Rappels de Docker Compose

Docker Compose est un outil qui permet de définir et de gérer des applications multi-conteneurs. Il simplifie le déploiement de services complexes en permettant de définir plusieurs conteneurs, leurs configurations et leurs interconnexions dans un fichier unique, appelé `docker-compose.yml`. Ce fichier est écrit en YAML et contient des informations sur la configuration des services, des réseaux et des volumes, ce qui facilite le démarrage, l'arrêt et la mise à l'échelle d'applications contenant plusieurs services.

---

1. Introduction à Docker Compose

Docker Compose permet de définir des applications multi-conteneurs en utilisant un seul fichier de configuration. Contrairement à la gestion manuelle des conteneurs avec la commande `docker run`, Docker Compose permet de déployer des applications qui nécessitent plusieurs conteneurs avec une configuration complexe.

Les principales fonctionnalités de Docker Compose incluent :

- **Définir plusieurs services** : Docker Compose permet de configurer plusieurs services dans un seul fichier, ce qui rend la gestion des applications complexes plus simple.
- **Définir des réseaux et des volumes** : Vous pouvez spécifier des réseaux entre les services et des volumes partagés pour la persistance des données.
- **Faciliter le développement** : Docker Compose permet de démarrer des environnements de développement locaux en quelques commandes, rendant le travail avec des applications multi-conteneurs plus rapide et plus fluide.
- **Isoler les environnements** : Chaque projet Docker Compose fonctionne de manière isolée, ce qui permet de créer des environnements d'application indépendants et reproductibles.

---

2. Structure d'un Fichier `docker-compose.yml`

Le fichier `docker-compose.yml` est au cœur de Docker Compose. Il définit l'ensemble des services, réseaux, volumes et configurations nécessaires à l'exécution de l'application.

Structure de base d'un fichier `docker-compose.yml` :

Voici un exemple simple d'un fichier `docker-compose.yml` pour une application qui utilise une base de données MySQL et un serveur web Nginx.

```
version: '3'
services:
  web:
    image: nginx:latest
    ports:
      - "8080:80"
    networks:
      - my-network
```

```
  db:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: example
    networks:
      - my-network
volumes:
  db-data:
    driver: local
networks:
  my-network:
    driver: bridge
```

Explication de la structure :

- **version** : Spécifie la version du format de fichier Docker Compose (ici, version 3).
- **services** : Contient la configuration de tous les services de l'application. Chaque service peut spécifier une image Docker, des volumes, des variables d'environnement, des ports exposés et des réseaux.
  - Le service `web` utilise l'image **Nginx** et expose le port 80 du conteneur sur le port 8080 de l'hôte.
  - Le service `db` utilise l'image **MySQL 5.7** et définit la variable d'environnement `MYSQL_ROOT_PASSWORD` pour initialiser la base de données.
- **volumes** : Spécifie les volumes partagés entre les services pour la persistance des données.
  - Le volume `db-data` est utilisé pour stocker les données de la base de données.
- **networks** : Définit les réseaux sur lesquels les services peuvent communiquer entre eux. Dans cet exemple, les services `web` et `db` sont sur le même réseau `my-network`, ce qui leur permet de se connecter et de communiquer.

---

3. Commandes Docker Compose de Base

Docker Compose simplifie plusieurs tâches courantes associées à la gestion des conteneurs multi-services. Voici quelques commandes clés pour travailler avec Docker Compose.

- **Démarrer l'application** :
  Pour démarrer les services définis dans le fichier `docker-compose.yml`, utilisez la commande suivante :

  ```
  docker-compose up
  ```

  - Par défaut, cette commande lance les services en mode "foreground" (avant), affichant les logs de chaque service dans la console.
  - Ajoutez `-d` pour démarrer les services en arrière-plan (mode détaché) :

    ```
    docker-compose up -d
    ```

- **Arrêter l'application** :
  Pour arrêter les services en cours d'exécution, utilisez la commande :

  ```
  docker-compose down
  ```

  - Cela arrête tous les services et supprime les conteneurs créés, mais conserve les volumes et les réseaux par défaut. Utilisez `docker-compose down --volumes` pour supprimer également les volumes associés aux services.

