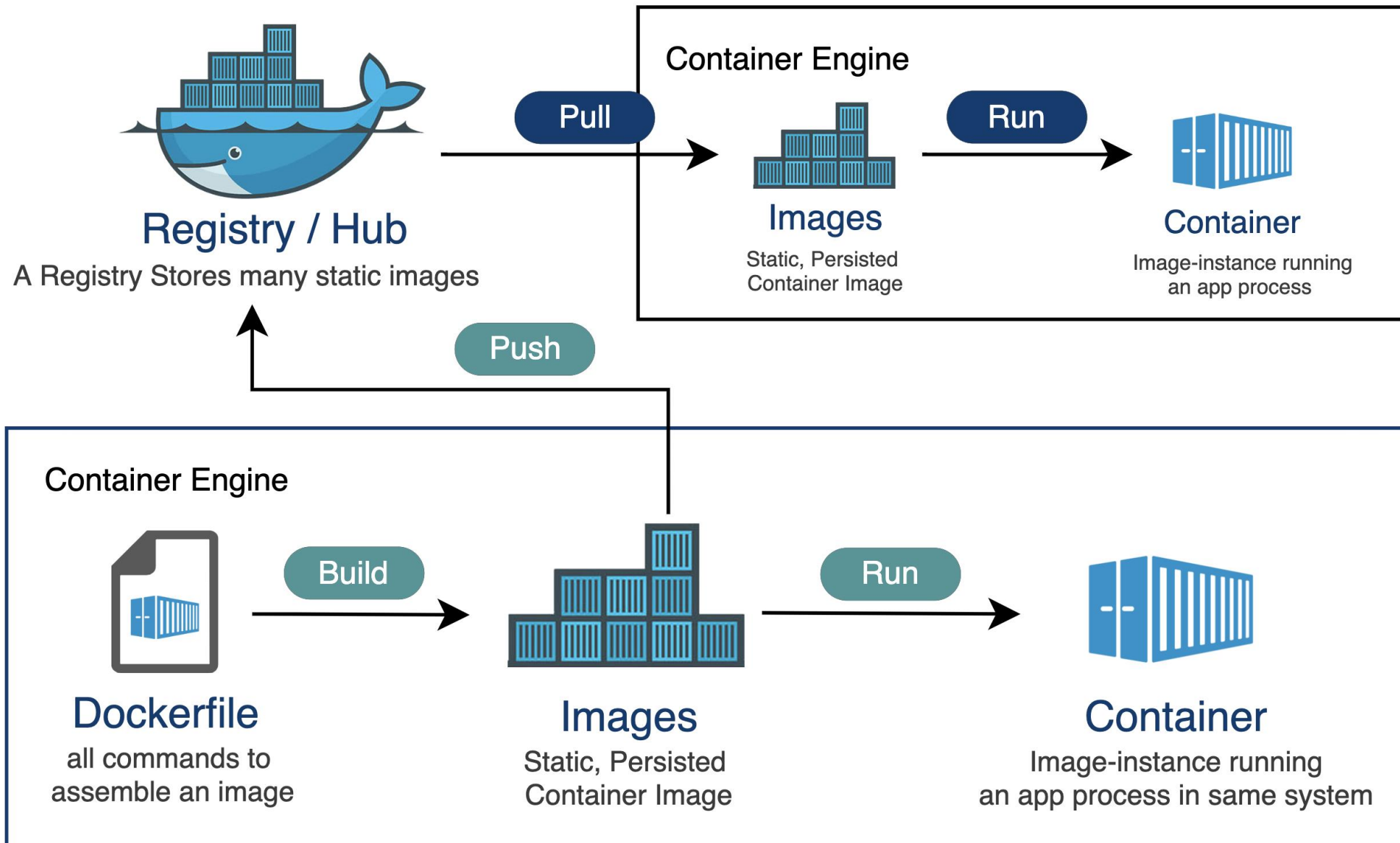


# Sécurité des images de Docker Hub



Dans le Docker Hub Registry une multitude d'images créées par des utilisateurs expérimentés indépendants.

Ces images sont parfois bien plus optimisées que les images officielles car certaines images n'utilisent que le strict minimum pour faire fonctionner leur application permettant ainsi de réduire la taille de l'image.

