Créer un site web avec apache2.2 sous Debian 7

- Objet : installation et configuration d'apache2.2
- Niveau requis :
  débutant, avisé
- Commentaires : Apprendre à configurer un serveur web sur son réseau local sous Debian 7.
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là ! 😀
- Suivi :
  obsolète
  - Création par Hypathie le 20/09/2014
  - Testé par Hypathie octobre 2014 sur Wheezy
- Commentaires sur le forum : Lien vers le forum concernant ce tuto [1)]
- Référence : http://www.apache.org/

Ce tuto traite d'Apache2.2, version proposée par Debian 7 (Wheezy).

La configuration de Apache2.4, proposée par Debian 8 (Jessie) est significativement différente. Il vaut donc mieux ne pas se référer à ce tutoriel à son sujet.

Introduction

Ce qui est mis en œuvre ici concerne une utilisation d'un serveur apache sur un réseau local qui ne pointe pas un nom de domaine acquis mais fictif.

Avant tout, il faut savoir que monter un serveur web pour de l'auto-hébergement présente des risques. Vous courrez par exemple le risque de donner un accès à tout votre réseau local à un pirate qui chercherait à prendre la main sur votre identité pour commettre des attaques illégales en votre nom.

Ce wiki a pour objet de proposer une initiation à apache2, et déploie son installation sur une machine virtuelle. Si vous choisissez de déployer ce qui suit sur une machine réelle faisant office de serveur personnel en vue d'auto-hébergement, l'auteur et debian-facile décline toute responsabilité sur les conséquences fâcheuses qui pourraient en découler.

Prenez le temps d'apprendre à sécuriser un serveur avant de vous lancer dans l'auto-hébergement ! 😎

Pré-requis

Créons une machine virtuelle pour s'exercer à mettre en place un serveur web.

Cette machine virtuelle doit être configurée côté réseau avec un accès par pont.
Pour faire tout comme en “vrai”, lors de son installation on a dé-sélectionné le choix de l'installation d'un “environnement de bureau”, on n'a pas sélectionné “serveur web”. On a simplement choisi les “outils debian”, et “serveur ssh”.

Après l'installation du système, la première chose à faire est de fixer l'IP de ce système virtuel. Par exemple, 192.168.x.xx.

Puis de relever son hostname, par exemple “debian-MV-server”.

Ensuite, on configure le serveur ssh, en prenant le temps de le sécuriser en changeant le port 22, en désactivant `PermitEmptyPasswords`, et côté client en créant des clés assymétriques, afin de pouvoir par la suite, configurer le serveur web depuis le client ssh.

Installation d'apache

```
apt-get update && apt-get install apache2
```

Après l'installation le serveur est fonctionnel. Si tout s'est bien passé, en tapant dans son navigateur `http://192.168.x.xx/`, il doit s'afficher ceci:

```
It works!

This is the default web page for this server.

The web server software is running but no content has been added, yet.
```

Comment apache est-il configuré ?

Afin de comprendre la mise en place d'un site web avec apache2, on va détailler la configuration par défaut d'apache.

- Le répertoire /etc/apache2

L'installation d'apache a mis en place sur le système plusieurs sous-répertoires de /etc/apache2.

```
cd /etc/apache2/ && ls
```

retour de la commande

```
apache2.conf  envvars  mods-available  ports.conf   sites-enabled
conf.d        magic    mods-enabled    sites-available
```

Il est à noter que le fichier /etc/apache2/apache2.conf est l'équivalent du fichier httpd.conf des distributions Linux à base de RedHat.
Il inclut d'autres fichiers de configuration qu'on n'a pas besoin de modifier pour une utilisation simple et personnelle d'apache2.

Les quatre sous-répertoires à relever dans un premier temps pour comprendre le fonctionnement d'apache sont :

1. /etc/apache2/mods-available/

2. /etc/apache2/mods-enabled/
3. /etc/apache2/sites-available/
4. /etc/apache2/sites-enabled/

- **/etc/apache2/sites-available/** :

Ce répertoire contient des fichiers qui indiquent les sites hébergés par apache2.
Puisque après l'installation, il a été possible d'afficher une page web d'accueil, c'est que ce répertoire contient un fichier qui le permet :

```
cd /etc/apache2/sites-available/ && less default
```

retour de la commande

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost

        DocumentRoot /var/www
        <Directory />
                Options FollowSymLinks
                AllowOverride None
        </Directory>
        <Directory /var/www/>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride None
                Order allow,deny
                allow from all
        </Directory>

        ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
        <Directory "/usr/lib/cgi-bin">
                AllowOverride None
                Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
                Order allow,deny
                Allow from all
        </Directory>
```

On voit la ligne **<Directory /var/www/>**. C'est là le chemin du fichier qui contient le code html de la page d'accueil.

Vérifions cela.

```
cd /var/www && ls
```

retour de la commande

```
index.html
```

```
less index.html
```

retour de la commande

```
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added,
yet.</p>
</body></html>
```

Donc le site “default” est configuré de sorte à mettre à disposition tout ce qui est dans /var/www. Mais ce n'est pas tout. Si la page d'accueil s'affiche sur le navigateur, c'est que le site de test de apache2 est activé. Cette activation dépend d'un lien symbolique entre un fichier du répertoire /etc/apache2/sites-enabled/ et un fichier du répertoire /etc/apache2/sites-available/default.

- **/etc/apache2/sites-enabled/** :

Ce répertoire contient des liens symboliques qui pointent vers des fichiers de /etc/apache2/sites-available.

