Télécharger et Lancer une Image

Avant de créer nos propres images, nous allons en chercher des existantes et les utiliser.

En tant que développeur web, ce qu'il nous faudrait comme outil, c'est bien entendu un serveur web.

Nous allons apprendre à télécharger une image, pour cela ouvrons n'importe quel invite de commande et tapons :

```
docker pull php
```

Ici nous allons retrouver les mots clefs :

- **docker** qui indique que l'on va lancer une commande docker.
- **pull** qui permet comme avec git de télécharger une image.
- **php** le nom de l'image que l'on veut télécharger.

Nous pouvons retrouver les différentes images téléchargeable sur :

https://hub.docker.com

Par défaut, PHP indique qu'il utilise le tag "latest", cela signifie qu'il télécharge la dernière version. Mais on peut souhaiter préciser la version que l'on souhaite.

```
docker pull php:8.2-apache
```

les ":" qui suivent le nom de l'image permettent d'indiquer un "**tag**", une étiquette permettant plus de précision.

On pourra vérifier quelles sont les images que l'on possède avec :

```
docker images
```

Supprimons l'image qui ne nous sert pas avec :

```
docker rmi php:latest
```

Nous avons donc maintenant une image de PHP accompagné d'un serveur apache, mais aucun conteneur l'utilisant. Pour créer un conteneur nous allons écrire :

```
docker run --rm -p 8080:80 php:8.2-apache
```

Plusieurs nouveautés ici :

- **run** indique que l'on va lancer un conteneur.
- **--rm** est une option qui supprime le conteneur une fois celui ci arrêté.
- **-p** permet d'indiquer le port sur lequel le conteneur sera accessible.
- Et enfin nous trouvons le nom de l'image que le conteneur doit copier.

Si vous tapez "**run**" sans avoir fait de "**pull**" au préalable, Docker s'occupera lui-même de **pull** l'image.

Ici nous indiquons pour le port "8080:80" En fait ce sont deux ports que l'on indique, le premier est celui de la machine virtuelle, celui qui nous permettra d'accéder à celle ci, et le second est le port du conteneur.

Pour résumer si nous allons sur localhost:8080 nous allons être redirigé vers le port 80 de notre conteneur. Ce qui actuellement entrainera une erreur 403 Forbidden car nous n'avons aucun fichier ici.

Arrêtons notre serveur, si notre invite de commande est occupée avec le serveur, un **ctrl+c** suffira. Sinon on utilisera :

```
# Pour voir les conteneurs actuellement actif :
docker ps
# Pour arrêter le conteneur :
docker stop nomDuConteneur
# Pour démarrer le conteneur :
docker start nomDuConteneur
# Pour supprimer le conteneur :
docker container rm nomDuConteneur
```

On notera l'option "**-a**" que l'on peut ajouter à "**docker ps**" pour lui demander d'afficher tous les conteneurs, même ceux éteints.

C'est bien beau tout cela mais on aimerait avoir nos propres fichiers dans ce conteneur.

Pour cela trois solutions s'offrent à nous.

La première est de créer notre propre **image** que l'on pourrait mettre en conteneur.

Cette solution serait parfaite pour mettre en production notre application, mais pour la développer, cela nous forcerait à créer une nouvelle image au moindre changement que l'on fait.

La seconde est de créer un **volume**. Un dossier à l'intérieur de la machine virtuelle qui sera accessible depuis plusieurs conteneur si on le souhaite. Là encore, les manipulations que l'on serait forcé à faire prendraient du temps pour voir le moindre changement.

La troisième est de monter un dossier (**bind mount**) avec notre conteneur. Monter un dossier va lier un dossier que nous possédons réellement sur notre ordinateur à un dossier de notre conteneur, permettant d'indiquer que lorsque ce conteneur souhaite atteindre ce dossier, c'est en fait celui sur notre ordinateur qui doit être utilisé.

Nous irons nous placer via l'invite de commande dans le dossier que l'on souhaiter "monter" (ici "*01-dev*"), puis :

```
docker run -d -p 8080:80 -v ${PWD}:/var/www/html --name exempleWeb php:8.2-apache
```

Ici plusieurs nouveautés s'offrent à nous :

- **-d** permet de ne pas lier notre terminal au conteneur.
- **-v** suivi de "*${PWD}*" (ou "*%cd%*" si vous utilisez windows CMD) qui indique le dossier où l'on se trouve actuellement, puis ":" et enfin le chemin vers le dossier à lier dans le conteneur.
- **--name** Permet d'indiquer le nom du conteneur, sans lui le nom est choisi au hasard.

Maintenant si nous nous rendons sur "**localhost:8080**" nous aurons accès à notre site.

Installer des extensions PHP

L'installation de base de PHP contient de nombreuses extensions, mais pas forcément toute celles que nous voulons utiliser. Pour en ajouter, nous allons nous connecter à l'invite de commande de notre conteneur.

```
docker exec -it nomDuConteneur bash
# ou bin/bash
```

- **exec** permet d'exécuter une commande dans un conteneur.
- **-it** indique de garder le conteneur ouvert.
- **bash** est la commande pour lancer l'invite de commande.

Une fois connecté à celui-ci, c'est une bonne pratique d'améliorer et mettre à jour le conteneur :

```
apt update && apt upgrade
```

La mise à jour terminé, nous pouvons ajouter nos extensions :

```
# Pour se connecter à la BDD :
docker-php-ext-install pdo_mysql
# pour générer des images avec php :
apt install -y libfreetype6-dev
docker-php-ext-configure gd --with-freetype=/usr/include/freetype2/
docker-php-ext-install gd
```

La prochaine étape sera de créer notre propre image.