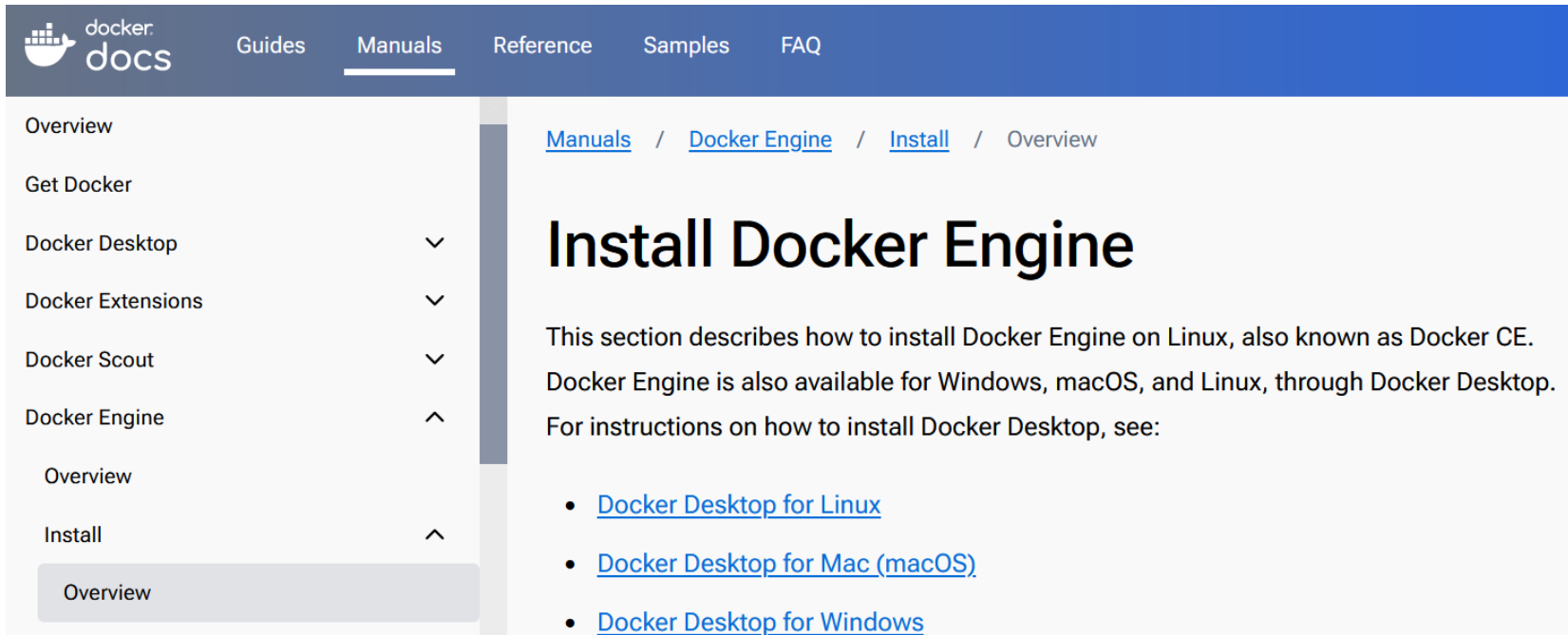


# Mettre en place Docker

# Installation

Vous devez installer Docker Engine sur votre système d'exploitation

<https://docs.docker.com/engine/install/>



The screenshot shows the Docker documentation website. The top navigation bar includes the Docker logo and links for Guides, Manuals, Reference, Samples, and FAQ. The left sidebar lists various topics, with 'Docker Engine' expanded to show 'Overview' and 'Install'. The main content area is titled 'Install Docker Engine' and provides an overview of the installation process for Docker Engine on Linux, also known as Docker CE. It mentions that Docker Engine is also available for Windows, macOS, and Linux through Docker Desktop. Below the text, there are three links: 'Docker Desktop for Linux', 'Docker Desktop for Mac (macOS)', and 'Docker Desktop for Windows'.

docker: docs

Guides Manuals Reference Samples FAQ

Overview

Get Docker

Docker Desktop

Docker Extensions

Docker Scout

Docker Engine

Overview

Install

Overview

[Manuals](#) / [Docker Engine](#) / [Install](#) / Overview

## Install Docker Engine

This section describes how to install Docker Engine on Linux, also known as Docker CE. Docker Engine is also available for Windows, macOS, and Linux, through Docker Desktop. For instructions on how to install Docker Desktop, see:

- [Docker Desktop for Linux](#)
- [Docker Desktop for Mac \(macOS\)](#)
- [Docker Desktop for Windows](#)

# Editions de Docker Engine

Docker est disponible en deux éditions

 **Enterprise Edition**



Windows Server



conçue pour les équipes de développement d'entreprise et les équipes système qui créent, expédient et exécutent des applications critiques pour la production à grande échelle. Version payante

 **Community Edition**



idéale pour les développeurs individuels et les petites équipes cherchant à se familiariser avec Docker et à expérimenter des applications basées sur des conteneurs. version gratuite.

# Versions de Docker CE

Docker CE dispose de trois types de canaux de mise à jour, **stable**, **test** et **Nightly** :

## Stable

cette version vous donne les dernières releases pour une disponibilité générale

## Test

cette version vous fournit des pré-versions prêtes à être testées avant la disponibilité générale

## Nightly

cette version vous présente les dernières versions de build en cours pour la prochaine release, elle fournit donc un accès plus rapide aux nouvelles fonctionnalités et correctifs pour les tests.

# Versions de Docker EE

Docker EE dispose de trois niveaux

## Basic

Plate-forme Docker pour l'infrastructure certifiée, avec le soutien de Docker Inc. et des conteneurs et plugins certifiés de Docker Store

## Standard

Ajout d'une gestion avancée des images et des conteneurs, de l'intégration des utilisateurs LDAP/AD et d'un contrôle d'accès basé sur les rôles (Docker Datacenter)

## Advanced

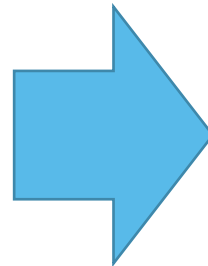
Docker Security Scanning et surveillance continue de la vulnérabilité.



# Vérification de l'installation

Une fois votre installation finie, l'étape suivante est de **vérifier que Docker CE est correctement installé** en vérifiant d'abord la version du moteur :

```
docker version
```



```
Client:
 Cloud integration: v1.0.35-desktop+001
 Version:          24.0.5
 API version:      1.43
 Go version:       go1.20.6
 Git commit:       ced0996
 Built:           Fri Jul 21 20:36:24 2023
 OS/Arch:          windows/amd64
 Context:          default

Server: Docker Desktop 4.22.0 (117440)
Engine:
 Version:          24.0.5
 API version:      1.43 (minimum version 1.12)
 Go version:       go1.20.6
 Git commit:       a61e2b4
 Built:           Fri Jul 21 20:35:45 2023
 OS/Arch:          linux/amd64
 Experimental:     false
 containerd:
 Version:          1.6.21
 GitCommit:        3dce8eb055cbb6872793272b4f20ed16117344f8
 runc:
 Version:          1.1.7
 GitCommit:        v1.1.7-0-g860f061
 docker-init:
 Version:          0.19.0
 GitCommit:        de40ad0
```

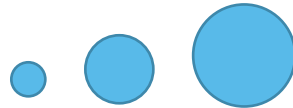


# Vérification de l'installation

## Activation du service

Maintenant il faut **activer le service docker** en tapant la commande suivante :

```
systemctl start docker
```



Pour activer le service docker automatiquement après un boot

```
systemctl enable docker
```

## Lancement de son premier conteneur

Pour finir nos tests de vérifications d'install, on va lancer la commande

```
docker run hello-world
```

Cette commande télécharge une image depuis le Docker Hub Registry et l'exécute dans un conteneur. Lorsque le conteneur s'exécute, il vous affiche un petit message d'information et se ferme directement.

