

# INTRODUCTION DOCKER

UTOPIOS



ANTHONY DI PERSIO



## DECOUVERTE DOCKER

Comprendre le concept de conteneurs, avantages et inconvénients

01

## INSTALLATION DE DOCKER

Installation des outils de Docker, découverte de l'écosystème

02

## LES IMAGES & CONTENEURS

Comprendre le concept d'image et de conteneur Docker

03

## LE DOCKERFILE

Créer une image et l'instancier en un conteneur Docker

04

## TABLE DES MATIÈRES

05

## DOCKER VOLUME

Comprendre la persistance des données dans Docker

06

## DOCKER-COMPOSE

Comprendre le fonctionnement d'une application multi conteneurs

07

## LE NETWORK DOCKER

Appréhender la communication réseau avec Docker

08

## LES ORCHESTRATEURS

Introduction à la notion d'orchestrateur de conteneurs



01

# DECOUVERTE DOCKER

Comprendre le concept de conteneurs, avantages et inconvénients

# DECOUVERTE DE DOCKER

- Qu'est-ce que Docker ?
  - Le logiciel « Docker » est une technologie de conteneurisation qui permet la création et l'utilisation de conteneurs Linux.
- Le terme « Docker » désigne plusieurs choses
  - le projet d'une communauté Open Source
  - les outils issus de ce projet Open Source
  - l'entreprise Docker Inc. qui constitue le principal soutien de ce projet
  - les outils que l'entreprise prend officiellement en charge

# DECOUVERTE DE DOCKER

- Donc...? Qu'est-ce que **Docker** ?
  - Le logiciel « Docker » est une technologie de conteneurisation qui permet la création et l'utilisation de conteneurs Linux®
  - La communauté Open Source Docker travaille à l'amélioration de cette technologie disponible gratuitement pour tout le monde.
  - L'entreprise **Docker Inc.** s'appuie sur le travail de la communauté Docker, sécurise sa technologie et partage ses avancées avec tous les utilisateurs. Elle prend ensuite en charge les technologies améliorées et sécurisées pour ses clients professionnels.

# DECOUVERTE DE DOCKER

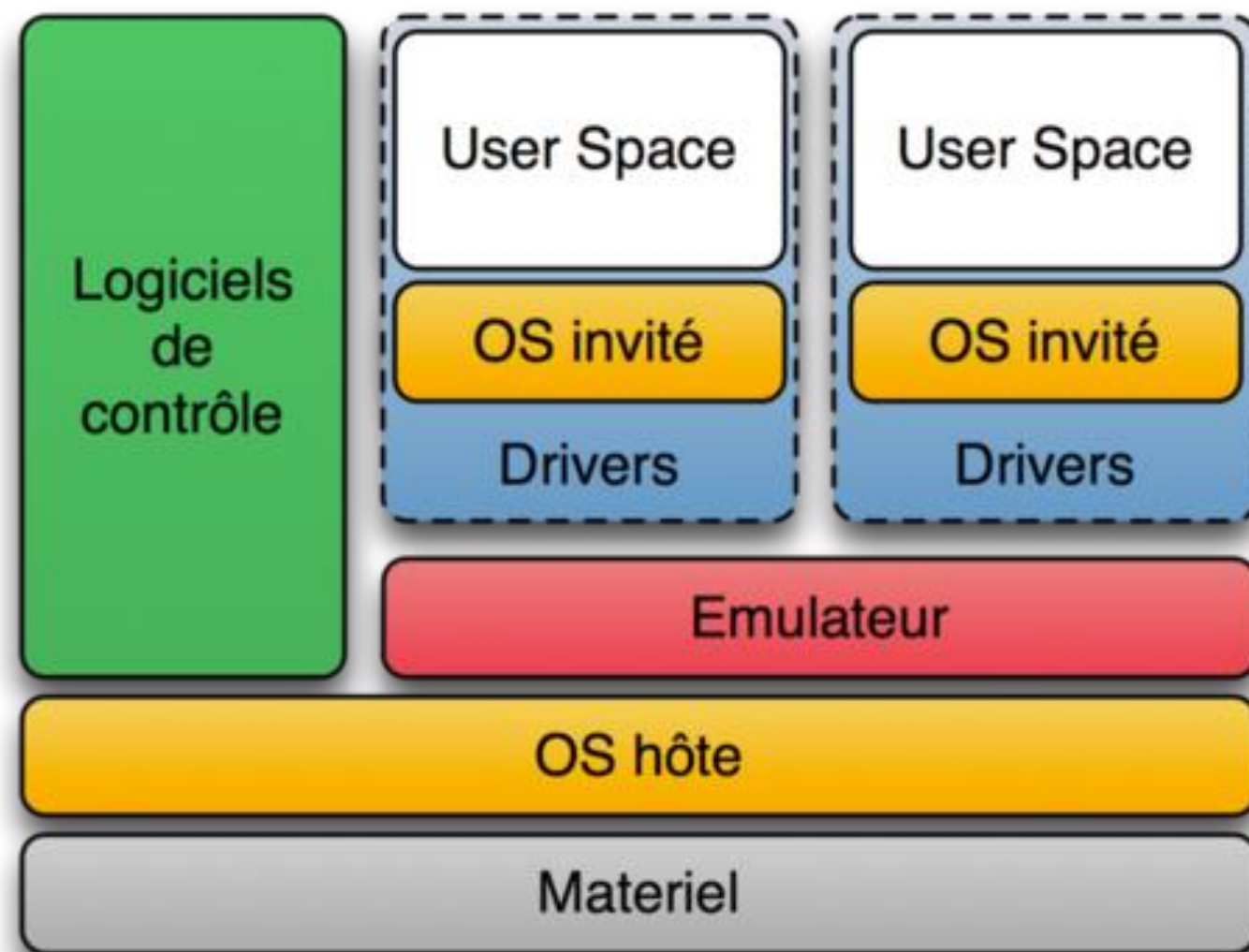
- À quoi sert **Docker** ?
  - Avec la technologie **Docker**, vous pouvez traiter les conteneurs comme des machines virtuelles très légères et modulaires
  - Ces conteneurs vous offrent une grande flexibilité :
    - ✓ Vous pouvez les créer, déployer des environnements très rapidement
    - ✓ Vous pouvez les copier et déplacer d'un environnement à un autre très facilement
    - ✓ Il vous permet d'optimiser vos applications pour le cloud

# DECOUVERTE DE DOCKER

- Virtualisation vs Conteneurisation?
  - La virtualisation est la capacité de faire tourner un, ou plusieurs serveur virtuel sur une seule machine physique grâce à un hyperviseur
  - L'hyperviseur permet d'émuler les différentes ressources matérielles d'un serveur physique et permet à des machines virtuelles de les partager
  - Une machine virtuelle possède ses propres ressources matérielles et son propre système d'exploitation

# DECOUVERTE DE DOCKER

- Détails d'une Machine Virtuelle (VM)