```
cd /etc/apache2/sites-enabled/ && ls
```

retour de la commande

```
000-default
```

Ce fichier est un lien symbolique : il pointe vers /etc/apache2/sites-available/default.

```
ls -l /etc/apache2/sites-enabled/000-default
```

retour de la commande

```
lrwxrwxrwx 1 root root 26 sept. 19 06:06\
 /etc/apache2/sites-enabled/000-default -> ../sites-available/default
```

Pour l'instant, le serveur apache se contente d'envoyer du code HTML, CSS, javascript au navigateur, c'est-à-dire du code écrit dans un langage compris par le navigateur (côté client). Pour le PHP interprété par le serveur, il faudra mettre en place un module php. Pour mettre cela en place il faudra considérer cette fois les répertoires /etc/apache2/mods-available/ et /etc/apache2/mods-enabled/ dont il sera question plus bas.

Revenons sur la configuration par défaut d'apache2, et la mise en service de la page d'accueil.

C'est le fait de la présence de ce lien symbolique /etc/apache2/sites-enabled/000-default qui pointe vers le site déclaré, et correctement configuré, /etc/apache2/sites-available/default, qui quant à lui, attribue à apache la prise en charge du fichier de code web /var/www/index.html, qui active le site.

Pour activer un site ou le désactiver, il suffit donc de créer ou de supprimer le lien symbolique qui relie /etc/apache2/sites-enabled/000-default et /etc/apache2/sites-available/

Cela peut se faire avec les commandes `ln -s` et `rm`. Mais on utilise pour ce faire les utilitaires :

- `a2ensite` : (apache2 enable site) : active un site,
- `a2dissite` : (apache2 disable site) : désactive un site.

Puisque tout est bien clair, créons notre propre site web.

Configurations d'un site web

Création du site web dans /var/www/

- On crée un dossier dans /var/www/ :

Par exemple “monsite.com” qui va pouvoir accueillir le site internet.

```
mkdir -p /var/www/monsite.com/public_html
```

- On attribue à root les fichiers nouvellement créés :

```
chown -R $USER:$USER /var/www/monsite.com/public_html
```

- On change les droits pour que le site puisse être lu par tous :

```
chmod -R 755 /var/www
```

- On crée sa première page index.html

```
vim /var/www/monsite.com/public_html/index.html
```

Contenant par exemple le code suivant:

index.html

```
<html>
 <body>
 <h1>Bravo !</h1>
  <p>La mise en place d'un Virtualhost est réussie !</p>
 </body>
```

```
</html>
```

On enregistre !

- On crée un VirtualHost d'apache2ls pour ce site :

```
cp /etc/apache2/sites-available/default /etc/apache2/sites-
available/monsite.com
```

```
vim /etc/apache2/sites-available/monsite.com
```

On inscrit ceci :

monsite.com

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        ServerName monsite.com
        ServerAlias www.monsite.com

        DocumentRoot /var/www/monsite.com/public_html
  <...>
```

On peut laisser le reste tel quel.

- Activer ce site en créant un lien symbolique dans /etc/apache2/sites-enabled/

```
a2ensite monsite.com
```

- Prendre en compte les modifications effectuées en redémarrant Apache :

```
service apache2 reload
```

- Vérification :

Dans le navigateur : **http://192.168.x.xx/monsite.com**

```
Index of /monsite.com
[ICO]   Name    Last modified    Size    Description
[DIR]   Parent Directory         -
[DIR]   public_html/    19-Sep-2014 16:38     -
Apache/2.2.22 (Debian) Server at 192.168.x.xx Port 80
```

Si on clique sur `public_html/` on voit un problème d'encodage :

```
Bravo !

La mise en place d'un Virtualhost est rÃ©ussie !
```

Cela va être vite corrigé !

Solutionner le problème d'encodage

On va forcer l'encodage au niveau du serveur apache. Il suffit que tous les fichiers utilisent le même encodage utf8.

- Mais avant vérifions les locales générées sur le système.

Elles apparaissent quand on tape la commande :

```
grep -v "^#" /etc/locale.gen
```

retour de la commande

```
fr_FR.UTF-8 UTF-8
```

- Corriger le fichier /etc/apache2/conf.d/charset :

```
vim /etc/apache2/conf.d/charset
```

On dé-commente la ligne `#AddDefaultCharset UTF-8`

```
AddDefaultCharset UTF-8
```

Et on enregistre !

- Corriger le fichier /etc/apache2/envvars :

```
vim /etc/apache2/envvars
```

Pour dé-commanter la ligne `. /etc/default/locale`

```
## Uncomment the following line to use the system default locale instead:
. /etc/default/locale
```

- Faire prendre en compte les modifications à apache2 :

```
service apache2 reload
```

- Vider le cache du navigateur :

Par exemple, avec iceweasel :

`Historique → Supprimer l'historique récent`

Et quand on recharge la page le problème est réglé :

```
Bravo !
```

```
La mise en place d'un Virtualhost est réussie !
```

Installer le module php

Installer libapache2-mod-php5

Ce paquet casse le MPM worker[2)] et engendre l'installation du MPM prefork[3)].

```
apt-get install libapache2-mod-php5
```

