

Installation d'un service WEB sous docker.



Installation de la solution DOCKER et utilisation des conteneurs pour un premier environnement de tests.

I. Introduction.

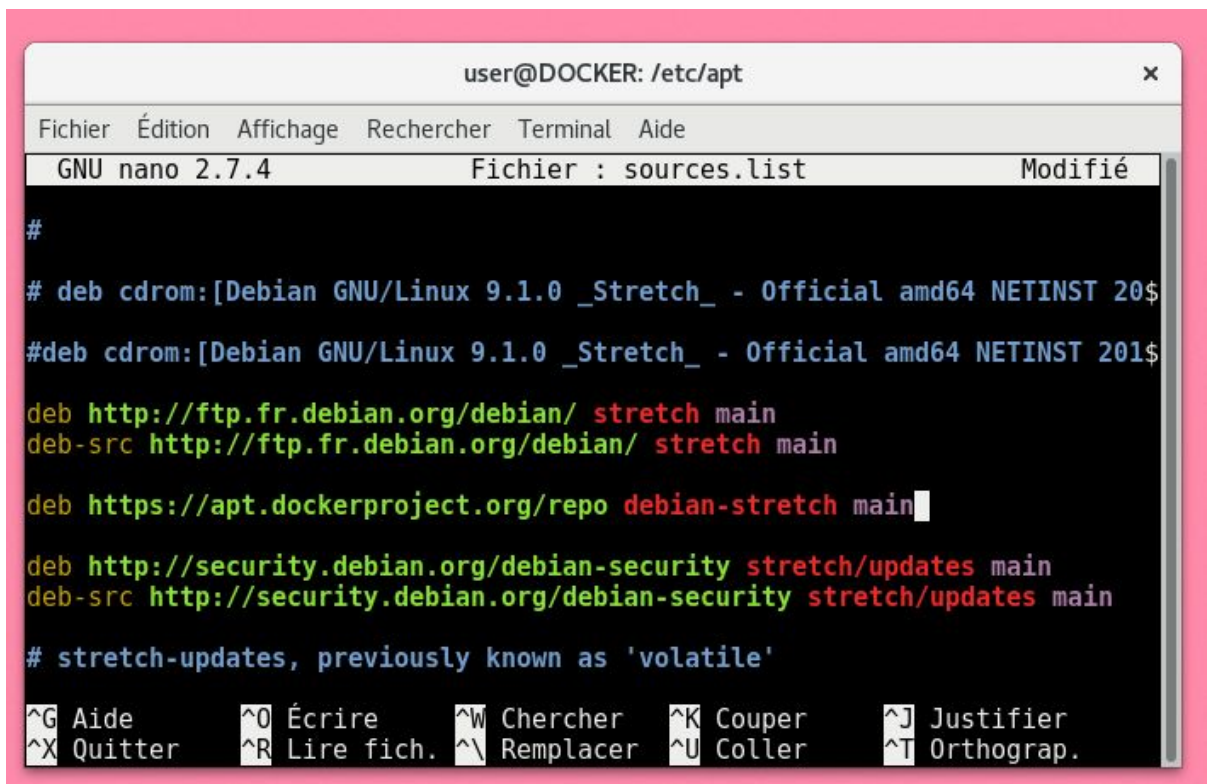
Nous allons mettre en place un environnement de test sous docker, composé d'un service web, d'une base de donnée et d'un environnement PHP. Une application à été créé spécialement pour l'occasion.

Cette application est un simple page d'authentification en PHP, avec une liaison vers une base de donnée contenant les couples d'identifiants valides.

I. Installation

Dans un premier temps, nous ajoutons le dépôt officiel docker dans notre fichier /etc/apt/sources.list et nous récupérons la clé publique correspondante.

Pour mettre à jour la liste des paquets : apt-get update.



```
user@DOCKER: /etc/apt
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
GNU nano 2.7.4      Fichier : sources.list      Modifié

#
# deb cdrom:[Debian GNU/Linux 9.1.0 _Stretch_ - Official amd64 NETINST 20$
#deb cdrom:[Debian GNU/Linux 9.1.0 _Stretch_ - Official amd64 NETINST 201$
deb http://ftp.fr.debian.org/debian/ stretch main
deb-src http://ftp.fr.debian.org/debian/ stretch main
deb https://apt.dockerproject.org/repo debian-stretch main
deb http://security.debian.org/debian-security stretch/updates main
deb-src http://security.debian.org/debian-security stretch/updates main
# stretch-updates, previously known as 'volatile'

^G Aide      ^O Écrire    ^W Chercher  ^K Couper    ^J Justifier
^X Quitter   ^R Lire fich.^N Remplacer ^U Coller    ^T Orthograp.
```

Nous installons ensuite les paquets nécessaires, à savoir : “apt-transport-https” et “docker-engine”.

