Infogérance et virtualisation

Commandes de base Docker

Hassen CHEFFI

Institut Supérieur des Études Technologiques de SFAX

October 19, 2024

Commandes de base Docker

Commandes de base Docker

Introduction à Docker

Docker est une plateforme open-source qui permet de créer, déployer et gérer des applications dans des conteneurs, offrant ainsi une solution efficace pour l'isolation et la portabilité des environnements.

- Isolation : Exécute chaque application dans son propre conteneur, avec ses dépendances.
- Portabilité : Fonctionne de manière identique sur différents environnements (développement, test, production).
- Légèreté : Les conteneurs partagent le noyau de l'hôte, ce qui les rend plus légers que les machines virtuelles.
- Rapidité : Les conteneurs démarrent rapidement, améliorant la productivité.
- Ecosystème : Intégration facile avec des outils comme Docker Compose et Docker Swarm pour l'orchestration.

Recherche d'Images Docker

- Rechercher une image sur Docker Hub

```
docker search <image>
```

- Exemple

```
$ docker search debian
NAME              DESCRIPTION                     STARS   OFFICIAL
debian            Debian is a Linux distrib...    5088    [OK]
jitesoft/debian   Debian base image.              0
ustclug/debian    Official Debian Image           2
```

Gestion des Images Docker

- Télécharger une image :

```
docker pull <image>
```

- Lister les images disponibles :

```
docker images
```

- Supprimer une image :

```
docker rmi <image>
```

Gestion des Conteneurs

- Créer et exécuter un conteneur :

```
docker run <image>
```

- Créer et exécuter un conteneur en arrière-plan (mode détaché) :

```
docker run -d <image>
```

- Créer un conteneur avec des ports exposés :

```
docker run -p <port_hôte>:<port_conteneur> <image>
```

- Attacher un volume à un conteneur :

```
docker run -v
    <nom_volume>:<chemin_dans_conteneur> <image>
```

Gestion des Conteneurs

- Lister les conteneurs en cours d'exécution :

```
docker ps
```

- Lister tous les conteneurs (y compris ceux arrêtés) :

```
docker ps -a
```

Gestion des Conteneurs

- Arrêter un conteneur :

```
docker stop <nom_conteneur>
```

- Démarrer un conteneur :

```
docker start <nom_conteneur>
```

- Supprimer un conteneur :

```
docker rm <nom_conteneur>
```

Docker Compose

Qu'est-ce que Docker Compose ?

Docker Compose est un outil permettant de définir et de gérer des applications multi-conteneurs. Il utilise un fichier YAML (docker-compose.yml) pour définir les services, réseaux et volumes nécessaires à une application.

Exemple de fichier docker-compose.yml

```yaml
version: '3'
services:
  web:
    image: httpd
    ports:
      - "8080:80"
    volumes:
      - ./html:/usr/share/nginx/html
    networks:
      - my-network
  db:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: example
    volumes:
      - db-data:/var/lib/mysql
    networks:
      - my-network
networks:
  my-network:
volumes:
  db-data:
```

Docker Compose

- Démarrer les services :

```
docker-compose up
```

- Optionnel : Ajouter l'option -d pour exécuter en mode détaché.

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

- Arrêter tous les conteneurs en cours d'exécution.

```
docker-compose down
```

- Redémarrer tous les services définis.

```
docker-compose restart
```

Docker Swarm

Pourquoi utiliser Docker Swarm ?

- Scalabilité : Déploie et gère des conteneurs sur plusieurs serveurs pour facilement augmenter ou réduire les ressources.
- Tolérance aux pannes : Redéploie les services automatiquement en cas de panne d'un nœud.
- Simplicité : Utilise la même syntaxe que Docker et Docker Compose pour une configuration facile.
- Réseautage : Crée un réseau virtuel pour connecter les conteneurs sur tous les nœuds du cluster.
- Load balancing : Répartit le trafic entre les conteneurs pour une performance optimale.
- Sécurité : Chiffre les communications et gère les certificats pour protéger le cluster.

Docker Swarm est idéal pour orchestrer des conteneurs à grande échelle avec une configuration simple.

Docker Swarm

- Initialiser un Swarm (sur le nœud principal) :

```
docker swarm init
```

- Ajouter un nœud à un Swarm (obtenu via le nœud principal) :

```
docker swarm join --token <token>
    <adresse_ip>:<port>
```

- Lister les nœuds du Swarm :

```
docker node ls
```

Docker Swarm

- Créer un service dans le Swarm :

```
docker service create --name <nom_service> <image>
```

- Lister les services dans le Swarm :

```
docker service ls
```

- Redimensionner un service (scaling) :

```
docker service scale
    <nom_service>=<nombre_réplicas>
```

- Supprimer un service :

```
docker service rm <nom_service>
```