Stockage avec Docker

1. Stockage par défaut dans un conteneur (Non-Persistant)

- Couche Modifiable : Les fichiers que vous créez ou modifiez à l'intérieur d'un conteneur sont stockés sur une couche modifiable spécifique à ce conteneur. C'est une partie de ce que l'on appelle le *Container Layer*.
- Non-Persistance : Le problème majeur de cette couche est qu'elle est étroitement liée à la machine hôte et au conteneur lui-même. Lorsque le conteneur est supprimé, cette couche modifiable disparaît, et toutes les données qu'elle contenait sont perdues. C'est pourquoi ce n'est pas la bonne solution pour les données importantes (bases de données, logs, configurations, etc.).
- Latence : L'écriture sur cette couche nécessite un pilote de stockage (comme overlay2 sur Linux) qui gère le système de fichiers en couches. Cela peut générer plus de latence que l'écriture directe sur le système de fichiers de l'hôte.

2.Volume Géré par Docker (volumes):

Ce type de montage est recommandé pour la persistance des données de production (comme les bases de données). Il garantit que les données ne sont pas perdues lorsque le conteneur est supprimé.

exemple:

```
services:
  db:
    image: postgres:14
    environment:
      POSTGRES_USER: user
      POSTGRES_PASSWORD: password
    volumes:
      - db_data:/var/lib/postgresql/data
    restart: always

volumes:
  db_data:
    # Docker gérera l'emplacement de ce volume sur l'hôte
    # Vous n'avez pas besoin de spécifier un chemin sur l'hôte
```

- Explication :
    - Le volume nommé db_data est déclaré à la fin du fichier sous volumes:.
    - Dans la section services.db.volumes, la syntaxe nom_du_volume:chemin_dans_le_conteneur lie le volume géré par Docker (db_data) au répertoire de données interne du conteneur (/var/lib/postgresql/data).

3. Montage de Liaison (bind mounts)

Ce type de montage est idéal pour le développement ou l'injection de fichiers de configuration spécifiques de l'hôte.

Exemple:

```
services:
  app:
    build: .
    ports:
      - "80:8000"
    volumes:
        # Pour lier un fichier de configuration spécifique de l'hôte :
      - ./config/settings.ini:/etc/app/settings.ini
```

- Explication :
    - Dans la section services.app.volumes, la syntaxe est chemin_sur_l'hôte:chemin_dans_le_conteneur.
    - ./config/settings.ini:/etc/app/settings.ini

      Le dossier local ./app_code (relatif à l'emplacement du docker-compose.yml) est monté dans le conteneur sous /app. Toute modification de code sur votre hôte est immédiatement visible dans le conteneur.

4. Montage Temporaire (tmpfs mounts)

Ce type de montage est utilisé pour les données qui ne doivent jamais persister sur le disque et doivent être stockées dans la mémoire vive (RAM) du système hôte.

Exemple:

```
services:
  processor:
    image: my-processing-image
    tmpfs:
      - /tmp/cache_data:size=100m
      - /var/log/app_logs
```

- Explication :
    - La clé tmpfs: est utilisée pour créer ces montages.
    - /tmp/cache_data:size=100m : Crée un montage tmpfs dans le conteneur à /tmp/cache_data et lui alloue une taille maximale de 100 Mo dans la RAM.
    - /var/log/app_logs : Crée un autre montage tmpfs simple.
    - ⚠️ Attention : Lorsque le conteneur est arrêté ou redémarré, toutes les données dans ces répertoires sont perdues.

Synthèse:

Type de Montage	Clé docker-compose	Chemin Spécifié	But Principal
Volume	volumes (avec définition racine)	nom_du_volume:/chemin/conteneur	Persistance des données (BDD, etc.)
Bind Mount	volumes	/chemin/hôte:/chemin/conteneur	Développement ou Configuration de l'hôte
tmpfs Mount	tmpfs	/chemin/conteneur	Temporaire, Stockage en RAM (Cache)