Docker

Docker permet de créer, déployer et gérer des applications dans des conteneurs. Contrairement à une machine virtuelle, un conteneur Docker partage le noyau du système d'exploitation hôte, ce qui le rend plus léger et rapide. Docker encapsule les applications avec toutes leurs dépendances, assurant ainsi leur portabilité.

1. Installation de Docker

Sur Ubuntu :

```
sudo apt update
```

```
sudo apt install docker.io
```

2. Commandes de base pour les conteneurs

Vérifier la version de Docker

```
docker --version
```

Affiche la version de Docker installée.

Télécharger une image

```
docker pull <nom_image>
```

Télécharge une image depuis Docker Hub (par exemple `docker pull nginx`).

Lister les images locales

```
docker images
```

Affiche toutes les images disponibles sur votre machine.

Exécuter un conteneur

```
docker run <nom_image>
```

Crée et lance un conteneur à partir d'une image (par exemple `docker run nginx`).

Exécuter un conteneur en arrière-plan (mode détaché) :

```
docker run -d <nom_image>
```

Exécuter avec des ports spécifiques :

```
docker run -d -p <port_hôte>:<port_conteneur> <nom_image>
```

Mappe les ports du conteneur à ceux de l'hôte (par exemple `docker run -d -p 8080:80 nginx`).

Lister les conteneurs actifs

```
docker ps
```

Affiche les conteneurs en cours d'exécution.

Voir tous les conteneurs (même arrêtés) :

```
docker ps -a
```

Arrêter un conteneur

```
docker stop <ID_conteneur>
```

Arrête un conteneur en cours d'exécution.

Démarrer un conteneur arrêté

```
docker start <ID_conteneur>
```

Supprimer un conteneur

```
docker rm <ID_conteneur>
```

Supprime un conteneur arrêté.

Supprimer une image

```
docker rmi <ID_image>
```

Supprime une image locale.

3. Gestion des images

Construire une image à partir d'un Dockerfile

```
docker build -t <nom_image>
```

Construit une image à partir d'un fichier `Dockerfile` situé dans le répertoire courant.

Afficher l'historique d'une image

```
docker history <nom_image>
```

Affiche les différentes couches de construction d'une image.

4. Volumes et persistance des données

Créer un volume

```
docker volume create <nom_volume>
```

Crée un volume pour stocker des données persistantes.

Monter un volume dans un conteneur

```
docker run -d -v
<nom_volume>:<répertoire_dans_conteneur>
<nom_image>
```

Monte un volume dans un conteneur pour persister les données (par exemple `docker run -d -v mon_volume:/data nginx`).

Lister les volumes

```
docker volume ls
```

Affiche tous les volumes Docker créés.

Supprimer un volume

```
docker volume rm <nom_volume>
```

Supprime un volume.

5. Réseaux Docker

Lister les réseaux

```
docker network ls
```

Affiche les réseaux Docker disponibles.

Créer un réseau

```
docker network create <nom_reseau>
```

Crée un nouveau réseau Docker.

Attacher un conteneur à un réseau

```
docker network connect <nom_reseau>
<ID_conteneur>
```

Déconnecter un conteneur d'un réseau

```
docker network disconnect <nom_reseau>
<ID_conteneur>
```

6. Commandes avancées

Entrer dans un conteneur en cours d'exécution

```
docker exec -it <ID_conteneur> /bin/bash
```

Ouvre un terminal interactif à l'intérieur du conteneur.

Voir les logs d'un conteneur

```
docker logs <ID_conteneur>
```

Affiche les logs d'un conteneur en cours d'exécution.

Supprimer tous les conteneurs et images non utilisés

```
docker system prune
```

Nettoie toutes les ressources Docker inutilisées (conteneurs, images, volumes).

7. Gestion des conteneurs via Docker Compose

Docker Compose permet de gérer des applications multi-conteneurs avec un fichier YAML.

Exécuter un projet Docker Compose

```
docker-compose up
```

Lance tous les services définis dans le fichier `docker-compose.yml`.

Arrêter et supprimer tous les conteneurs d’un projet

```
docker-compose down
```

Arrête et nettoie tous les services d'un projet.