# Vérification de l'installation

Cette commande télécharge une image de test et l'exécute dans un conteneur.

Lorsque le conteneur va s'exécuter, il vous affichera un message d'information et se fermera automatiquement

```
C:\Users\vsans>docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:6352af1ab4ba4b138648f8ee88e63331aae519946d3b67dae50c313c6fc8200f
Status: Downloaded newer image for hello-world:latest

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

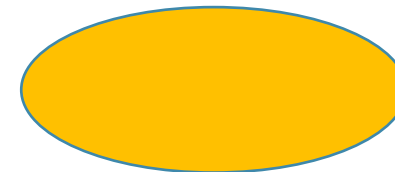
To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the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

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

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

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

<https://devopssec.fr/article/decouverte-et-installation-de-docker>



# Exécution d'un conteneur Docker

Une fois que vous avez créé une image Docker, vous pouvez exécuter un conteneur à partir de cette image en utilisant la commande docker run.

```
docker run nom_image
```

Cette commande démarre un nouveau conteneur à partir de l'image spécifiée.

# Gestion des conteneurs

# Gestion des conteneurs Docker

Récupérer la liste des commandes

```
docker help
```

```
Usage:  docker [OPTIONS] COMMAND
```

```
A self-sufficient runtime for containers
```

```
Options:
```

```
    -v, --version          Print version information and quit  
    ...
```

```
Management Commands:
```

```
    builder      Manage builds  
    ...
```

```
Commands:
```

```
    ...  
    info      Display system-wide information  
    ...
```

```
Run 'docker COMMAND --help' for more information on a command.
```

# Gestion des conteneurs Docker

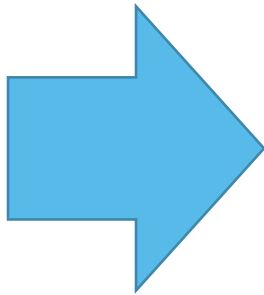
## Usage de l'autocomplétion

```
docker COMMAND
```

```
docker volumes
```



Tabulation



```
create  -- Create a volume
inspect -- Display detailed information on one or more volumes
ls      -- List volumes
prune   -- Remove all unused volumes
rm      -- Remove one or more volumes
```

cela affichera toutes les options possibles pour la sous-commande volumes, ainsi que leurs descriptions.

# Gestion des conteneurs Docker

Plus d'infos

```
docker info
```

Cette commande, nous fournit plusieurs informations concernant les spécifications du moteur Docker. Elle nous transmet aussi d'autres informations telles que le nombre total de conteneurs tournant sur notre machine ainsi que leur état :

```
Containers: 3  
Running: 0  
Paused: 0  
Stopped: 3
```

# Gestion des conteneurs Docker

## Lister les images téléchargées

```
docker image ls
```

commande qui permet de répertorier les images Docker téléchargées sur votre ordinateur

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hello-world	latest	fce289e99eb9	5 months ago	1.84kB

le nom de l'image.

...

façon de faire référence à votre image, ils sont utilisés principalement pour affecter une version à une image

Identifiant de l'image (unique pour chaque image téléchargée)

Date de la dernière modification de l'image

Taille de l'image



# Supprimer une image Docker

Maintenant si on souhaite supprimer une image Docker, on aura besoin soit de son IMAGE ID soit de son nom. Une fois qu'on aura récupéré ces informations, on peut passer à la suite en lançant la commande suivante :

avec l'id de l'image :

```
docker rmi fce289e99eb9
```

avec le nom de l'image :

```
docker rmi hello-world
```

# Supprimer une image Docker

Si vous lancez cette commande et avez le message d'erreur suivant

```
Error response from daemon: conflict: unable to remove repository reference "hello-world" (must force) - container 3e444920f82d is using its referenced image fce289e99eb9
```

On ne peut pas supprimer notre image Docker car des conteneurs ont été instanciés depuis notre image "hello-world".

On peut **forcer la suppression** pour éliminer aussi les conteneurs liés à notre image (must force)

Pour forcer la supprimer on va utiliser l'option --force ou -f.

```
docker rmi -f hello-world
```

Si plusieurs images avec le même nom mais avec des tags différents, vous devez préciser le tag dans votre commande rmi, pour ainsi être sûr de supprimer la bonne image. Pour information par défaut Docker prend le tag latest.

# Supprimer toutes les images

Commande qui permet de supprimer toutes les images disponibles sur notre machine :