Une fois l'installation effectuée on peut vérifier que php5 est apparu dans /etc/apache2/mod-available.

```
ls -l /etc/apache2/mods-available/php5*
```

retour de la commande

```
-rw-r--r-- 1 root root 898 août  21 10:49 /etc/apache2/mods-
available/php5.conf
-rw-r--r-- 1 root root  59 août  21 10:49 /etc/apache2/mods-
available/php5.load
```

Si après l'installation le module php5 ne figure pas dans la liste des modules du fichier /etc/apache2/mods-available/

- essayer :

```
apt-get install php5 libapache2-mod-php5
```

Et si cela n'est pas suffisant, suivre les directives de ce lien :

- http://digitizor.com/2012/09/03/how-to-fix-module-php5-does-not-exist-error-in-apache-linux/

Activer le module php5

```
a2enmod php && /etc/init.d/apache2 restart
```

Dans /etc/apache2/mod-enabled, ce sont des liens symboliques qui activent ce module:

```
ls -l /etc/apache2/mods-enabled/php5*
```

retour de la commande

```
lrwxrwxrwx 1 root root 27 sept. 20 11:40\
  /etc/apache2/mods-enabled/php5.conf   ->   ../mods-
available/php5.conf
lrwxrwxrwx 1 root root 27 sept. 20 11:40\
  /etc/apache2/mods-enabled/php5.load   ->   ../mods-
available/php5.load
```

Quand les liens sont là, comme ci-dessus, c'est que le module est activé.

Activer/désactiver un module

- a2enmod : (apache2 enable module) : active un module apache2
- a2dismod : (apache2 disable module) : désactive un module apache2

Le module est activé par défaut, mais si ce n'était pas le cas :

```
a2enmod php5
```

Si on active ou désactive un module ne pas oublier après l'opération de réactiver apache : `service apache2 restart`

Créer la page de test

```
mv /var/www/index.html /var/www/index.php
```

```
vim /var/www/index.php
```

index.php

```
<html>
 <body>
  <h1>It works!</h1>
  <p>This is the default web page for this server.</p>
  <?php
echo "La date du jour est " . date("d/m/Y") . "!\n";
  ?>
 </body>
</html>
```

- Recharger apache :

```
service apache2 restart
```

- Tester en tapant dans le navigateur :

http://ip-du-serveur/

```
It works!

This is the default web page for this server.
La date du jour est 20/09/2014!
```

Tous les outils sont en place pour apprendre les langages html et php ! 😎

Sécuriser son site web

Sécuriser Apache2

Ce n'est là qu'un minimum et non une sécurisation optimale.

Diffuser le minimum d'information sur apache

- Éditer le fichier /etc/apache2/conf.d/security.

```
vim /etc/apache2/conf.d/security
```

Modifier ceci :

> La ligne “ServerTokens OS” doit être commenté : `#ServerTokens OS`
>
> Ajouter : `ServerTokens Prod`
>
> Dé-commenter : `ServerSignature Off`
>
> Et commenter : `#ServerSignature On`
>
> Enfin vérifier que `TraceEnable Off` soit dé-commenté
>
> et que `#TraceEnable On` soit commenté

- On redémarre apache :

```
service apache2 restart
```

Modifier certaines options <Directory> de son Virtualhost

- Désactiver le site “sites-available/default” :

```
a2dissite default
```

- Désactiver le site “monsite.com” :

```
a2dissite monsite.com
```

- Ensuite on va modifier les directives du site “/etc/apache2/sites-available/monsite.com” :

On édite “/etc/apache2/sites-available/monsite.com” :

```
vim /etc/apache2/sites-available/monsite.com
```

monsite.com

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        ServerName monsite.com
        ServerAlias www.monsite.com

        DocumentRoot /var/www/monsite.com
        <Directory />
                Options -FollowSymLinks
                AllowOverride None
        </Directory>
        <Directory /var/www/>
                Options -Indexes
                Options -FollowSymLinks
                Options -Includes
                Options -ExecCGI
                Options MultiViews
                AllowOverride None
                Order deny,allow
                deny from all
                allow from 192.168.0.0/24
        </Directory>
```

> `Options -Indexes` : Pour désactiver l'option permettant le parcours d'un répertoire
>
> `Options -FollowSymLinks` : Pour désactiver l'option permettant à apache de suivre des liens symboliques (qui pourraient permettre de quitter /var/www).
>
> `Options -Includes` : Pour désactiver l'option permettant à apache de faire des inclusions côté serveur.
>
> `Options -ExecCGI` : Pour désactiver l'option permettant à apache l'utilisation de scripts CGI. Désactiver cette option seulement si on n'utilise pas de script CGI[4].
>
> Si on ne peut pas mettre l'option `-FollowSymLinks` dans le fichier “sites-available/default”, on peut dans celui de son site.
>
> S'il s'agit d'un usage personnel local, on peut restreindre l'accès au serveur avec les IP des clients du réseau local.

Pour connaître la signification des différentes options à mettre soit dans “sites-avaible” soit dans le fichier de son site voir : http://httpd.apache.org/docs/2.2/mod/core.html

Création d'un nouvel utilisateur du système Linux

On développera son site dans le répertoire de cet utilisateur. Il est déconseillé de développer son site dans /var/www, surtout si la partition est plus petite que /home/ !

Cela évitera aussi de mettre en place le module userdir, ce qui est déconseillé par la documentation d'apache[5)]

- Par exemple du nom de “web”

```
adduser --system web --ingroup www-data
```

- On lui crée un mot de passe système :

```
 passwd web
```

- On crée les fichiers du site web, “monsite.com” dans /home/web/:

```
cd /home/web/
```

```
mkdir -p monsite.com/public_html
```

- On édite un index pour le dossier /home/web/monsite.com/public_html/ :

Le module php5 est en place et a été testé, donc on peut créer un “index.php”.

