Migration environnement LAMP vers Docker

Commande pour se connecter au conteneur :

docker exec -it (nomduconteneur) /bin/bash

❖ Prérequis

Avoir des conteneurs apache avec php d'installé (attention de bien avoir la bonne version de php) et des conteneurs mysql/mariadb (même remarque)

❖ Vérification du bon fonctionnement des conteneurs docker (voir documentation Connection aux conteneurs)

❖ Faire un DUMB des bases de données

- Conversion des bases de données en fichier .sql.

Nous allons utiliser mysqldumb pour exporter les bases de données vers le conteneur mysql en docker.

```
mysqldump -u <utilisateur> -p <nom_de_la_base> > <fichier_de_sauvegarde>.sql
```

Cette commande sert à convertir la bdd en fichier sql.

- Conversion du dossier ou se situe tous les fichiers .sql en en dossier compressé.

Nous allons utiliser la commande tar pour convertir ce dossier.

```
root@srvweb1804:/home/nathan# tar -czvf dump.tar.gz dump
```

Migration environnement LAMP vers Docker

- ## Transfère du fichier vers le serveur.

Nous allons utiliser la commande scp pour transférer tout d'abord le fichier depuis le serveur vers notre machine local, puis de notre machine locale vers notre serveur docker.

La commande :

```
scp -i C:\Users\n.levavasseur\Documents\id_rsa root@172.16.66.43:/home/nathan/dump.tar.gz C:\Users\n.levavasseur\Documents\
```

Avec le -i la clé privée qui doit être convertie en format .rsa et non .ppk l'user root@172.16.66.43 : le dossier compressé et en fin le chemin où se situe la destination du transfère.

- ## Conversion du dossier compressé en décompressé.

Nous utiliserons la commande tar pour décompresser notre dossier qui est en .tar.gz.

```
root@srvwebpproddock2204:/home# tar -xzvf dump.tar.gz
```

Et normalement le dossier est maintenant décompressé.

- ## Importation des bases de données via le dossier dump.

Nous allons utiliser la commande mysql pour restaurer la base de données.

```
mysql -u root -p BlogGamacFr < bloggamacfr.sql
```

Le -p est le nom de la base de données ou le fichier sera injecté (il faut qu'elle soit créée avant cette commande), le bloggamacfr.sql correspond au fichier où se situe les données de la bdd (en dump).

Normalement la commande s'exécute et si l'on va vérifier le contenu de la bdd en question, les données sont présentes.

Migration environnement LAMP vers Docker

❖ Copie de l'ensemble des fichiers (configuration…)

- Copie des fichiers principaux.

Il faut copier tous les fichiers de conf qui se trouvent dans /etc/apache2/sites-available/ vers le même répertoire mais dans le serveur apache docker.

Même chose pour les pages dans /var/www/html.

Ainsi que pour les dépendances php, les logs etc :
/usr/lib/apache2 /etc/ssl /etc/php /usr/lib/php /var/lib/php

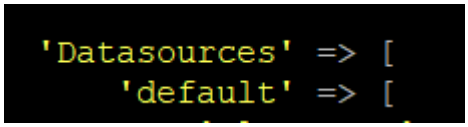
❖ Les droits d'accès

- Vérification des fichiers conf apache.

Il faut vérifier que les répertoires /var/www/ , /var/log/apache2/ /etc/apache2/ aient les bons propriétaires et droits c'est-à-dire www-data:www-data et droits d'écriture, tout ceci dans le conteneur.

- Modification des fichiers d'accès à la base de données pour les projets (les sites cakephp / wordpress).

Il faut modifier un fichier du projet pour permettre l'accès à la base de données, si c'est un cakephp alors le fichier est cheminduprojet/config/app.php ou bien /app_local.php et modifier la section



Si c'est un wordpress alors cela se situe dans ce répertoire /wp-config.app à la ligne suivante :

```
define( 'DB_HOST',
```

Dans les deux il y a normalement un paramètre ‘host’ ou il faut y attribuer le nom de notre « serveur » mysql défini dans le docker-compose.yml de mysql

```
version: '3.1'

services:
  db:
```

Ici db

Cependant il faut aussi modifier les droits d’accès dans le serveur mysql, avec les informations qui sont dans la suites du fichier de conf, il faut récupérer l’user, le mdp et la database et on tape ensuite cette commande dans mysql/mariadb :

GRANT ALL PRIVILEGES ON database.* TO 'user'@'192.168.144.3' IDENTIFIED BY 'mdp';

FLUSH PRIVILEGES;

(Ce qui est en rouge est à modifier avec les infos du fichiers app.php)

Et normalement le projet peut maintenant récupérer les informations dans la bdd.

❖ Test du bon fonctionnement des sites

- ### Vérification du bon fonctionnement d’apache :

On utilise la commande docker logs « nom-du-conteneur »

Normalement cela ne retourne aucune erreur (si erreur il y a, voir fin de la documentation dans la partie problèmes possibles)
On peut également aller dans le conteneur (docker exec -it « nom-du-conteneur » /bin/bash) et aller voir les logs dans /var/log/apache2/error.log ou access.log ce qui affichera les logs de chaque site en temps réel.