# DECOUVERTE DE DOCKER

- La **virtualisation** apporte des **avantages** :
  - Ressources adaptées aux besoins de l'application.
  - Faciliter de manipulation (Sauvegarde, bascule,...)
  - Réduction des dépenses et réduction d'équipements nécessaires
  - Facilités pour l'administration
- La virtualisation a des inconvénients également :
  - Réduction des performances
  - Multiplication des couches OS

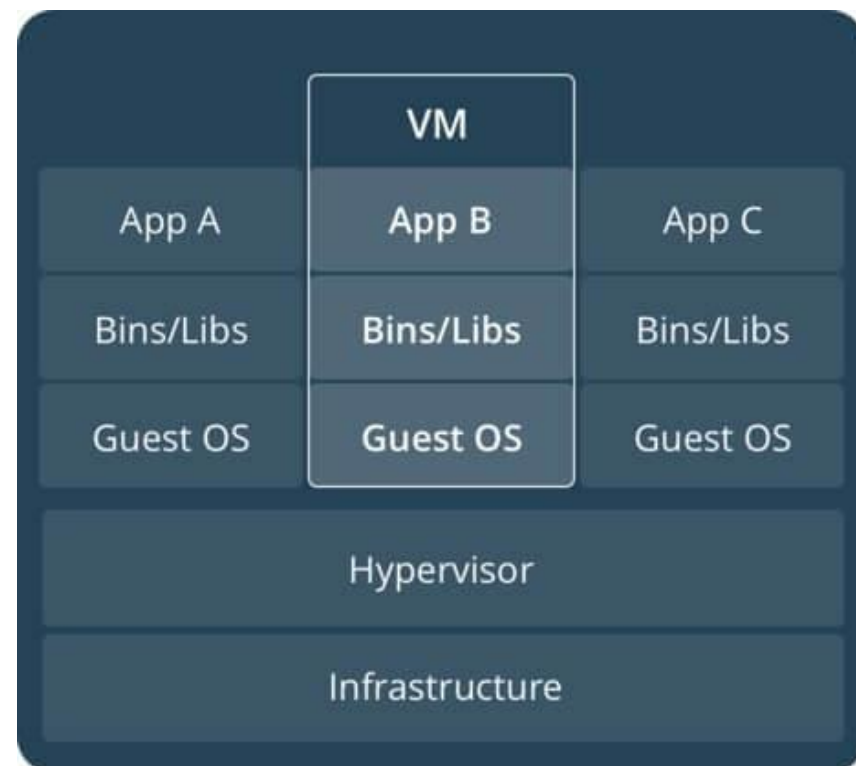
# DECOUVERTE DE DOCKER

- Virtualisation vs Conteneurisation?
  - Isolation en VM
    - ✓ Se fait au niveau matérielle
    - ✓ Accès virtuel aux ressources via l'hôte à l'aide de l'hyperviseur
  - Isolation Dans la conteneurisation
    - ✓ Se fait au niveau du système d'exploitation
    - ✓ Exécution native sur linux et partage du noyau hôte avec les conteneurs

# DECOUVERTE DE DOCKER

- Virtualisation vs Conteneurisation?

Machines Virtuelles



Conteneurs Dockers



- Avantages de Docker

- ✓ Flexibilité et légèreté grâce au partage du noyau de l'hôte
- ✓ Scalabilité ou Extensibilité ...



# DECOUVERTE DE DOCKER

- Comment fonctionne la technologie **Docker** ?
  - La technologie Docker utilise le noyau Linux et des fonctions de ce noyau pour :
    - ✓ Séparer les processus afin qu'ils puissent s'exécuter de façon indépendante
  - Elle utilise des fonctionnalités nativement disponibles sur Linux
    - ✓ La création de groupes de contrôle « **cgroups** »
    - ✓ La création des espaces de noms « **namespaces** »

# DECOUVERTE DE DOCKER

- Un **conteneurs** et un **processus linux isolé** à l'aide de...
  - **Namespaces linux** qui sont un mécanisme permettant de limiter l'accès d'un processus
  - **Les cgroups linux** qui sont des mécanismes permettant de limiter l'accès aux ressources d'un processus
- Quelques exemples de **namespaces**:
  - Namespace **PID**
  - Namespace **USER**
- Quelques exemples de **cgroups**:
  - cgroup **cpuset**
  - cgroup **memory**

