

? Utiliser Docker Compose

```
version: '3'

# On peut définir plusieurs conteneurs d'un seul coup en utilisant docker-compose
# Et si on modifie ce fichier, cela va supprimer les anciens docker pour les mettre à jour
services:
  db:
    # On définit l'image
    # Sinon on peut aussi utiliser build: <chemin> pour demander de build l'image par soi même
    image: mariadb:10.6.4-focal

    # On peut définir les arguments de la commande CMD
    command: '--default-authentication-plugin=mysql_native_password'

    # On peut définir des volumes, soit par bind : <chemin sur l'hôte>:<chemin sur le docker>
    #                               soit par nom : <nom du volume>:<chemin sur le docker> où
    dans quel cas cela sera géré par Docker
    # Ici on va lier /var/lib/mysql au volume du nom de "db_data"
    volumes:
      - db_data:/var/lib/mysql

    # On peut définir une condition pour que le conteneur redemarre automatiquement (par
    défaut: no)
    restart: always

    # On peut définir des variable d'environnement
    # Quand il y en a vraiment beaucoup et que l'on ne veut pas les mettre ici on peut les
    mettre dans un fichier à part et utiliser "env_file" à la place pour les importer (voir la
    doc pour plus d'info)
    environment:
      - MYSQL_ROOT_PASSWORD=somewordpress
      - MYSQL_DATABASE=wordpress
      - MYSQL_USER=wordpress
      - MYSQL_PASSWORD=wordpress

    # Ou encore des ports
```

```
# Les ports dans docker suivent la syntaxe HOST_PORT:CONTAINER_PORT
ports:
  - "3306:3306"
  - "33060:33060"

# Pareil ici mais avec un autre service du nom de "wordpress"
wordpress:
  image: wordpress:latest
  volumes:
    - wp_data:/var/www/html
  ports:
    - 80:80
  restart: always
  environment:
    - WORDPRESS_DB_HOST=db
    - WORDPRESS_DB_USER=wordpress
    - WORDPRESS_DB_PASSWORD=wordpress
    - WORDPRESS_DB_NAME=wordpress

# Enfin ici on peut définir les noms des volumes (seulement dans le cas par nom, pas
nécessaire dans les cas de bind)
volumes:
  db_data:
  wp_data:
```

Un docker compose comme celui ci, permet d'automatiquement déployer plusieurs conteneurs/services sur *une* machine très simplement. C'est très pratique pour expérimenté (si le fichier est mis à jour puis redémarré, cela va automatiquement remettre les conteneurs à jours)

Il suffit juste d'être dans un dossier où un fichier du nom de `docker-compose.yml` existe et de lancer la commande :

```
# Pour le lancer en mode debug :
sudo docker-compose up

# Pour le lancer pour de bon
sudo docker-compose up -d

# Pour le stopper
sudo docker-compose down
```

Ainsi maintenant on peut faire des configurations plus complexes beaucoup plus simplement qu'avec les lignes de commandes.

Pour en savoir plus sur la syntaxe des fichiers Docker Compose (version 3) vous pouvez cliquer [ici](#)

Revision #3

Created 26 April 2023 17:09:38 by SnowCode

Updated 1 May 2023 14:26:13 by SnowCode