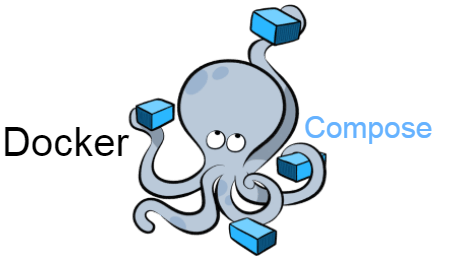
Vous avez un nouveau projet de site avec Wordpress. Pour **simplifier la gestion de l'infrastructure**, vous souhaitez déployer l'ensemble des composants dans des conteneurs Docker. Pour cela, nous allons avoir besoin de deux conteneurs :

* un conteneur **MySQL** ;
* un conteneur **Wordpress**.

Docker Compose va vous permettre d'orchestrer vos conteneurs, et ainsi de simplifier vos déploiements sur de multiples environnements. Docker Compose est un outil écrit en Python qui permet de décrire, dans un **fichier YAML**, plusieurs conteneurs comme un **ensemble de services**.



Docker Compose permet d'orchestrer plusieurs conteneurs

**Installation de Docker Compose**

Si vous avez utilisé Docker for Mac ou Docker for Windows, vous avez déjà la dernière version de Docker Compose installée dans votre système.

Sur un poste Linux, cela ne sera pas le cas. Vous devez donc le télécharger puis l'installer avec cette ligne de commande (un peu longue, oui !) :

**sudo curl -L "https://github.com/docker/compose/releases/download/1.23.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/bin/docker-compose && sudo chmod +x /usr/bin/docker-compose**

Vous pouvez vérifier la version de Docker Compose installée sur votre poste via :

> docker-compose --version

docker-compose version 1.23.2, build 1110ad01

**À la découverte du CLI de Docker Compose**

Pour utiliser le CLI (Command Line Interface) de Docker Compose, nous avons besoin d'un fichier **docker-compose.yml** que nous allons créer dans le prochain chapitre ! Mais avant de le créer, nous allons commencer par découvrir ensemble l'interface en ligne de commande (CLI) qui nous permet d'utiliser le fichier **docker-compose.yml**.

**Le CLI de Docker Compose et celui de Docker sont très proches**.

Par exemple, si vous souhaitez récupérer l'ensemble des images décrites dans votre fichier **docker-compose.yml** et les télécharger depuis le Docker Hub, vous devez faire un **docker-compose pull**. Du côté de Docker, la commande serait un **docker pull**.

Le fait que les deux interfaces en lignes de commandes soit très similaires nous évite d'apprendre un nouveau CLI. Si vous connaissez celui de Docker, vous savez globalement utiliser celui de Docker Compose !

Cependant, nous allons quand même voir ensemble **les principales commandes** que vous pourriez avoir besoin d'utiliser et de connaître pour votre utilisation de Docker Compose.

**Démarrer une stack Docker Compose**

Si vous souhaitez lancer la création de l'ensemble des conteneurs, vous devez lancer la commande **docker-compose up** (pour rappel, vous faites un **docker run** pour lancer un seul conteneur). Vous pouvez ajouter l’argument **-d** pour faire **tourner les conteneurs en tâche de fond**.

Nous appelons **stack** un ensemble de conteneurs Docker lancés via un seul et unique fichier Docker Compose.

**Voir le statut d'une stack Docker Compose**

Après avoir démarré une stack Docker Compose, vous aurez certainement besoin de voir si l'ensemble des conteneurs sont bien dans un état fonctionnel, et prêts à rendre un service.

Pour cela, vous allez utiliser la commande **docker-compose ps** qui vous affichera le retour suivant :

ADD CONTENT

**Voir les logs d'une stack Docker Compose**

Votre stack Docker Compose est maintenant fonctionnelle, et l'ensemble des services répondent bien ; mais vous pourriez avoir besoin de voir les **logs** de vos conteneurs.

Pour cela vous devez utiliser la commande **docker-compose logs -f --tail 5**

Celle-ci permet de voir l'ensemble des logs sur les différents conteneurs de façon continue, tout en limitant l'affichage aux 5 premières lignes.

Ainsi, si nos conteneurs fonctionnent depuis longtemps, nous n'aurons pas à attendre plusieurs secondes, ni voir de nombreux logs qui ne nous intéressent pas.

**Arrêter une stack Docker Compose**

Si vous souhaitez **arrêter** une stack Docker Compose, vous devez utiliser la commande **docker-compose stop**. Cependant, celle-ci ne supprimera pas les différentes ressources créées par votre stack.

Ainsi, si vous lancez à nouveau un **docker-compose up -d**, l'ensemble de votre stack sera tout de suite à nouveau fonctionnelle.

Si vous souhaitez supprimer l'ensemble de la stack Docker Compose, vous devez utiliser la commande **docker-compose down** qui **détruira l'ensemble des ressources** créées.

**Valider une stack Docker Compose**

Lors de l'écriture d'un fichier docker-compose, nous ne sommes pas à l’abri d'une **erreur**. Pour éviter au maximum cela, vous devez utiliser la commande **docker-compose config** qui vous permettra de **valider la syntaxe de votre fichier**, ainsi d'être certain de son bon fonctionnement.

Si nous créons une erreur dans notre stack, en remplaçant "image" par "images", par exemple, nous aurons le résultat suivant :

➜ docker-compose config

ERROR: The Compose file './docker-compose.yml' is invalid because:

Unsupported config option for services.db: 'images'

**Pour Résumer**

Vous connaissez maintenant les commandes principales pour utiliser une stack Docker Compose. Voici les commandes les plus importantes :

* **docker-compose up -d** vous permettra de **démarrer** l'ensemble des conteneurs en arrière-plan ;
* **docker-compose ps** vous permettra de voir le **status** de l'ensemble de votre stack ;
* **docker-compose logs -f --tail 5** vous permettra d'afficher les **logs** de votre stack ;
* **docker-compose stop** vous permettra d'**arrêter** l'ensemble des services d'une stack ;
* **docker-compose down** vous permettra de **détruire** l'ensemble des ressources d'une stack ;
* **docker-compose config** vous permettra de **valider** la syntaxe de votre fichier **docker-compose.yml**.

Mais passons sans plus attendre à un exemple concret ! Et créons notre première stack Docker Compose !