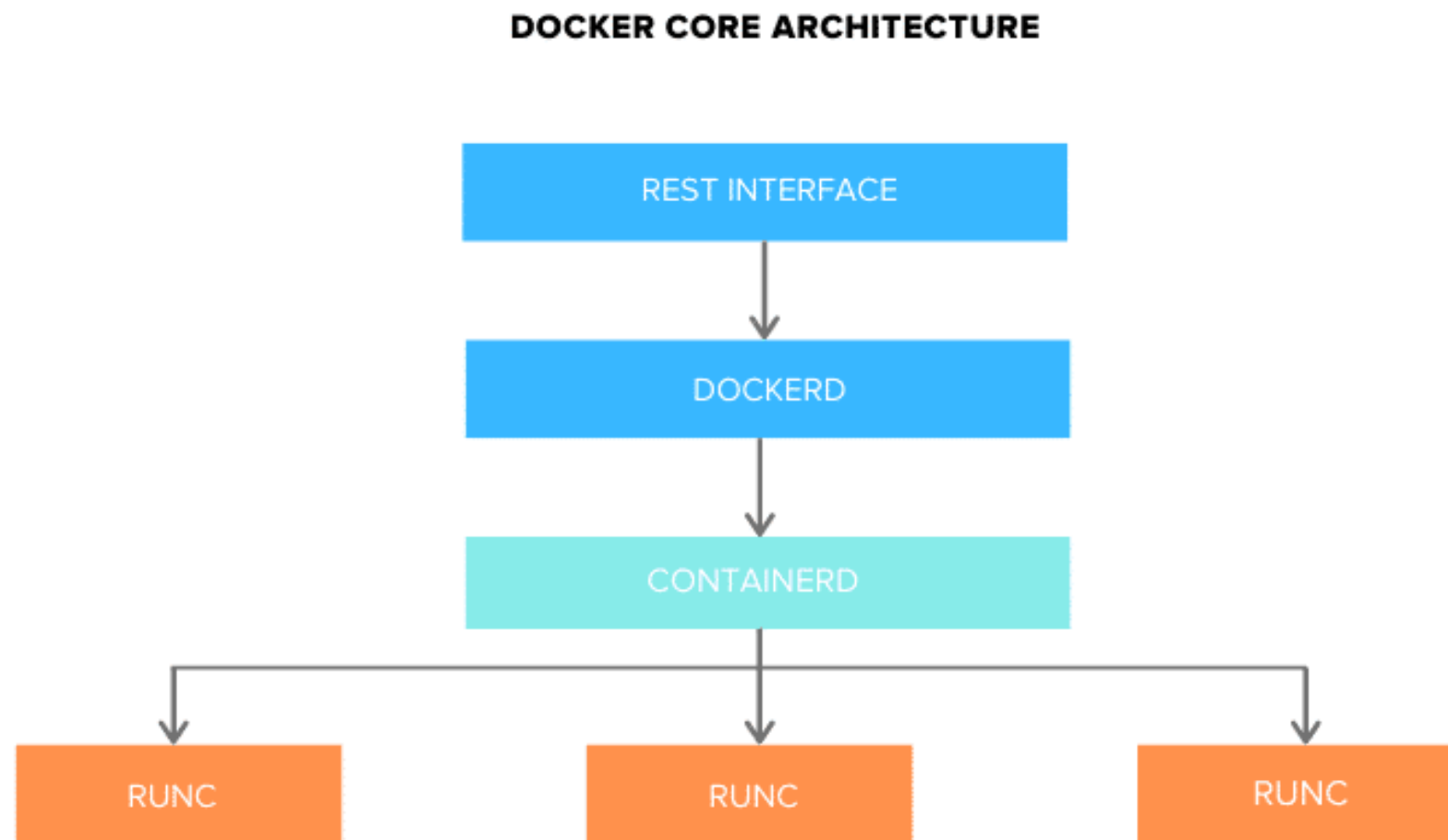
# DECOUVERTE DE DOCKER

- L'utilisation de conteneur n'est pas une technologie récente
  - Il y a de nombreuses applications qui utilisent ce concept
    - ✓ Chroot sur Unix (1982)
    - ✓ Jail sur BSD (2000)
    - ✓ Conteneurs sur Solaris (2004)
    - ✓ LXC (Linux conteneurs) sur Linux (2008)
- Les conteneurs ne sont pas nouveaux, mais leur utilisation pour déployer facilement des applications l'est.
  - La notoriété de docker vient du fait qu'il a su permettre aux utilisateurs de gérer facilement leurs conteneurs avec une interface en ligne de commande (CLI) très simple



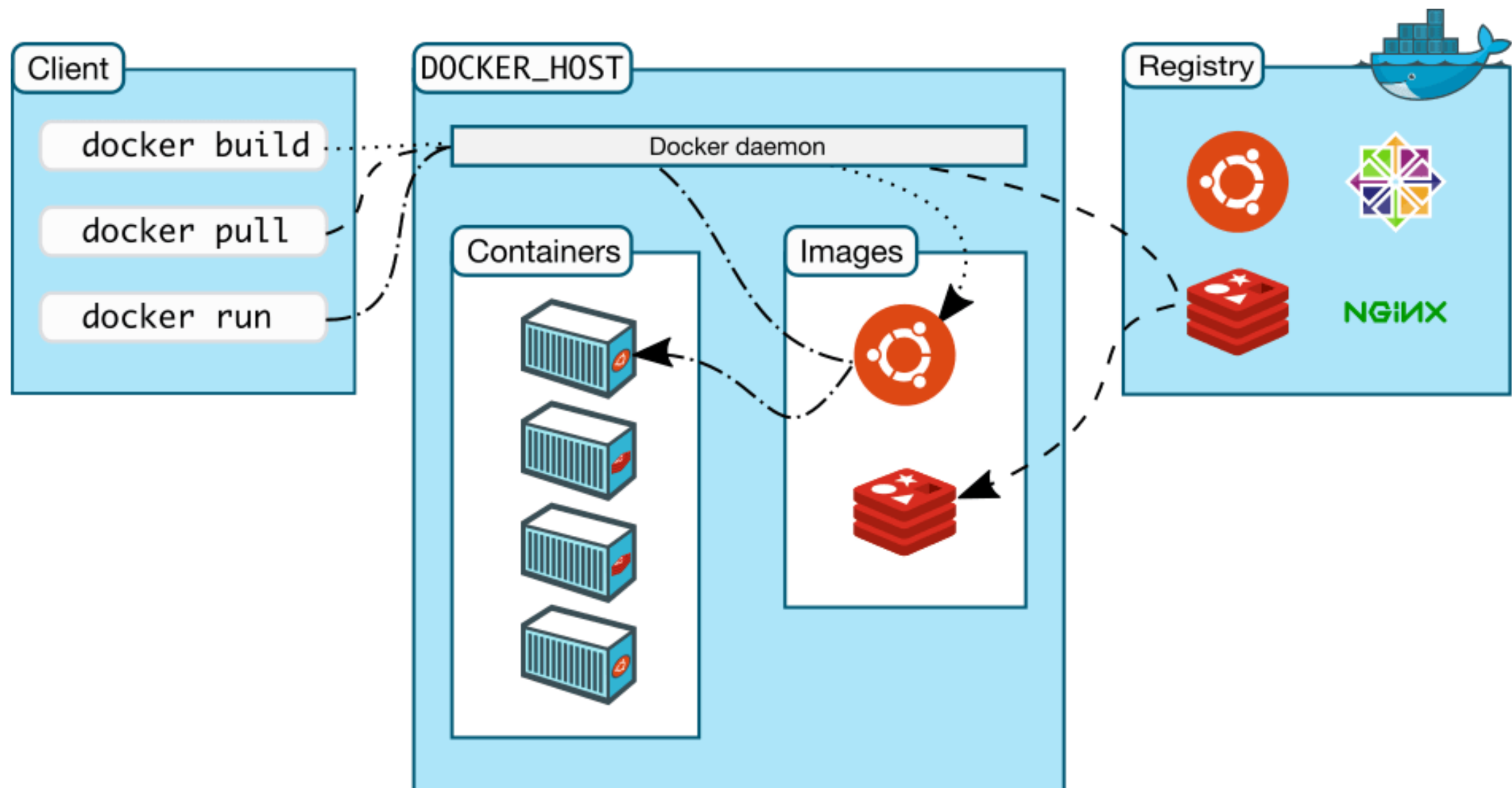
# DECOUVERTE DE DOCKER

- Docker est composé de :
  - Docker engine
  - Docker-containerd
  - Docker-runc



