Docker

Version	V0
Type	Exercice
Date de création	@24 juillet 2023 11:50
Dernière modification	@14 décembre 2023 16:25

🐳 Présentation

Docker et Docker-compose sont deux outils puissants et populaires qui facilitent le développement, le déploiement et la gestion d'applications. Docker est une plateforme open-source qui automatise le déploiement d'applications dans des conteneurs logiciels légers et portables.

Les conteneurs sont des environnements isolés qui incluent l'application et toutes ses dépendances, ce qui garantit une cohérence entre les différents environnements de développement, de test et de production. En utilisant Docker, vous pouvez créer une image conteneurisée de votre application une fois et la déployer sur n'importe quel environnement compatible Docker, sans souci de compatibilité.

Les avantages :

- Isolation : chaque application est isolée dans son propre conteneur, évitant les conflits de dépendances et de configurations.
- Portabilité : les conteneurs Docker peuvent être exécutés sur n'importe quelle plateforme compatible, qu'il s'agisse d'un ordinateur local, d'un serveur distant ou même dans le cloud.
- Léger et rapide : les conteneurs sont plus légers que les machines virtuelles traditionnelles, ce qui leur permet de démarrer rapidement et d'utiliser moins de ressources système.
- Versionnement et reproductibilité : les images Docker peuvent être versionnées, assurant ainsi la reproductibilité des déploiements.

🔧 Fonctionnement

Docker utilise un système client-serveur. Le Docker Daemon s'exécute sur l'hôte et gère les conteneurs. Le client Docker permet aux utilisateurs d'interagir avec le Daemon via une un CLI ou une interface graphique.

Les étapes clés pour utiliser Docker sont les suivantes :

1. Créer une image : Vous spécifiez les dépendances et configurations nécessaires pour votre application dans un fichier appelé Dockerfile. Ce fichier décrit les étapes pour créer une image

conteneurisée de votre application.

2. Construire l'image : Le Docker Daemon utilise le Dockerfile pour créer l'image conteneurisée en installant les dépendances et en configurant l'environnement.
3. Exécuter le conteneur : Une fois que l'image est créée, vous pouvez exécuter un conteneur basé sur cette image. Le conteneur est un environnement isolé et exécutable où votre application s'exécutera.

🤝 Docker-compose

Docker-compose est un outil qui simplifie la gestion de plusieurs conteneurs. Il vous permet de définir toutes vos applications, services, dépendances et configurations dans un fichier YAML, appelé docker-compose.yml. En exécutant une seule commande, vous pouvez lancer et arrêter tous vos conteneurs avec leurs configurations associées.

Les avantages :

- Gestion centralisée : Docker-compose facilite le déploiement et la gestion d'applications composées de plusieurs conteneurs, en les gérant ensemble de manière centralisée.
- Réseau et communication : Docker-compose crée automatiquement un réseau interne pour les conteneurs, leur permettant de communiquer facilement entre eux.

🏁 Pré-requis

Installer Docker sur votre machine

```
sudo apt update
sudo apt upgrade
sudo apt install docker.io
```

Vérifier que le service Docker est bien exécuté

```
sudo systemctl status docker
```

Par défaut, le binaire Docker nécessite des privilèges de super utilisateur (root) pour être exécuté. Cependant, en ajoutant votre utilisateur au groupe "docker", vous pouvez lui permettre d'utiliser Docker sans avoir à utiliser `sudo` à chaque fois

Pour vérifier si votre utilisateur appartient déjà au groupe "docker" :

```
whoami # pour connaitre votre num d'utilisateur
groups <nom_utilisateur> # liste les groupes auxquels appartient votre utilis
```

Si votre utilisateur n'appartient pas encore au groupe "docker" :

```
sudo usermod -aG docker <nom_utilisateur>
```

Pour vérifier que la configuration précédente est effective exécuter la commande suivante, elle ne contient pas `sudo`, si cela fonctionne la configuration est validée

```
docker-compose --version
```

🗃️ Préparation

Créer un dossier `exo_docker`

Les services exploités par docker utilisent les mêmes ports que ceux des services de votre machine (exemple MySQL utilise le port 3306 sur votre machine ainsi que dans docker)

