🐳 Docker — Premiers pas

6TQ

Objectifs

À la fin de cette page, tu seras capable de :

- Installer Docker Desktop sous Windows
- Lancer un conteneur avec `docker run`
- Lister, inspecter, arrêter et supprimer un conteneur
- Orchestrer deux services qui coopèrent avec Docker Compose

💻 Installation sous Windows

Prérequis

- Windows 10 version 21H2 ou Windows 11 (64 bits)
- WSL 2 activé (Windows Subsystem for Linux)
- Virtualisation activée dans le BIOS (VT-x / AMD-V)

Étape 1 : Activer WSL 2

Ouvre PowerShell **en administrateur** et exécute :

```
wsl --install
```

Redémarre le PC si demandé.

Étape 2 : Télécharger Docker Desktop

Télécharge l'installeur sur : **docker.com/products/docker-desktop**

Lance l'installeur et coche "**Use WSL 2 instead of Hyper-V**".

Étape 3 : Vérifier l'installation

Ouvre un terminal (PowerShell ou CMD) et tape :

```
docker --version
docker run hello-world
```

Si tu vois `Hello from Docker!`, c'est bon ! 🎉

> 💡 L'interface graphique **DOCKER DESKTOP** est détaillée dans l'article suivant.

⌨️ Commandes essentielles

Gestion des images

```
# Télécharger une image depuis Docker Hub
docker pull nginx

# Lister les images locales
docker images

# Supprimer une image
docker rmi nginx
```

Gestion des conteneurs

```
# Lancer un conteneur (télécharge l'image si besoin)
docker run nginx

# Lancer en arrière-plan (-d = detached)
docker run -d nginx

# Lancer avec un nom et un port (hôte:conteneur)
docker run -d --name mon-serveur -p 8080:80 nginx

# Lister les conteneurs actifs
docker ps

# Lister TOUS les conteneurs (actifs + arrêtés)
docker ps -a

# Arrêter un conteneur
docker stop mon-serveur
```

```
18
19  # Supprimer un conteneur
20  docker rm mon-serveur
21
22  # Arrêter ET supprimer automatiquement (--rm)
23  docker run --rm -p 8080:80 nginx
```

Inspecter et déboguer

```
1   # Voir les logs d'un conteneur
2   docker logs mon-serveur
3
4   # Logs en temps réel (suivi)
5   docker logs -f mon-serveur
6
7   # Ouvrir un terminal dans un conteneur actif
8   docker exec -it mon-serveur bash
9
10  # Inspecter les détails d'un conteneur
11  docker inspect mon-serveur
```

Nettoyage

```
1   # Supprimer tous les conteneurs arrêtés
2   docker container prune
3
4   # Supprimer toutes les images inutilisées
5   docker image prune
6
7   # Nettoyage global (images, conteneurs, réseaux non utilisés)
8   docker system prune
```

Docker Compose

```
1   # Démarrer tous les services (en arrière-plan)
2   docker compose up -d
3
4   # Arrêter tous les services
5   docker compose down
6
7   # Voir les logs de tous les services
8   docker compose logs -f
9
10  # Reconstruire les images et redémarrer
11  docker compose up -d --build
```

🧪 Exercice 1 — Ton premier serveur web avec Nginx

Objectif : Lancer un serveur web Nginx en une commande et y accéder depuis ton navigateur.

Contexte

Nginx est un serveur web très utilisé en production. Normalement, l'installer sous Windows est compliqué. Avec Docker, c'est une seule commande.

Instructions

1. Ouvre un terminal PowerShell.

2. Lance le conteneur :

```
docker run -d --name mon-nginx -p 8080:80 nginx
```

Décortiquons cette commande :

Argument	Signification
`run`	Crée et démarre un conteneur
`-d`	En arrière-plan (detached)
`--name mon-nginx`	Donne un nom au conteneur
`-p 8080:80`	Redirige le port 8080 de ta machine vers le port 80 du conteneur
`nginx`	Le nom de l'image à utiliser (téléchargée depuis Docker Hub)

3. Ouvre ton navigateur et va sur : **http://localhost:8080**

Tu devrais voir la page de bienvenue de Nginx ✅

4. Vérifie que le conteneur tourne :

```
docker ps
```

5. Lis les logs du serveur :

```
docker logs mon-nginx
```

Recharge la page du navigateur et regarde les logs se mettre à jour.

6. Arrête et supprime le conteneur :

```
docker stop mon-nginx
docker rm mon-nginx
```

Questions de réflexion

- Que se passe-t-il si tu essaies d'accéder à http://localhost:8080 après avoir arrêté le conteneur ?
- Que se passerait-il si tu lançais deux fois la même commande avec le même `--name` ?
- Quel est le port par défaut d'un serveur web HTTP ?

🧪 Exercice 2 — Une base de données MySQL sans installation

Objectif : Lancer un serveur MySQL complet via Docker, s'y connecter et créer une base de données.

Contexte

Tu connais MySQL depuis tes cours. L'installer sur un PC prend du temps et "pollue" le système. Avec Docker, tu lances une instance MySQL propre en quelques secondes, que tu peux supprimer après utilisation.

Instructions

1. Lance MySQL dans un conteneur :

```
docker run -d \
  --name mon-mysql \
  -p 3306:3306 \
  -e MYSQL_ROOT_PASSWORD=motdepasse \
  -e MYSQL_DATABASE=mabase \
  mysql:8.0
```

> **NOTE WINDOWS** : Sur PowerShell, remplace les `\` par des backticks `` ` `` pour les retours à la ligne, ou écris tout sur une seule ligne.