```
docker rmi -f $(docker images -q)
```

# Quelques commandes à connaître

- **docker ps** : Affiche une liste des conteneurs Docker en cours d'exécution sur votre système.
- **docker ps -a** : Affiche une liste de tous les conteneurs Docker, y compris ceux qui sont arrêtés.
- **docker run** : Démarre un nouveau conteneur Docker à partir d'une image spécifiée.
- **docker stop** : Arrête un conteneur Docker en cours d'exécution.
- **docker start** : Démarre un conteneur Docker arrêté.
- **docker restart** : Redémarre un conteneur Docker en cours d'exécution.
- **docker rm** : Supprime un ou plusieurs conteneurs Docker. Vous pouvez spécifier le nom ou l'ID du conteneur à supprimer.
- **docker kill** : Force l'arrêt d'un conteneur Docker en cours d'exécution en envoyant un signal KILL.
- **docker logs** : Affiche les journaux d'un conteneur Docker spécifique..

# Quelques commandes à connaître

- **docker exec** : Exécute une commande à l'intérieur d'un conteneur Docker en cours d'exécution.
- **docker inspect** : Affiche les détails d'un conteneur Docker, y compris les informations sur les réseaux, les volumes, etc.
- **docker pause** : Met en pause l'exécution d'un conteneur Docker en cours.
- **docker unpause** : Reprend l'exécution d'un conteneur Docker mis en pause.
- **docker rename** : Renomme un conteneur Docker.
- **docker wait** : Attends que le processus d'un conteneur Docker se termine, puis retourne le code de sortie.
- **docker stats** : Affiche des informations en temps réel sur l'utilisation des ressources par les conteneurs Docker en cours d'exécution.

Ces commandes sont essentielles pour la gestion quotidienne des conteneurs Docker, vous permettant de contrôler, surveiller et interagir avec vos conteneurs de manière efficace.

# Créer des images Docker

# Création d'une image Docker

Ecrire un Dockerfile qui définit les étapes nécessaires à la construction de l'image.

Pour construire une image Docker à partir d'un Dockerfile, placez-le dans un répertoire contenant les fichiers de votre application et exécutez la commande dans le terminal :

```
docker build -t nom_image chemin_du_dockerfile
```

# INSTRUCTIONS DOCKER FILE

Ces instructions peuvent être combinées dans un Dockerfile pour construire une image Docker personnalisée. Chaque instruction est exécutée séquentiellement, et l'état du système de fichiers est conservé entre chaque instruction, ce qui permet de construire une image en couches (layer).

**FROM** : Spécifie l'image de base à utiliser. Par exemple : FROM ubuntu:20.04 pour utiliser l'image Ubuntu 20.04 comme base.

**MAINTAINER** : Facultatif. Indique le nom et l'email du mainteneur de l'image. Par exemple : MAINTAINER John Doe <john@example.com>.



# INSTRUCTIONS DOCKER FILE

**RUN** : Exécute des commandes shell pendant le processus de construction de l'image. Par exemple : RUN apt-get update && apt-get install -y curl.

**COPY** : Copie des fichiers ou des répertoires depuis l'hôte dans l'image. Par exemple : COPY app.py /app/.

**ADD** : Similaire à COPY, mais permet également de récupérer des URL et d'extraire des fichiers tar. Par exemple : ADD https://example.com/file.tar.gz /tmp/.

**WORKDIR** : Définit le répertoire de travail pour les instructions suivantes. Par exemple : WORKDIR /app.

# INSTRUCTIONS DOCKER FILE

**EXPOSE** : Indique les ports sur lesquels l'application à l'intérieur du conteneur écoute. Par exemple : EXPOSE 8080.

**CMD** : Définit la commande par défaut à exécuter lorsque le conteneur est démarré. Cette instruction peut être remplacée par la commande spécifiée lors du démarrage du conteneur. Par exemple : CMD ["python", "app.py"].

**ENTRYPOINT** : Définit la commande à exécuter lorsque le conteneur est démarré. Elle ne peut pas être remplacée lors du démarrage du conteneur. Par exemple : ENTRYPOINT ["python", "app.py"].

# ECRIRE UN DOCKER FILE

**ENV** : Définit des variables d'environnement dans le conteneur. Par exemple : ENV ENV\_VARIABLE=value.

**ARG** : Définit des arguments qui peuvent être passés lors de la construction de l'image Docker. Par exemple : ARG VERSION=latest.

**LABEL** : Ajoute des métadonnées à une image. Par exemple : LABEL version="1.0" description="Ceci est une image Docker d'exemple".

# EXEMPLE DE DOCKER FILE

## Construire une image Docker pour une application web Python Flask

```
# Utilisez une image de base avec Python
FROM python:3.9-slim

# Définir le répertoire de travail dans le conteneur
WORKDIR /app

# Copier le fichier requirements.txt dans le conteneur
COPY requirements.txt .

# Installer les dépendances Python
RUN pip install --no-cache-dir -r requirements.txt
```

```
# Copier le code source de l'application dans le conteneur
COPY . .

# Exposer le port 5000 pour que l'application puisse être accessible
EXPOSE 5000

# Commande pour exécuter l'application lorsque le conteneur démarre
CMD ["python", "app.py"]
```

# EXEMPLE DE DOCKER FILE

Dockerfile pour une image pour un serveur web Nginx

```
# Utilisez une image de base avec Nginx
FROM nginx:latest

# Copier les fichiers de configuration personnalisés dans le conteneur
COPY nginx.conf /etc/nginx/nginx.conf

# Exposer le port 80 pour que le serveur web puisse être accessible
EXPOSE 80

# Commande pour démarrer Nginx lorsque le conteneur démarre
CMD ["nginx", "-g", "daemon off;"]
```

# EXEMPLE DE DOCKER FILE

Exemple de Dockerfile pour exécuter MySQL avec phpMyAdmin dans des conteneurs distincts

```
# Dockerfile pour MySQL

# Utilisez une image de base MySQL
FROM mysql:8.0

# Définir le mot de passe root pour MySQL
ENV MYSQL_ROOT_PASSWORD=password

# Exposez le port MySQL
EXPOSE 3306
```

Ce Dockerfile crée une image Docker basée sur l'image officielle de MySQL 8.0 et définit un mot de passe root pour MySQL. L'instruction EXPOSE expose le port MySQL (par défaut, 3306) pour que d'autres conteneurs puissent y accéder.

# EXEMPLE DE DOCKER FILE

Exemple de Dockerfile pour exécuter MySQL avec phpMyAdmin dans des conteneurs distincts

