INSTALLATION ET CONFIGURATION SERVEUR WEB SOUS LINUX

PRESENTATION

Un serveur web, c'est le logiciel qui répond et renvoie les pages HTML à votre navigateur quand vous consultez un site Internet. Comme de plus en plus de services passent par le Web, c'est vraiment une composante essentielle à connaître pour un administrateur.

Apache existe depuis plus de 20 ans, et même s'il perd des parts de marché depuis quelques années, il reste le serveur web le plus utilisé au monde : c'est *la* référence à connaître. Son principal concurrent aujourd'hui est un autre logiciel libre appelé *Nginx*

Commencez par installer Apache par la commande :

```
$ sudo apt-get install apache2
```

Sous Ubuntu, vous trouverez la configuration d'Apache dans le répertoire `/etc/apache2/` . La configuration est ensuite découpée en un grand nombre de fichiers. Le fichier principal se nomme `apache2.conf` . Comme c'est souvent le cas pour les fichiers de configuration, les lignes commençant par un `#` sont des commentaires, et ce fichier est abondamment commenté. Il y a beaucoup de paramètres, parmi ceux-ci vous trouverez :

```
User ${APACHE_RUN_USER}
Group ${APACHE_RUN_GROUP}
```

qui définissent l'utilisateur et le groupe sous lesquels tourne Apache. Vous voyez que les valeurs sont des variables. La valeur réelle est configurée à part dans le fichier `envvars` , c'est dans ce fichier que vous pourrez la changer.

En fait, le processus principal d'Apache est lancé en tant que *root*, mais ce processus a pour seul rôle de lancer des processus fils appartenant à l'utilisateur indiqué ci-dessus. Ce sont alors ces processus fils qui s'occuperont de répondre aux requêtes des clients. Ces processus fils ont des droits limités aux fonctions qu'ils doivent accomplir (accès réseau, accès aux fichiers du serveur web, etc.). Si vous devez changer l'utilisateur et le groupe par défaut, il est important en termes de sécurité de garder un utilisateur dédié à Apache.

Vous trouverez ensuite la configuration des logs :

```
ErrorLog ${APACHE_LOG_DIR}/error.log
```

```
LogLevel warn
LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" vhost_combined
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %O" common
```

Vous pouvez définir le fichier de log utilisé pour stocker les logs d'erreur Apache (directive `ErrorLog`), le niveau de verbosité des logs (`LogLevel`) et différents formats de logs définis par la directive `LogFormat` . Pour cette dernière, différents formats sont préconfigurés, leur nom se trouve en second paramètre en fin de ligne.

Ce fichier de configuration inclut le fichier `ports.conf` qui contient la fameuse directive :

```
Listen 80
```

Elle indique à Apache d'écouter sur toutes les interfaces sur le port 80 (port par défaut pour HTTP). Si vous voulez restreindre votre service à une interface en particulier, vous pouvez par exemple écrire :

```
Listen 127.0.0.1:8080
```

pour écouter seulement sur l'interface locale sur le port 8080. Il peut y avoir plusieurs directives `Listen` .

Finalement, ce fichier inclut aussi tous les fichiers de configuration placés dans :

- `conf-enabled/`
- `mods-enabled/`
- `sites-enabled/`

Le principe de fonctionnement est que vous trouverez dans :

- `conf-available/` les fichiers de configuration additionnels installés sur le système ;
- `mods-available/` les modules installés et leur configuration ;
- `sites-available/` les configurations des différents sites.

Pour chacun de ces fichiers et pour activer la fonction correspondante, il faudra faire un lien symbolique du type `XXXX-enabled/YYYY.conf` -> `XXXX-available/YYYY.conf` . Comme ça, pour activer/désactiver une fonction, il suffit de faire/supprimer des liens symboliques. En plus, vous verrez plus loin qu'Ubuntu vous fournit des commandes pour faire ces liens automatiquement.

La conception d'Apache est modulaire et permet d'ajouter ou d'enlever à la volée les fonctionnalités nécessaires. Ce principe s'applique même aux fonctions qui forment le cœur du logiciel. Ainsi, vous trouverez dans `mods-available` trois modules du type `mpm_*.load` . Ces **MultiProcessing Modules** définissent la manière dont Apache va gérer les connexions HTTP. Pour fonctionner, Apache a besoin qu'**un et un seul** de ces modules soit chargé. Voici le détail de ces modules :

- **prefork** : avec ce module, Apache crée un processus fils pour chaque nouvelle connexion. C'est le plus sûr mais pas toujours le plus performant, car il est long et gourmand en mémoire de gérer autant de processus.
- **worker** : avec ce module, chaque processus fils d'Apache peut gérer plusieurs connexions dans des "threads". C'est généralement mieux en terme de performances, mais ça peut poser des problèmes avec les applications web qui gèrent mal ces threads.
- **event** : c'est le module le plus récent et celui choisi par défaut sous Ubuntu. C'est une évolution de worker qui permet de mieux gérer les connexions "keepalive". Ces connexions permettent de faire plusieurs requêtes HTTP à travers une même connexion réseau.