Pour tester que DOCKER à bien été installé sur le poste, il est possible d’effectuer un simple `docker --version` ou un `docker --help`. Si la version est retournée ou que l’aide s’affiche correctement, DOCKER est installé.

```
root@DOCKER:/# docker -v
Docker version 17.05.0-ce, build 89658be
root@DOCKER:/#
```

Nous sommes maintenant prêts pour chercher des images dockers correspondants à nos besoins.

Pour rechercher un container DOCKER, les commandes `docker search <nom>` et `docker pull <nom>` sont utilisées. En recherchant un paquet par mot clé, le nom complet du container nous est retourné, tout comme sa popularité et la description de son contenu.

Une fois les containers correspondants à phpmyadmin, apache et mysql trouvés, nous les installons avec la commande “`docker install <nom>`”

```
root@DOCKER:/# docker search apache
```

NAME	STARS	OFFICIAL	AUTOMATED	DESCRIPTION
tomcat				Apache Tomcat is an open source implementa
...	1477	[OK]		
httpd				The Apache HTTP Server Project
	1237	[OK]		

Pour vérifier la liste des containers installés via DOCKER : `docker image ls`.

```
root@DOCKER:/# docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED
eboraas/apache	latest	4d05a7265505	23 hours ago
phpmyadmin/phpmyadmin	latest	3405a88394e3	13 days ago
mysql	latest	c73c7527c03a	6 weeks ago

Exécutons maintenant les différents containers dans leurs environnement respectifs.

Pour exécuter le container apache en tâche de fond :

```
docker run -v /var/www/html/testlogin: /var/www/html/testlogin -d -p 80:80 -p 443:443
eboraas/apache
```

L’argument “-v” est suivis du chemin relatif jusqu’aux dossiers de la machine hôte concerné.

L’argument “-d” indique que le container tourne en tant que tâche de fond.

L’argument “-p” est suivis du port machine puis du port container relatif.

Enfin, la commande se termine par le nom du container à lancer.

Pour MySQL :

```
docker run -d --name mysql -e MYSQL_ROOT_PASSWORD:0000 mysql
```

L’argument “-p” est suivis du port machine puis du port container relatif.

L’argument “-d” indique que le container tourne en tant que tâche de fond.

L’argument “--name” permet de nommer le container pour éviter de se retrouver avec des noms par défauts pour tous nos containers (et donc de ne pas savoir explicitement lesquels démarrer/arrêter/stopper)

L’argument “-e” est suivit du mot de passe MySQL admin. Celui-ci sera

Pour PhpMyAdmin :