# DECOUVERTE DE DOCKER

- Le fonctionnement de Docker



02

# INSTALLATION DE DOCKER

Installation des outils de Docker, découverte de l'écosystème



# INSTALLATION DE DOCKER

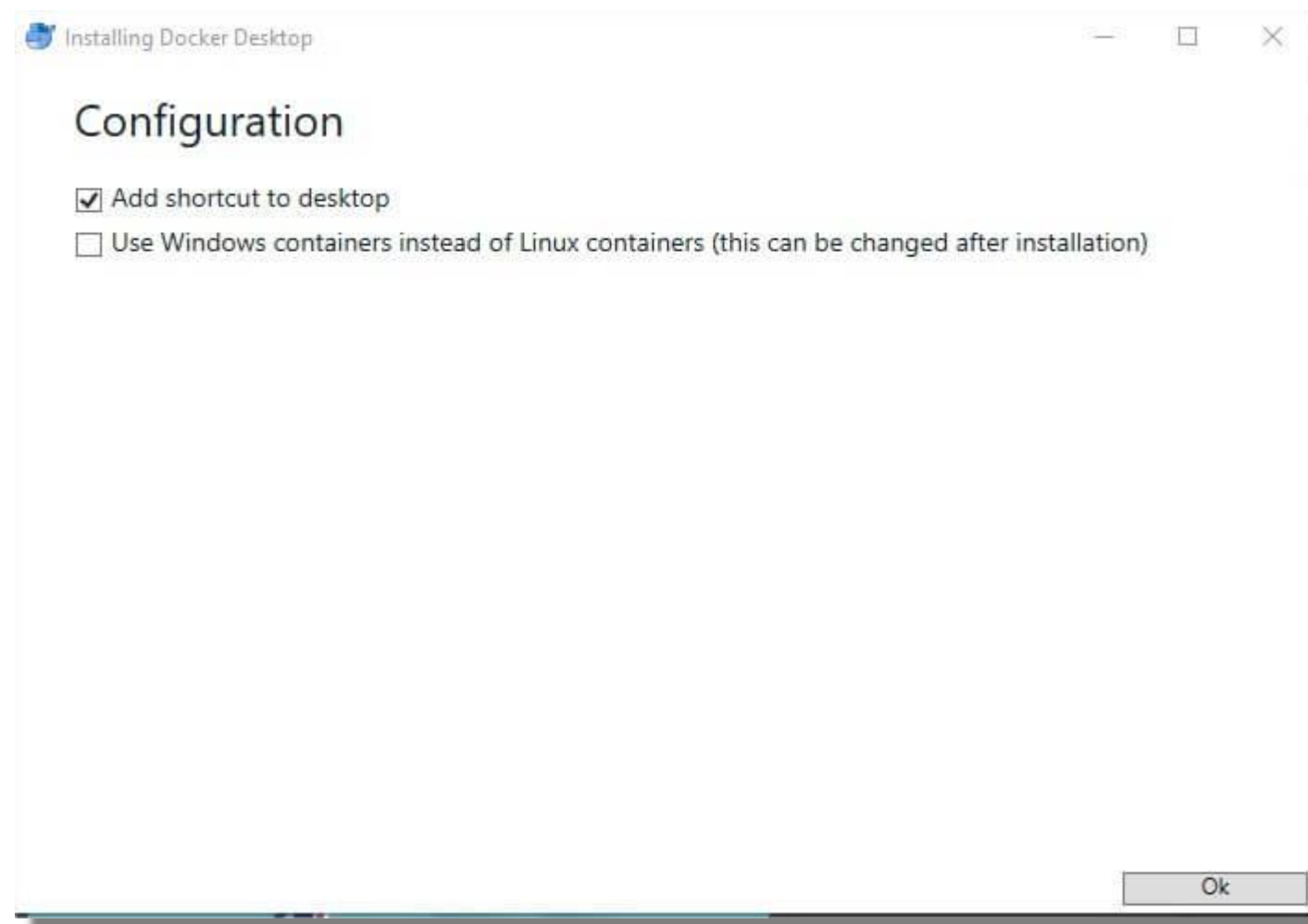
- Docker peut être installer sur toute type de distribution
  - Windows
  - Linux
  - MacOS
- Docker est disponible en deux éditions
  - **Docker Community Edition (CE)**
    - ✓ Idéale pour les développeurs individuels et les petites équipes cherchant à se familiariser avec Docker
  - **Docker Enterprise Edition (EE)**
    - ✓ Conçue pour les équipes de développement d'entreprise et les équipes système.

# INSTALLATION DE DOCKER

- Installation de **Docker** sur Windows
  - [hub.docker.com/](https://hub.docker.com/)
- Configuration minimale requise pour une installation Windows
  - **Windows 10 64 bits**: Pro, Entreprise ou Education (version 15063 ou ultérieure)
  - La **virtualisation** est **activée** dans le **BIOS** (normalement elle est activée par défaut, sinon activer **HyperV**)
  - Au moins **4Go** de **RAM**
- Pour utiliser **Docker** il vous faut un **compte Docker-Hub**
  - Une fois inscrit et connecté, Cliquer sur « **get Docker** » (CE)

# INSTALLATION DE DOCKER

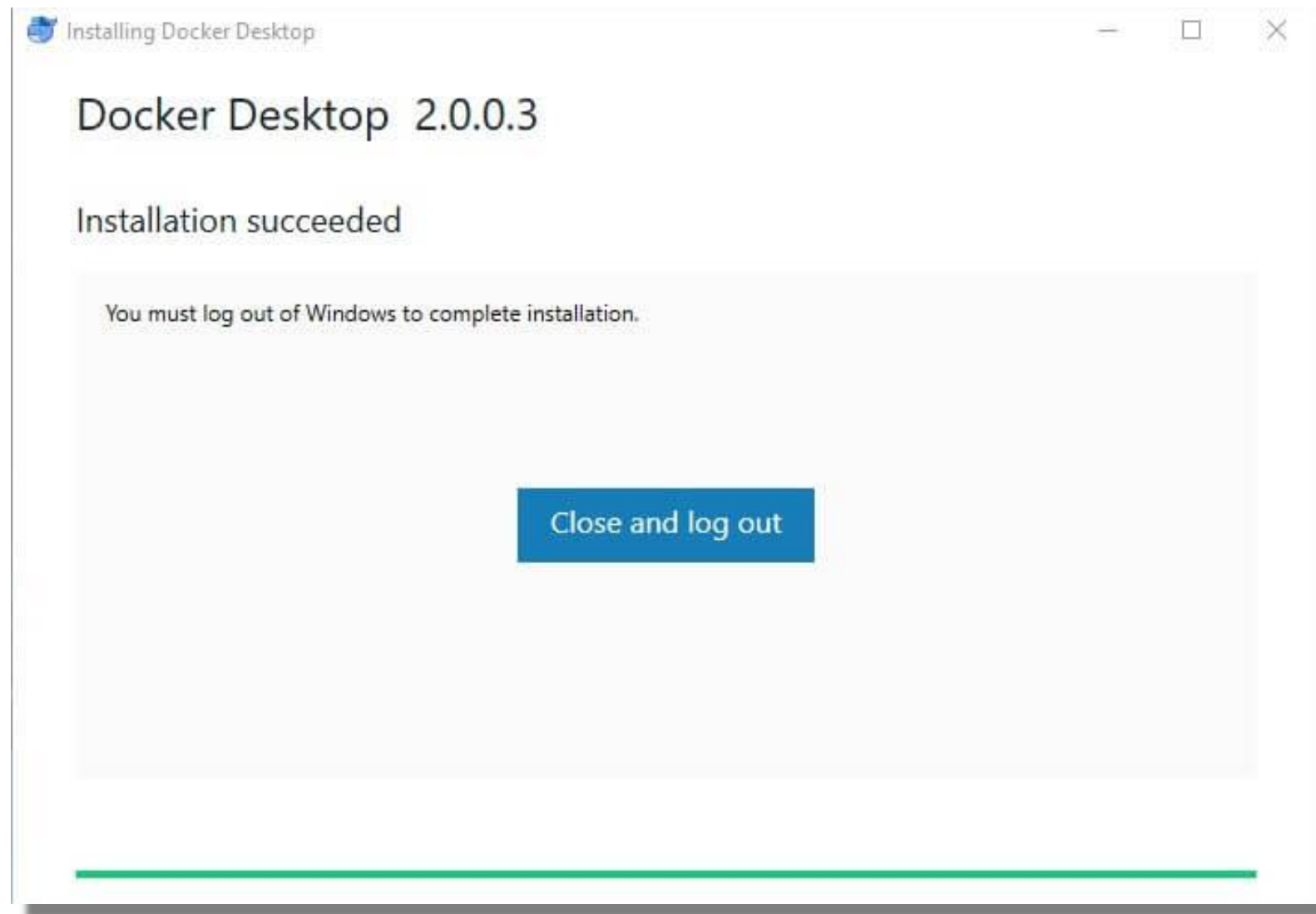
- Une fois Docker CE téléchargé, procéder à son installation