```
vim /home/web/monsite.com/public_html/index.php
```

index.php

```
<html>
 <body>
 <h1>Bienvenue sur monsite.com</h1>
  <p>Site en cours de réalisation !</p>
  <?php
  echo "La date du jour est " . date("d/m/Y") . "!\n";
  ?>
 </body>
</html>
```

Restreindre l'accès à "monsite.com" par login et mot de passe apache2

La création d'un mot de passe sécurise un peu l'accès du site qu'on va crée pour le nouvel virtualhost. **Pour ce faire on va créer en correspondance au utilisateur “web” du système Linux, un mot de passe apache avec la commande `htpasswd`.**

Création d'un mot de passe pour accéder à "monsite.com"

Le dossier /etc/local est un bon endroit pour créer les fichiers de mots de passe apache.

```
cd /usr/local/ && ls
```

retour de la commande

```
bin  etc  games  include  lib  man  sbin  share  src
```

- Il faut créer un dossier apache du nom de son choix :

Par exemple “passwd”

```
mkdir -p apache/passwd && cd apache/passwd
```

- Il faut générer des mots de passe pour l'utilisateur apache2:

La commande `htpasswd` va créer un fichier qui contiendra utilisateur apache /mot de passe.

```
htpasswd -c passwords web
```

séquence interactive

```
New password:
Re-type new password:
Adding password for user web
```

→ `-c` pour la première fois qu'on crée un mot de passe. Si on l'utilisait une deuxième fois pour le même fichier “passwords”, pour un deuxième utilisateur, on écrasera le contenu du fichier, et on perdrait le mot de passe du premier utilisateur. Donc pour un éventuel deuxième utilisateur “toto”: `htpasswd passwords toto`.
→ le mot de passe est chiffré.

Pour supprimer un utilisateur :

```
htpasswd -D /usr/local/apache/passwd/passwords utilisateur
```

- Pour créer un groupe par exemple aussi dans le même dossier /usr/local/apache/passwd/

Par exemple du nom de “groups” :

```
vim groups
```

```
goupe1: web
```

Modifier les fichiers des sites actifs de /etc/apache2/sites-available

Pour restreindre l'accès à “monsite.com”, il faut modifier d'abord le virtualhost “/etc/apache2/sites-available/default”

Pour "/etc/apache2/sites-available/default"

Cela est nécessaire pour restreindre l'accès du site qui est développé dans le répertoire de l'utilisateur /home/web/monsite.com qu'on a créé.

- On modifie “/etc/apache2/sites-available/default” comme ceci :

```
vim /etc/apache2/sites-available/default
```

default

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost

        DocumentRoot /var/www
        <Directory />
                Options FollowSymLinks
                AllowOverride AuthConfig
                Order deny,allow
                deny from all
                allow from 192.168.0.0/24
        </Directory>
        <Directory /var/www/>
                AuthName "Ma zone est restreinte !"
                AuthBasicProvider file
                AuthUserFile /usr/local/apache/passwd/passwords
                AuthGroupFile /usr/local/apache/passwd/groups
                Require user web
                require group group1

        </Directory>

        ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
        <Directory "/usr/lib/cgi-bin">
                AllowOverride None
                Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
                Order allow,deny
                Allow from all
        </Directory>
        ErrorLog ${APACHE_LOG_DIR}/error.log

        # Possible values include: debug, info, notice, warn, error,
crit,
        # alert, emerg.
```

```
        LogLevel warn

        CustomLog ${APACHE_LOG_DIR}/access.log combined
```

Bien remarquer que pour pouvoir ajouter cette directive

→ on a mis `AllowOverride AuthConfig` à la place de `AllowOverride None` non seulement dans la directive “<Directory /var/www/>”.

mais aussi dans “<Directory />”.

Pour autoriser l'accès à tous les utilisateurs qui ont un mot de passe apache :

→ `Require user web`

→ `Require group groupe1`

On aurait pu mettre :

→ `Require valid-user`

Voir aussi la directive “Satisfy” : http://httpd.apache.org/docs/2.2/howto/auth.html

Pour sécuriser l'accès au système de fichier :

`Require all denied` : interdire aux clients de parcourir l'ensemble du système de fichiers. Ceci va interdire l'accès par défaut à tous les fichiers du système de fichiers; `Ensuite on autorise section par section.` Voir http://httpd.apache.org/docs/trunk/fr/misc/security_tips.html et require directive
Pour l'utiliser avec une zone restreinte, il faut alors l'inclure dans <Directory /> et créer la zone restreinte dans une directive séparée :

`<Directory /var/www/restricted/> … </Directory>`

Pour "/etc/apache2/sites-available/monsite.com"

```
vim /etc/apache2/sites-available/monsite.com
```

monsite.com

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        ServerName monsite.com
        ServerAlias www.monsite.com

        DocumentRoot /var/www/monsite.com
```

```
        <Directory />
                Options -FollowSymLinks
                AllowOverride AuthConfig
                Order deny,allow
                deny from all
                allow from 192.168.0.0/24
        </Directory>
        <Directory /var/www/monsite.com>
                Options -Indexes
                Options -FollowSymLinks
                Options -Includes
                Options -ExecCGI
                Options MultiViews
                AllowOverride AuthConfig
                AuthType Basic
                AuthName "Ma zone est restreinte !"
                AuthBasicProvider file
                AuthUserFile /usr/local/apache/passwd/passwords
                AuthGroupFile /usr/local/apache/passwd/groups
                Require user web
        </Directory>
        ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
        <Directory "/usr/lib/cgi-bin">
                AllowOverride None
                Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
                Order allow,deny
                Allow from all
        </Directory>