```
docker run -d -p 8080:80 --link mysql:db --name phpmyadmin phpmyadmin/phpmyadmin
```

L'argument "-p" est suivis du port machine puis du port container relatif.

L'argument "-d" indique que le container tourne en tant que tâche de fond.

L'argument "--link" est suivi du nom de container à lier à ce service. Ici, le container MySQL.

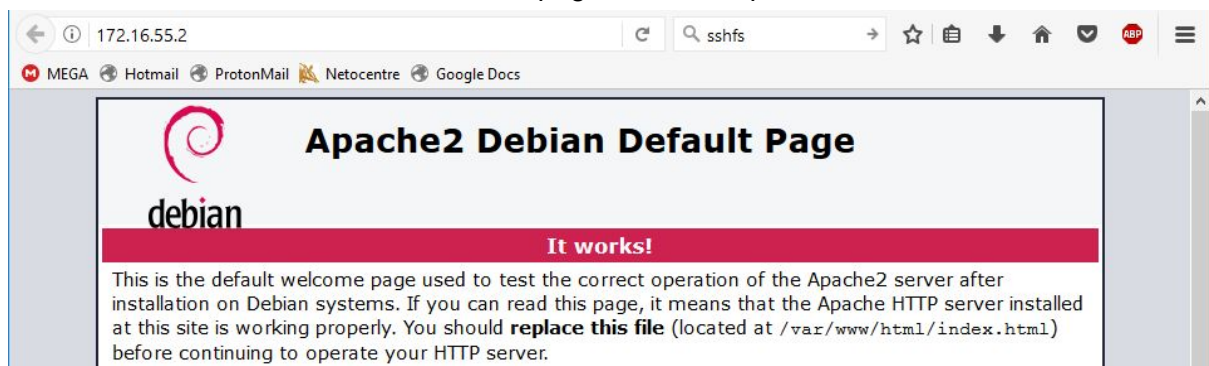
Pour entrer dans un conteneur, la commande est "*docker exec -it *id du container**".

Une autre commande très utile : La commande "*docker ps -a*", qui retourne la liste des containers en action, ainsi que les informations listés ci-dessus.

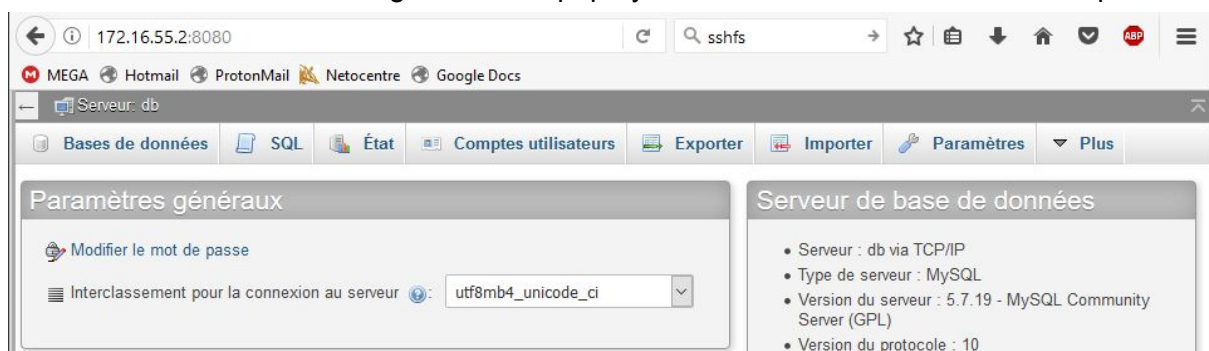
```
root@DOCKER:/home/user# docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        NAMES
STATUS        PORTS
8bd26d7fdb3    phpmyadmin/phpmyadmin  "/run.sh phpmyadmin"    4 seconds ago  myadmin
Up 3 seconds   0.0.0.0:8080->80/tcp
55dc90fd3ae1   mysql          "docker-entrypoint..." 41 seconds ago mysql
Up 41 seconds  3306/tcp
4087372bd9d0   eboraas/apache  "/usr/sbin/apache2..." 6 days ago    apache
Up 24 minutes  0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp
```

Pour tester si les trois containers sont biens configurés, il suffit de naviguer avec un poste client sur le même réseau jusqu'à l'ip de la machine hébergeant docker.

Ici, 172.16.55.2:80 nous retourne bien la page d'accueil apache :



Et 172.16.55.2:8080 nous dirige bien vers phpmyadmin et la base de donnée créée plus tôt :



Nous sommes désormais en mesure de sauvegarder nos images pour les redéployer plus tard.

Les commandes de sauvegarde d'images est la suivante pour les trois containers :

