Administration des Stacks dans Docker Swarm avec Docker Compose

Une **stack** dans Docker Swarm est un ensemble de services, réseaux et volumes déployés en tant qu'unité. Les stacks sont particulièrement utiles pour déployer des applications complexes qui nécessitent plusieurs services, avec des configurations de réseau et de stockage. Docker Compose permet de gérer facilement les stacks dans Swarm, en utilisant un fichier `docker-compose.yml` pour décrire et déployer les services, ainsi que leurs configurations respectives.

---

1. Déployer une Stack avec Docker Compose

Dans Docker Swarm, vous pouvez déployer une stack en utilisant **Docker Compose** avec la commande `docker stack deploy`. Cette commande permet de déployer l'ensemble des services définis dans un fichier Compose au sein du cluster Swarm. Le fichier `docker-compose.yml` utilisé pour les stacks dans Swarm peut inclure des configurations de services, réseaux et volumes, de manière similaire à l'utilisation traditionnelle de Docker Compose.

a. Créer un fichier `docker-compose.yml` pour une Stack

Voici un exemple de fichier `docker-compose.yml` qui définit une application simple avec deux services : un service web utilisant Nginx et un service de base de données utilisant MySQL.

Exemple de `docker-compose.yml` pour une stack :

```
version: '3.8'
services:
  web:
    image: nginx
    ports:
      - "8080:80"
    networks:
      - frontend
    depends_on:
      - db

  db:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: example
    networks:
      - backend
    volumes:
      - db-data:/var/lib/mysql

volumes:
  db-data:
    driver: local

networks:
```

```
  frontend:
  backend:
```

- **web** : Service Nginx, exposant le port 80 sur l'hôte avec le port 8080, et connecté au réseau `frontend`.
- **db** : Service MySQL, avec une variable d'environnement pour le mot de passe root et un volume persistant pour stocker les données.
- **Volumes** : Volume `db-data` utilisé pour persister les données de la base de données MySQL.
- **Réseaux** : `frontend` et `backend` sont des réseaux distincts auxquels les services peuvent être connectés. Le service `web` est connecté au réseau `frontend` et le service `db` au réseau `backend`.

b. Déployer la Stack dans Docker Swarm

Pour déployer cette stack dans Docker Swarm, utilisez la commande `docker stack deploy`. Cette commande permet de déployer tous les services définis dans le fichier `docker-compose.yml` sur le cluster Swarm.

1. **Déployer la Stack** :
   Utilisez la commande suivante pour déployer la stack :

   ```
   docker stack deploy -c docker-compose.yml my_stack
   ```

   - `-c` : Spécifie le fichier Docker Compose à utiliser.
   - `my_stack` : Le nom de la stack (ici `my_stack`).

   Cette commande crée une stack dans Swarm, déploie tous les services définis dans le fichier `docker-compose.yml`, et crée les réseaux et volumes nécessaires. Docker Swarm s'assurera que les services sont déployés et répartis sur les nœuds du cluster de manière optimale.

c. Vérifier le Déploiement de la Stack

Après avoir déployé la stack, vous pouvez vérifier son statut à l'aide des commandes suivantes :

- **Lister les stacks déployées** :

  ```
  docker stack ls
  ```

  Cela vous montre les stacks actuellement déployées dans votre cluster Swarm.

- **Lister les services dans la stack** :

  ```
  docker stack services my_stack
  ```

Cela affiche les services associés à la stack `my_stack`, ainsi que leur statut (ex. `replicated`, `running`, etc.).

- **Voir les conteneurs dans la stack** :

```
docker stack ps my_stack
```

Cela vous permet de voir les tâches (ou conteneurs) de chaque service dans la stack et leur état actuel (par exemple, `running`, `shutdown`).

---

2. Mettre à Jour une Stack

Une des puissantes fonctionnalités de Docker Swarm avec Docker Compose est la possibilité de mettre à jour une stack sans interruption. Lorsque vous modifiez le fichier `docker-compose.yml` (par exemple, pour changer une image de service ou ajuster des configurations), Docker Swarm peut appliquer ces changements de manière transparente à la stack.

a. Mise à jour d'une Stack

Pour appliquer les changements à une stack déjà déployée, utilisez la commande `docker stack deploy` avec le même nom de stack et le fichier `docker-compose.yml` mis à jour.

