

# INTRODUCTION À DOCKER

---

4eme année IIR

# BENTALBA

## Salah eddine



Consultant expert  
transformation digitale

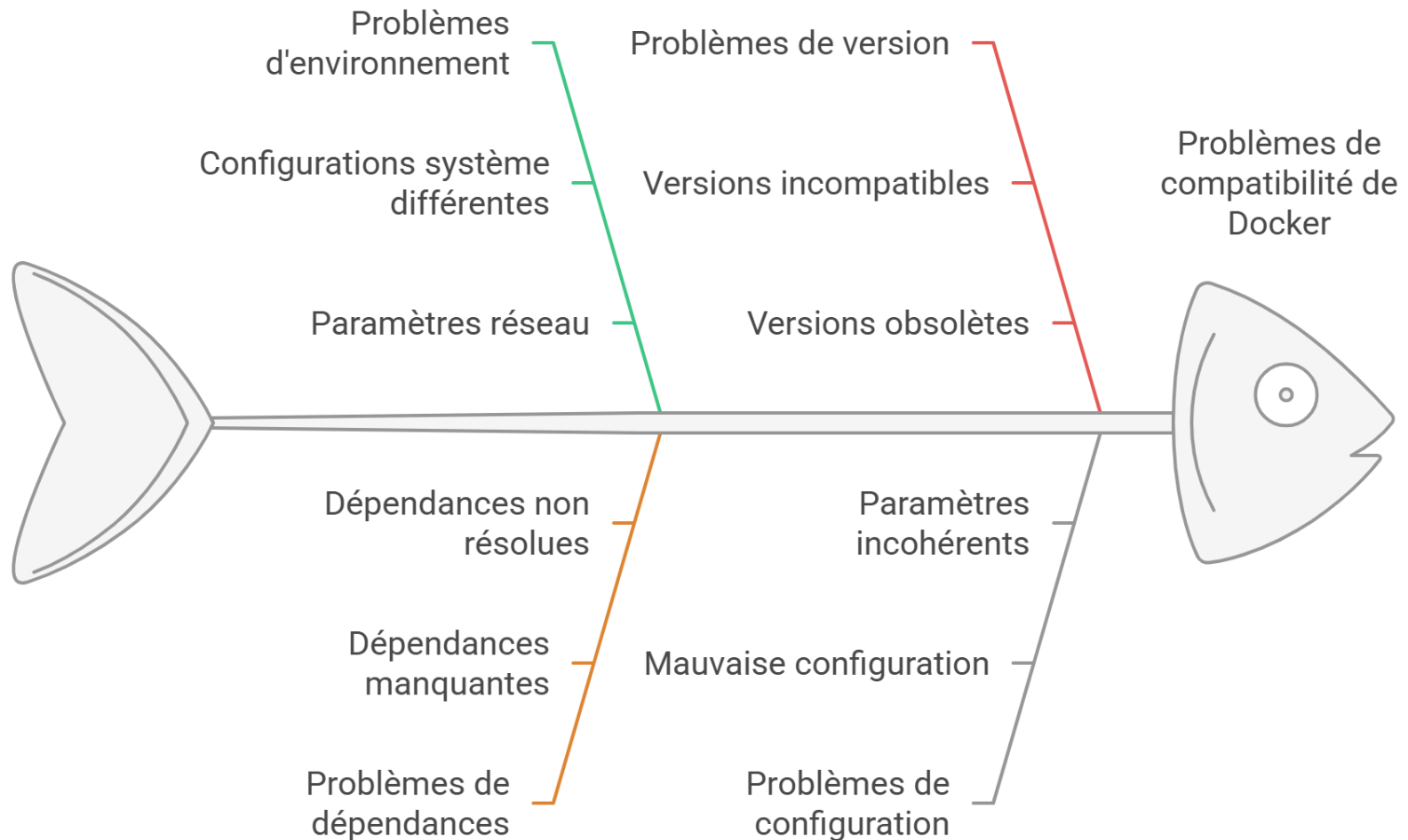
+212 601 100 600

Salaheddine.bentalba@gmail.com

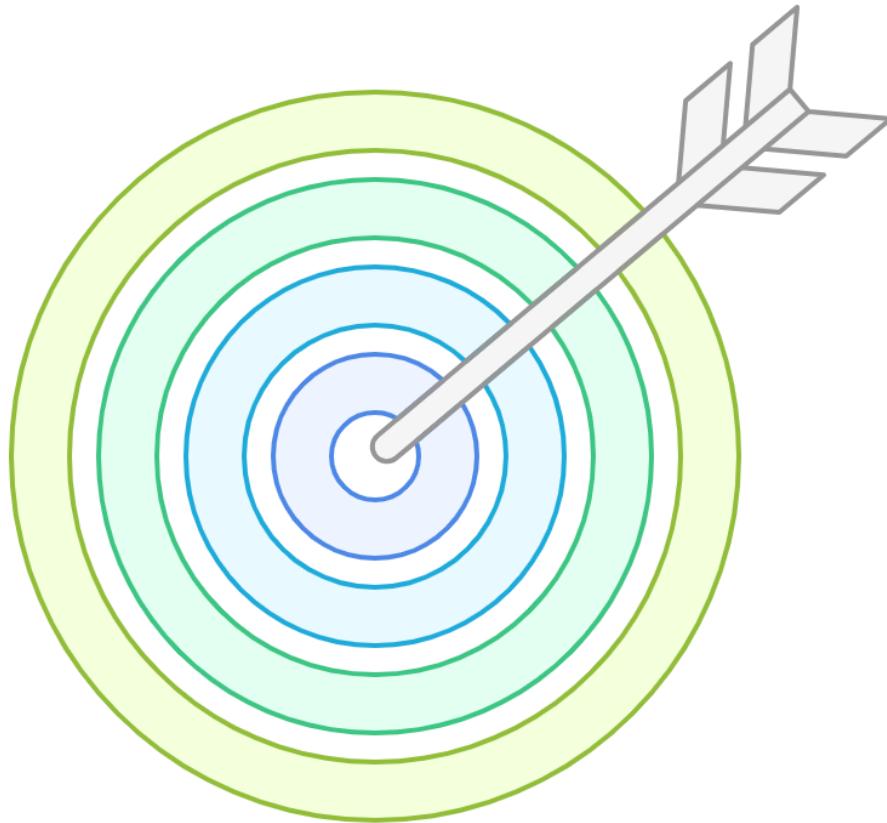
- Lauréat EMSI Casablanca 2012
- Co-Fondateur de **Mobiconnect Technologies**
- **13+ ans d'expérience** en gestion de projets IT et transformation digitale.
- Spécialiste des **services cloud**.
- Collaborations avec **ATOS**, **Capgemini**, **42Gears** et **SOTI**.
- Interventions lors de **conférences et formations** :
  - Coach & Expert dans **EMSI NextGen Hackathon** et d'autres.
  - Organisateur de plusieurs **TEDx** et conférencier.

# Problématiques et besoins

« Tous ces problèmes nous les avons rencontrés... jusqu'à ce qu'on découvre Docker. »



# Conteneurisation avec docker



## Conteneurisation

Le processus d'encapsulation des applications



## Docker

L'outil qui facilite la conteneurisation



## Images

Modèles pour créer des conteneurs



## Conteneurs

Instances en cours d'exécution d'images

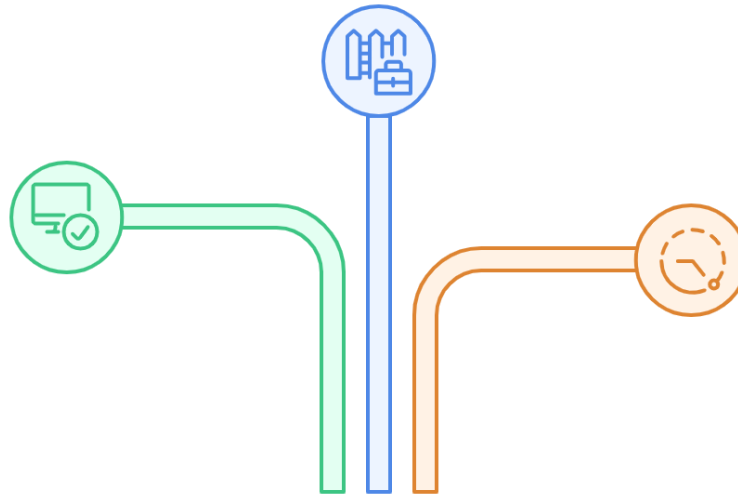
# Pourquoi docker ?

## Gérer les dépendances

Docker permet de regrouper et de gérer les dépendances, assurant que toutes les bibliothèques nécessaires sont disponibles.