```
# Dockerfile pour phpMyAdmin

# Utilisez une image de base phpMyAdmin
FROM phpmyadmin/phpmyadmin

# Définissez les variables d'environnement pour la connexion à la base de données
ENV PMA_HOST=mysql
ENV PMA_PORT=3306
ENV PMA_USER=root
ENV PMA_PASSWORD=password

# Exposez le port phpMyAdmin
EXPOSE 80
```

Ce Dockerfile utilise l'image officielle de phpMyAdmin et définit les variables d'environnement pour la connexion à la base de données MySQL, y compris l'hôte, le port, le nom d'utilisateur et le mot de passe. L'instruction EXPOSE expose le port 80 pour accéder à phpMyAdmin depuis le navigateur.

# EXEMPLE DE DOCKER FILE

Pour exécuter MySQL et phpMyAdmin ensemble, vous pouvez utiliser un fichier docker-compose.yml.

```
version: '3'

services:
  mysql:
    build:
      context: .
      dockerfile: Dockerfile.mysql
    container_name: mysql
    restart: always
```

```
phpmyadmin:
  build:
    context: .
    dockerfile: Dockerfile.phpmyadmin
  container_name: phpmyadmin
  ports:
    - "8080:80"
  restart: always
  depends_on:
    - mysql
```

Ce fichier docker-compose.yml définit deux services, l'un pour MySQL et l'autre pour phpMyAdmin. Il lie également le port 8080 de l'hôte à phpMyAdmin pour y accéder via le navigateur.



## EXEMPLE DE DOCKER FILE

Assurez-vous d'avoir les Dockerfiles (Dockerfile.mysql et Dockerfile.phpmyadmin) et le fichier docker-compose.yml dans le même répertoire, puis exécutez la commande suivante pour démarrer les conteneurs