Exemple de mise à jour de la stack :

1. Modifiez le fichier `docker-compose.yml` pour modifier, par exemple, l'image du service web.
2. Déployez à nouveau la stack :

```
docker stack deploy -c docker-compose.yml my_stack
```

Docker Swarm gère la mise à jour des services, effectuant un **rolling update**, c'est-à-dire qu'il met à jour les réplicas du service progressivement sans interrompre l'ensemble de l'application.

b. Stratégie de Rolling Update

Docker Swarm permet de définir des stratégies pour les mises à jour des services, permettant de contrôler le nombre de réplicas mis à jour en même temps et de garantir une disponibilité continue.

Exemple de commande avec stratégie de mise à jour :

```
docker service update --update-parallelism 2 --image nginx:latest
my_stack_web
```

- `--update-parallelism 2` : Met à jour deux réplicas en même temps.

- `--image nginx:latest` : Définit l'image à mettre à jour (ici, `nginx:latest`).

Cela permet de mettre à jour le service tout en minimisant l'impact sur la disponibilité.

---

3. Supprimer une Stack

Si vous n'avez plus besoin d'une stack, vous pouvez la supprimer du cluster Swarm en utilisant la commande suivante. Cela arrête et supprime tous les services, réseaux et volumes associés à la stack.

```
docker stack rm my_stack
```

Cela supprime la stack `my_stack` et tous ses services associés. Les volumes et réseaux ne sont pas supprimés par défaut, mais vous pouvez les supprimer manuellement si nécessaire.

---

4. Gestion des Secrets et Configurations dans une Stack

Docker Swarm permet également de gérer des **secrets** et des **configurations** dans une stack pour garantir la sécurité des applications et services.

a. Ajouter des Secrets à une Stack

Les **secrets** sont utilisés pour stocker de manière sécurisée des informations sensibles, comme des mots de passe ou des clés API. Les secrets sont stockés dans le gestionnaire de secrets de Docker Swarm et peuvent être utilisés par les services.

Exemple de création d'un secret :

```
echo "my_secret_password" | docker secret create db_password -
```

Vous pouvez ensuite référencer ce secret dans votre fichier `docker-compose.yml` :

```
version: '3.8'
services:
  db:
    image: mysql:5.7
    secrets:
      - db_password
    environment:
      MYSQL_ROOT_PASSWORD_FILE: /run/secrets/db_password

secrets:
  db_password:
    external: true
```

Cela permettra au service `db` d'utiliser le secret `db_password` dans ses variables d'environnement de manière sécurisée.

b. Ajouter des Configurations à une Stack

Les **configurations** sont utilisées pour stocker des fichiers de configuration ou des données non sensibles qui doivent être partagées entre les services.

Exemple de création d'une configuration :

```
docker config create my_config /path/to/config-file
```

Ensuite, vous pouvez référencer cette configuration dans le fichier `docker-compose.yml` :

```
version: '3.8'
services:
  web:
    image: nginx
    configs:
      - source: my_config
        target: /etc/nginx/nginx.conf

configs:
  my_config:
    external: true
```

Cela permet au service `web` d'utiliser le fichier de configuration partagé `my_config` dans le conteneur.

Résumé des Commandes pour Administrer des Stacks dans Docker Swarm

Action	Commande
Déployer une stack	`docker stack deploy -c docker-compose.yml my_stack`
Lister les stacks déployées	`docker stack ls`
Lister les services d'une stack	`docker stack services my_stack`
Lister les tâches de la stack	`docker stack ps my_stack`
Mettre à jour une stack	`docker stack deploy -c docker-compose.yml my_stack`
Supprimer une stack	`docker stack rm my_stack`
Ajouter un secret à une stack	`docker secret create <secret_name> <file>`

Action	Commande
Ajouter une configuration	`docker config create <config_name> <file>`

Conclusion

L'administration des stacks dans Docker Swarm avec Docker Compose simplifie le déploiement et la gestion d'applications multi-services. Avec des fonctionnalités comme les mises à jour sans interruption, la gestion des secrets et des configurations, et la possibilité de gérer des volumes et des réseaux partagés, Docker Compose dans Swarm permet de déployer et de gérer des applications complexes de manière cohérente et efficace.