## Résoudre les problèmes de compatibilité

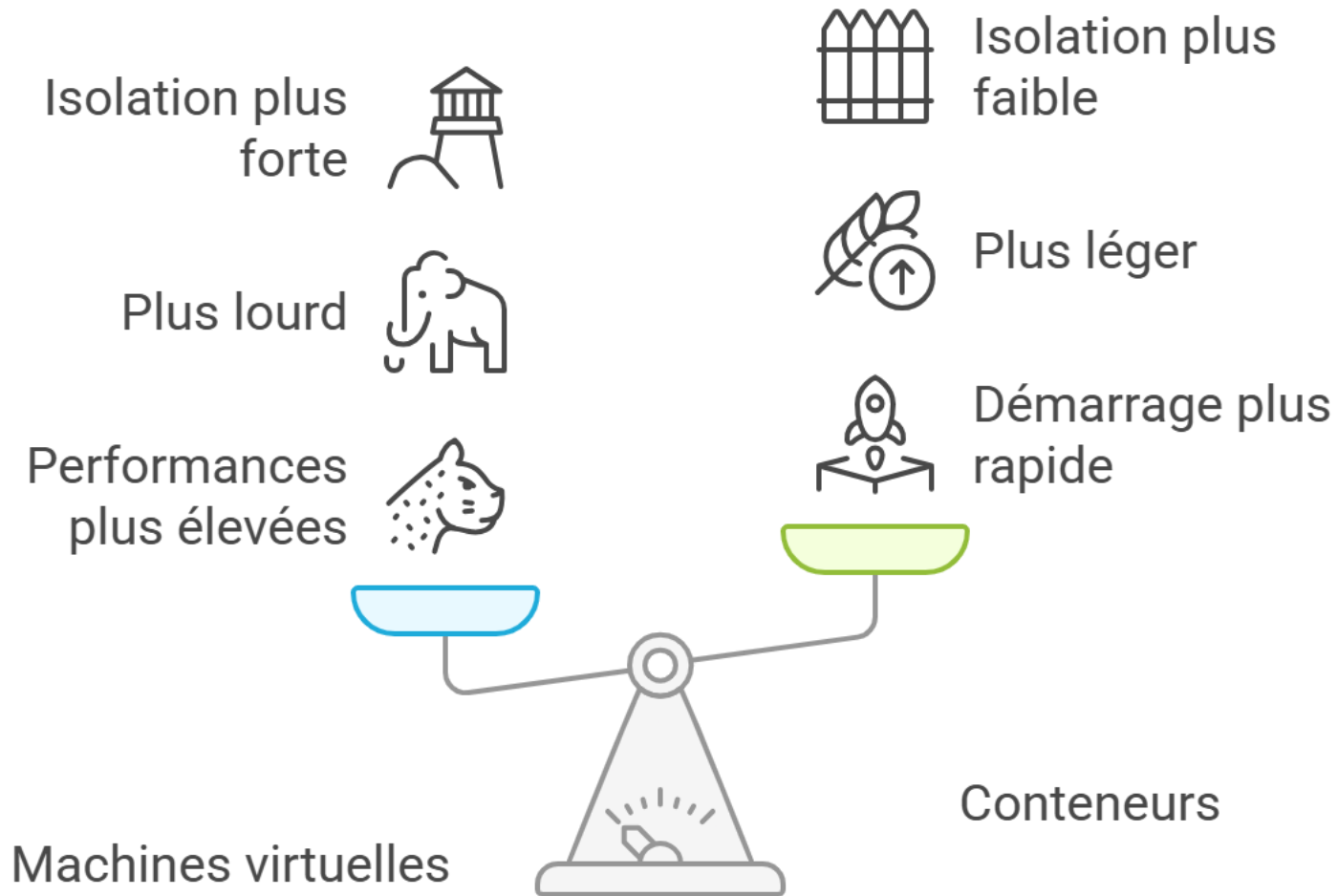
Docker garantit que les applications fonctionnent de manière cohérente dans différents environnements, évitant les problèmes de compatibilité.



## Éviter les conflits de version

Docker aide à maintenir des versions cohérentes des logiciels, empêchant les conflits entre différents environnements.

# Conteneurisation vs virtualisation



# Conteneur vs Image

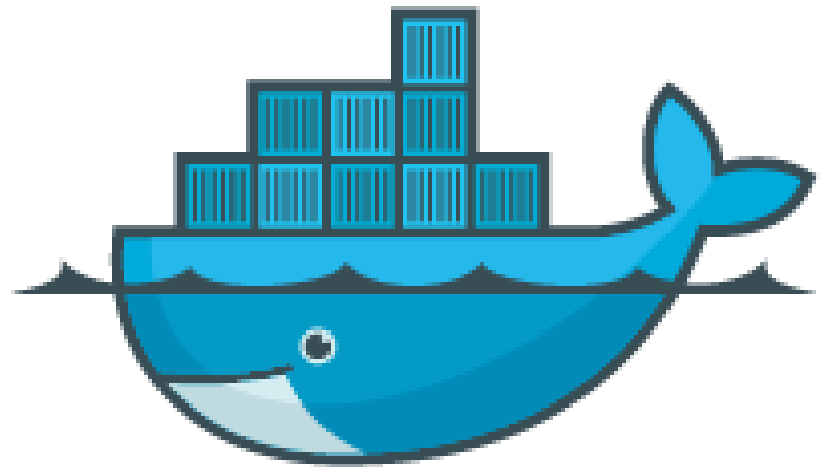


**vs**



# Image docker

Une image est une photo statique de l'état d'une application à un moment donné. Elle contient tout le nécessaire pour exécuter l'application : le code, les bibliothèques, les dépendances et la configuration. -->L'image est utilisée pour créer des conteneurs.

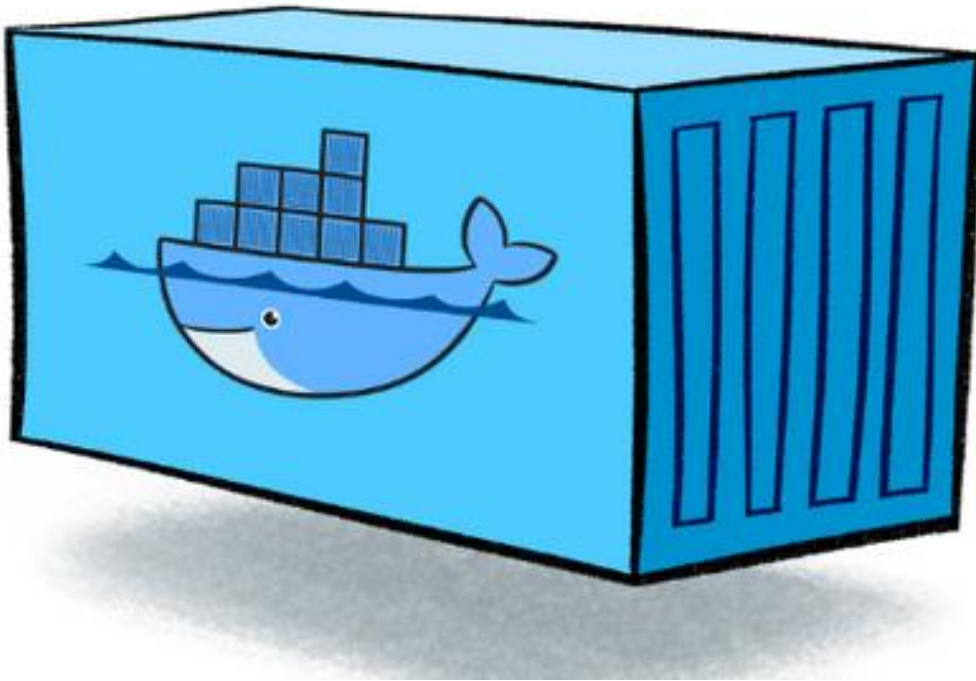


# docker



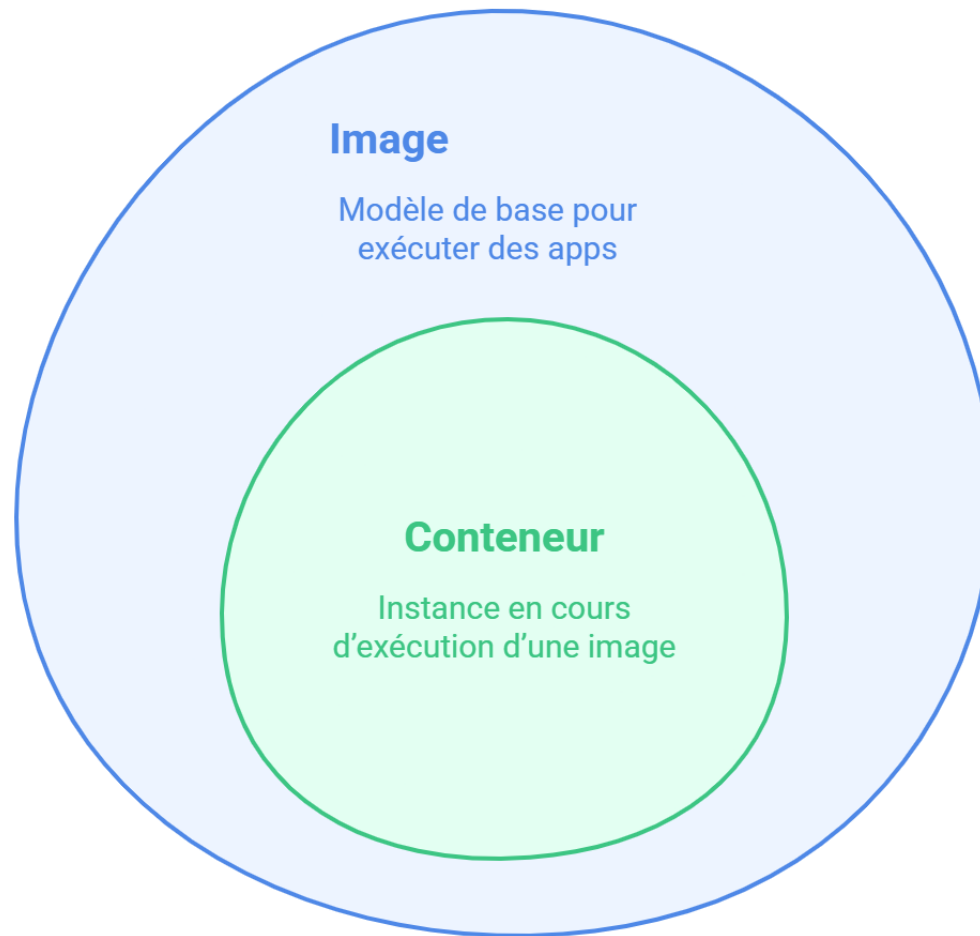
# Conteneur docker

Un conteneur Docker est comme une boîte virtuelle où ton application et toutes ses dépendances sont isolées des autres applications et de l'environnement global. Cela garantit que l'application fonctionne de manière identique, peu importe l'endroit où elle est exécutée.



