TP - Installation d'un serveur LAMP

Généralités

LAMP signifie Linux Apache MySQL PHP.

Nous allons installer l'ensemble des composants nécessaires à la création d'un serveur Web. Les différents paquets sont interchangeables mais cette combinaison est la plus courante.

Linux sera le système d'exploitation (près de 96% des serveurs de la planète l'utilise !).

Apache va être le logiciel qui va répondre aux requêtes HTTP. Son travail en version très résumée consiste à distribuer des fichiers quand on lui demande (exemple : je tape google.fr, Apache me renvoi le fichier index.html du site). Une autre version consiste à dire que c'est un explorateur de fichier mais qui fonctionne via requête HTTP.

MySQL sera notre base de donnée. Nous utiliserons très exactement MariaDB qui est la version communautaire et open source de MySQL. Il existe bien d'autres systèmes de base de données mais MySQL reste un des plus répandu.

PHP va nous permettre d'interpréter du code php afin de rendre les sites web beaucoup plus intéressant. En utilisant uniquement de l'HTML on ne va pas bien loin ... Apache en lui même n'a pas pour vocation d’exécuter du PHP ou n'importe quel autre langage d'ailleurs, il faut donc ajouter un autre paquet pour réaliser cette tâche.

Linux

Nous travaillerons sous une distribution Linux **Debian 9**. Sous Debian les versions majeures portent un nom. Ainsi la version 8 s'appelle Jessie, la 9 Stretch et la 10 Buster.

Une machine vous ait mis à disposition. Téléchargez et installez Putty . Sélectionnez **SSH**, dans le champs IP tapez l'IP qui vous a été attribuée et connectez vous. Si tout va bien, vous devez voir apparaître un message (une vague histoire de clé), cliquez sur "Oui". Un écran noir affiche alors "login". Tapez "root", puis à "password" tapez "epsi123456". Première chose à faire : changer son mot de passe pour éviter qu'un de vos camarades se connecte par erreur à votre machine et pour

sécuriser l'installation. Pour ce faire, tapez "passwd" puis rentrer le nouveau mot de passe (noter vous le quelque part, nous en aurons besoin jusqu'à la fin du module !).

Nous allons maintenant utiliser "**apt**" qui est le gestionnaire de paquet sous Debian. Son rôle est de gérer les différents paquets (les logiciels si vous voulez) sur votre système : installation, désinstallation, gestion de mise à jour ... Le scénario d'utilisation est complètement différent de ce que vous connaissez sur Windows ou MacOS et s'apparente beaucoup plus à ce qu'on croise sur mobile. En effet, apt va interroger ce qu'on appelle des **dépôts** (que l'on va configurer) pour savoir quels paquets ils proposent et dans quelles version (ce qui ressemble plutôt à un store sur smartphone). apt s'appuie sur un fichier tampon qui sert se registre. Il stock dedans le listing des paquets disponibles sur les différents dépôts enregistrés, mais aussi la liste des paquets que vous avez installé et leurs versions. Il faut donc mettre à jour ce fichier avant toute opération pour avoir un listing à jour. Cette action d'effectue via la commande :

`apt update`

Un certain nombre de ligne se met à défiler sur l'écran, apt interroge un à un les dépôts enregistrés. Ces dépôts sont configurés via le fichier /etc/apt/sources.list et tous les fichiers éventuellement présents dans /etc/apt/sources.list.d

Ces fichiers sont de simples fichiers textes répondant à une syntaxe précise demandée par apt. Ainsi le fichier /etc/apt/sources.list contient par défaut :

```
deb http://ftp.debian.org/debian stretch main contrib

deb http://ftp.debian.org/debian stretch-updates main contrib

deb http://security.debian.org stretch/updates main contrib
```

La syntaxe se décompose en deb URL distribution composant1 composant2

Plus d'infos ici

Nous pouvons maintenant installer apache via la commande :

`apt install apache2`

> Note : il est possible de faire du "**piping**" et de cumuler des commandes. Nous aurions par exemple plus utiliser :
>
> `apt update && apt install apache2`
>
> Le && permet d’exécuter une commande si la précédente a fonctionné. On peut également faire du piping pour utiliser la sortie d'une commande dans une

> autre. Par exemple la commande ps aux permet de lister les processus en cours. La commande **grep** permet de son côté de faire un filtre par mot clés. Ainsi si je veux voir uniquement les processus d'apache, il me suffit de taper :
>
> `ps aux | grep apache2`
>
> Cette fois c'est l'opérateur | qui est utilisé.

Voyons voir maintenant si Apache est fonctionnel. La commande *service* permet d'exécuter des commandes de base sur des processus. La syntaxe est simple :

`service nomduservice action`

action peut-être : start, stop, restart, reload ou status. Il existe d'autres commandes mais nous ne les verrons pas (et elles ne servent pas souvent ...).

start : lance le service.

stop : l'arrête.

restart : équivalent de stop + start.

reload : recharge la configuration sans arrêter le processus.

status : retourne l'état du processus.

Pour en revenir à Apache, on tape :

`service apache2 status`

Si tout est ok vous devez lire dans les premières lignes :

`Active: active (running)`

Si ce n'est pas le cas, tapez :

`service apache2 start`

Puis revérifier l'état du processus.