```
docker-compose up -d
```

Cela va construire les images et démarrer les conteneurs MySQL et phpMyAdmin.

Vous pourrez alors accéder à phpMyAdmin via <http://localhost:8080> dans votre navigateur.

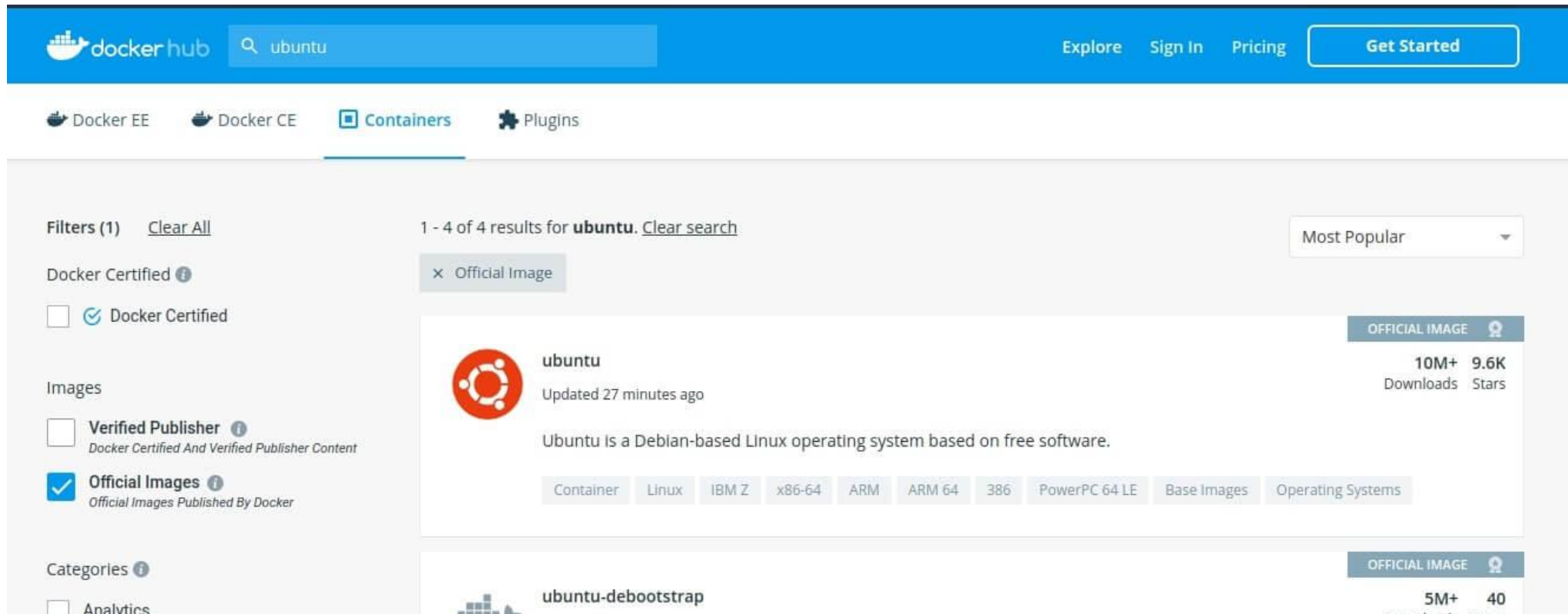
# Manipuler des images Docker

## Télécharger une image depuis le Docker Hub Registry

Il existe deux façons pour voir si une image est disponible dans le Docker Hub Registry,

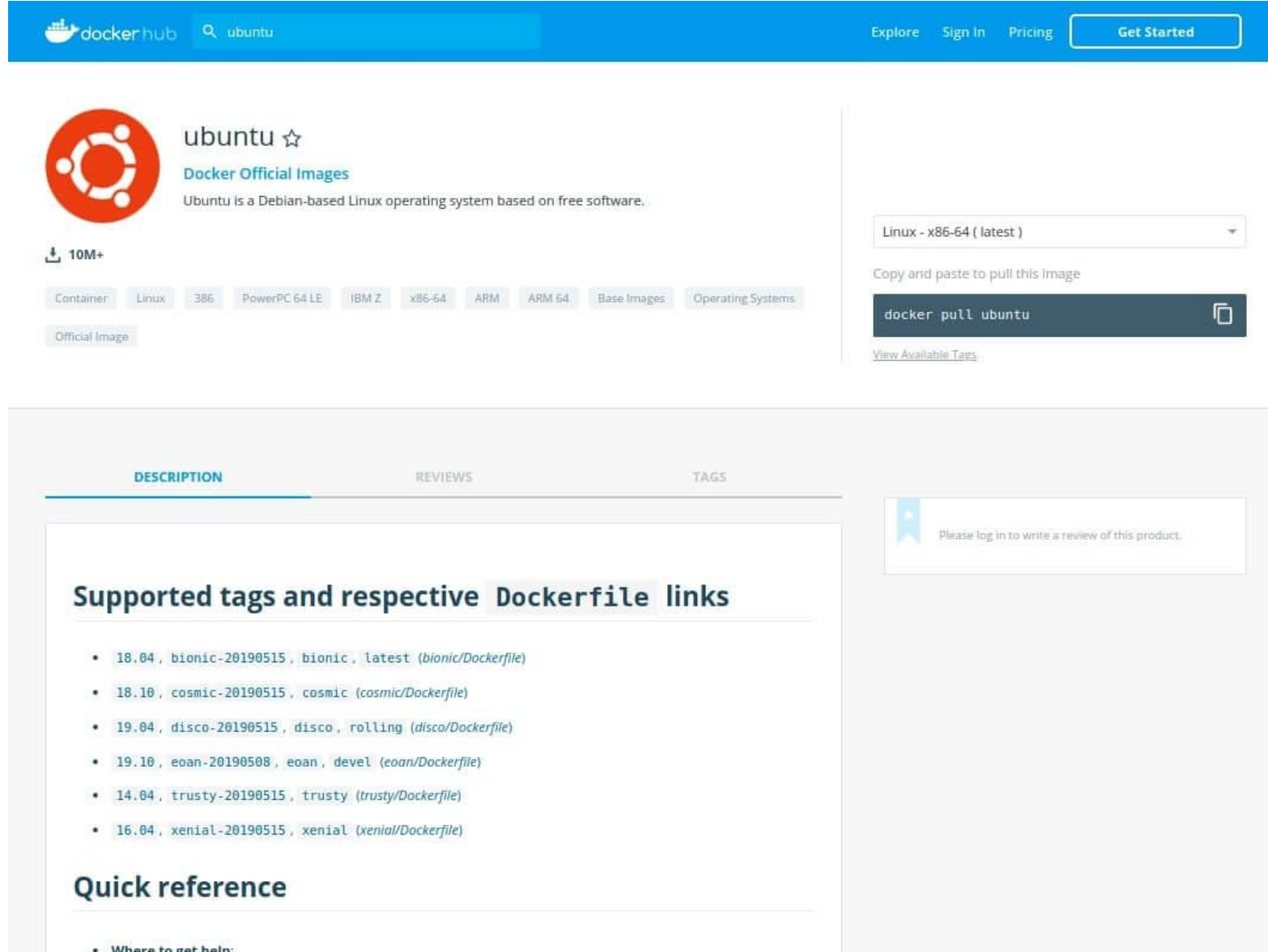
Depuis le site <https://hub.docker.com/>

case "Official  
Image" pour  
n'afficher que les  
images officielles



The screenshot shows the Docker Hub interface. At the top, there's a blue header with the Docker Hub logo, a search bar containing 'ubuntu', and links for 'Explore', 'Sign In', 'Pricing', and a 'Get Started' button. Below the header, there are tabs for 'Docker EE', 'Docker CE', 'Containers' (which is selected), and 'Plugins'. The main content area shows search results for 'ubuntu'. On the left, there are filters: 'Filters (1) Clear All', 'Docker Certified' (unchecked), 'Images' (with 'Verified Publisher' unchecked and 'Official Images' checked), and 'Categories' (with 'Analytics' unchecked). The search results show '1 - 4 of 4 results for ubuntu. Clear search'. A dropdown menu is set to 'Most Popular'. The first result is 'ubuntu', marked as an 'OFFICIAL IMAGE' with a shield icon. It shows '10M+ Downloads' and '9.6K Stars', and was 'Updated 27 minutes ago'. Below the title, it says 'Ubuntu is a Debian-based Linux operating system based on free software.' and lists various architectures: 'Container', 'Linux', 'IBM Z', 'x86-64', 'ARM', 'ARM 64', '386', 'PowerPC 64 LE', 'Base Images', and 'Operating Systems'. The second result, 'ubuntu-debootstrap', is partially visible at the bottom, also marked as an 'OFFICIAL IMAGE' with '5M+' downloads and '40' stars.

si on clique sur l'image officielle d'ubuntu, on tombe sur la page suivante :



The screenshot shows the Docker Hub interface for the 'ubuntu' image. The top navigation bar is blue with the Docker Hub logo, a search bar containing 'ubuntu', and links for 'Explore', 'Sign In', 'Pricing', and a 'Get Started' button. The main content area features the Ubuntu logo, the name 'ubuntu' with a star, and the text 'Docker Official Images'. Below this, it states 'Ubuntu is a Debian-based Linux operating system based on free software.' and shows a download count of '10M+'. A row of tags includes 'Container', 'Linux', '386', 'PowerPC 64 LE', 'IBM Z', 'x86-64', 'ARM', 'ARM 64', 'Base Images', and 'Operating Systems'. A button labeled 'Official Image' is also present. On the right, a dropdown menu shows 'Linux - x86-64 ( latest )', a text box for copying the command 'docker pull ubuntu', and a 'View Available Tags' link. Below the main content, there are tabs for 'DESCRIPTION', 'REVIEWS', and 'TAGS'. The 'DESCRIPTION' tab is active, showing a section titled 'Supported tags and respective Dockerfile links' with a list of tags and their corresponding Dockerfile links. A 'Quick reference' section is also visible at the bottom.

docker hub ubuntu Explore Sign In Pricing Get Started

ubuntu ☆  
Docker Official Images  
Ubuntu is a Debian-based Linux operating system based on free software.

10M+

Container Linux 386 PowerPC 64 LE IBM Z x86-64 ARM ARM 64 Base Images Operating Systems

Official Image

Linux - x86-64 ( latest )

Copy and paste to pull this image