Un conteneur est une instance en cours d'exécution d'une image Docker. Lorsque tu lances une image, un conteneur est créé et exécute l'application. --> Chaque conteneur est une application active fonctionnant dans un environnement isolé

# Conteneur vs Image



# Docker – Images : Commande de bases

Build an image from a dockerfile

```
> docker build -t image_name path_to_dockerfile

# EXAMPLE

> docker build -t myapp .
```

List all local images

```
> docker images

# EXAMPLE

> docker image ls
```

Inspect details of an image

```
> docker image inspect image_name:tag

# EXAMPLE

> docker image inspect myapp:v1
```

Remove a local image

```
> docker rmi image_name:tag

# EXAMPLE

> docker rmi myapp:latest
```

# Docker – Conteneur : Commande de bases

Run a container from an image

```
Terminal
> docker run container_name image_name

# EXAMPLE
> docker run myapp
```

Run a named container from an image

```
Terminal
> docker run --name container_name image_name:tag


# EXAMPLE
> docker run --name my_container myapp:v1
```

List all running containers

```
Terminal
> docker ps
```

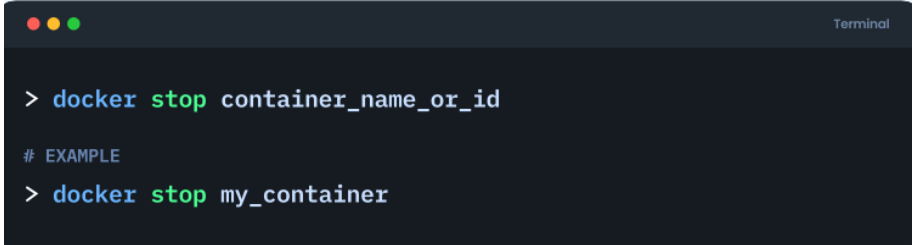
# Docker – Conteneur : Commande de bases

List all containers (including stopped ones)

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The word "Terminal" is in the top-right corner. The command `> docker ps -a` is entered in the terminal.

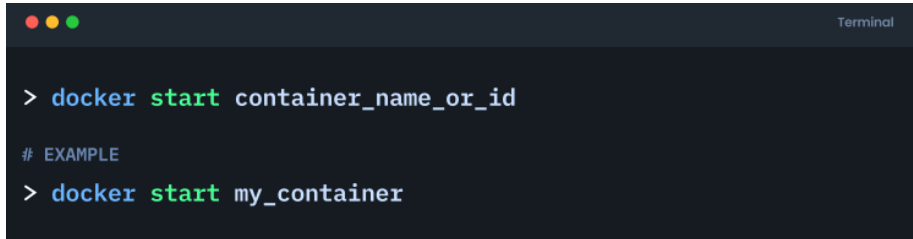
```
> docker ps -a
```

Stop a running container

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The word "Terminal" is in the top-right corner. The command `> docker stop container_name_or_id` is entered, followed by a comment `# EXAMPLE` and an example command `> docker stop my_container`.

```
> docker stop container_name_or_id  
  
# EXAMPLE  
> docker stop my_container
```

Start a stopped container

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The word "Terminal" is in the top-right corner. The command `> docker start container_name_or_id` is entered, followed by a comment `# EXAMPLE` and an example command `> docker start my_container`.

```
> docker start container_name_or_id  
  
# EXAMPLE  
> docker start my_container
```

# Docker – Conteneur : Commande de bases

Remove a stopped container

```
Terminal
> docker rm container_name_or_id

# EXAMPLE
> docker rm my_container
```

Inspect details of a container

```
Terminal
> docker inspect container_name_or_id

# EXAMPLE
> docker inspect my_container
```

View container logs

```
Terminal
> docker logs container_name_or_id

# EXAMPLE
> docker logs my_container
```

# Docker Hub : le magasin d'images Docker

Docker Hub est une plateforme de stockage et de partage d'images Docker, un registre cloud où les développeurs peuvent stocker et récupérer leurs images de conteneurs. C'est un service centralisé qui facilite la collaboration et la distribution des applications en conteneurs.



## Rechercher des images

Trouver des images Docker officielles comme MySQL.



## Télécharger des images


Télécharger des images existantes en utilisant la commande docker pull.



## Publier des images

Publier vos propres images en ligne en utilisant docker push.

# Docker Hub : le magasin d'images Docker

 **dockerhub**

[CtrlK](#)

[?](#)

[🔍](#)

[☰](#)

[Sign in](#)

[Sign up](#)

**Filter by**1 - 25 of 10,000 results for **wordpress**.

Best match

▼

**Products**

▼

☐ Images

☐ Extensions

☐ Plugins

**Trusted content**

▼

☐  Docker Official Image ⓘ

☐  Verified Publisher ⓘ

☐  Sponsored OSS ⓘ

**wordpress** 

↓1B+ · ☆5.8K

Pulls: 679,131  
Last week

Updated 9 days ago

The WordPress rich content management system can utilize plugins, widgets, and themes.

CONTENT MANAGEMENT SYSTEM

[Learn more](#) 

**bitnami/wordpress** 

↓100M+ · ☆267

Pulls: 91,021  
Last week

By [VMware](#) · Updated a day ago

Bitnami container image for WordPress

CONTENT MANAGEMENT SYSTEM

[Learn more](#) 



# Docker Hub : le magasin d'images Docker

Push an image to Docker Hub

```
Terminal

> docker push image_name:tag

# EXAMPLE

> docker push myapp:v1
```