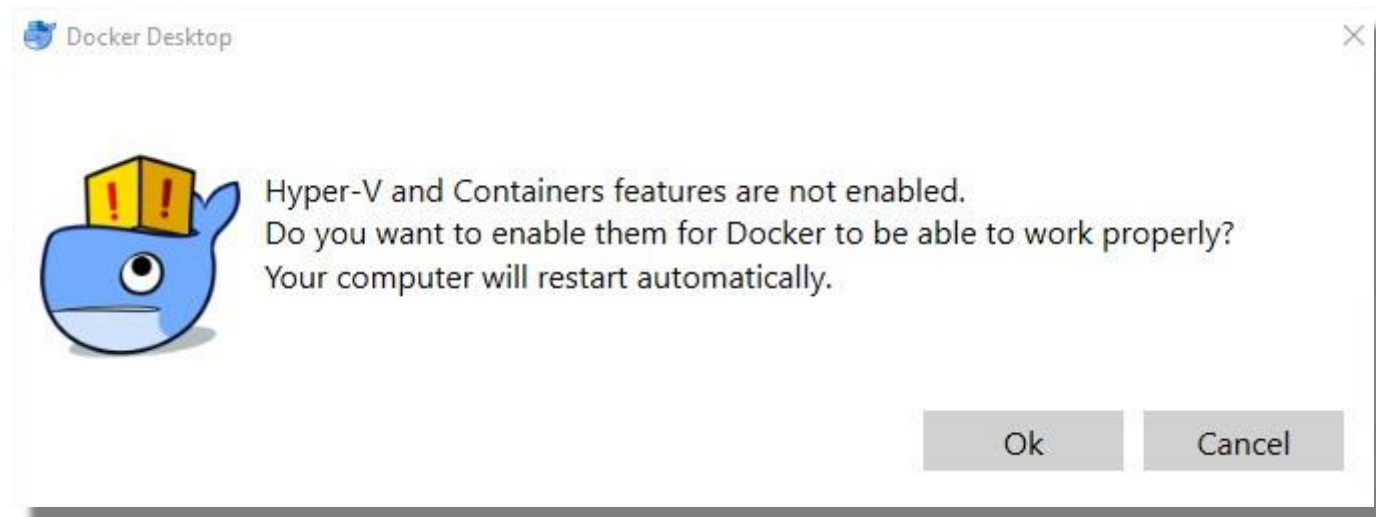
# INSTALLATION DE DOCKER

- Une fois terminée, vous recevrez le message suivant :



# INSTALLATION DE DOCKER

- Si vous n'avez pas activé Hyper-V, alors Docker s'en chargera



- Cliquez sur "OK" pour activer **Hyper-V**. Par la suite votre machine va **automatiquement se redémarrer** à la fin de l'activation d'Hyper-V

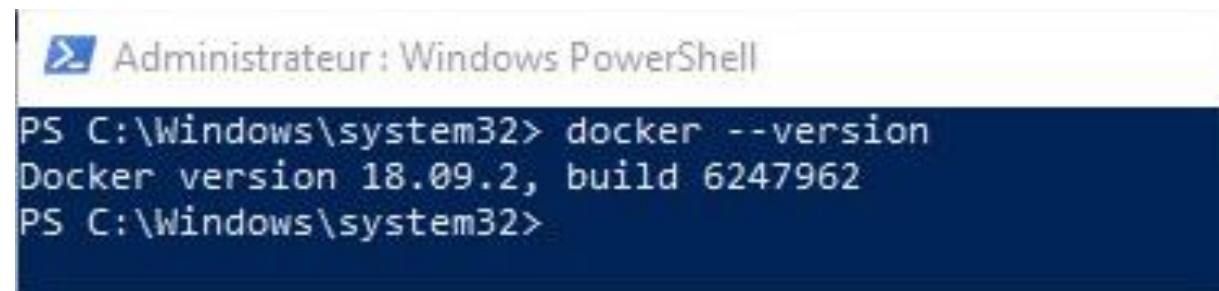
# INSTALLATION DE DOCKER

- Après votre redémarrage, Vous verrez la fenêtre suivante indiquant que le moteur Docker est bien installé



# INSTALLATION DE DOCKER

- Dernière étape, lancez votre powershell (ou terminal, CMD) en tant qu'administrateur et exécutez la commande suivante afin de vérifier que votre Docker CE c'est correctement installé
  - `$ docker --version`



```
Administrateur : Windows PowerShell
PS C:\Windows\system32> docker --version
Docker version 18.09.2, build 6247962
PS C:\Windows\system32>
```