```
docker pull ubuntu
```

[View Available Tags](#)

DESCRIPTION REVIEWS TAGS

### Supported tags and respective Dockerfile links

- 18.04, bionic-20190515, bionic, latest ([bionic/Dockerfile](#))
- 18.10, cosmic-20190515, cosmic ([cosmic/Dockerfile](#))
- 19.04, disco-20190515, disco, rolling ([disco/Dockerfile](#))
- 19.10, eoan-20190508, eoan, devel ([eoan/Dockerfile](#))
- 14.04, trusty-20190515, trusty ([trusty/Dockerfile](#))
- 16.04, xenial-20190515, xenial ([xenial/Dockerfile](#))

### Quick reference

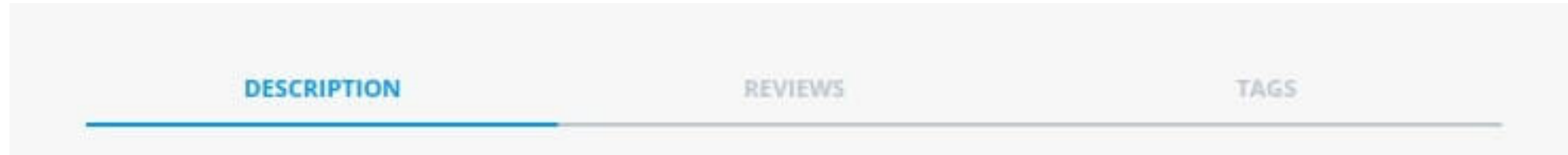
- Where to get help:

Please log in to write a review of this product.

en haut à droite, on retrouve le nom de l'image avec une toute petite description



Plus bas, un menu de navigation contenant :



Description de l'image, souvent on retrouve quelques tags, la configuration de votre conteneur (par exemple la config de votre base de données pour une image basé sur du mysql) et les liens github vers les sources du projet.

l'avis des utilisateurs

les différents tags disponible pour cette image

En haut à droit , la commande à lancer permettant de télécharger l'image

Copy and paste to pull this image

```
docker pull ubuntu
```



[View Available Tags](#)

# Télécharger une image depuis le Docker Hub Registry

Chercher l'image en ligne de commande

```
docker search ubuntu
```

NAME	DESCRIPTION	STARS
ubuntu	Ubuntu is a Debian-based Linux operating sys...	9616
...		
pivotaldata/ubuntu-gpdb-dev	Ubuntu images for GPDB development	0

Si vous souhaitez n'afficher que les images officielles, il est possible de filtrer le résultat avec la commande suivante

```
docker search --filter "is-official=true" ubuntu
```

# Télécharger une image depuis le Docker Hub Registry

## Installer l'image en ligne de commande

Pour télécharger une image depuis le Docker hub Registry il faut utiliser la commande suivante pull avec éventuellement un tag

