R4.08 - TD 3
Stack Web + PHP + BDD

Introduction

Docker est un outil très présent dans le domaine du Web.

Une “stack” classique dans le Web est constituée d’un serveur Web et d’un moteur de BDD. On peut aussi y trouver d’autres outils/services tels qu’un reverse proxy, un mécanisme de cache ou encore des API de WebServices.

Le serveur Web est généralement **Apache** ou **nginx**[1]. Ca peut aussi être un serveur intégré à un autre outil, comme avec **NodeJS** par exemple.

Le moteur de BDD peut être SQL avec **PostgreSQL** ou **MySQL/MariaDB**, ou peut être du NoSQL tel que **MongoDB** ou encore **Redis**.

Le reverse proxy peut, par exemple, être aussi un **nginx** ou un **Traefik**. Le proxy sait souvent gérer la création et le renouvellement de certificats SSL, avec notamment **Let’s Encrypt** qui offre des certificats gratuits faciles à utiliser.

Le cache peut être géré par **Redis** aussi.

Dans ce TD, nous allons mettre en place une Stack **Apache + PHP + MySQL**.

> Comme tous les objets de l’environnement Docker sont liés à votre machine et pas à votre session, il est possible que vous ayez déjà tout ou partie du travail oublié d’un·e étudiant·e qui vous a précédé·e sur votre poste de travail. Vous ne gagnerez rien à conserver son travail et ça vous sera plutôt préjudiciable de ne pas opérer un nettoyage préalable.
>
> Donc, partout où il vous est demandé de créer un conteneur, un réseau ou un volume, commencez par exécuter une commande **`docker <target> rm <nom>`** où **`<target>`** est le type d’objet et **`<nom>`** est son nom. Par exemple, pour supprimer un volume nommé **`vol_td3_web`**, vous devrez faire un **`docker volume rm vol_td3_web`** et ainsi de suite pour chaque objet Docker qu’on vous demande de créer dans le sujet, avant de pouvoir poursuivre.
>
> Pensez à faire ce même nettoyage avant de quitter votre poste en fin de séance.

[1] Prononcer “Engine X”

Exercice 1 - Stack “à la main”

Comme pour l’exercice sur **Dockerfile**, nous allons commencer par faire les choses à la main et, dans un second temps, nous allons voir comment les automatiser en écrivant aussi une sorte de recette.

Serveur Web + PHP

Vous savez déjà, depuis le TD précédent, créer manuellement un conteneur avec un serveur Web **nginx**.

L’image **nginx** est celle d’un simple serveur Web, qui délivre des pages statiques, or il nous faut un serveur Web intégrant **PHP**.

Faire fonctionner **PHP** sur **nginx** est tout à fait possible mais, pour varier les plaisirs et vous faire découvrir d’autres outils, nous allons plutôt opter pour un serveur **Apache**. Il existe déjà une image **PHP** prête pour **Apache**.

L’image que nous allons utiliser est **`r408-php:8.2-apache`**. Vous noterez, comme son nom le présume, qu’il s’agit ici d’une image **PHP** qui est construite pour **Apache** (notez la présence du nom **apache** dans le tag) et non pas une image **Apache** contenant **PHP**. Ce détail est important pour savoir où se rendre sur le Docker Hub : allez donc voir du côté de **PHP** (et non de **Apache**).

> Cette démarche qui vous est demandée ici, celle d’aller sur le Docker Hub, correspond à la démarche naturelle que vous devez avoir en général. Cependant, comme vous le savez, à l’IUT les images issues du Docker Hub ne vous sont pas accessibles. A l’IUT nous avons un Hub privé et c’est celui-là que vous devez utiliser en prévoyant de lancer les **`docker image pull`** nécessaires pour chaque image que vous souhaitez utiliser.
>
> Cela dit, faites quand même la démarche d’aller sur le Docker Hub pour y consulter la documentation des images à utiliser.

Créez un conteneur avec les caractéristiques suivantes :

- Image **`r408-php:8.2-apache`**
- Attacher un réseau **`rezo_td3_web`** que vous aurez créé au préalable
- Router le port **`8050`** de votre machine vers le port **`80`** du conteneur
- Créez un volume **`vol_td3_web`** et montez-le sur **`/var/www/html`** dans le conteneur.

- Nommer le conteneur **td3_web** en ajoutant l'option **--name td3_web** sur la ligne **docker run** (avant le nom de l'image)

À l'aide de VSC, placez un fichier **index.php** dans **/var/www/html** avec ce contenu :

```
<?php
    phpinfo();
?>
```

et testez ce script en ouvrant votre navigateur sur **http://localhost:8050**

Exécution en mode "detached" (arrière-plan, daemon)

Une fois que vous avez testé que ça fonctionne bien, arrêtez votre conteneur, puis relancez la même commande en ajoutant une option **-d** (comme **detached**) n'importe où sur la ligne de commande, mais avant d'indiquer le nom de l'image.

Avec cette option **-d**, le conteneur va vivre sa vie en autonomie, vous n'avez plus les logs qui s'affichent et vous récupérez immédiatement la main dans le terminal.

Pour afficher les logs, vous devez utiliser la commande :

```
docker container logs -f ID_du_conteneur
```

Pour interrompre l'affichage sans fin des logs obtenus avec cette commande, faites simplement un **CTRL**+**C**

Serveur BDD MySQL

L'image MySQL que nous allons utiliser est **r408-mysql:5.7**

Cette image offre la possibilité de créer une BDD vierge et un utilisateur (et son mot de passe associé), dès la création du conteneur. Pour ce faire, il faut passer en paramètres quelques variables d'environnement grâce à des options **-e** sur la ligne de commande **docker run**.