- Docker est maintenant installé sur votre machine

03

# LES IMAGES & CONTENEURS

Comprendre le concept d'image et de conteneur Docker



# LES IMAGES DOCKER

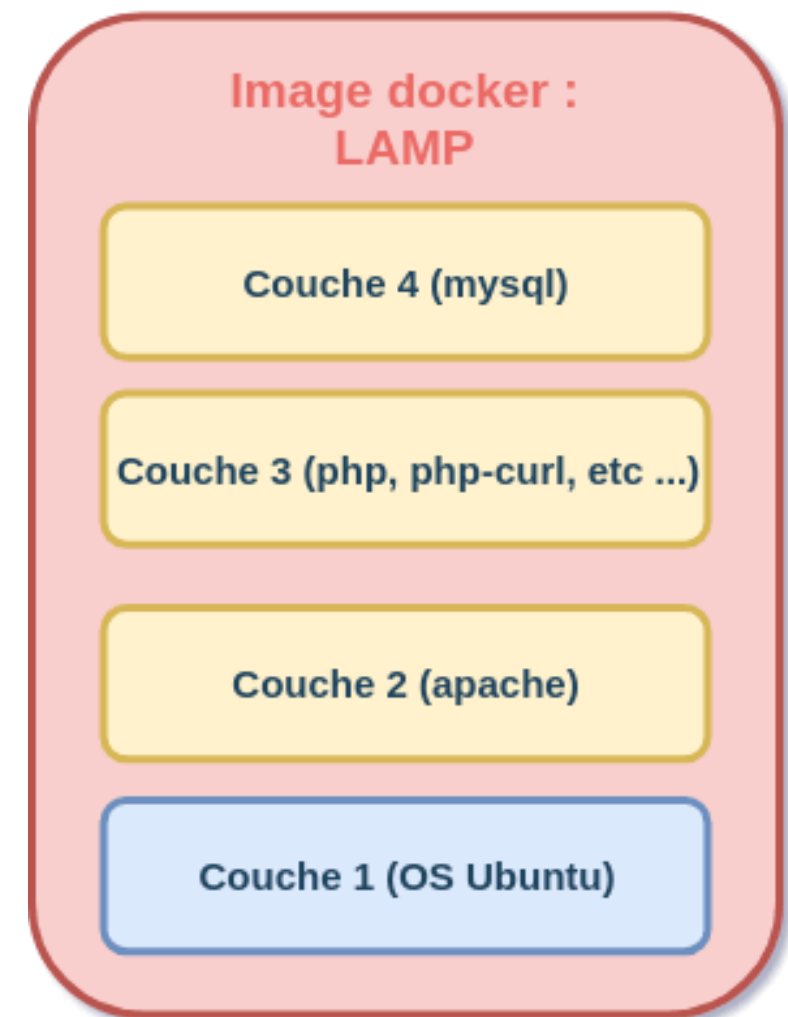
- Qu'est qu'une image Docker ?
  - Sur Docker, un conteneur est lancé en exécutant une image.
- Une image est un package qui inclut tout ce qui est nécessaire à l'exécution d'une application
  - Le code
  - L'exécution
  - Les variables d'environnement
  - Les bibliothèques
  - Les fichiers de configuration

# LES IMAGES DOCKER

- Une **image** est un modèle composé de plusieurs couches
  - Ces **couches** contiennent notre **application** ainsi que les **fichiers binaires** et les **bibliothèques requises**
- Lorsqu'une **image** est instanciée, son nom est un **conteneur**
  - Un **conteneur** est donc une **image** en cours d'exécution
- Pour mieux **comprendre** le système de **couche**, imaginons par exemple qu'on souhaite **déployer** notre **application web** dans un **serveur LAMP** (*Linux Apache MySQL PHP*)
  - Cette **image** sera **composé** de **4 couches**
    - ✓ Ces couches sont des layers (calques) en lecture seule

# LES IMAGES DOCKER

- Exemple de **serveur LAMP** en détail
  - Une couche **OS**
    - ✓ Pour exécuter Apache, MySQL...
  - Une couche **Apache**
    - ✓ Pour exécuter le serveur Web
  - Une couche **PHP**
    - ✓ Interpréteur et Library PHP
  - Une couche **MySQL**
    - ✓ Contiendra notre SGBD Mysql



# LES IMAGES DOCKER

- Premières commandes Docker
  - Pour commencer on va d'abord récupérer la liste des commandes possibles
    - ✓ \$ docker help
- Sur l'**output** de la commande **help**, nous avons une information d'une grande utilité et vous permettra de gagner beaucoup de temps
  - Run 'docker COMMAND --help' *for more information on a command*. Exemple la commande docker volume

✓ \$ docker volume

```
Usage:  docker volume COMMAND

Manage volumes

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

# LES IMAGES DOCKER

- Commandes Docker pour de l'information

- Petit rappel de la commande exécutée précédemment pour vérifier le fonctionnement

- ✓ \$ docker --version

```
Docker version 20.10.5, build 55c4c88
```

- La commande info permet d'afficher encore plus de détails sur votre installation de Docker

- ✓ \$ docker info

```
Client:
Context:    default
Debug Mode: false
Plugins:
  app: Docker App (Docker Inc., v0.9.1-beta3)
  buildx: Build with BuildKit (Docker Inc., v0.5.1-docker)
  scan: Docker Scan (Docker Inc., v0.6.0)

