🚀 Objectif du cours

Comprendre la différence entre **Bind Mounts** et **Volumes** dans Docker, pourquoi et quand les utiliser, et les manipuler à travers un petit exemple pratique.

---

1. 🔍 Concepts de base

📦 Volume

- Géré **par Docker lui-même**
- Stocké dans le répertoire `/var/lib/docker/volumes`
- Sécurisé, persistant même après la suppression du conteneur
- Idéal pour les données d'une **base de données, logs, etc.**

```
bash
CopyEdit
docker volume create mon_volume
```

🗂️ Bind Mount

- Permet de **lier** un répertoire spécifique de l’hôte au conteneur
- Ex: synchroniser un dossier local avec celui du conteneur
- Très utile en **développement** pour voir les changements en live

```
bash
CopyEdit
docker run -v /chemin/vers/repertoire:/app mon_image
```

---

2. 📐 Différences clés

Caractéristique	Volume	Bind Mount
Gestion	Docker	OS de l’hôte
Sécurité	Haute	Moins sécurisé
Portabilité	Haute	Dépend de l'OS
Cas d’usage	Prod (DB, logs)	Dev (code local)

---

3. ⚙️ Mini pratique

□ Objectif :

Créer un fichier dans notre machine et le voir en **live** dans un conteneur, et vice versa.

Étape 1 – Préparer un fichier local

```
bash
CopyEdit
mkdir ~/bind_test
echo "Bonjour Docker" > ~/bind_test/message.txt
```

Étape 2 – Lancer un conteneur avec un bind mount

```
bash
CopyEdit
docker run --rm -it -v ~/bind_test:/app alpine sh
```

- `--rm` : supprime le conteneur à la fin
- `-it` : mode interactif
- `-v ~/bind_test:/app` : on monte le dossier local

👉 Dans le terminal du conteneur :

```
sh
CopyEdit
cat /app/message.txt
```

☑ Vous devriez voir "Bonjour Docker"

Étape 3 – Modifier depuis le conteneur

Toujours dans le conteneur :

```
sh
CopyEdit
echo "Ajouté depuis le conteneur" >> /app/message.txt
exit
```

Étape 4 – Vérifier côté hôte

```
bash
CopyEdit
cat ~/bind_test/message.txt
```

☑ Le texte modifié est visible : preuve que c'est **synchronisé en temps réel**.

Étape 5 – Tester avec un volume Docker

```
bash
CopyEdit
docker volume create mon_volume
docker run -it --rm -v mon_volume:/app alpine sh
```

Dans le conteneur :

```
sh
CopyEdit
echo "Contenu du volume" > /app/volume.txt
exit
```

Étape 6 – Voir le contenu du volume

```
bash
CopyEdit
docker run -it --rm -v mon_volume:/app alpine cat /app/volume.txt
```

☑ Le fichier est persistant dans le volume même si le conteneur a été supprimé.

🧠 Résumé

Volume	Bind Mount
Gestion interne Docker	Lien direct avec système
Idéal pour production	Idéal pour développement
Plus sûr et plus isolé	Plus souple mais risqué

🛠 Ce que fait exactement cette commande :

Partie de la commande	Rôle
`docker run`	Lance un conteneur
`--rm`	Supprime le conteneur après son exécution
`-it`	Mode interactif avec terminal
`-v ~/bind_test:/app`	Monte le dossier local dans le conteneur
`alpine`	**Image utilisée** pour créer le conteneur
`sh`	Shell exécuté à l'intérieur du conteneur