Commandes Docker

Voici une liste des commandes Docker les plus couramment utilisées et leurs fonctions.

Notez que cette liste n'est pas exhaustive, car Docker offre un large éventail de commandes pour diverses opérations, mais cela couvrira les plus fondamentales.

Gestion des Conteneurs

- **`docker run`** : Lance un conteneur à partir d'une image spécifiée. Vous pouvez utiliser diverses options avec cette commande pour contrôler le comportement du conteneur (comme `-d` pour le mode détaché, `-p` pour la publication des ports, etc.).

- **`docker ps`** : Liste tous les conteneurs en cours d'exécution. Utilisez l'option `-a` pour voir tous les conteneurs, y compris ceux qui sont arrêtés.

- **`docker stop`** : Arrête un ou plusieurs conteneurs en cours d'exécution.

- **`docker start`** : Démarre un ou plusieurs conteneurs arrêtés.

- **`docker restart`** : Redémarre un ou plusieurs conteneurs.

- **`docker rm`** : Supprime un ou plusieurs conteneurs.

- **`docker create`** : Crée un nouveau conteneur mais ne le démarre pas immédiatement.

- **`docker exec`** : Exécute une commande dans un conteneur en cours d'exécution.

- `docker logs` : Affiche les logs d'un conteneur.

- `docker pause` / `docker unpause` : Met en pause les processus dans un conteneur, et les reprend ensuite.

Gestion des Images

- `docker images` ou `docker image ls` : Liste les images Docker disponibles localement.

- `docker pull` : Télécharge une image ou un dépôt d'images depuis un registre.

- `docker push` : Envoie une image ou un dépôt vers un registre.

- `docker build` : Crée une image à partir d'un Dockerfile.

- `docker rmi` : Supprime une ou plusieurs images Docker.

- `docker history` : Montre l'historique des modifications d'une image.

Réseau et Stockage

- `docker network create` / `docker network rm` : Crée ou supprime des réseaux.

- `docker network ls` : Liste les réseaux Docker.

- **`docker volume create`** / **`docker volume rm`** : Crée ou supprime des volumes pour la persistance des données.

- **`docker volume ls`** : Liste les volumes Docker.

Docker Compose (outil distinct)

- **`docker-compose up`** : Démarre et exécute l'ensemble d'une application définie par un fichier `docker-compose.yml`.

- **`docker-compose down`** : Arrête et supprime les ressources spécifiées dans le fichier `docker-compose.yml`.

Informations et Configuration

- **`docker version`** : Affiche la version de Docker qui est installée.

- **`docker info`** : Affiche des informations sur le système Docker, y compris le nombre de conteneurs et d'images, les informations sur le réseau et le stockage.

- **`docker login`** : Permet de s'authentifier sur un registre Docker.

- **`docker system prune`** : Supprime les conteneurs, les réseaux, les images (non utilisées par au moins un conteneur), et le cache de build inutilisés.

Gestion des Stacks

- `docker stack ls` : Liste toutes les stacks déployées dans le Swarm. Une stack est un groupe de services qui sont liés entre eux et qui partagent des dépendances, définis dans un fichier `docker-compose.yml` ou équivalent.

- `docker stack deploy` : Déploie une nouvelle stack ou met à jour une stack existante. C'est la commande utilisée pour lancer une application conteneurisée sur un Swarm en utilisant un fichier de composition Docker Compose. Elle prend souvent l'option `-c` pour spécifier le fichier de composition.

- `docker stack rm` : Supprime une stack du Swarm. Cela arrête et supprime tous les services associés à la stack, mais conserve les volumes de données, à moins que des options spécifiques ne soient utilisées pour les supprimer également.

Gestion des Services dans une Stack

Ces commandes offrent une interface de haut niveau pour la gestion des applications distribuées sur un cluster Docker Swarm, simplifiant le déploiement et la maintenance d'applications conteneurisées à grande échelle.

Elles sont particulièrement utiles pour les opérations de déploiement continu et pour la gestion de l'infrastructure en tant que code, permettant aux développeurs et aux administrateurs système de décrire l'état souhaité de leur application dans des fichiers de composition et de laisser Docker s'occuper du déploiement et de la gestion.

- `docker stack services` : Affiche les services dans une stack. Cette commande fournit une vue d'ensemble des services qui composent la stack, y compris leur état, le nombre de répliques, et les ports exposés.

- `docker stack ps` : Liste les tâches (containers) d'une stack. Cela permet de voir le détail de chaque conteneur lancé par les services de la stack, y compris leur état et sur quel nœud

du Swarm ils sont déployés.

Inspection et Configuration

- **`docker stack inspect`** : Affiche des informations détaillées sur une stack, en format JSON. Cela inclut la configuration des services, les réseaux utilisés, et d'autres paramètres.