- Modification du fichier hosts de Windows pour simuler une résolution DNS sur notre serveur web en docker vers les url des sites.

Il faut avoir les droits d'administrateur sur son pc pour pouvoir exécuter l'invite de commandes en administrateurs, une fois cela fait il suffit d'aller dans le répertoire C:\Windows\System32\drivers\etc\hosts

On modifie ensuite le fichier et on ajoute les noms DNS de nos sites comme www.exemple.fr et on met à côté notre ip.

```
192.168.1.50 exemple.fr
```

On peut maintenant essayer notre site exemple.fr dans notre url et cela redirigera directement vers notre serveur avec notre conteneur docker et non pas le serveur apache que l'on a migré car le fichier hosts a la priorité sur le DNS résolveur.

❖ Erreurs possibles

- Problèmes d'injection du dump dans le mysql (docker).

Il est possible d'avoir un problème lorsque l'on injecte le fichier .sql du dump dans mysql notamment lors du redémarrage du conteneur, celui-ci peut ne pas garder en mémoire les bdd ainsi que les tables et leur contenu.

Cela est dû à l'absence d'un volume dans le docker-compose.yml, en effet il faut ajouter un volume qui pointera directement vers le dossier

```
      - /home/mysql/dump:/docker-entrypoint-initdb.d
```

Il faudra également ajouter nos fichiers .sql dans le répertoire /home/mysql/dump et lorsque le conteneur démarrera cela permettra d'exécuter automatiquement les scripts .sql dans le conteur mysql.

- ## Problèmes d'extensions php.

Il est possible que lorsque l'on exécute la commande docker logs (« nom de conteneur apache ») une multitude d'erreurs php apparaissent avec des extensions comme intl, gd ...

Ceci est un problème relativement mineur car soit il suffit d'exécuter la commande : php -m | grep nom_extension et cela si elle est installée affichera l'extension.

Si l'erreur d'extension persiste alors il faut ajouter l'extension dans le dockerfile pour qu'elle soit installée directement avec l'image.

Et si l'erreur est toujours présente alors il faut changer de version de php vers quelque chose de plus récent.

- ## Problème de variable manquante iconv :

Il se peut qu'une erreur avec l'extension php iconv apparaisse et cela même si elle est présente dans le conteneur, cela est dû à l'absence de cette variable dans le composer, il suffit de taper la commande suivante dans le conteneur au répertoire du projet cake.php :

```
composer require symfony/polyfill-iconv
```

Cela va installer le module et l'erreur devrait disparaître.

Une autre solution plus « propre » consiste passer à une version de php plus récente.

- ## Problème de la variable JsonSerializable manquante dans le projet cakephp.

Cela est dû à un souci de version de php qui ne reconnait pas la variable, ce souci peut être réglée en passant à une version de php >= 7.4.

Migration environnement LAMP vers Docker

- ## Problème de changement de version de php.

Il est possible que durant la modification de php vers une version plus récente cela ne fonctionne pas et cela pour une raison, c'est qu'il y a une différence entre la version de php CLI (ligne de commande) et la version de php qu'utilise apache, on peut voir ces versions avec ces méthodes : php -v pour php CLI et pour php apache il faut créer un fichier info.php avec le code suivant à l'intérieur :
< ?php

phpinfo()

?>

Ensuite il faut l'afficher sur un navigateur et on la version de php s'affiche en haut de page :

Cependant, il se peut que les versions diffèrent, cela est dû au fait que lorsque l'on a monté le répertoire :/etc/php dans le docker-compose.yml les fichiers ajoutés à l'intérieur ont supprimé la conf de base de php à savoir celle du Dockerfile, il faut donc commenter la ligne du volume en question, on recrée le conteneur et cela devrait fonctionner (! il faut bien que toutes les extensions nécessaires aux projets soient présentes dans le Dockerfile !).

- ## Problème de droits d'accès.

Il se peut que des soucis de permissions s'affichent en haut de page, cela est dû à l'impossibilité pour l'utilisateur apache d'ouvrir les pages web, il suffit de taper régulièrement la commande chown -R www-data:www-data html/ , cette commande est à exécuter dès lors qu'un fichier est créé avec n'importe quel autre utilisateur que www-data:www-data (root...).

- Autres soucis.

S'il y a d'autres soucis, de nombreux outils existent, comme docker logs (nomduconteneur) ou bien directement le dossier /var/log/ avec les logs de tous les outils/services installés.

❖ Commandes pratiques

- docker exec -it « namecontainer » /bin/bash
- docker logs « namecontainer »
- apachectl -S
- apachectl restart
- mysqldumb -u … -p databases > databases.sql
- mysql -u … -p … databases < databases.sql
- php -m
- php -v
- chmod +x