Architecture de Docker Swarm

Docker Swarm est la solution native de Docker pour la gestion de clusters de conteneurs. Il permet de coordonner plusieurs hôtes Docker pour les faire fonctionner comme un seul cluster logique. Avec Docker Swarm, vous pouvez facilement déployer et gérer des applications distribuées et des microservices à grande échelle.

Swarm fournit des fonctionnalités comme l'orchestration des conteneurs, la gestion de la haute disponibilité, la scalabilité, le load balancing, ainsi que la distribution automatique des tâches sur les nœuds du cluster.

1. Présentation de Docker Swarm

Docker Swarm transforme plusieurs hôtes Docker en un seul cluster cohérent et géré de manière centralisée. Cela permet aux développeurs de déployer des applications conteneurisées sur des clusters Docker sans avoir à se soucier des détails de gestion de l'infrastructure. Swarm orchestre les conteneurs à travers les nœuds du cluster en utilisant une architecture maître-esclave.

Les principaux avantages de Docker Swarm incluent :

- **Orchestration native** : Docker Swarm est intégré directement dans Docker, il n'y a pas besoin d'installer ou de configurer des outils supplémentaires (comme Kubernetes).
- **Simplicité de configuration** : La mise en place d'un cluster Swarm est simple et nécessite quelques commandes Docker de base.
- **Haute disponibilité** : Les services sont répartis sur plusieurs nœuds et peuvent être redondants pour garantir une tolérance aux pannes.
- **Scalabilité** : Vous pouvez facilement ajouter ou supprimer des nœuds et mettre à l'échelle les services en fonction des besoins.
- **Load balancing** : Swarm gère automatiquement le load balancing pour distribuer les demandes réseau entre les conteneurs et services.

Swarm utilise un modèle basé sur deux types de nœuds : les nœuds **manager** et les nœuds **worker**. Chaque nœud joue un rôle spécifique dans le fonctionnement du cluster.

---

2. Rôles des Nœuds dans Docker Swarm : Manager et Worker

Docker Swarm repose sur un système de **nœuds**, chacun jouant un rôle spécifique. Il existe deux types de nœuds : **manager** et **worker**.

a. Nœud Manager

Un **nœud manager** est responsable de la gestion du cluster Swarm et de la coordination des services Docker sur l'ensemble du cluster. Le manager prend des décisions concernant le placement des services, le routage des requêtes, et la gestion des nœuds. Il gère également l'état global du cluster, les configurations des services et les communications entre les nœuds.

Les principales fonctions des nœuds **manager** incluent :

- **Planification et orchestration** : Les managers planifient les tâches (conteneurs et services) à exécuter sur les nœuds workers. Ils attribuent les tâches en fonction des ressources disponibles et des contraintes de placement.
- **Coordination du cluster** : Les managers prennent en charge la gestion du cluster, y compris la distribution des mises à jour, la gestion des états et la synchronisation de l'état du cluster.
- **Gestion des services** : Ils définissent, déploient, et gèrent les services Docker dans le cluster.
- **Suivi de l'état du cluster** : Les nœuds managers surveillent l'état des autres nœuds et s'assurent que les services sont en bonne santé.
- **Vote pour la gestion du cluster** : Dans un environnement multi-manager, un consensus est utilisé pour assurer la cohérence du cluster (élection de leader, gestion des pannes, etc.).

Les nœuds manager peuvent être **réduits** à un nombre impair pour garantir une majorité dans les votes et éviter une situation de "égalité" en cas de panne de nœud.

- **Création d'un nœud manager** :
  Pour initialiser un cluster Swarm et désigner un manager, on utilise la commande suivante :

  ```
  docker swarm init
  ```

  Cette commande démarre un cluster Swarm avec le nœud courant comme manager principal.

- **Exemple de commande pour ajouter un manager supplémentaire** :
  Pour ajouter un autre nœud manager, récupérez le jeton de join du nœud manager initial et utilisez-le sur le nouveau nœud :

  ```
  docker swarm join --token <manager-join-token> <manager-ip>:
  <manager-port>
  ```

b. Nœud Worker

Les **nœuds workers** sont les nœuds qui exécutent les tâches, c'est-à-dire les conteneurs Docker associés aux services. Les workers reçoivent les instructions des nœuds manager sur les tâches à accomplir (comme le déploiement de conteneurs, la mise à l'échelle des services, etc.).

Les principales fonctions des nœuds **worker** incluent :

- **Exécution des tâches** : Les nœuds workers exécutent les conteneurs en fonction des plans reçus des nœuds managers.
- **Gestion des conteneurs** : Ils gèrent le cycle de vie des conteneurs (démarrage, arrêt, mise à jour).
- **Réplication des services** : Les workers répliquent les services qui ont été définis par les managers. Si un service est mis à l'échelle, les workers s'occupent de lancer ou de supprimer des instances de conteneurs.

Les nœuds workers ne peuvent pas modifier l'état global du cluster ni prendre des décisions sur le placement des services. Ils sont uniquement responsables de l'exécution des conteneurs.

- **Ajout d'un nœud worker** :
  Pour ajouter un nœud worker à un cluster Swarm existant, exécutez la commande suivante sur le nœud worker, avec le jeton fourni par le manager :

  ```
  docker swarm join --token <worker-join-token> <manager-ip>:
  <manager-port>
  ```

3. Exemple d'Architecture d'un Cluster Docker Swarm

Dans une architecture Docker Swarm typique, vous pouvez avoir plusieurs nœuds managers et workers. Voici un exemple simple :

- **Nœuds Managers** : Un ou plusieurs nœuds managers qui gèrent l'état global du cluster, la planification des services, la mise à l'échelle, et la gestion du routage.
- **Nœuds Workers** : Plusieurs nœuds workers qui exécutent les conteneurs et services attribués par les managers.

Exemple de configuration :

```
docker swarm init  # Initialise un manager
docker swarm join --token <manager-token> <manager-ip>:2377  # Ajoute un
autre manager
docker swarm join --token <worker-token> <manager-ip>:2377  # Ajoute un
worker
```

L'architecture finale pourrait ressembler à ceci :

- **Manager Node 1**
  - Gère l'état global du cluster.
  - Prend les décisions sur la distribution des services.
- **Manager Node 2**
  - Fournit une redondance pour la gestion du cluster.
- **Worker Node 1**
  - Exécute des services Docker.
- **Worker Node 2**
  - Exécute des services Docker.

4. Avantages de Docker Swarm

- **Haute disponibilité** : En cas de défaillance d'un manager, un autre manager prend le relais grâce au consensus entre nœuds managers.
- **Scalabilité** : Vous pouvez ajouter ou retirer des nœuds à la volée, et Swarm s'assurera que les services sont automatiquement mis à l'échelle.
- **Simplicité** : Swarm est simple à configurer, et sa configuration est native dans Docker, sans outils externes comme Kubernetes.
- **Load balancing intégré** : Swarm fournit un load balancing automatique pour les services, afin de distribuer le trafic réseau de manière équitable entre les conteneurs.

Résumé des Rôles dans Docker Swarm

- **Nœuds Manager** : Gèrent le cluster, planifient les tâches, et orchestrent l'exécution des services sur les nœuds workers. Ils prennent également des décisions de gestion de l'état et du routage du cluster.
- **Nœuds Worker** : Exécutent les services Docker et gèrent le cycle de vie des conteneurs. Ils sont contrôlés par les nœuds managers.

Docker Swarm permet ainsi de créer et de gérer des clusters de conteneurs de manière simple et efficace, en garantissant haute disponibilité, scalabilité et sécurité.