Pull an image from Docker Hub

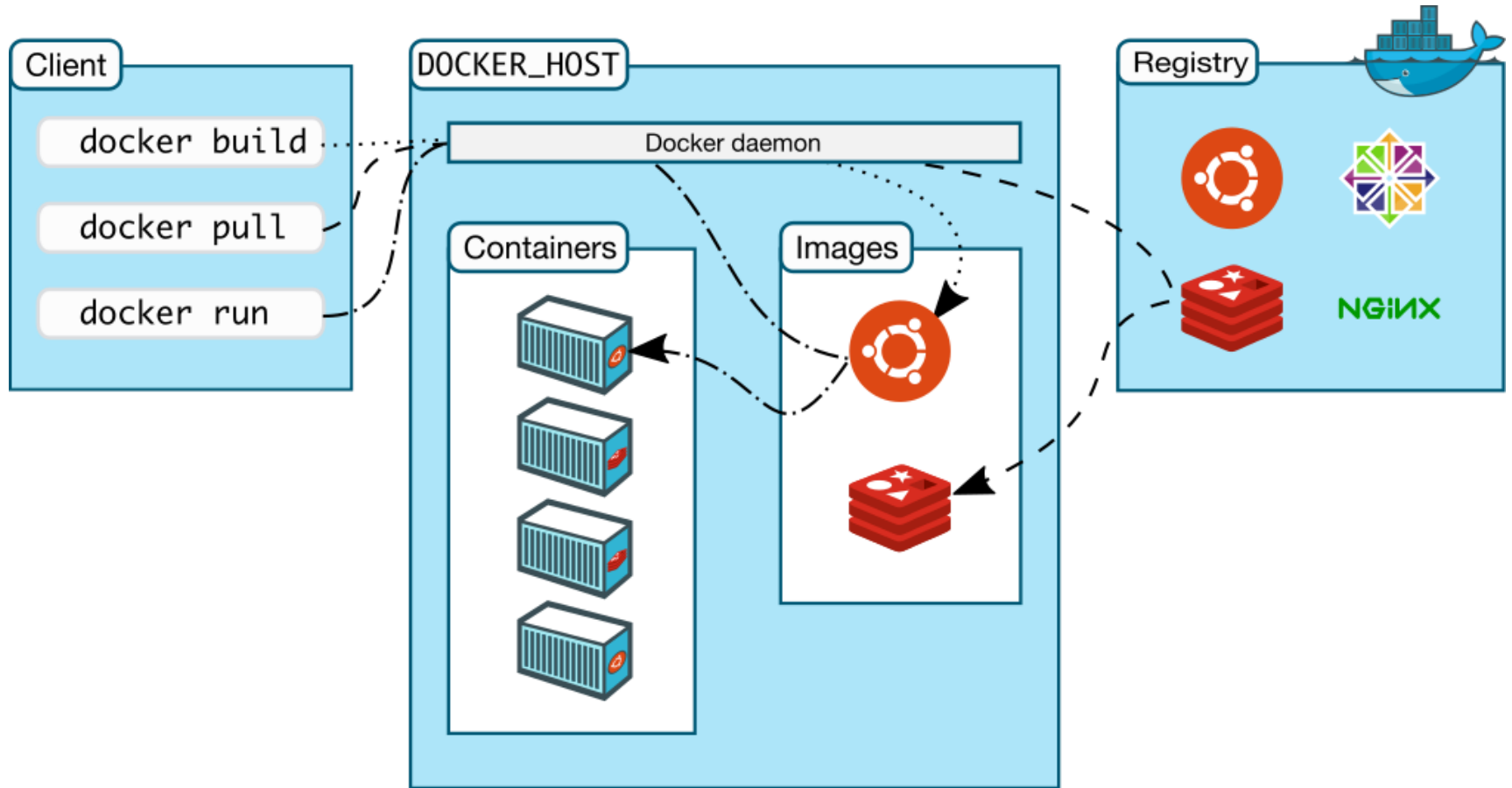
```
Terminal

> docker pull image_name:tag

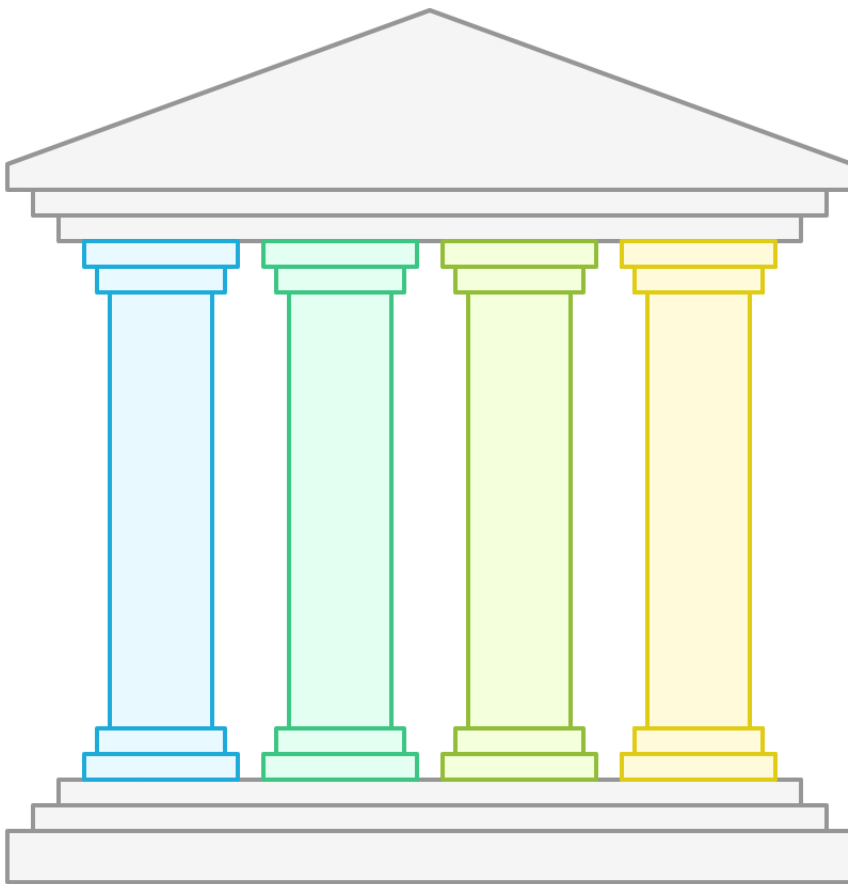
# EXAMPLE

> docker pull nginx:latest
```

# Architecture docker



# Composants Docker



## Docker Engine

Le cœur de Docker, gérant les conteneurs et les images.



## Docker Daemon

Un processus en arrière-plan qui écoute les requêtes de l'API Docker.



## Docker CLI

L'interface de ligne de commande pour interagir avec Docker.

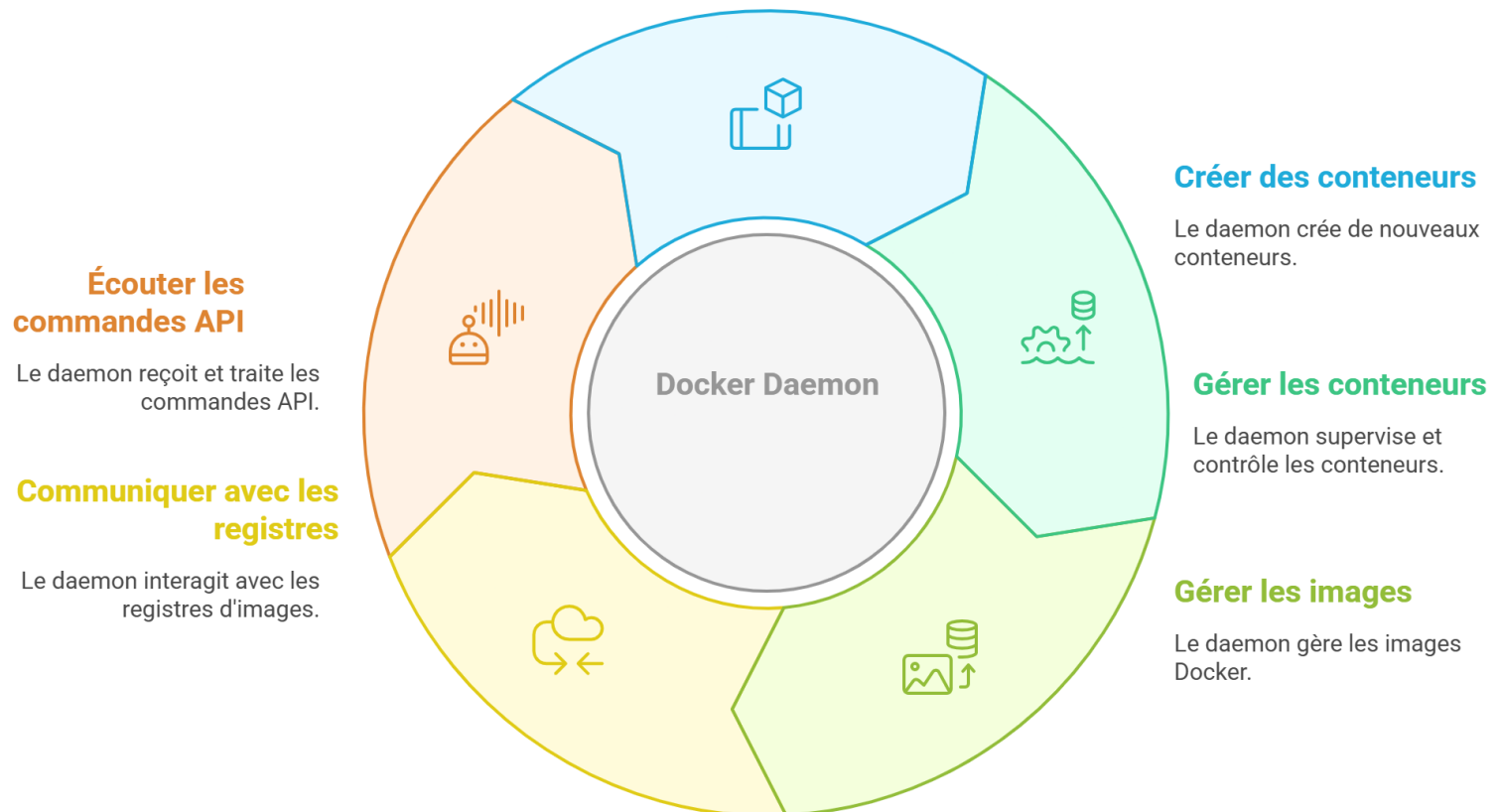


## Images & Conteneurs

Les modèles et les instances de conteneurs Docker.

# Docker Daemon

Docker Daemon (dockerd) est le serveur qui gère les conteneurs, les images, et les volumes. Il reçoit les commandes du client et les exécute. Il peut également communiquer avec d'autres démons Docker sur différentes machines pour gérer un cluster Docker. Le démon est responsable de l'exécution réelle des applications dans les conteneurs.