        ErrorLog ${APACHE_LOG_DIR}/error.log

        # Possible values include: debug, info, notice, warn, error,
crit,
        # alert, emerg.
        LogLevel warn
        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

On supprime le fichier "monsite.com" de /var/www/

(Si on a suivi précédemment : création de "monsite.com" dans /var/www/)

```
cd /var/www/
```

```
rm -r monsite.com
```

On crée un lien symbolique à la place

Le nom symbolique doit avoir pour nom celui du site (“monsite.com”) et il faut le placer dans /var/www/ et le faire pointer vers les répertoires et fichiers où le site est développé (/home/web/monsite.com/) :

```
ln -s /home/web/monsite.com/ /var/www/monsite.com
```

Vérifier l'appartenance et les droits des fichiers utilisés par "apache"

Modifier le groupe de /var/www

Le groupe www-data ne doit pas être propriétaire de /var/www/ mais ce répertoire peut appartenir au groupe www-data. L'idée est ainsi de diminuer au maximum les droits du groupe www-data, tout en laissant possible la consultation du site web.

```
chown -R root:www-data /var/www/*
```

On vérifie les droits POSIX de /var/www/

A priori le fichier /var/www/index.php, au même titre que n'importe quelle page web accessible depuis un navigateur web, devrait avoir les droits suivants :

```
-rw-r----- 1 root www-data 4096 sept. 22 11:47 index.php
```

On utilise les droits suivants :

- Sur les fichiers réguliers : pour l'utilisateur root, droits en lecture (r) écriture (w) ; pour le groupe, droit de lecture seulement.
- Sur les répertoires : pour l'utilisateur root, droits en lecture (r) écriture (w) exécution (x) ; pour le groupe et les autres droit en lecture et exécution.

- **Pour les répertoires /var/www et /home/web** :

```
chmod -R 755 /var/www/ /home/web/
```

- **Pour les fichiers d'index :**

```
chmod -R 644 /var/www/index.php /home/web/monsite.com/public_html/index.php
```

- **Pour les propriétaires des fichiers d'index** :

```
chown root:www-data /var/www/index.php\
 /home/web/monsite.com/public_html/index.php
```

- **Pour le lien symbolique /var/www/monsite.com** : on ne peut pas modifier ces droits mais ce n'est pas grave. Ce qui compte c'est le fichier vers lequel pointe un lien symbolique.

On vérifie depuis son navigateur

- On ré-active le site “default”

```
a2ensite default
```

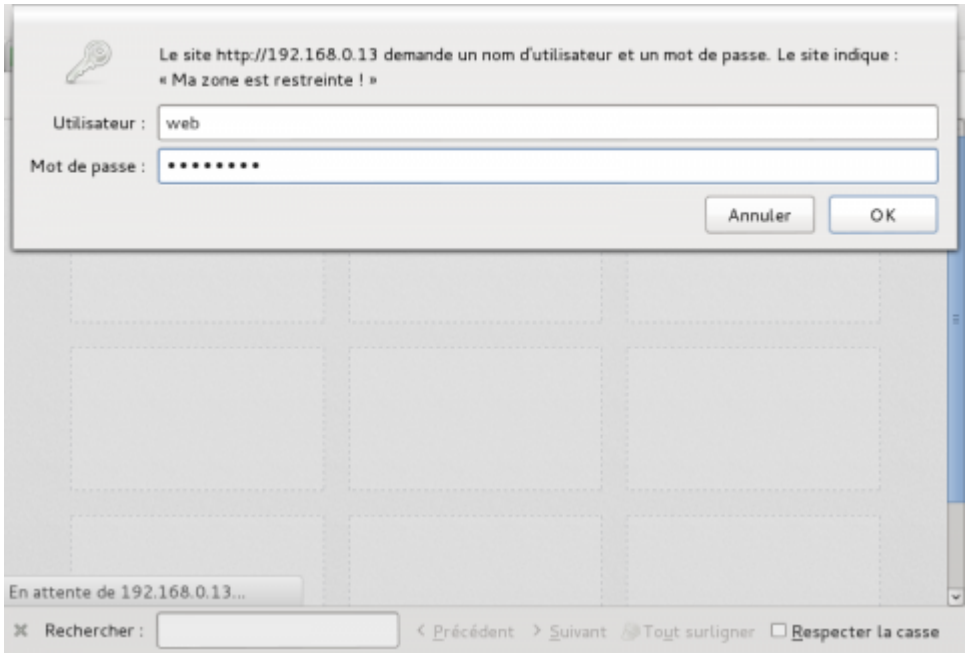
- On ré-active le site “monsite.com”

```
a2ensite monsite.com
```

- On recharge apache2 :

```
service apache2 reload
```

Enfin depuis son navigateur :
http://ip-serveur/monsite.com/



→

Le login du compte apache (nom de l'utilisateur du système Linux) est demandé ainsi que son mot de passe et on peut accéder à ses pages d'index !

La journalisation (Logging)

Il faut consulter régulièrement au moins deux : Les logs d'erreur et les logs d'accès au serveur.