```
docker commit apache apache_bak
```

```
docker commit mysql mysql_bak
```

```
docker commit myadmin phpmyadmin_bak
```

```
root@DOCKER:/home/user# docker commit apache apache_bak
sha256:41de597aca09c67f0a352e68edb591b1d4042d9328a692f0c252caaf9eddb20a
root@DOCKER:/home/user# docker commit myadmin phpmyadmin_bak
sha256:a41b22a6abba20b55ec4b87624aec40794eec685916be7c8130621779a251ae0
```

III. Ajout de l'application

Après avoir récupéré les fichiers de l'application, les avoirs dé-compressés et placés sous `/var/www/html`, avec la commande `"docker exec -it *name* bash -c ' cat > /var/www/html/testlogin.zip ' < /testlogin.zip"`, nous pouvons entrer dans le container avec la commande `"docker exec -it *id du container* bash"`. Le prompt du terminal viens de changer, nous précisant désormais que nous nous trouvons dans le container.

En se déplaçant jusqu'au répertoire, nous vérifions que les fichiers sont bien présents dans le répertoire en question. Nous en profitons pour éditer le fichier connexion.php, en y ajoutant l'adresse IP du container mysql, et le bon login et mot de passe correspondant à la base de donnée.

Pour trouver l'IP du container mysql, il suffit d'entrer dedans et d'installer le paquet relatif à ifconfig, soit `"apt-get install iputils-tool"`.

Nous obtenons 172.17.0.3 pour le container mysql et 172.17.0.2 pour le container pache.

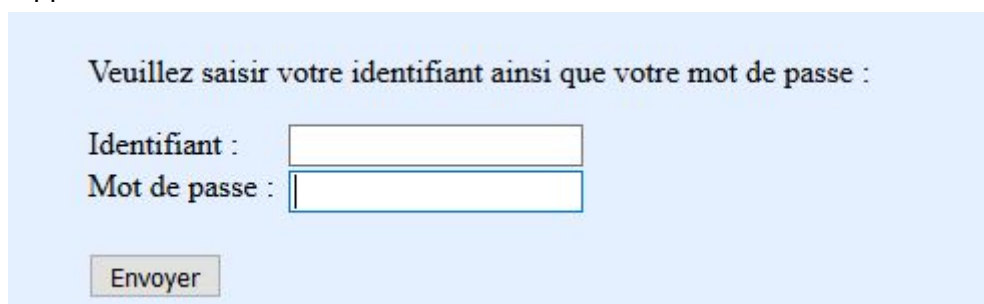
```
<?PHP
//Connexion la base de donn e
try
{
    $bdd = new PDO('mysql:host=172.17.0.1;dbname=motdepasse;charset=utf8', '$
}
catch (Exception $e) // Si erreur
```

Toujours dans le container apache, les paquets relatifs à l'interprétation php sont à installer, soit `php-common libapache2-mod-php php-cli` ainsi que `php7.0-mysql` pour le PDO.

Une fois les paquets installés, il faudra redémarrer le service apache2 avec `"service apache2 restart"`.

Nous allons être déconnectés du container après avoir redémarré le service.

Si tout s'est bien déroulé, nous avons maintenant accès au formulaire d'identification de l'application.



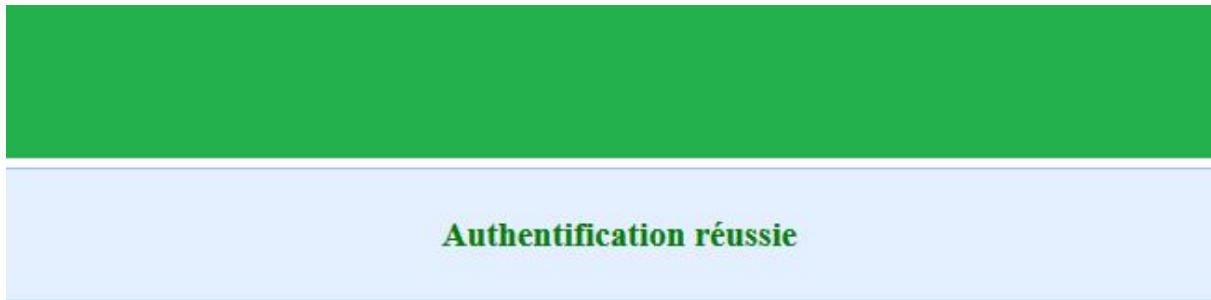
Veuillez saisir votre identifiant ainsi que votre mot de passe :

Identifiant :

Mot de passe :

Enfin, il ne reste plus qu'à créer la base de donnée `"motdepasse"`, via l'interface graphique, en se connectant à l'adresse 172.16.55.2:8080. Nous importons directement la base avec le fichier SQL intégré dans l'application.

Le formulaire de connexion fonctionne désormais correctement, et nous retourne les informations de logins :



Ayant utilisé l'argument `-v` en lançant le container apache, nous pouvons directement exporter les fichiers de l'application dans la machine hôte, sous `/var/www/html/testlogin`.

Nous continuons en exportant les images de nos trois containers :

```
CONTAINER ID   STATUS    IMAGE                                  COMMAND                  CREATED
eb0133e43736   Exited (0) 2 minutes ago                mysql                    "docker-entrypoint..." About an ho
417c9c296253   Exited (137) 2 minutes ago                eboraas/apache          "/usr/sbin/apache2..." 20 hours ag
8bd26d7fdb3    Exited (0) 2 minutes ago                phpmyadmin/phpmyadmin   "/run.sh phpmyadmin"     23 hours ag
root@DOCKER:/home/user# docker commit eb0133e43736 mysql_1
sha256:dac4b625e090276fd46a413956849bbe9ea8db9b574ace30ffc03c7a342fa31d
root@DOCKER:/home/user# docker commit 417c9c296253 apache_1
sha256:dff61da21bac98c72bb9b6080b0c735ca9d99bf065262330d8fa4e65c2bd45f1
root@DOCKER:/home/user# docker commit 8bd26d7fdb3 phpmyadmin_1
sha256:cff17595b3187e88aceb71d7d35f1557ff947ce1447b92fbb3cca845a1e06b72
root@DOCKER:/home/user#
```

Par la suite, il faudra stopper les containers, les supprimer, et les ré-importer via les images :

- `docker run -v /var/www/html -d -p 80:80 -p 443:443 --name apache apache_ok`
- `docker run -d --name mysql -e MYSQL_ROOT_PASSWORD:toor mysql_ok`
- `docker run -d -p 8080:80 --link mysql:db --name phpmyadmin myadmin_ok`

Finalement, en vérifiant dans notre navigateur, nous obtenons bien l'application fonctionnel. l'argument `-v` fonctionne en fait comme un lien logique entre le conteneur et la machine hôte