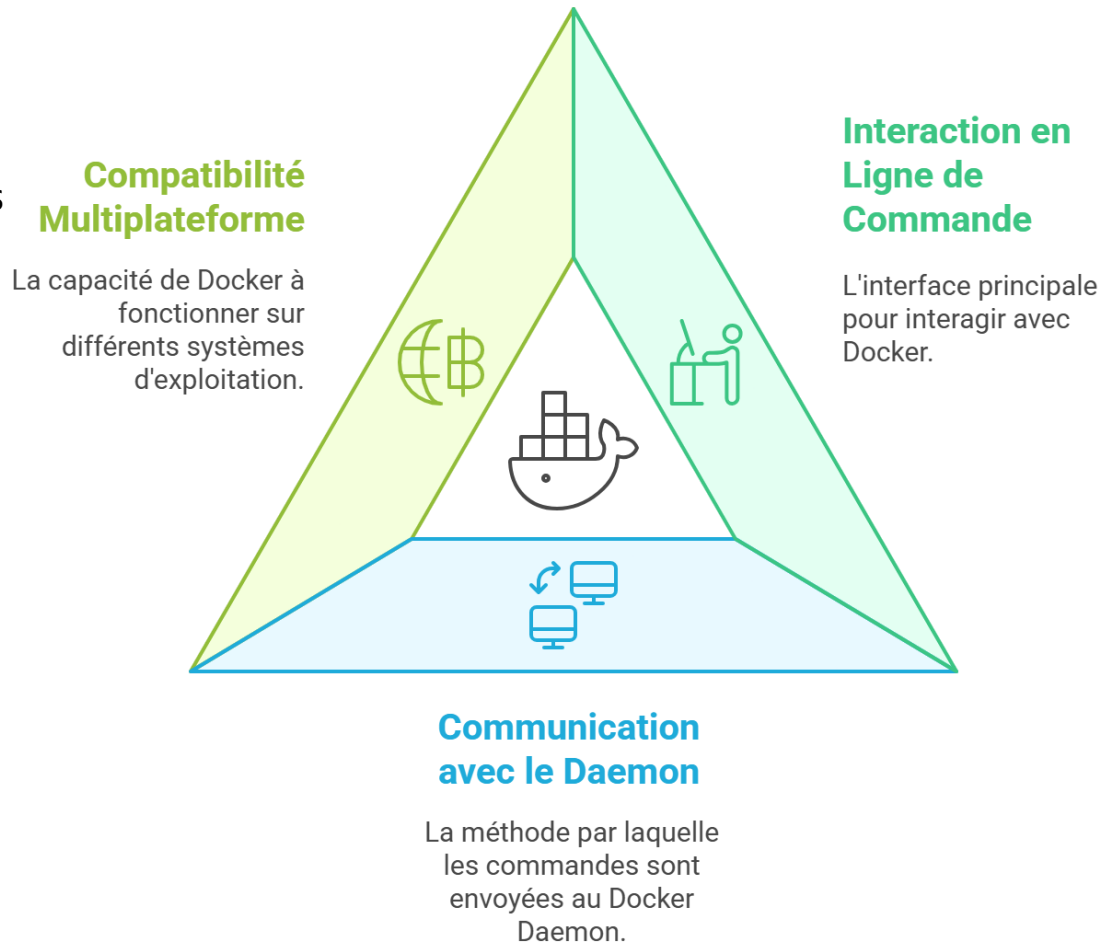


# Docker CLI

Docker CLI c'est l'outil en ligne de commande qui permet de communiquer avec le moteur Docker (Docker Engine).

Il te permet de :

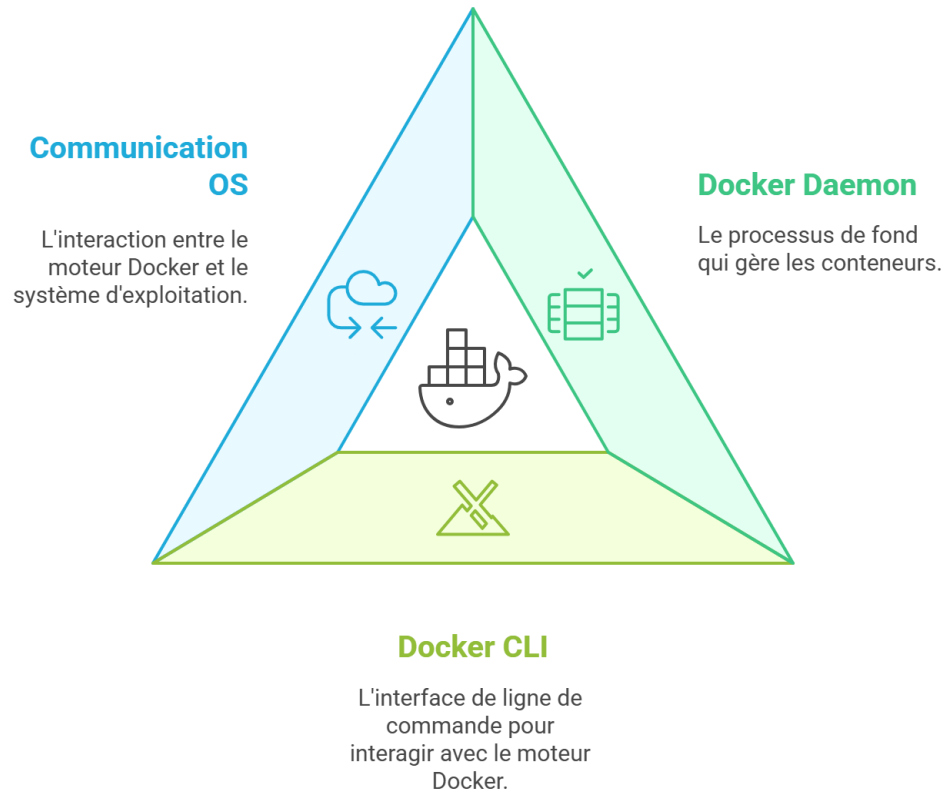
- Créer et lancer des conteneurs
- Gérer des images
- Supprimer, stopper ou visualiser les conteneurs
- Automatiser le déploiement d'applications



# Docker Engine

**Docker Engine** est le cœur de la plateforme Docker : c'est le logiciel qui permet de créer, exécuter et gérer les conteneurs.

Il fonctionne en arrière-plan sur votre machine et exécute toutes les commandes Docker que vous tapez via le terminal.