Paramétrer la journalisation

Le niveau de consignation souhaité dépend de la directive `LogLevel`, dans le fichier de configuration par défaut `/etc/apache2/apache2.conf` qui est réglé par défaut sur :

```
LogLevel warn
```

Ainsi paramétrer apache inscrit les événements anormaux dans l'un des fichiers du répertoire /var/log/apache2/error.log. On peut mettre l'un des neuf niveau d'alerte, pour être alerter de la moindre information jusqu'au alerte grave :

-`trace` : traçage des informations de différents niveaux (produit une grande quantité d'informations);

- `debug` : informations de débogage qui peut être utile pour repérer où un problème ;

- `info`: message d'information qui pourrait être bon à savoir;

- `notice` : signal un événement normal, mais à noter;

- `warn` : signal un événement anormal, mais pas très préoccupant;

- `error` : signal que quelque chose a échoué;

- `crit` : problèmes importants qui doivent être pris en compte;

- `alert` : situation grave qui nécessite rapidement une action;

- `emerg` : urgence de la situation, le système est dans un état inutilisable.

→ Par défaut, on est informé à partir du niveau choisi jusqu'au niveau le plus grave.

Comment le système de journalisation est-il configuré ?

Dans le fichier général **/etc/apache2/apache2.conf** une ligne indique le fichier où sont consigné les logs :

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

> Cette directive nomme le fichier où Apache tiendra ses messages d'erreur. Comme vous pouvez le voir, il utilise une variable d'environnement appelée “APACHE_LOG_DIR” pour obtenir le préfixe du chemin de répertoire.
> Pour savoir vers quel fichier renvoie cette variable d'environnement il faut consulter le fichier **/etc/apache2/envvars**

```
export APACHE_LOG_DIR=/var/log/apache2$SUFFIX
```

> Cela signifie que lorsqu'il est combiné avec la directive (variable “SUFFIX”) dans le fichier “apache2.conf”, **Apache enregistre les logs d'erreur dans un fichier appelé /var/log/apache2/error.log**.

```
ls /var/log/apache2
```

```
access.log  error.log  other_vhosts_access.log
```

Les logs d'accès au serveur

Très important pour la sécurisation du serveur. On peut savoir s'il y a eu des tentatives de piratage.

Ces logs sont consignés par défaut dans le fichier **/var/log/apache2/access.log**.

Le paramétrage des logs d'accès n'est pas dans le fichier de configuration générale /etc/apache2/apache2.conf, mais dans le fichier de l'hôte par défaut.

La déclaration du journal d'accès se trouve donc dans : **/etc/apache2/sites-available/default**.

Nous pouvons y trouver trois valeurs distinctes concernant l'exploitation des logs d'accès.

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

LogLevel warn

CustomLog ${APACHE_LOG_DIR}/access.log combined
```

La définition du journal des erreurs (ErrorLog et Loglevel) correspond à celle dans le fichier de configuration par général. Il n'est pas nécessaire d'avoir cette ligne dans les deux fichiers, mais ce doublon permet à modifier ici l'emplacement des logs pour celui qu'on souhaite.

Paramétrer la directive "CustomLog" pour améliorer la sécurité

```
CustomLog ${APACHE_LOG_DIR}/access.log combined
```

Elle est constituée de deux partie :

- ${APACHE_LOG_DIR}/access.log : localisation des logs d'accès
- `combined` : le format des logs par défaut
  il s'agit d'une étiquette qui renvoie à la directive `LogFormat` (personnalisable) dans le fichier de configuration par général **`/etc/apache2/apache2.conf`**.

La directive "LogFormat" : /etc/apache2/apache2.conf

Par défaut :

```
LogFormat "%h %l %u %t \"%r\" %>s %O \"{Referer}i\" \"%{User-Agent}i\""
combined
```

Pour comprendre chacune de ces variables voir la documentation d'apache.

- Voici par exemple ma configuration :