Décortiquons les nouveaux arguments :

Argument	Signification
`-p 3306:3306`	Expose le port MySQL standard
`-e MYSQL_ROOT_PASSWORD=motdepasse`	Variable d'environnement : mot de passe root

Argument	Signification
`-e MYSQL_DATABASE=mabase`	Crée automatiquement une base "mabase" au démarrage
`mysql:8.0`	Image MySQL version 8.0

2. Attends quelques secondes que MySQL démarre. Surveille les logs :

```
docker logs -f mon-mysql
```

Quand tu vois `ready for connections`, appuie sur `Ctrl+C` pour quitter les logs.

3. Connecte-toi à MySQL depuis l'intérieur du conteneur :

```
docker exec -it mon-mysql mysql -u root -pmotdepasse
```

Tu es maintenant dans le shell MySQL.

4. Teste quelques commandes :

```
SHOW DATABASES;
USE mabase;
CREATE TABLE eleves (id INT AUTO_INCREMENT PRIMARY KEY, nom VARCHAR(100));
INSERT INTO eleves (nom) VALUES ('Alice'), ('Bob');
SELECT * FROM eleves;
EXIT;
```

5. Connecte-toi aussi depuis **MySQL Workbench** ou **HeidiSQL** sur ta machine hôte :

- Hôte : `127.0.0.1`
- Port : `3306`
- Utilisateur : `root`
- Mot de passe : `motdepasse`

6. Supprime le conteneur :

```
docker stop mon-mysql
docker rm mon-mysql
```

⚠️ Observation importante

Recrée le conteneur et retourne voir la table `eleves`. Elle a **disparu**. C'est normal : sans **volume**, les données ne sont pas persistées.

Pour garder les données, on utilise un volume :

```
docker run -d \
  --name mon-mysql \
  -p 3306:3306 \
  -e MYSQL_ROOT_PASSWORD=motdepasse \
```

```
  -v mysql_data:/var/lib/mysql \
  mysql:8.0
```

Le `-v mysql_data:/var/lib/mysql` crée un volume nommé `mysql_data` qui survit à la suppression du conteneur. Les volumes sont détaillés dans un article dédié plus loin.

🧪 Exercice 3 — MySQL + phpMyAdmin avec Docker Compose

Objectif : Lancer deux services qui coopèrent (MySQL + phpMyAdmin) avec un seul fichier de configuration.

Contexte

Dans la vraie vie, les applications ont souvent besoin de plusieurs services. **Docker Compose** permet de les définir ensemble, de gérer leur démarrage dans l'ordre, et de les faire communiquer sur un réseau interne.

> 💡 Cet exercice est un **AVANT-GOÛT** de Compose. La syntaxe complète est détaillée dans l'article dédié plus loin dans le chapitre.

Instructions

1. Crée un dossier `docker-mysql-lab` sur ton Bureau.

2. Dans ce dossier, crée un fichier `docker-compose.yml` :

```yaml
services:

  # Service 1 : la base de données MySQL
  db:
    image: mysql:8.0
    container_name: lab_mysql
    restart: unless-stopped
    environment:
      MYSQL_ROOT_PASSWORD: secret
      MYSQL_DATABASE: monprojet
      MYSQL_USER: dev
      MYSQL_PASSWORD: devpass
    volumes:
      - mysql_data:/var/lib/mysql
    networks:
      - lab_network

  # Service 2 : phpMyAdmin pour gérer la base via navigateur
  phpmyadmin:
```

```
    image: phpmyadmin:latest
    container_name: lab_phpmyadmin
    restart: unless-stopped
    ports:
      - "8080:80"
    environment:
      PMA_HOST: db          # Le nom du service MySQL dans le réseau Docker
      PMA_PORT: 3306
    depends_on:
      - db
    networks:
      - lab_network

# Déclaration du volume persistant
volumes:
  mysql_data:

# Déclaration du réseau interne
networks:
  lab_network:
```

3. Ouvre un terminal dans ce dossier et lance tout :

```
docker compose up -d
```

Docker va :

- Télécharger les images manquantes
- Créer le réseau `lab_network`
- Créer le volume `mysql_data`
- Démarrer MySQL en premier (car `depends_on`)
- Démarrer phpMyAdmin ensuite

4. Ouvre **http://localhost:8080** dans ton navigateur.

Connecte-toi avec :

- Serveur : `db`
- Utilisateur : `root`
- Mot de passe : `secret`

5. Crée une table, insère des données.

6. Arrête les services :

```
docker compose down
```

Relance avec `docker compose up -d`. Tes données sont **toujours là** grâce au volume.

7. Pour tout supprimer y compris les volumes :

```
docker compose down -v
```

Pour aller plus loin

- Ajoute un troisième service : une application PHP qui se connecte à la base MySQL
- Modifie le port de phpMyAdmin pour utiliser `8888:80` et relance
- Explore l'onglet **Containers** de Docker Desktop pour visualiser les deux services

---

📊 Récap des commandes de base

Commande	Action
`docker pull <image>`	Télécharger une image
`docker run -d -p X:Y <image>`	Lancer un conteneur
`docker ps`	Voir les conteneurs actifs
`docker stop <nom>`	Arrêter un conteneur
`docker rm <nom>`	Supprimer un conteneur
`docker logs <nom>`	Voir les journaux
`docker exec -it <nom> bash`	Entrer dans un conteneur
`docker compose up -d`	Démarrer les services Compose
`docker compose down`	Arrêter les services Compose