```
docker pull ubuntu
```

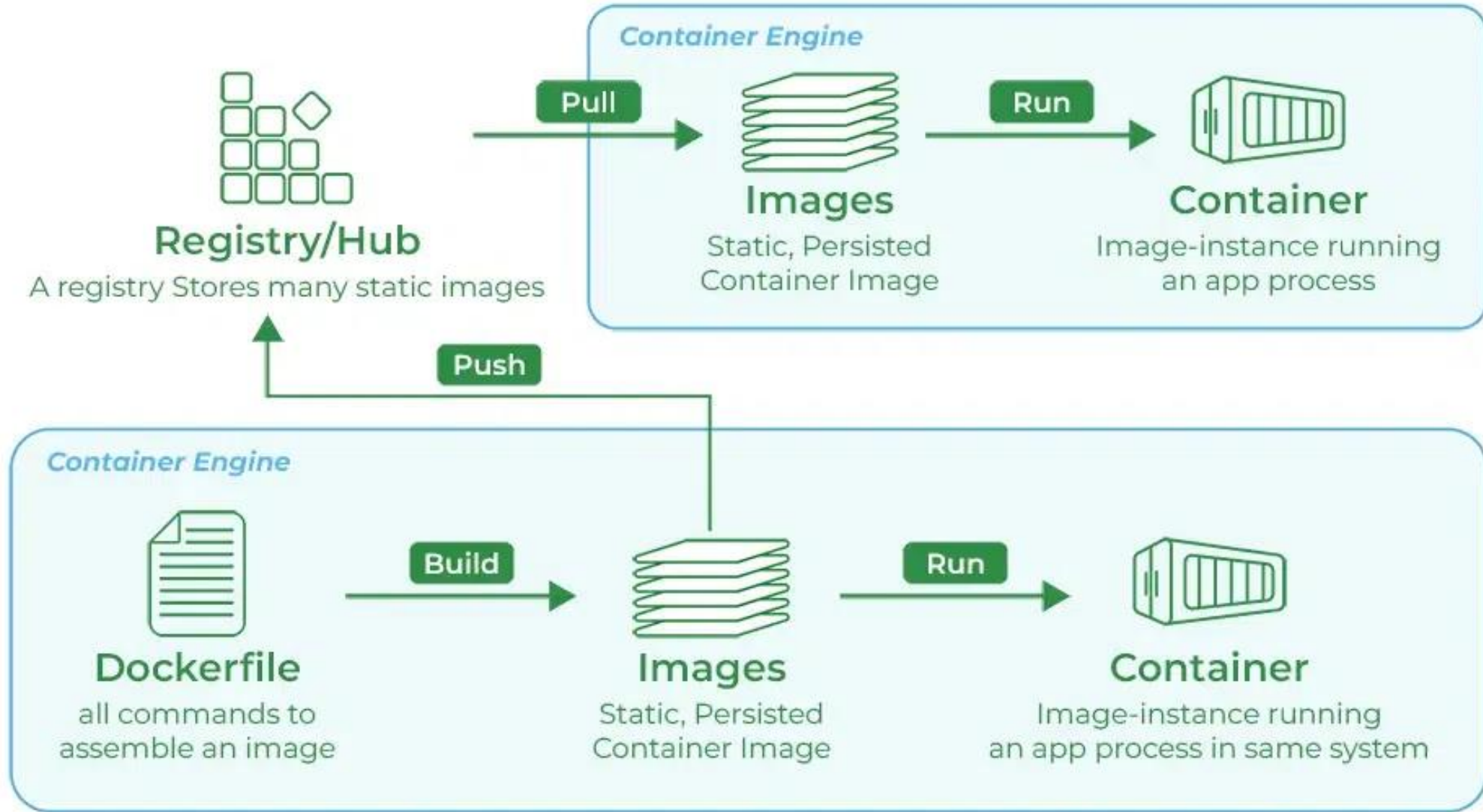
Pour télécharger une image ubuntu avec un autre tag différent de latest par exemple le tag 16.04:

```
docker pull ubuntu:16.04
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ubuntu	16.04	2a697363a870	3 weeks ago	119MB
ubuntu	latest	7698f282e524	3 weeks ago	69.9MB
hello-world	latest	fce289e99eb9	5 months ago	1.84kB

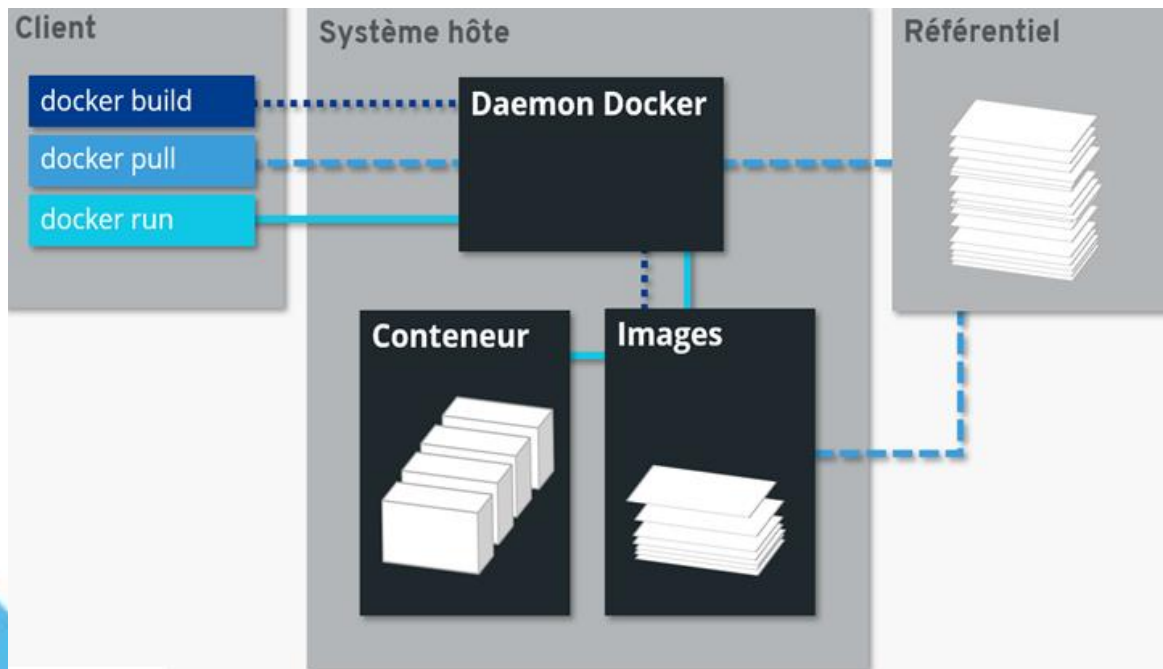


- <https://devopssec.fr/article/fonctionnement-manipulation-conteneurs-docker>



# En savoir plus sur le Docker Engine

Le cœur du projet est le moteur Docker. Il s'agit d'une **application client-serveur ouverte à tous**, dont la version actuelle est disponible pour les utilisateurs sur toutes les plateformes établies.



Le moteur Docker est composé de trois éléments fondamentaux :

- un **daemon** doté de fonctions de serveur,
- une **interface de programmation (API)** basée sur le principe de programmation REST (*Representational State Transfer*),
- le **terminal du système d'exploitation** (*Command-Line Interface, CLI*), servant d'interface utilisateur

# Conclusion

Docker simplifie le développement, le déploiement et la gestion des applications en utilisant des conteneurs logiciels.

En comprenant les principes de base de Docker et en apprenant à utiliser ses fonctionnalités, vous pouvez améliorer l'efficacité de votre processus de développement et de déploiement d'applications.