```
vim /etc/apache2/apache2.conf
```

```
LogFormat "%h %l %u %t \"%r\" %s %b" common
```

> %h : sera remplacée par le nom de l'hôte distant
> %l : sera remplacée par logname a distance. Il faut que le module "mod_ident" soit présent et que la directive "IdentityCheck" soit activé (“On”).
> u% : sera remplacée par utilisateur distant.
> %t : sera remplacée par moment où la demande a été reçue (format standard anglais)
> \“%r\” : sera remplacée par la ligne de demande.
> %s : sera remplacée par le statut de la demande.
> %b : sera remplacée par la taille de la réponse en octets, à l'exception des en-têtes HTTP. Dans le format CLF, I.S. un «-» plutôt que d'un 0 si aucune octets sont envoyés.

Pour aller plus loin, par exemple pour la rotation des journaux voir
http://httpd.apache.org/docs/trunk/fr/logs.html

OpenSSL : créer un certificat avec apache2

Le protocole ssl

Le protocole ssl (Secure Socket Layer) permet la sécurisation des communications entre un serveur et un client, au moyen :

1. d'une authentification mutuelle ;
2. d'une vérification de l'intégrité des communication par signature digitale
3. par un chiffrement par clés asymétriques afin que la communication soit privée

- Le port par défaut est 443 :
- Son utilisation se fait pour apache2 via OpenSSL, avec le module mod_ssl:

Il est installé par défaut mais il n'est pas chargé.

À quoi sert ssl ?

Il permet la signature et la mise en place de certificat au moyen de chiffrement par clés asymétriques.

- Chiffrement par clé asymétrique :

On crée deux clés, une publique qu'on donne à l'utilisateur Tout_le_monde. Cette clé chiffre les informations que l'utilisateur Tout_le_monde veut nous envoyer. La deuxième clé, la clé privée, qu'on garde précieusement secrète, permet de déchiffrer les informations que l'utilisateur Tout_le_monde a chiffré avec la clé publique qu'on lui a donné.

- La signature :

À l'inverse du chiffrement, avec notre clé privée ont va cette fois, non pas déchiffrer ce qu'on envoie, mais chiffrer un document avec cette clé privée. La clé qui permet de déchiffrer ce qu'on va chiffrer avec notre clé privée pourra être déchiffrer par tous ceux qui ont notre clé publique. Et c'est le but de la signature, qui n'est faite pour rendre secret quoique ce soit, mais pour certifier qu'on est l'auteur du document qu'on a chiffré avec notre clé privée.

Pour ce faire, on prend nos données à certifier, un effectue un hashage sur nos données, puis on fait un chiffrement du hashage (cela s'appelle la signature).

Ainsi pour être sûr que notre document est authentique, l'utilisateur Tout_le_monde va appliquer sur notre document le même algorithme de hashage qu'on a utilisé et compare ce résultat avec celui issu du déchiffrement qu'il obtient sur ce même document au moyen de la clé publique qu'on lui a donné. Si les deux sont identiques, alors l'auteur du document est authentifié ; il est aussi garanti d'être intacte du point de vue de son contenu.

Certificat

Il faut encore que la clé publique que reçoit l'utilisateur Tout_le_monde, soit garantie comme étant bien la mienne. Pour éviter que quelqu'un se fasse passer pour moi auprès de l'utilisateur Tout_le_monde, j'ai recours à un certificat.

Quand un certificat est mis en place, l'utilisateur Tout_le_monde peut alors se renseigner auprès de l'organisme de certification[6] que je suis bien l'auteur la clé publique qu'il a reçue. Cela garantit à l'utilisateur Tout_le_monde qui est bien sur mon serveur quand il s'y connecte, y télécharge des fichiers, etc...

Pour faire certifier une clé publique, il faut qu'une organisation appelé “Autorité de certification” (CA) me vende [7] une signature pour mon certificat.

Concrètement mon serveur apache2 crée un certificat en même temps que la paire de clés, et je fais signer ce certificat par une autorité de certification.

Les certificats auto-signés

Ce sont des certificats à usage interne. Signés par un serveur local, ce type de certificat permet de garantir la confidentialité des échanges au sein d'une organisation locale, par exemple pour le besoin d'un intranet. Il est ainsi possible d'effectuer une authentification des utilisateurs grâce à des certificats auto-signés.

Créer un certificat auto-signé pour le site "default-ssl"

Cela ne garantira pas d'une éventuelle usurpation d'identité.
Le certificat auto-signé c'est bien pour l'apprentissage en local !

ssl dans la configuration d'Apache2

- Vérifier la présence mod_ssl et comprendre la configuration sous debian :

Le fichier de configuration du module se trouve dans :

```
ls /etc/apache2/mods-available | grep ssl.*
```

Il doit y avoir `ssl.conf` et `ssl.load`. Sinon il faut **installer** (ne pas l'activer tout de suite).

Création des clé et certificat

Pour cela on va se servir de l'utilitaire openssl.

- Création du répertoire où l'on rangera sa clé privée :

```
cd /etc/apache2/ && mkdir ssl
```

- Création de la clé privée et du certificat :

La commande qui suit cette note, raccourcit ce qui suit:

1) **Création de la clé privée** : par exemple `openssl genrsa 1024 > /etc/apache2/ssl/apache.key`

2) **À partir de cette clé privée, on crée un certificat CSR qui contient une clé publique** à faire signer :

```
openssl req -new -key /etc/apache2/ssl/apache.key > /etc/apache2/ssl/apache.csr
```

⇒ C'est ce certificat qu'il faut faire signer par une autorité de certification (CA) Ou qu'il faudra auto-signer.

3) **On décide d'auto-signer ce certificat donc on crée une clé privée qui serait celle du CA officielle** :

```
openssl genrsa -des3 1024 > ca.key
```

-on peut ajouter -des3 qui introduit l'usage d'une “passphrase”
-c'est cette clé privée qui signera tous les certificats que l'on émettra ; cette “passphrase” sera donc demandée à chaque utilisation de la clé.

4) **Puis à partir de la clé privée, on crée un certificat x509 pour une durée de validité d'un an auto-signé**

```
openssl req -new -x509 -days 365 -key ca.key > ca.crt
```

-On répondrait à nouveau aux questions, mais on changerait la réponse relative à “Common Name”.

5) **Enfin il faudrait faire signer notre certificat de demande de signature par le certificat du CA (qui a été fait par nous-même) :**

```
openssl x509 -req -in /etc/apache2/ssl/apache.csr -out /etc/apache2/ssl/apache.crt -CA ca.crt -CAkey ca.key\ -CAcreateserial -CAserial ca.srl
```

Pour créer une connexion privée et sécurisée entre le serveur et les clients qui s'y connecteraient, il faudrait procurer de façon sécurisé le fichier ca.crt (qu'on aurait fait nous-même en suivant toutes ces étapes) aux clients (ssh par exemple), afin qu'il soit installer dans leur navigateur.
Par exemple avec Iceweasel :

→ Edition → préférence → Avancé → Afficher les certificats → (Bouton)importer