Apache2 lors de la première installation fourni une configuration de test. Lancez votre navigateur web favori et tapez XXX.mydil.lan/ XXX étant le numéro de votre machine. Une page web comme celle-ci devrait apparaître :

Résultat de recherche d'images pour "default debian apache page"

Image not found or type unknown

win GIF **Victory !**

Comment configurer Apache ?
Image not found or type unknown

Le fonctionnement d'Apache dans son concept est très simple : à une URL donnée correspond un emplacement dans les dossier de ma machine. Pour la configuration de base, Apache répond à n'importe quelle requête HTTP et pointe dans le dossier **/var/www/html.** Ainsi si vous vous rendez dans le dossier /var/www/html vous trouverez un fichier index.html correspondant à la page de test qu'il vous a affiché.

La configuration des sites servis par Apache se trouve dans deux dossiers : **/etc/apache2/sites-available** et **/etc/apache2/sites-enabled**. Les bonnes pratiques veulent que les fichiers de configuration se trouvent dans sites-available et qu'on crée des liens symboliques (des genres de raccourci) dans le dossier sites-enabled pour les activer. Ainsi si je veux désactiver un site il me suffit de supprimer son lien symbolique, et le fichier de configuration est conservé (au cas où on voudrait le réactiver plus tard).

Ainsi sur une installation fraîche, si je vais dans /etc/apache2/sites-available je vais trouver un fichier 000-default.conf et un fichier ssl-default.conf, et dans le dossier /etc/apache2/sites-enabled un lien symbolique vers 000-default.conf (par contre pas de lien vers ssl-default.conf, qui n'est donc pas actif !). En tapant la commande suivante, je vais donc aller voir à quoi ressemble ce fichier de configuration :

```
nano /etc/apache2/sites-available/000-default.conf
```

Voici ce qu'on y trouve :

```
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        # ServerName example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html
```

```
        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host
                #Include conf-available/serve-cgi-bin.conf
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Toutes les lignes commençant par un # sont des commentaires. Nous allons d'abord nous intéresser aux balises *<VirtualHost *:80>* et *</VirtualHost>* : elles permettent de créer un **host** (un site quoi) virtuel. A l'interieur de ces balises nous allons mettre ces différents paramètres. Nous allons surtout nous attarder sur deux d'entre eux :

- **ServerName** : c'est l'URL sur laquelle va répondre notre host.
- **DocumentRoot** : c'est l'emplacement sur notre machine qui va être "servi" par ce host.

Attention : dans la suite de ce tutorial, je vais utiliser MA machine qui porte l'ID

Je vais par exemple créer deux hosts sur ma machine :

- coucou.100.mydil.lan va servir le dossier /var/www/coucou
- test.100.mydil.lan va servir le dossier /var/www/html (la page de test d'origine quoi !)

```
<VirtualHost *:80>
ServerName coucou.100.mydil.lan
DocumentRoot /var/www/coucou
</VirtualHost>

<VirtualHost *:80>
ServerName test.mydil.lan
DocumentRoot /var/www/html
```

```
</VirtualHost>
```

La configuration est modifiée, il faut que Apache la prenne en compte :

```
service apache2 reload
```

Si une erreur aparaît c'est que vous avez fait une erreur : **service apache2 status** vous affichera quelques infos à ce sujet.

Apache est bien configuré, il faut maintenant créer les contenus. Et oui, autant /var/www/html existe déjà, autant le dossier /var/www/coucou ne contient rien, mais mieux encore, il n'existe pas ! Résolvons ce premier problème :

```
mkdir /var/www/html/coucou
```

Créons maintenant notre index.hmtl

```
nano /var/www/coucou/index.html
```

Et tapons une page bateau :

```
<h1> Coucou ! </h1>
```

Si tout va bien, nous pouvons maintenant nous rendre sur nos deux URLs (coucou.100.mydil.lan et test.100.mydil.lan) et constater le résultat.

PHP

Nous allons installer PHP7.3 en mod de Apache, c'est à dire comme un plugin d'Apache. Il est possible d'installer PHP au travers de PHP-FPM qui est un paquet séparé d'Apache.

Installation des paquets additionnels pour apt

apt ne gérant pas nativement l'https, et les dépôts pour php7.3 étant en https, il nous faut installer des paquets supplémentaires :

```
apt install ca-certificates apt-transport-https
```

Ajouter les sources

On ajoute tout d'abord la signature du dépôt pour augmenter la sécurité (apt vérifiera si les paquets comportent bien la même signature, ce qui évite certains types de piratage).

```
wget -q https://packages.sury.org/php/apt.gpg -O- | sudo apt-key add -
```

On va ensuite ajouter le dépôt.

nano /etc/apt/sources.list.d/php.list

Et ajouter :

```
deb https://packages.sury.org/php/ stretch main
```

Sauvegardez et lancer l'installation de php7.3 :

`apt update && apt install php7.3`

Taper php -v et contrôlez que vous êtes bien en 7.3

MariaDB

Nous allons utiliser la version contenue dans les dépôts de base :

`apt install mariadb-server`

A l'issue de l'installation, testez votre base de données en tapant mysql, la console mysql se lance alors et vous pouvez lancer des requêtes SQL (par exemple *SHOW DATABASES*;). Sortez en tapant *exit;*

Revision #8
Created 9 December 2019 13:30:13 by Admin
Updated 9 December 2019 14:57:12 by Admin