Les 4 variables d'environnement à renseigner sont :

- **MYSQL_ROOT_PASSWORD** : le mot de passe de l'administrateur de la BDD
- **MYSQL_DATABASE** : le nom de la BDD vierge à créer au 1er lancement
- **MYSQL_USER** : le nom de l'utilisateur gestionnaire de la base créée (cf **MYSQL_DATABASE**)
- **MYSQL_PASSWORD** : le mot de passe de l'utilisateur gestionnaire (cf **MYSQL_USER**)

Voici un exemple raccourci, à compléter, pour passer en paramètres ces variables d'environnement :

```
docker run -e MYSQL_USER=toto -e MYSQL_PASSWORD=lulu...
```

Créez maintenant un conteneur avec les caractéristiques suivantes :

- Image **`r408-mysql:5.7`**
- Attachez au même réseau **`rezo_td3_web`**
- Lancez en mode **`detached`**
- Indiquez que la BDD vierge à créer s'appelle **`td3`**
- Indiquez **`toto`** comme nom à créer pour le gestionnaire de la BDD et que son mot de passe associé est **`lulu`**
- Choisissez un mot de passe pour l'administrateur et indiquez-le dans la variable **`MYSQL_ROOT_PASSWORD`**
- Nommez le conteneur **`td3_db`** sur la ligne **`docker run`**

PHPMyAdmin

Pour administrer votre BDD, vous avez besoin d'un client **PHP**.

Pour cela il existe **PHPMyAdmin**.

On pourrait l'installer sur la machine hôte mais restons dans l'esprit Docker. Cet outil fonctionnant comme un site Web, il est aussi possible de le dockeriser, et d'ailleurs il en existe déjà une image sur le Docker Hub. On va donc créer un conteneur pour lui !

Pour information, **PHPMyAdmin** utilise lui-même **PHP** (c'est écrit dans son nom) et requiert donc un serveur Web pour l'héberger. L'image que nous allons utiliser intègre donc aussi un serveur Web en plus du logiciel **PHPMyAdmin** ! Et tout ceci est packagé dans un conteneur. Au final, vous aurez donc deux conteneurs avec un serveur **Web** + **PHP**, mais chacun sera isolé dans son conteneur, un pour votre site et un pour PHPMyAdmin.

Créez un conteneur avec les caractéristiques suivantes :

- Image **`r408-phpmyadmin:5.2`**
- Attachez encore au même réseau **`rezo_td3_web`**
- Router le port **`8060`** de votre machine vers le port **`80`** du conteneur
- Lancez en mode **`detached`**
- Indiquez le nom du conteneur qui héberge la BDD en passant en option du **`docker run`**, une variable d'environnement qui s'appelle **`PMA_HOST`** et en lui affectant le nom du conteneur, qui doit être **`td3_db`** (qui doit être le nom passé au **`--name`** lors de la création du conteneur BDD)
- Nommez le conteneur **`td3_pma`** sur la ligne **`docker run`**

Testez votre interface d'administration sur **`http://localhost:8060`**

Exercice 2 - Une petite application

Maintenant que vous avez tout préparé, vous allez créer une petite page Web avec sa BDD associée.

> Comme votre BDD sera stockée dans un volume (**`td3_db`** en principe) et que les volumes restent physiquement sur la machine hôte et ne vous suivent pas dans votre profil d'une séance à l'autre, prenez soin de sauvegarder le script de création de votre BDD avant la fin de la séance. Idéalement faites-le à chaque modification de votre BDD. Il y a un onglet **Export** dans l'interface de PHPMyAdmin prévu à cet usage.
>
> Cet export vous servira pour redémarrer rapidement à la prochaine séance.

Étape 1

Via l'interface de PHPMyAdmin, créez une BDD avec la table **`articles`** suivante :

- **`id`** : entier auto incrément (sert de clé primaire)
- **`nom`** : chaîne de 40 caractères
- **`prix`** : réel
- **`tva`** : réel (taux de TVA)

Remplissez votre table **`articles`** avec quelques enregistrements.

Étape 2

À l'aide de VSC, créez une page **`articles.php`**, dans le dossier **`/var/www/html`** de votre conteneur Web. Ce script doit accéder à la BDD MySQL avec **PDO** et doit afficher les articles.

À ce stade, vous devriez rencontrer un problème avec **PDO**. Il devrait vous manquer le driver **PDO** de MySQL dans PHP.

Nous allons corriger ce problème, à l'étape suivante, par la création d'une image personnalisée, comme nous avons appris à le faire au TD précédent, avec un **`Dockerfile`**.

Étape 3

Comme vous avez appris à le faire en TD 1, créez un **`Dockerfile`** avec les caractéristiques suivantes :

- Image de base : **`r408-php:8.2-apache`**
- Un seul ajout au **`Dockerfile`** : **`RUN docker-php-ext-install pdo pdo_mysql`**

Construisez votre nouvelle image en la nommant (avec un **`-t`**, allez voir vos notes du TD 2) : **`r408-php:8.2-apache-pdo`**

C'est cette image que vous allez maintenant utiliser à la place de **`r408-php:8.2-apache`**. Arrêtez le conteneur **`td3_web`** et relancez-en un nouveau, à l'identique (voir début de ce TD) mais sur la base de cette nouvelle image.

Testez maintenant que l'**Étape 2** de cet exercice fonctionne mieux (sous réserve de mise au point de votre script bien sûr !). Vous ne devez plus avoir d'erreur liée à **PDO+MySQL**.

Exercice 3 - docker-compose.yml

Comme pour le **`Dockerfile`**, nous allons voir ensemble comment créer un fichier **`docker-compose.yml`** qui va permettre d'automatiser la création de ces trois conteneurs.