Docker – TP
Virtualisation par containers

Objectif

Le but de ce TP est de se familiariser à l'utilisation de containers pour le développement de vos applications Web. L'utilisation des containers, va permettre par la suite, de pouvoir déployer vos applications dans les mêmes conditions d'exécution utilisées pendant la phase de développement.

Machine virtuelle à configurer

Utiliser la VM VirtualBox nommée ubuntu-tli

- Dossier sous Linux : sync/VMs ; si non disponible, la récupérer sur le serveur softwares/sync/VMs/
- OS : Ubuntu 64 bits
- RAM : 2 Go au minimum
- Augmenter la taille de la mémoire vidéo
- Login (root) : tp ; mot de passe : tp
- Activer le presse-papier partagé (bidirectionnel)
- Si l'affichage de la VM est en faible résolution, installer les Additions Invité (drivers) : Menu *Périphériques* puis *Insérer l'image CD des Additions Invité...* L'image d'un CD va alors être chargée et les drivers installés. Redémarrer si nécessaire la machine virtuelle et changer la résolution de l'affichage.

1. Création de votre environnement de travail

Dans cette étape vous allez installer un environnement docker à partir de votre image de base ubuntu-tli.

Exercice

Vous allez installer docker en suivant la guide officielle sur le site :
https://docs.docker.com/install/linux/docker-ce/ubuntu/#install-using-the-repository

Voici les pas à suivre :

1. Ajouter les clés GPG de docker:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

2. Ajouter le dépôt docker:

```
$ sudo add-apt-repository \
  "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) \
  stable"
```

3. Installer docker:

```
$ sudo apt-get update
$ sudo apt-get install docker-ce
```

4. Par défaut seul l'utilisateur root a le droit d'exécuter docker, si vous voulez éviter d'utiliser la commande sudo à chaque appel de docker, vous devez donner les permissions pour l'exécution de containers à votre utilisateur :

```
$ sudo usermod -aG docker tp
```

5. Redémarrer la machine virtuelle pour mettre à jour les permissions pour votre utilisateur et

tester que docker est bien installé en exécutant le container de test « hello-world » :

```
$ docker run hello-world
```

6. Configurer docker pour être démarré avec le démarrage de la machine :

```
$ sudo systemctl enable docker
```

A la fin de cette étape l'utilisateur « tp » est en mesure d'exécuter docker dans la machine virtuelle.

2. Configurer un serveur apache avec php dans un container

Dans cette étape vous allez exécuter une instance apache avec php activé. Pour tester le bon fonctionnement de votre container vous allez utiliser un fichier index.php pour afficher les informations de votre container.

Exercice

1. Créez un dossier où vous allez garder la configuration pour le container :

```
$ mkdir mon-serveur-php
```

2. Créez un dossier où vous allez stocker les fichiers de votre site web, il aura le rôle du dossier /var/www/html que vous utilisiez habituellement dans apache :

```
$ mkdir www
```

3. Dans le dossier web, créez et éditez un fichier index.php qui aura le contenu suivant :

```
<?php phpinfo; ?>
```

Cette fonction va montrer l'information sur l'environnement PHP qui est configuré dans le container.

4. Dans l'écosystème docker, plusieurs images sont disponibles pour être utilisées à partir du registre public *docker hub*. Pour php, plusieurs configurations sont possibles, vous pouvez regarder sur le site https://hub.docker.com/_/php/ une liste des différentes versions et configurations disponibles. Dans ce TP nous allons utiliser l'image de php en version 7.0 et qui vient avec apache déjà intégré, php:7.0-apache :

```
$ docker run -d -p 80:80 –-name serveur-php \
        -v /home/tp/mon-serveur-php/www:/var/www/html php:7.0-apache
```

5. Allez regarder dans la documentation de docker et expliquer les fonctions de chaque partie de la commande docker run :

```
docker : _____
run : _____
-d : _____
-p 80:80 : _____
-v /home/tp/mon-serveur-php/www:/var/www/html: _____
php:7.0-apache:_____
```

6. Vérifiez dans le navigateur de la machine virtuelle que vous avez accès aux informations de la configuration php du container, allez sur le site http://localhost.
7. Démarrez un deuxième container avec la version 5.6 de php sur le port 81. Vérifiez que les informations de la nouvelle configuration sont bien affichées dans le navigateur.
8. Pour voir les containers en cours d'exécution, la commande est :

```
docker ps
```

Cette commande va montrer les noms et id des containers.
Les containers peuvent être arrêtés ou démarrés avec les mêmes paramètres d'initialisation avec les commandes :

```
docker stop [id ou nom]
docker start [id ou nom]
```

9. Les containers ont leur propre environnement et système d'exploitation séparés de la machine hôte, dans le cas de l'image php, c'est une instance linux debian qui est en exécution et avec laquelle on peut interagir et exécuter des commandes telles que apt-get.