- **Voir les logs des services** :
  Vous pouvez visualiser les logs des services en cours d'exécution avec :

  ```
  docker-compose logs
  ```

  - Vous pouvez aussi spécifier un service particulier pour afficher uniquement les logs de ce service :

    ```
    docker-compose logs web
    ```

- **Mettre à jour l'application** :
  Si vous avez modifié le fichier `docker-compose.yml` (par exemple, pour modifier une image ou un port), vous pouvez mettre à jour les services en exécutant :

  ```
  docker-compose up -d
  ```

  Cela arrêtera les services existants, appliquera les modifications et redémarrera les services mis à jour.

- **Lister les services actifs** :
  Pour afficher l'état des services gérés par Docker Compose, utilisez :

  ```
  docker-compose ps
  ```

---

4. Volumes et Réseaux dans Docker Compose

Une des forces de Docker Compose est de permettre une gestion facile des **volumes** et des **réseaux**, assurant ainsi que les services peuvent persister leurs données et communiquer entre eux de manière

transparente.

a. Volumes :

Les volumes sont utilisés pour persister les données au-delà de la durée de vie des conteneurs. Ils peuvent être définis dans le fichier `docker-compose.yml` pour chaque service nécessitant une persistance de données.

Exemple de volume pour un service de base de données :

```
services:
  db:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: example
    volumes:
      - db-data:/var/lib/mysql
volumes:
  db-data:
```

Cela crée un volume nommé `db-data` qui persiste les données MySQL, permettant de conserver les données même si le conteneur est supprimé.

b. Réseaux :

Docker Compose permet de définir des réseaux pour organiser la communication entre les services. Par défaut, Docker Compose crée un réseau privé pour les services définis dans le fichier `docker-compose.yml`. Cependant, vous pouvez spécifier un réseau personnalisé pour mieux gérer la communication entre les services.

Exemple de configuration de réseau :

```
services:
  web:
    image: nginx:latest
    networks:
      - front-end
  db:
    image: mysql:5.7
    networks:
      - back-end
networks:
  front-end:
  back-end:
```

Ici, le service `web` est connecté au réseau `front-end`, tandis que le service `db` est connecté au réseau `back-end`, ce qui empêche les services de se communiquer entre eux directement, sauf si des réseaux

communs sont définis.

---

5. Utilisation de Docker Compose avec des Environnements Multiples

Docker Compose permet également de travailler avec plusieurs environnements de déploiement, ce qui est utile lorsque vous avez des environnements de développement, de test et de production.

a. Fichier `docker-compose.override.yml`

Docker Compose prend en charge un fichier `docker-compose.override.yml` qui permet de définir des configurations spécifiques pour un environnement local ou un environnement de développement.

Par exemple, vous pouvez définir un fichier de configuration différent pour la base de données ou les variables d'environnement en fonction de l'environnement.

Exemple de fichier `docker-compose.override.yml` :

```
services:
  db:
    environment:
      MYSQL_ROOT_PASSWORD: devpassword
```

Cela remplace la valeur par défaut de `MYSQL_ROOT_PASSWORD` dans le fichier principal `docker-compose.yml` lorsque vous utilisez l'override.

b. Utiliser plusieurs fichiers Compose :

Vous pouvez également spécifier plusieurs fichiers Compose pour gérer différents environnements. Pour combiner plusieurs fichiers Compose, vous pouvez utiliser la commande `-f` :

```
docker-compose -f docker-compose.yml -f docker-compose.prod.yml up -d
```

Cela applique les configurations définies dans les deux fichiers `docker-compose.yml` et `docker-compose.prod.yml`, en permettant des configurations spécifiques pour la production.

Résumé des Concepts Docker Compose

Concept	Description
docker-compose.yml	Fichier de configuration principal où sont définis les services, les réseaux, les volumes et les configurations des conteneurs.
`docker-compose up`	Démarre les services définis dans le fichier `docker-compose.yml`.

Concept	Description
`docker-compose down`	Arrête et supprime les conteneurs et services, mais conserve les volumes et réseaux (avec `--volumes` pour les supprimer aussi).
`docker-compose ps`	Affiche l'état des services en cours d'exécution.
Volumes	Permet de persister les données de conteneurs, avec des volumes partagés entre services et réplicas.
Réseaux	Définit comment les conteneurs peuvent communiquer entre eux, avec des réseaux isolés ou partagés.
Override Files	Permet de modifier les paramètres pour des environnements spécifiques (développement, production).

Conclusion

Docker Compose simplifie la gestion des applications multi-conteneurs en offrant un moyen cohérent de définir, configurer et gérer plusieurs services, réseaux et volumes dans un fichier unique. Cela facilite le développement, le test, et la mise en production des applications conteneurisées.