# Cycle de vie d'un conteneur

## Création d'image

Construire une image de conteneur à partir d'un Dockerfile



## Création de conteneur

Créer un conteneur à partir de l'image



## Exécution de conteneur

Exécuter le conteneur pour effectuer des tâches



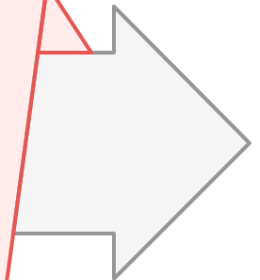
## Arrêt de conteneur

Arrêter le conteneur en cours d'exécution

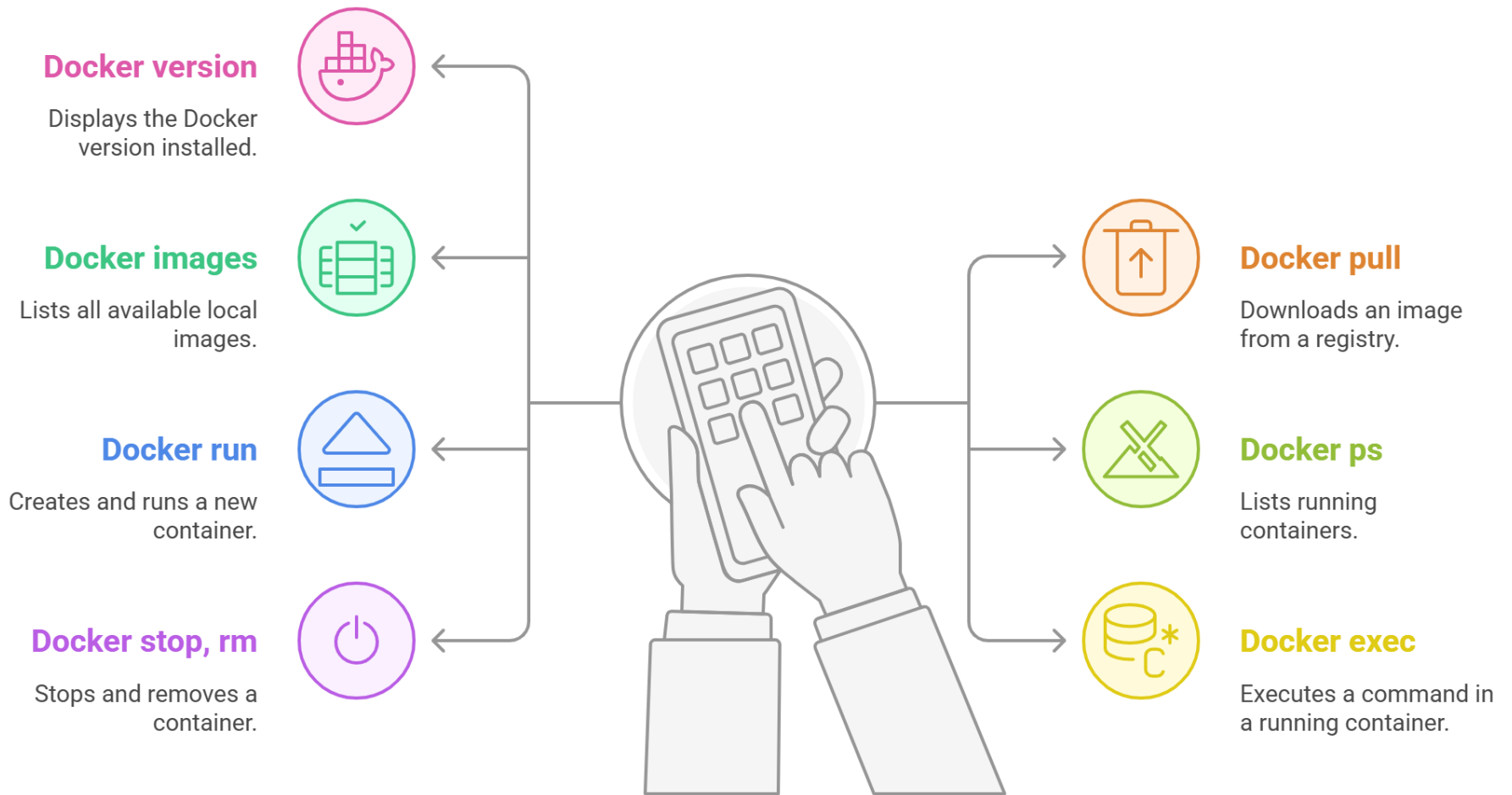


## Suppression de conteneur

Supprimer le conteneur du système

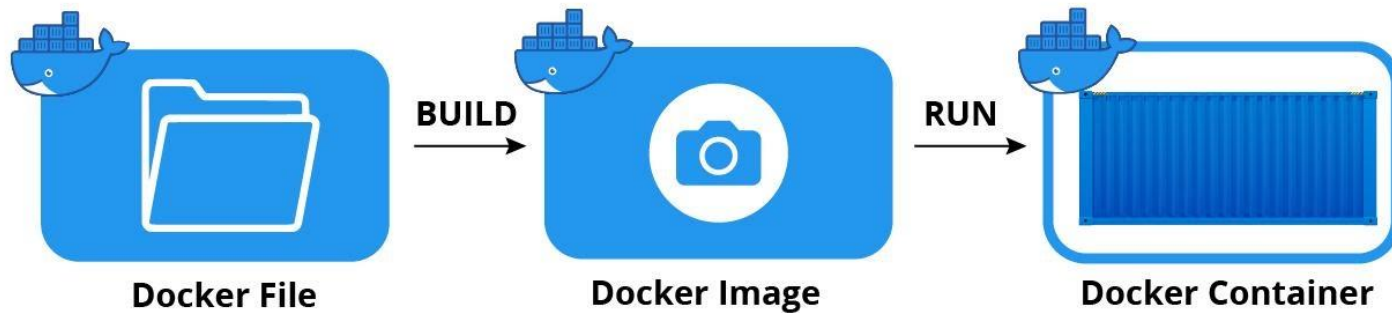


# Commandes de bases Docker





# De la recette au plat : Dockerfile → Image → Conteneur



# Docker file



## Définition

Un fichier texte avec des instructions pour créer des images Docker.



## Automatisation

Automatise la construction d'environnements fiables et reproductibles.



## Exemple

Installe Python 3.9 sur Ubuntu 20.04 de manière automatisée.

```
> docker build -t image_name path_to_dockerfile
```

```
# EXAMPLE
```

```
> docker build -t myapp .
```

Terminal

# Docker file : exemple de création d'image

**FROM python:3.9-slim-buster**

Sélection de l'image de base

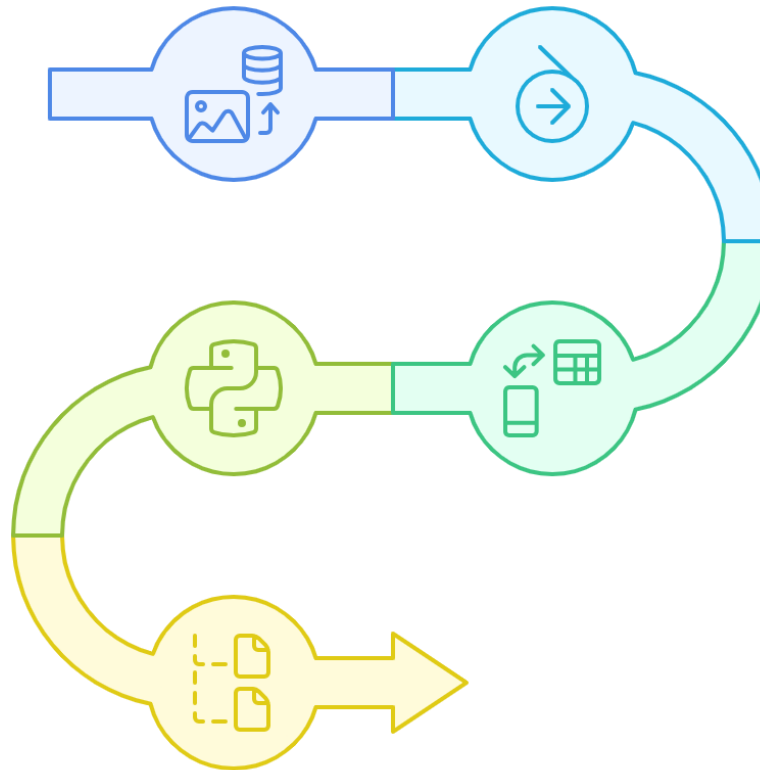
**RUN pip install**

Installation des dépendances

**COPY ..**

Copie des fichiers du projet

```
FROM ubuntu:20.04
RUN apt update && apt install -y python3.9
```



**WORKDIR /app**

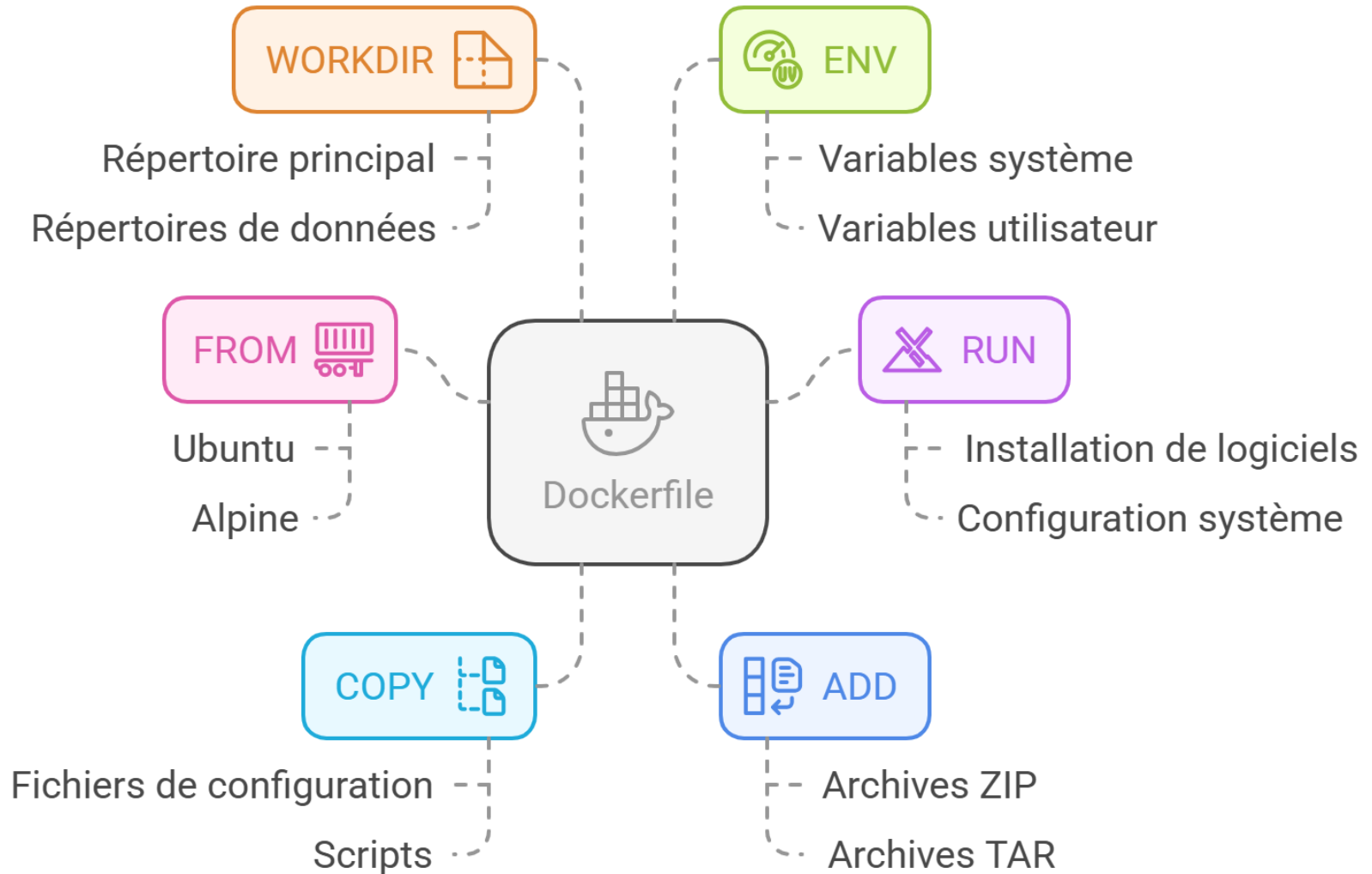
Définir le répertoire de travail

**COPY requirements.txt .**

Copie des dépendances

```
`FROM python:3.9-slim-buster`
`WORKDIR /app`
`COPY requirements.txt .` et `RUN pip install`
`COPY ..` puis `CMD ["python", "app.py"]`
```