Je vous conseille de garder le MPM “event”, sauf si vous utilisez une application non compatible avec les threads (non-threadsafe) ; dans ce cas, utilisez “prefork”.

Configurez votre premier site web sous Apache

Apache est capable de gérer plusieurs sites web sur la même machine, c’est ce qu’on appelle des **hôtes virtuels** ou **virtual hosts**. Dans la requête que fera le client sur la machine, un en-tête HTTP précisera si le client veut plutôt consulter toto.com ou tata.org, qui sont tous les deux hébergés sur la machine. La configuration de chaque virtual host se fait dans un fichier `*.conf` dans `/etc/apache2/sites-available/` . Créez donc le fichier `/etc/apache2/sites-available/01-www.example.com.conf` contenant les paramètres suivants :

```
<VirtualHost *:80>
ServerName www.example.com
ServerAlias example.com
ServerAdmin webmaster@example.com
DocumentRoot /var/www/html/www.example.com
CustomLog ${APACHE_LOG_DIR}/www.example.com-access.log combined
ErrorLog ${APACHE_LOG_DIR}/www.example.com-error.log
  <Directory /var/www/html/www.example.com>
   Options All
   AllowOverride None
   </Directory>
</VirtualHost>
```

Tout d’abord, remarquez que la configuration d’un virtual host est comprise dans des balises “type HTML” `<VirtualHost *:80></VirtualHost>` . Le `*:80` de la balise ouvrante indique que le virtual host peut être utilisé sur le port 80 sur toutes les interfaces. Voici ensuite le détail des directives utilisées :

- `ServerName` : précise l'hôte pour lequel cette configuration sera utilisée. Il ne peut y avoir qu'un `ServerName` par VirtualHost.
- `ServerAlias` : il ne peut y avoir qu'un ServerName mais par contre, vous pouvez compléter par autant de ServerAlias que vous voulez, séparés par des espaces.
- `ServerAdmin` : indique une adresse mail de contact et qui peut être affichée sur certains messages d'erreur.
- `DocumentRoot` : c'est la racine de l'arborescence de votre site web, là où vous mettrez tous vos fichiers.

Une configuration minimale pourrait s'arrêter là, les autres directives sont des options supplémentaires.

- `CustomLog` : demande de stocker les fichiers de logs au format `combined` dans un fichier à part plutôt que dans le fichier de logs général.
- `ErrorLog` : demande de stocker aussi les logs d'erreur dans un fichier à part, mais leur format est fixe.
- `<Directory />` : dans la balise Directory, vous pouvez définir des règles qui s'appliquent uniquement au contenu d'un répertoire. Ici, vous activez toutes les options. Voici une partie des options activées :
  - `ExecCGI` : l'exécution de scripts à l'aide du module CGI est permise.
  - `FollowSymlinks` : le serveur va suivre les liens symboliques. Cette option est la seule active par défaut.
  - `Includes` : les inclusions côté serveur à l'aide du module `mod_include` sont autorisées.
  - `Indexes` : si aucun fichier par défaut type `index.html` n'est présent, le module `mod_autoindexes` va présenter une liste des fichiers et répertoires formatée par Apache.

- `AllowOverride None` : cette directive placée dans `<Directory />` indique qu'aucune option ne peut être surchargée par les options qui seraient contenues dans un fichier appelé `.htaccess` placé dans cette arborescence. Vous auriez pu préciser une à une les options qui peuvent être surchargées, ou utiliser `All` pour autoriser la surcharge de toutes les options : dans ce cas, les options du fichier `.htaccess` auront la priorité sur celles du Virtual Host.

Pour pouvoir tester votre site web, il vous reste à :

- Créer la racine de votre arborescence et y déposer un premier fichier `.html` :

```
$ sudo mkdir /var/www/html/www.example.com/
$ sudo cp /var/www/html/index.html /var/www/html/www.example.com
```

L'utilisateur Apache (par défaut www-data) doit avoir les droits de lecture sur ce répertoire et ses fichiers. C'est le cas par défaut.

- **Activer la configuration de votre VirtualHost** :

```
$ sudo a2ensite 01-www.example.com
```

Cette commande crée automatiquement le lien symbolique du répertoire `sites-enabled` vers le répertoire `sites-available` .

De la même façon, pour supprimer le lien symbolique, vous pouvez utiliser la commande `a2dissite` . De même, pour activer/désactiver des modules, vous pouvez utiliser `a2enmod` et `a2dismod` .

Recharger maintenant la configuration d'Apache pour prendre en compte vos modifications :

```
$ sudo systemctl reload apache2
```

Votre serveur est prêt. Maintenant si vous ajoutez `www.example.com` en alias à votre serveur dans le fichier `/etc/hosts` de votre client, vous pouvez admirer la page d'accueil de votre site dans votre navigateur.