Docker

Last updated by | Cerena Hostains | 23 juin 2025 at 13:55 UTC+2

Qu'est-ce que Docker ?

Docker est une plateforme open-source conçue pour simplifier la création, le déploiement et l'exécution d'applications à l'aide de **conteneurs**.
Un conteneur est une unité légère, portable et isolée, qui regroupe l'application et toutes ses dépendances, garantissant une exécution cohérente quel que soit l'environnement hôte.

Concepts clés

Terme	Description
Image	Instantané immuable d'un environnement logiciel (application, dépendances, etc.).
Conteneur	Instance active d'une image.
Dockerfile	Fichier de script permettant de construire une image Docker.
Docker Hub	Registre public permettant de stocker et partager des images Docker.
Volume	Système de stockage persistant attaché à un ou plusieurs conteneurs.
Network	Mécanisme de communication entre plusieurs conteneurs.

Installation (Debian)

1. Installer Docker

```
sudo apt update
sudo apt install -y docker.io
sudo systemctl start docker
sudo systemctl enable docker
```

2. Vérifier l'installation

```
docker --version
```

Commandes de base

Commande	Description
`docker pull <image>`	Télécharge une image depuis Docker Hub
`docker images`	Liste les images locales
`docker run <image>`	Lance un conteneur à partir d'une image
`docker ps`	Affiche les conteneurs en cours d’exécution
`docker ps -a`	Affiche tous les conteneurs, y compris arrêtés
`docker stop <id nom>`	Arrête un conteneur
`docker rm <id nom>`	Supprime un conteneur
`docker rmi <image>`	Supprime une image
`docker exec -it <id> bash`	Lance un shell interactif dans un conteneur

Exemple : Hello World

```
docker run hello-world
```

Cette commande exécute un test rapide pour vérifier que Docker est bien installé et fonctionnel.

Créer une image avec un Dockerfile

1. Exemple de Dockerfile

```
# Image de base
FROM python:3.11-slim
 # Dossier de travail
WORKDIR /app
 # Copier les fichiers de l’application
COPY . .
 # Installer les dépendances Python
RUN pip install -r requirements.txt
 # Lancer l'application
CMD ["python", "app.py"]
```

2. Construire l’image

```
docker build -t monapp .
```

3. Lancer le conteneur

```
docker run -d -p 5000:5000 monapp
```

Utilisation des volumes (stockage persistant)

```
docker run -v /chemin/local:/app/data myimage
```

Les volumes permettent de partager des fichiers entre le système hôte et le conteneur tout en assurant la persistance des données.

Docker Compose

Docker Compose permet d'orchestrer plusieurs conteneurs à l'aide d'un fichier de configuration YAML.

Exemple de `docker-compose.yml`

```yaml
version: '3'

services:
  web:
    image: nginx
    ports:
      - "80:80"

  app:
    build: .
    volumes:
      - .:/app
    ports:
      - "5000:5000"
```

Commandes utiles

`docker-compose up -d` # Démarre les services en arrière-plan
`docker-compose down` # Arrête et supprime les conteneurs

Réseaux entre conteneurs

Docker crée automatiquement un réseau virtuel pour les services définis dans un même fichier `docker-compose.yml`.
Les conteneurs peuvent alors communiquer entre eux via leurs noms de service (par exemple : `app`, `web`, etc.).

Bonnes pratiques

- Utiliser des **images officielles** lorsque cela est possible.
- Nettoyer régulièrement les ressources inutilisées avec :

  ```
  docker system prune
  ```

- Utiliser un fichier `.dockerignore` pour exclure les fichiers non nécessaires à la construction de l'image.
- **Ne jamais stocker de secrets** (clés API, mots de passe) directement dans les Dockerfiles.