Arrêter le service MySQL de votre machine

```
sudo service mysql stop # pour arrêter le service
sudo service mysql start # pour démarrer le service
```

- [ ] Créer un fichier `docker-compose.yml` à la racine du dossier `exo_docker`
- [ ] Initialiser un projet Nextjs à la racine du dossier `exo_docker`

Image en lecture seule

Pour commencer nous allons configurer un container pour MySQL

Pour créer un container nous allons avoir besoin d'une "image", dans le cas de MySQL cette image n'aura pas besoin d'être configuré puisqu'elle est livrée prête à l'emploi. On appelle ces images des "images en lecture seule" car elles ne nécessitent aucunes configurations (hormis celles présentent dans le fichier `docker-compose.yml`

Exemples d'images en lecture seule :

- MySQL
- Mailhog
- PHPMyAdmin

🔗 Hub des images Docker

🔗 Image MySQL officielle +1 milliard de DL

Ajouter la configuration suivante dans le fichier `docker-compose.yml`

```
version: '3.1'

services:

  db:
    image: mysql
    # NOTE: use of "mysql_native_password" is not recommended: https://dev.my
    # (this is just an example, not intended to be a production configuration
    command: --default-authentication-plugin=mysql_native_password
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: root
```

Dès lors vous pouvez exécuter vos containers depuis le terminal, à la racine du dossier `exo_docker` avec la commande suivante :

```
docker-compose up # pour démarrer les containers
docker-compose stop # pour arrêter les containers
docker-compose down # pour arrêter et supprimer les containers
```

Si l'exécuter du container c'est bien déroulé vous pouvez dorénavant accéder au container MySQL

```
docker-compose exec <nom_du_container> bash # accéder à un container bash
docker-compose exec <nom_du_container> sh # accéder à un container sh

docker-compose exec db bash # pour notre exemple
```

Vous voila dans le container MySQL que vous avez créé et démarré, celui-ci contient le logiciel de gestion de base de données SQL MySQL. Pour accéder au logiciel la démarche est la même que sur n'importe quelle machine

```
mysql -u <nom_utilisateur> -p # accéder à MySQL avec un utilisateur + mdp

mysql -u root -p # pour notre exemple
```

Il ne vous reste plus qu'à renseigner votre mot de passe (voir votre configuration docker-compose) et vous aurez accès au MySQL de votre container Docker

Image personnalisée

Pour finir nous allons configurer un container pour notre application Nextjs

Contrairement au container Docker, ce container se basera sur une image en lecture seule à laquelle nous ajouterons des configurations pour exploiter notre application Nextjs. À proprement paler l'image sera celle de Nodejs, qui est l'environnement nécessaire à l'exécuter du serveur de Nextjs, puis nous viendrons indiquer à l'image une suite de commandes à exécuter pour démarrer notre application.

🔗 Images Nodejs +1 milliard de DL

Ajouter la configuration suivante dans le fichier `docker-compose.yml`

```
version: '3.1'

services:

  db:
        ...

    next:
        build:
          context: next # nom du dossier de votre app Nextjs
        volumes:
          - ./next:/app # remplacer "next" par le nom du dossier de votre app
        ports:
          - "3000:3000"
```

Cette configuration n'indique rien à propos de l'image que nous souhaitons utiliser, cela se passera dans le fichier `Dockerfile` à la racine de notre projet NextJs. Cependant nous indiquons au `build` et au `volumes` le chemin vers notre projet NextJs

☐ Créer un fichier `Dockerfile` à la racine du dossier de votre application Nextjs

Ajouter le code suivant dans le fichier `Dockerfile` fraîchement créé

```
FROM node:18-alpine # l'image que l'on souhaite utiliser comme environnement

# Défini le working directory
WORKDIR /app

# Ajoute les path des CLI présents dans les node_modules à notre environnemen
ENV PATH /app/node_modules/.bin:$PATH
```

```
# Installe les librairies et exécuter le serveur de développement de NextJs
CMD yarn install && yarn install --dev && yarn build && yarn dev
```

Si vous souhaitez accéder au container Nextjs

```
docker-compose exec next bash
```