Server:
Containers: 9
  Running: 1
  Paused: 0
  Stopped: 8
Images: 13
Server Version: 20.10.5
Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Cgroup Version: 1
Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentr
ies splunk syslog
Swarm: inactive
Runtimes: runc io.containerd.runc.v2 io.containerd.runtime.v1.linux
```



# LES IMAGES DOCKER

- Commandes **Docker** pour la gestion des images
  - Pour lister l'ensemble des images présentes sur votre repos local

✓ \$ docker image ls

| REPOSITORY  | TAG    | IMAGE ID     | CREATED    | SIZE   |
|-------------|--------|--------------|------------|--------|
| myapp       | latest | c13d22c29096 | 3 days ago | 415MB  |
| my_lamp     | latest | 8b2d8ac6b226 | 3 days ago | 552MB  |
| volume_test | latest | 6df9cc60aaf6 | 3 days ago | 72.7MB |

✓ \$ docker images

- Elle nous donne différentes informations

| REPOSITORY  | TAG   | IMAGE ID  | CREATED                                     | SIZE              |
|---|---|---|---|-------------------|
| Le titre REPOSITORY peut porter à confusion, c'est essentiellement le nom de l'image. | un tag ici est une façon de faire référence à votre image, ils sont utilisés principalement pour affecter une version à une image | L'identifiant de l'image (unique pour chaque image téléchargée) | Date de la dernière modification de l'image | Taille de l'image |

# LES IMAGES DOCKER

- Commandes **Docker** pour la gestion des images
  - Supprimer une image (par nom ou id)
    - ✓ `$ docker rmi <nom_image ou id_image>`
  - Avec l'option **-f** pour **forcer** la suppression
    - ✓ `$ docker rmi -f <nom_image ou id_image>`
  - Supprimer toutes les images
    - ✓ `$ docker rmi -f $(docker images -q)`

# LES IMAGES DOCKER

- Commandes **Docker** pour la gestion des images
  - Rechercher une image sur le hub registry
    - ✓ `$ docker search <nom_image>`
  - Avec l'option **--filter** pour **trier** les images officielles
    - ✓ `$ docker search <nom_image> --filter "is-official=true"`

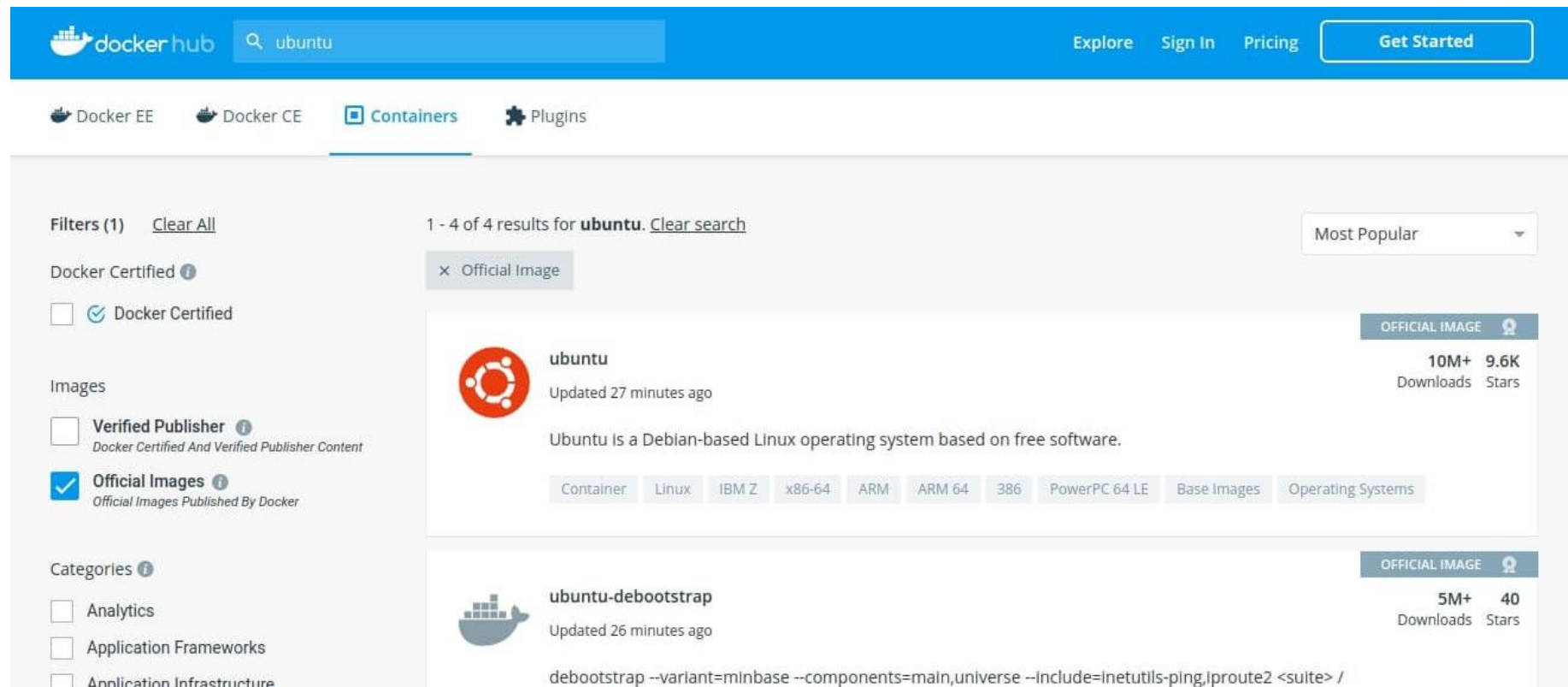
| NAME               | DESCRIPTION                                     | STARS | OFFICIAL | AUTOMATED |
|--------------------|---|-------|----------|-----------|
| ubuntu             | Ubuntu is a Debian-based Linux operating sys... | 12334 | [OK]     |           |
| websphere-liberty  | WebSphere Liberty multi-architecture images ... | 273   | [OK]     |           |
| ubuntu-upstart     | Upstart is an event-based replacement for th... | 110   | [OK]     |           |
| open-liberty       | Open Liberty multi-architecture images based... | 46    | [OK]     |           |
| ubuntu-debootstrap | debootstrap --variant=minbase --components=m... | 44    | [OK]     |           |

# LES IMAGES DOCKER

- Commandes **Docker** pour la gestion des images
  - Pour télécharger une image à partir du hub
    - ✓ `$ docker pull <nom_image>`
  - Télécharger une version précise (tag)
    - ✓ `$ docker pull ubuntu:16.04`
  - Télécharger la dernière version
    - ✓ `$ docker pull ubuntu:latest`

# LES IMAGES DOCKER

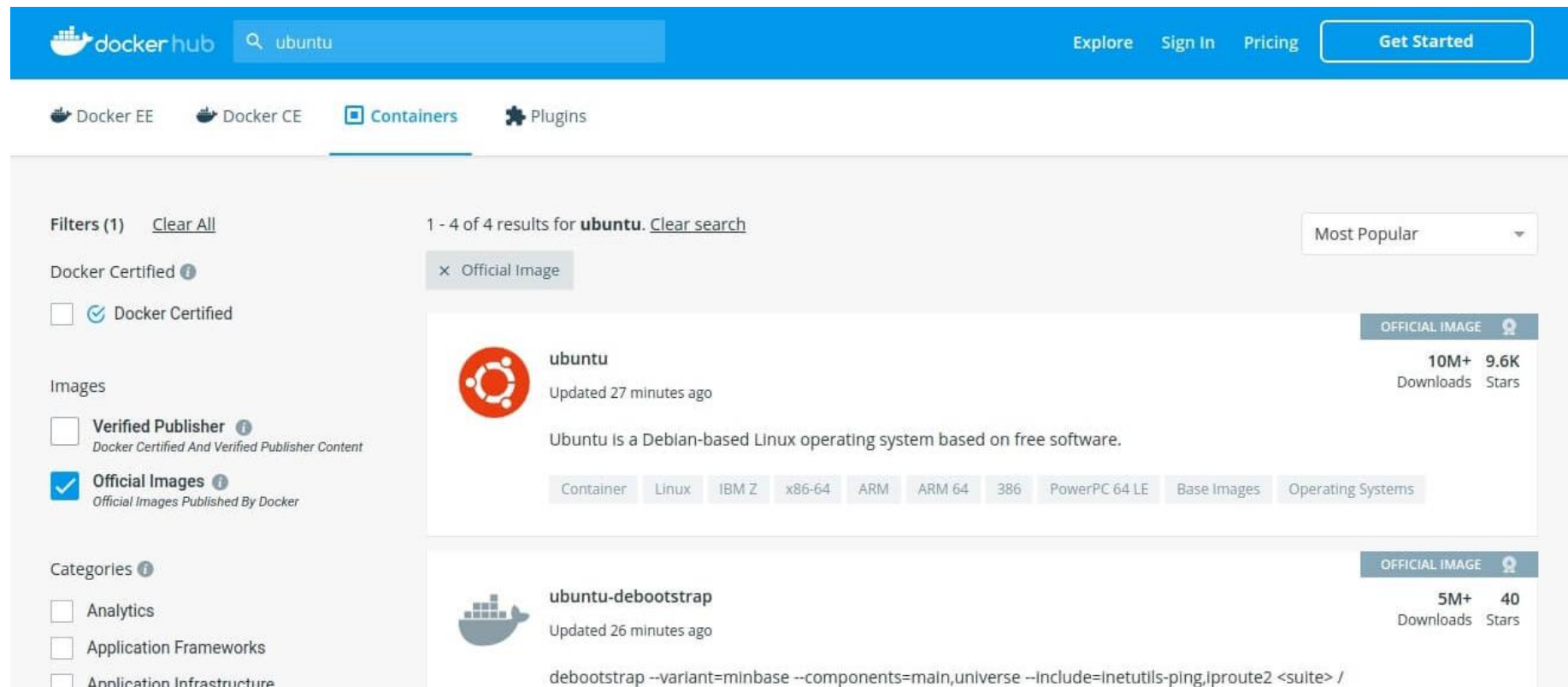
- Le Hub Registry Docker
  - Où est-ce que je peux retrouver la liste des images disponibles ?
- ✓ Le hub Registry





# LES IMAGES DOCKER

- Le Hub Registry Docker
  - Où est-ce que je peux retrouver la liste des images disponibles ?
- ✓ Le hub Registry



# LES IMAGES DOCKER

- Le Hub Registry Docker