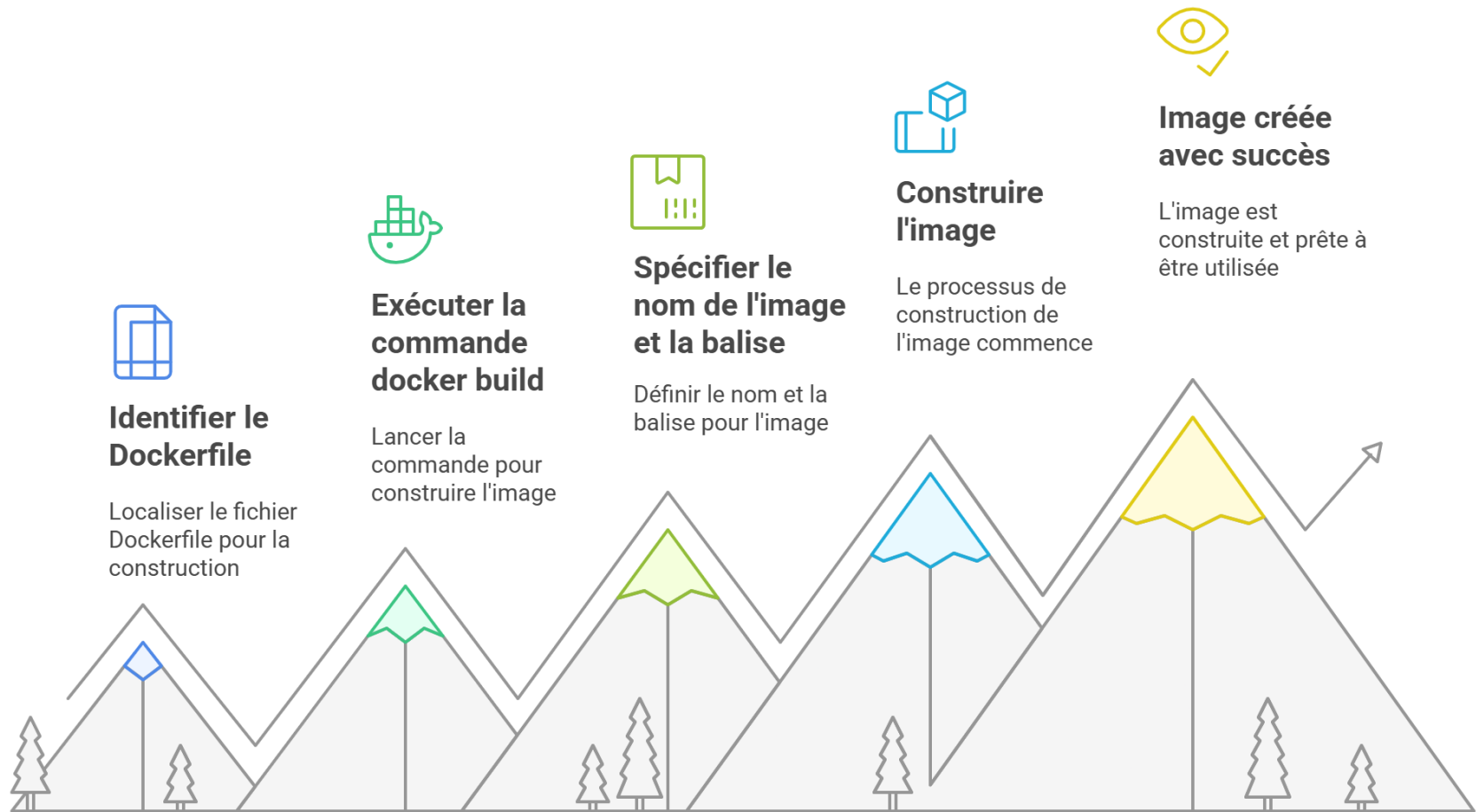
# Docker file : commande de base



*Les commandes de Dockerfile ne sont pas sensibles à la casse. On les note en majuscule par convention (facilite la lecture)*

# Docker build : créateur d'images

Exemple : `docker build . -t mon-image:tag` La commande construit une image Docker à partir du Dockerfile situé dans le dossier actuel et lui donne le nom mon-image avec l'étiquette tag.



# Docker compose : simplifier le multi-conteneur

Lorsque ton application a plusieurs services (ex. backend, base de données, frontend), il devient :

- Fatigant de lancer chaque conteneur à la main
- Difficile de gérer les configurations et la communication entre services



## Orchestration simple

Automatise le déploiement et la gestion de plusieurs conteneurs Docker.



## Fichier YAML

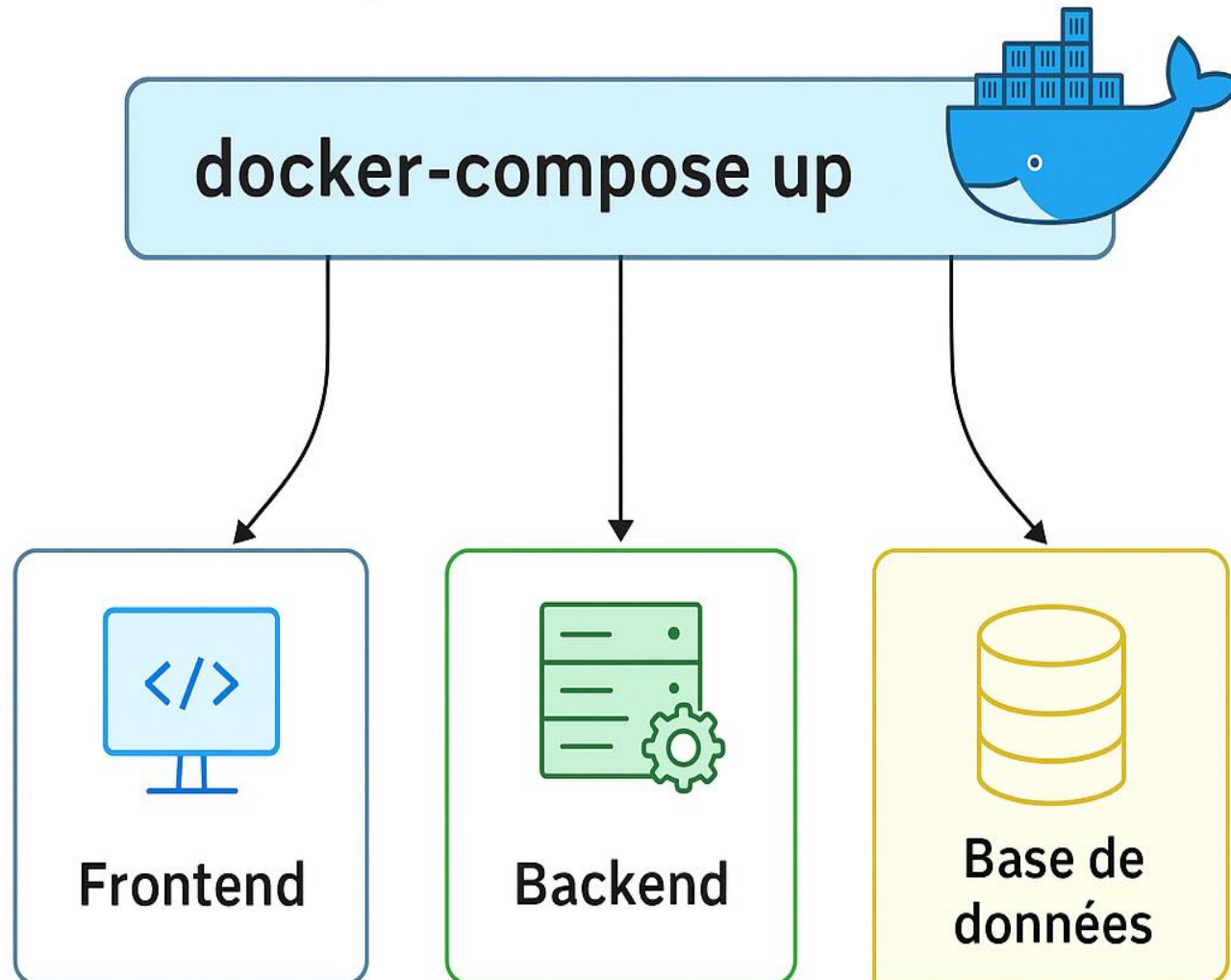
Définit des services, des réseaux et des volumes dans un format lisible.