Pour vous connecter en root, tapez la commande suivante :

```
$ docker exec -it [id ou nom] bash
```

Vous pouvez maintenant installer des applications, copier des fichiers, ou regarder les logs. Cependant tout ce que vous faites dans le container sera perdu une fois il est arrêté.

À la fin de cette étape vous devriez avoir deux instances du serveur apache, une avec la version php 7.0 sur le port 80 et une autre avec la version php 5.6 sur le port 81. En utilisant les containers de cette façon vous pourriez tester votre code avec différentes versions de php sans besoin de faire une installation complète d'un autre serveur ou machine virtuelle.

3. Personnaliser un serveur pour le développement dans un container

Les images disponibles dans docker hub n'ont pas toujours les modules que l'on voudrait utiliser dans l'environnement d'un site web.
Cependant, on peut les enrichir et construire de nouvelles images en spécifiant les modifications à apporter à l'image de base. On peut se baser par exemple sur l'image ubuntu et décrire un processus de configuration pour un service en spécial ou ajouter des modules à une image plus complète comme celle de php.
Dans cette étape on va rajouter des modules à l'image de base php pour permettre d'utiliser mysql en pdo.

Exercice

1. Dans un des dossiers créés à l'étape précédente (pas celui www), ajoutez et éditez un fichier nommé Dockerfile avec le contenu suivant :

```
FROM php:7-apache

# Installer des logiciel à l'aide d'apt-get
RUN apt-get update && apt-get install -y libpng-dev curl libcurl4-openssl-dev

# Ajoute l'extension pdo_mysql avec les outils docker
RUN docker-php-ext-install pdo_mysql gd

# Activer le module rewrite dans apache
RUN a2enmod rewrite
RUN service apache2 restart

# Donner un nom au serveur
RUN echo "ServerName localhost" >> /etc/apache2/apache2.conf

# Pour intégrer le dossier de votre site web directement dans l'image
COPY www/ /var/www/html

# Si on a une configuration particulière pour php.ini, on peut l'intégrer aussi
# directement dans l'image, le fichier php.ini doit être créé avant
# COPY php.ini /usr/local/etc/php/
```

   Ce fichier est un exemple de début pour un fichier plus complexe dans lequel vous pourriez ajouter plus tard un framework php et d'autres fichiers externes. Vous pouvez regarder la documentation officielle docker pour la liste complète de paramètres : https://docs.docker.com/engine/reference/builder/

2. Créez un fichier Dockerfile selon vos préférences de développement, quels modules serez pertinents pour votre projet ?

3. Pour créer la nouvelle image, utilisez la commande docker build -t [nom de votre image] depuis le dossier ou vous avez créé le Dockerfile :

```
$docker build -t mon-php-perso .
```

4. Démarrez le nouveau container :

```
$ docker run -d -p 82:80 –-name mon-serveur-php-perso mon-php-perso
```

5. Vérifiez que les nouveaux modules sont activés : https://localhost:82

4. Configurer un serveur mysql dans un container

Pour compléter votre environnement de travail vous aurez besoin d'une base de données et avec docker rien de plus facile que d'exécuter l'image mysql, elle est disponible dans https://hub.docker.com/_/mysql/.

Exercice

1. Allez sur le site docker hub pour la base de données myslq et démarrez un container avec les bons paramètres. N'oubliez pas d'associer le port d'écoute 3306 pour mysql et d'initialiser le mot de passe.

5. Configurer phpmyadmin dans un container

Créer un container docker dans lequel phpmyadmin sera configuré pour se connecter à votre base de données mysql. La bonne image est sur le docker hub :
https://hub.docker.com/r/phpmyadmin/phpmyadmin/.

Exercice

1. Allez sur le site docker hub pour phpmyadmin et démarrez un container avec les bons paramètres. N'oubliez pas d'associer un port d'écoute différent à 80 car il est déjà associé à votre container de développement apache.
   Évitez d'utiliser le paramètre « --link » car il n'est plus recommandé, utilisez la variable d'environnement PMA_HOST pour ajouter le nom de votre serveur mysql.
2. Créez une nouvelle base de données et ajouter quelques entrées.
3. Créez une page web php, dans votre container de développement, capable d'afficher des données à partir de la nouvelle base de données.

En cas de pannes

- Vérifier toujours que vous avez les permissions pour utiliser les fichiers, au cas où l'utilisateur n'a pas les permissions, vous pouvez toujours changer les permissions à l'aide de la commande :

```
$ sudo chown [-R] $(utilisateur):$(groupe) $(fichier ou dossier)
```

Ou

```
$ sudo chmod [-R] $(permissions) $(fichier ou dossier)
```

- Si un container est démarré avec les mauvais paramètres, vous pouvez l'arrêter et l'effacer avec les commandes :

```
$ docker stop [id ou nom container]

$ docker rm [id ou nom container]
```