**TP DOCKERCOMPOSE**

Dans ce TP nous allons voir comment configurer le déploiement de nos deux services, Nginx et Php, en un seul fichier. Cela a pour but de faciliter le déploiement de votre solution. Egalement le fait d'utiliser un fichier de configuration à appliquer au lieux de démarrer vos containers en ligne de commande vous permettra de plus facilement garder une trace de vos manipulations afin de débugger plus facilement si quelque chose se passe mal ou alors afin de modifier facilement votre solution. Pour ce TP vous aurez besoin de rechercher les éléments clefs des fichiers par vous-même. Servez-vous de google pour rechercher directement ce que vous voulez ou bien regarder des dépôts github de solution docker et docker-compose pour essayer de trouver des exemples et informations.

**I/Mise en place**

Pour faire ce TP vous devez avoir votre arborescence complète :

* /home/<votre\_utilisateur>/nginx/code/intra
* /home/<votre\_utilisateur>/nginx/code/wiki
* /home/<votre\_utilisateur>/nginx/conf

Vous aurez également besoin de modifier les fichiers de configuration de vos sites pour qu'ils acceptent le php. Rajoutez cette section dans chaque fichier en remplaçant <chemin\_du\_code> par le répertoire de code de chaque site :

location ~ \.php$ {  
 root <chemin\_du\_code>;  
 try\_files $uri =404;  
 fastcgi\_split\_path\_info ^(.+\.php)(/.+)$;  
 fastcgi\_pass php:9000;  
 fastcgi\_index index.php;  
 include fastcgi\_params;  
 fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;  
 fastcgi\_param PATH\_INFO $fastcgi\_path\_info;  
 }

Dans les dossiers intra et wiki mettez un fichier index.php et inscrivez y dedans le code suivant :

<?php  
 echo phpinfo()  
?>

**II/Rédaction du compose**

Dans votre répertoire personnel créez le fichier docker-compose.yml. Ce fichier va décrire le démarrage de tous les services composants notre solution. Pour chaque service nous allons indiquer le nom du service, le nom de l'image à utiliser, le nom du container à générer ainsi que les volumes et les ports à attribuer.

Nous aurons deux services : un service web comprenant la configuration du container nginx et le service php servant à l'interprétation des fichiers php.

Votre fichier docker-compose.yml commence par l'indication de la **version** du fichier yaml. Nous indiquerons ici '3.8'.   
Nous indiquons ensuite, par le mot clef **services**, que nous allons déclarer nos services.   
Puis nous déclarons notre premier service que nous appelons **web**.  
Nous indiquons ensuite que l'image à utiliser est en fait à compiler depuis le dossier ./nginx/.  
Nous déclarons ensuite deux volumes :  
 - le volume local ./nginx/code/intra lié au repertoire /intra du container  
 - le volume local ./nginx/code/wiki lié au repertoire /wiki du container  
Nous exposons ensuite les ports du container en les attribuant aux ports de la vm  
 - le port 80 de la vm au port 80 du container  
 - le port 8080 de la vm au port 8080 du container  
Nous indiquons ensuite que le service est lié au service php

Nous déclarons ensuite notre service php.  
Nous indiquons qu'il utilise l'image php:8-fpm  
Nous déclarons ensuite les deux volumes :  
 - le volume local ./nginx/code/intra lié au repertoire /intra du container  
 - le volume local ./nginx/code/wiki lié au repertoire /wiki du container

Une fois votre fichier compose rédigé vous pouvez, dans le dossier qui contiens docker-compose.yml, exécuter la commande docker-compose up -d --build

Si tout se passe bien vos deux container se lancent et vous pouvez accéder à la page d'informations de votre serveur php aux deux adresses de vos sites web.

**III/Ajout d'une base de donées**

Nous allons maintenant rajouter un service de base de données à notre docker-compose.

Rajoutez sur le fichier un service que vous appellerez db dont l'image est mariadb.

Definnisez-y des variables d'environnement :

* MARIADB\_ROOT\_PASSWORD: secret
* MARIADB\_DATABASE: project
* MARIADB\_PASSWORD: password

Avec cette configuration votre container base de données est prêt à l'utilisation. Votre container php ne l'est pas par contre. En effet si vous regardez sur la page d'informations de votre serveur php vous remarquerez dans la section pdo que le support pour mysql n'est pas assuré. Nous allons donc créer une image php pour notre projet sur laquelle nous installerons ce qu'il faut pour pouvoir interagir avec notre base de données.

Créez dans votre répertoire personnel, au même niveau que votre répertoire, un dossier php. Rédigez dedans un Dockerfile qui se base sur une image php:8-fpm. Indiquez dans le dockerfile de lancer la commande docker-php-ext-install pdo\_mysql qui va exécuter un script fournit par l'image de base permettant d'installer des modules. Finissez par indiquer que le répertoire de travail est /var/www.

Retournez maintenant dans votre fichier docker-compose.yml et indiquez dans le service php de build l'image présente dans le dossier php au lieu d'utiliser l'image php:8-fpm.

Vous pouvez maintenant re lancer la commande docker-compose up -d --build . Docker va redéployer vos containers qui ont été modifiés. Vous pouvez maintenant rédiger du code php pour l'un de vos sites afin qu'il exploite la base de donnée. Afin d'accéder à la base il faudra indiquer db comme nom d'hôte du serveur de base de donnée, étant donné que nous avons indiqué ce nom dans le service de notre docker-compose.