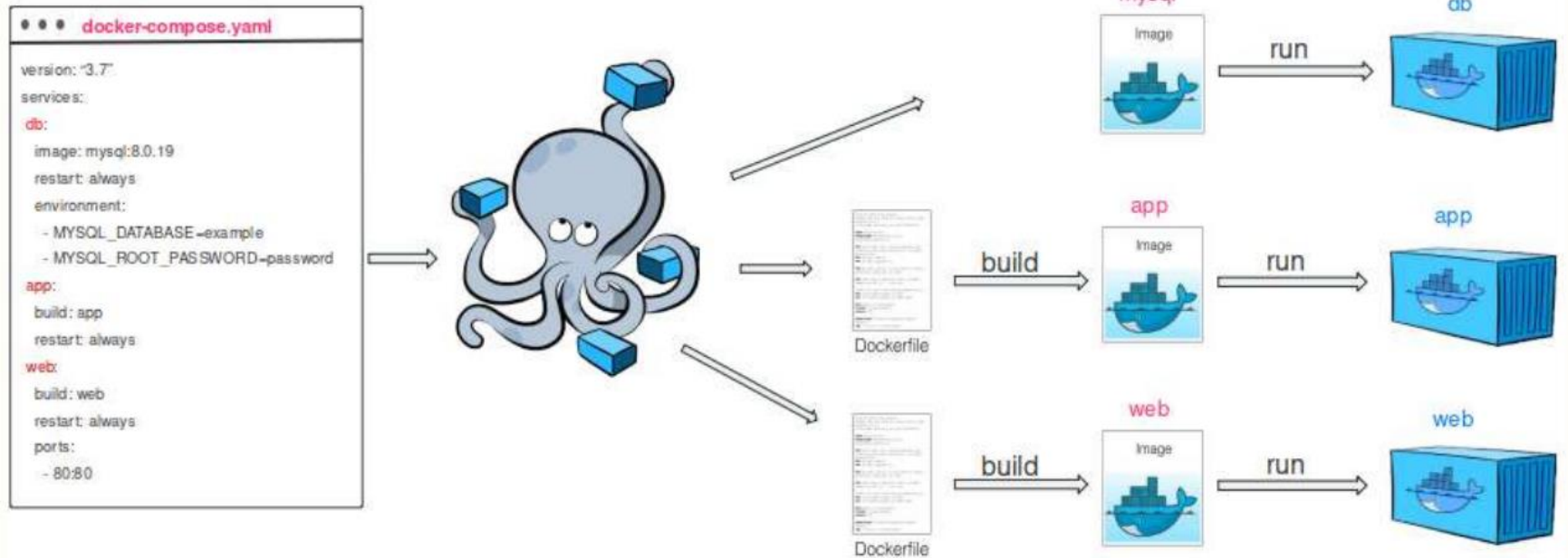
## Administration simplifiée

Réduit la complexité pour les développeurs et les opérateurs utilisant Docker Compose.

# Gestion d'une application multi-conteneur avec Docker Compose

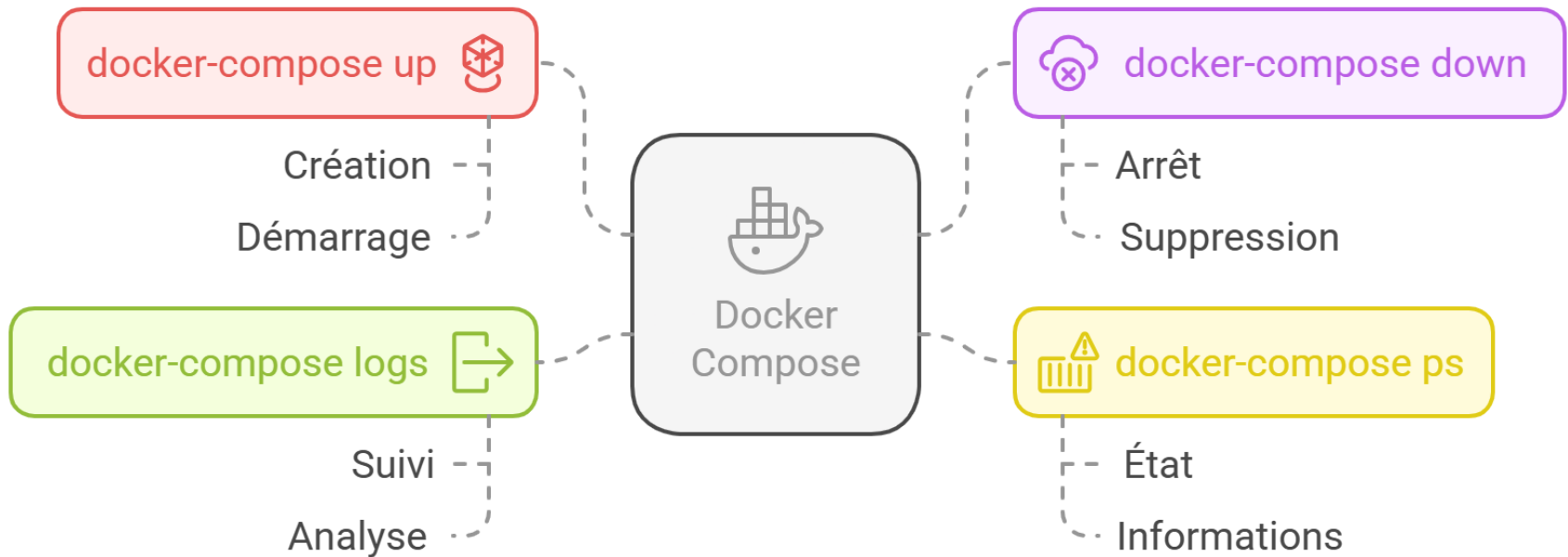


# Docker compose : simplifier le multi-conteneur





# Docker compose : Commandes de base



# Gestion des Dépendances et de l'Ordre de Démarrage

**depends\_on**  
Définit les dépendances entre les services.

**Cohérence Assurée**  
Garantit la bonne initialisation de l'application.



## Démarrage Séquentiel

Docker Compose démarre les dépendances en premier.

## Exemple Pratique

`depends_on: [database]` pour MySQL.

## Exemple d'un fichier YML

```
services:
  web:
    build: .
    ports:
      - "8080:80"
    depends_on:
      - db
  db:
    image: mysql:8.0
    environment:
      MYSQL_ROOT_PASSWORD: root1234
      MYSQL_DATABASE: dockercDb
    volumes:
      - db data:/var/lib/mysql
volumes:
  db-data:
```

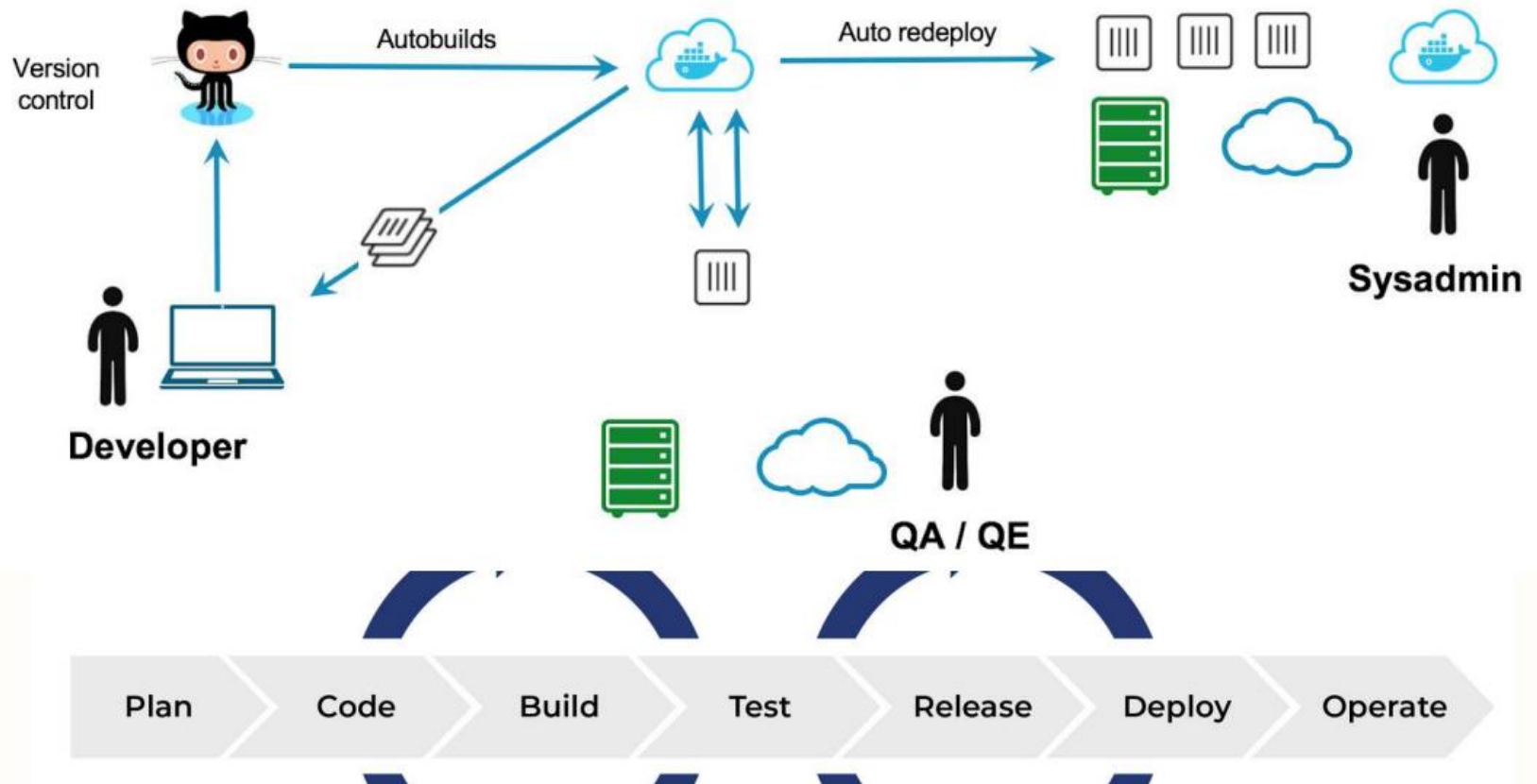
Voir: <https://docs.docker.com/compose/#compose-documentation>

# Docker dans un pipeline CI/CD

1. Development

2. Test

3. Stage / Production



# MERCI

---

[Salaheddine.bentalba@gmail.com](mailto:Salaheddine.bentalba@gmail.com)

+212 601 100 600