Gestion d'un cluster Swarm

Les commandes `docker swarm` *sont utilisées pour gérer et orchestrer un cluster Docker Swarm, qui est un groupe de machines Docker (nœuds) fonctionnant ensemble dans un mode de cluster.*

Docker Swarm fournit des fonctionnalités d'orchestration de conteneurs natives, permettant de déployer, mettre à l'échelle et gérer des applications conteneurisées sur plusieurs hôtes Docker.

Initialisation et Gestion du Swarm

- **`docker swarm init`** : Initialise un nouveau cluster Swarm sur l'hôte. Cette commande transforme la machine Docker sur laquelle elle est exécutée en un manager Swarm, ce qui lui permet de gérer et d'orchestrer le Swarm. Lors de l'initialisation, vous pouvez spécifier divers paramètres, comme l'adresse IP à utiliser pour la communication inter-nœuds.

- **`docker swarm join`** : Rejoint une machine au Swarm, soit en tant que nœud worker, soit en tant que manager supplémentaire, en fonction des tokens et des options spécifiés. Cette commande nécessite l'adresse d'un nœud manager existant et un token d'adhésion approprié.

- **`docker swarm leave`** : Fait quitter le Swarm à un nœud, le retirant ainsi de l'ensemble du cluster. Peut être utilisée sur des workers comme sur des managers, avec des considérations spéciales pour assurer que le Swarm continue de fonctionner correctement sans ce nœud.

- **`docker swarm update`** : Met à jour la configuration du Swarm. Cette commande permet de modifier des paramètres du Swarm, tels que la politique de certificat ou les paramètres de dispatcher.

Gestion et Sécurité des Nœuds

- **`docker node ls`** : Liste tous les nœuds dans le Swarm. Fournit des informations telles que l'ID, le nom, le rôle (manager ou worker), l'état, et la disponibilité.

- **`docker node inspect`** : Affiche des informations détaillées sur un ou plusieurs nœuds en format JSON. Utile pour obtenir des informations de configuration spécifiques ou l'état actuel d'un nœud.

- **`docker node rm`** : Supprime un nœud du Swarm. Il est important de noter que cette commande ne fait pas quitter le Swarm au nœud lui-même; elle doit être utilisée pour enlever un nœud qui a déjà quitté le Swarm ou qui est en panne.

- **`docker node update`** : Met à jour les paramètres d'un nœud. Cela peut inclure le changement de rôle d'un nœud (de worker à manager ou inversement) ou la modification de sa disponibilité.

Gestion des Tokens du Swarm

- **`docker swarm join-token`** : Gère les tokens utilisés pour rejoindre le Swarm. Vous pouvez créer de nouveaux tokens ou afficher les tokens existants pour les rôles de manager ou de worker, facilitant ainsi l'ajout sécurisé de nouveaux nœuds au cluster.

Sécurité et Configuration

- **`docker swarm ca`** : Affiche et met à jour le certificat d'autorité de certification (CA) du Swarm. Permet de gérer la façon dont les certificats sont émis et renouvelés au sein du Swarm pour sécuriser la communication inter-nœuds.

Promouvoir un Nœud Worker en Manager

- `docker node promote` : Cette commande est utilisée pour promouvoir un ou plusieurs nœuds workers à un rôle de manager dans le Swarm. Promouvoir un nœud en manager augmente le nombre de nœuds capables de gérer le cluster, ce qui peut améliorer la tolérance aux pannes et la disponibilité du cluster. La commande nécessite l'ID ou le nom du nœud (ou des nœuds) à promouvoir.

Rétrograder un Nœud Manager en Worker

- `docker node demote` : Utilisez cette commande pour rétrograder un ou plusieurs nœuds managers à un rôle de worker. Rétrograder un manager en worker peut être nécessaire pour réduire le nombre de managers et simplifier la gestion du cluster, ou lorsqu'un nœud manager n'est plus nécessaire ou ne doit plus avoir le rôle de gestion. Tout comme pour la promotion, cette commande prend l'ID ou le nom du nœud (ou des nœuds) à rétrograder.

Considérations Importantes

Équilibrage des Rôles : Lors de la promotion ou de la rétrogradation des nœuds, il est crucial de maintenir un équilibre entre le nombre de managers et de workers dans votre Swarm. Un nombre optimal de managers assure une bonne tolérance aux pannes sans introduire de complexité ou de latence de gestion inutiles.

Sécurité et Fiabilité : Les managers maintiennent l'état du cluster et prennent des décisions de gestion importantes. Il est donc essentiel de s'assurer que les nœuds promus en managers sont sécurisés et fiables.

Revision #2
Created 11 March 2024 13:26:41 by MASSON Romain
Updated 11 March 2024 14:13:46 by MASSON Romain