The screenshot shows the Docker Hub interface for the 'ubuntu' image. The top navigation bar includes the Docker Hub logo, a search bar with 'ubuntu' entered, and links for 'Explore', 'Sign In', 'Pricing', and 'Get Started'. The main content area features the Ubuntu logo, the text 'ubuntu ☆ Docker Official Images', and a description: 'Ubuntu is a Debian-based Linux operating system based on free software.' Below this, there are tags for 'Container', 'Linux', '386', 'PowerPC 64 LE', 'IBM Z', 'x86-64', 'ARM', 'ARM 64', 'Base Images', and 'Operating Systems'. A red box highlights the 'docker pull ubuntu' command, which is displayed in a dark blue box with a copy icon. To the right of the command, there is a dropdown menu showing 'Linux - x86-64 (latest)' and a link to 'View Available Tags'. Below the command, there is a section titled 'Supported tags and respective Dockerfile links' with a list of tags and their corresponding Dockerfile links. The list includes: 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); and 16.04, xenial-20190515, xenial (xenial/Dockerfile). Below this list is a 'Quick reference' section with links to 'Where to get help', 'Where to file issues', 'Maintained by', and 'Supported architectures'.

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](#)

**Supported tags and respective Dockerfile links**

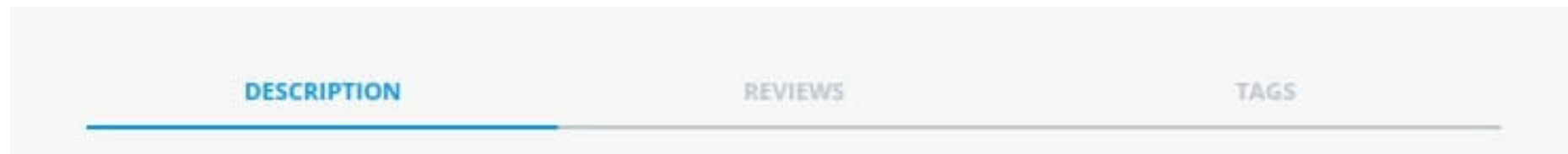
- 18.04, bionic-20190515, bionic, latest ([bionic/Dockerfile](#))
- 18.10, cosmic-20190515, cosmic ([cosmic/Dockerfile](#))
- 19.04, disco-20190515, disco, rolling ([disco/Dockerfile](#))
- 19.10, eoan-20190508, eoan, devel ([eoan/Dockerfile](#))
- 14.04, trusty-20190515, trusty ([trusty/Dockerfile](#))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

**Quick reference**

- Where to get help:**  
the Docker Community Forums, the Docker Community Slack, or Stack Overflow
- Where to file issues:**  
the cloud-images bug tracker (include the `docker` tag)
- Maintained by:**  
Canonical and Tianon (Debian Developer)
- Supported architectures:** (more info)  
`amd64`, `arm32v7`, `arm64v8`, `i386`, `ppc64le`, `s390x`

# LES IMAGES DOCKER

- Le Hub Registry Docker



- **DESCRIPTION** : 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.
- **REVIEWS** : l'avis des utilisateurs
- **TAGS** : les différents tags disponible pour cette image

# LES CONTENEURS DOCKER

## Différence entre image et conteneur dans Docker

- Rappel lorsque vous utilisez des **fonctionnalités** permettant une **isolation** du **processus**
  - namespaces
  - cgroups
    - ✓ On appelle cela des **conteneurs**
- Un **conteneur** est une **instance d'exécution** d'une **image**
  - Plus précisément un **conteneur** est ce que l'**image** **devient** en **mémoire** lorsqu'elle est **exécutée**
    - ✓ Avec un état, un processus utilisateur...

# LES CONTENEURS DOCKER

## Créer un conteneur

- Créer une instance de notre image avec **docker run**
  - `$ docker run [OPTIONS] <Image_Name ou ID>`
    - ✓ Exemple: `$ docker run debian:latest`
- Cette commande peut être complétées par des options
  - `$ docker run -tid ubuntu:latest`
  - ✓ `$ docker run --help // Documentation`



# LES CONTENEURS DOCKER

## Options de la commande **run**

- **-t** : Allouer un terminal tty (terminal virtuel)
- **-i** : Garder un STDIN ouvert (l'entrée standard, plus précisément l'entrée clavier)
- **-d** : Exécuter le conteneur en arrière-plan
- **-p** : Exposer un ou plusieurs ports (mapping)
- **--name** : donner un nom au container
- **--expose** : Exposer un port ou une plage de ports
  - on demande au firewall du conteneur de nous ouvrir un port ou une plage de port
- **-p** ou **-publish** : Mapper un port déjà exposé, plus simplement ça permet de faire une redirection de port

# LES CONTENEURS DOCKER

## Commandes pour administrer un conteneur

- Pour inspecter un conteneur
  - `$ docker inspect <nom_conteneur>`
- Pour stopper un conteneur Actif
  - `$ docker stop <nom_conteneur>`
- Pour supprimer un conteneur
  - `$ docker rm <nom_conteneur>`

# LES CONTENEURS DOCKER

## Commandes pour administrer un conteneur

- Pour **exécuter** une **commande** dans un **conteneur**
  - `$ docker exec [OPTIONS] <Id ou name> command`
- Docker **exec** peut être également complétée par des options
  - `$ docker exec -tid <nom_conteneur> bash`
- Lister les **conteneurs actifs**
  - `$ docker container ls`    *ou*    `$ docker ps`
- Lister tous les **conteneurs**
  - `$ docker container ls -a`    *ou*    `$ docker ps -a`