```
openssl req -x509 -nodes -days 365\
 -newkey rsa:1024 -keyout /etc/apache2/ssl/apache.key\
 -out /etc/apache2/ssl/apache.crt
```

> -x509 -nodes : type de certificat
> -days 365 : durée de vie du certificat (en jours)
> - newkey rsa:1024 : clé rsa de 1024 bits

> -out /chemin/fichier/certificat : crée le fichier du certificat
> -keyout /chemin/fichier/clé : chemin du fichier de sa clé privée

Il faut répondre à une suite de question : La plus importante est “Common Name”: il faut répondre par le nom de domaine ou l'ip public du serveur concerné. Dans le cas de cette exemple, l'ip locale (fixe) du serveur fera l'affaire.

Par exemple :
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:France
Locality Name (eg, city) []:SaVille
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:192.168.0.13
Email Address []:example@gmx.fr

- Restreindre les droits sur le fichier de clé privée, aux droits de lecture seulement, et pour root:

```
chmod 400 /etc/apache2/ssl/apache.key
```

Créer un site accessible en ssl

Cette fois on va mettre en place une méthode plus directe que précédemment. Par exemple un site s'appelant pegaseous.com

On va se servir des fichiers par défaut.
On commence par désactiver les deux fichiers /etc/apache2/sites-available/default et /etc/apache2/sites-available/default-ssl avec la commande `a2dissite`.

On crée ensuite les répertoires et fichiers où développer son site.

- Par exemple dans le répertoire d'un autre utilisateur.

```
mkdir -p /home/hypathie/www/pegaseous.com/public_html
```

- On crée un index dans /home/hypathie/www/

```
mv /var/www/index.html /home/hypathie/www/
```

On peut le modifier :

```
vim /home/hypathie/www/index.html
```

index.html

```
<html><body><h1>Ça marche!</h1>
<p>Voici ma page d'index.</p>
<p>YEP ! Déployée dans mon répertoire personnel !</p>
```

```
<p>Le serveur web apache2 fonctionne.</p>
</body></html>
```

- On crée un contenu en php dans “pegaseous” :

```
vim /home/hypathie/www/pegaseous.com/public_html/index.php
```

index.php

```
<html>
 <body>
 <h1>Bienvenue sur pegaseous.com</h1>
  <p>Site en cours de réalisation !</p>
  <?php
  echo "La date du jour est " . date("d/m/Y") . "!\n";
  ?>
 </body>
</html>
```

On pense aux droits unix sur ses répertoires et fichiers

```
chmod -R 755 /home/hypathie/www/
```

```
chmod 644 /home/hypathie/www/index.html\
 /home/hypathie/www/pegaseous.com/public_html/index.php
```

```
chown root:www-data /home/hypathie/www/
```

Le virtualhost "defaut"

On doit d'abord s'occuper du fichier /etc/apache2/sites-available/default

```
vim /etc/apache2/sites-available/default
```

default

```
<VirtualHost 192.168.0.13:80>
        ServerAdmin webmaster@localhost
        ServerName pegaseous.com
        DocumentRoot /home/hypathie/www
        <Directory />
                Options FollowSymLinks
                AllowOverride None
        </Directory>
        <Directory /home/hypathie/www/>
```

```
                Options Indexes FollowSymLinks MultiViews
                AllowOverride None
                Order allow,deny
                allow from all
        </Directory>

        ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
        <Directory "/usr/lib/cgi-bin">
                AllowOverride None
                Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
                Order allow,deny
                Allow from all
        </Directory>

        ErrorLog ${APACHE_LOG_DIR}/error.log
        # Possible values include: debug, info, notice, warn, error,
crit,
        # alert, emerg.
        LogLevel warn

        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

- On charge le site “default” :

```
a2ensite default
```

- On édite /etc/apache2/ports.conf

Pour s'éviter d'avoir au redémarrage d'apache2 le message suivant :

```
service apache2 reload
[....] Reloading web server config: apache2[Thu Sep 25 09:33:25 2014]
 [warn] NameVirtualHost 192.168.0.13:80 has no VirtualHosts
```

```
vim /etc/apache2/ports.conf
```

ports.conf

```
NameVirtualHost 192.168.0.13:80
Listen 80
```

Le fichier "default-ssl"

- On active le module ssl:

```
a2enmod ssl
```

- On édite vim /etc/apache2/sites-available/default-ssl pour ajouter clé et certificat ainsi que le site déclarer dans le fichier default“ :

```
vim /etc/apache2/sites-available/default-ssl
```

default-ssl

```
<IfModule mod_ssl.c>
<VirtualHost *:443>
        ServerAdmin webmaster@localhost

        ServerName pegaseous.com
        DocumentRoot /home/hypathie/www
        <Directory />
                Options FollowSymLinks
                AllowOverride None
        </Directory>
        <Directory /home/hypathie/www/>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride None
                Order allow,deny
                allow from all
        </Directory>
#plus bas
        SSLCertificateFile    /etc/apache2/ssl/apache.crt
        SSLCertificateKeyFile /etc/apache2/ssl/apache.key
#<...>

</VirtualHost>
</IfModule>
```

Il n'y a rien d'autre à modifier.

- On charge le fichier “default-ssl”

```
a2ensite default-ssl
```

On peut maintenant accéder au site “pegaseous.com” en tapant dans le navigateur : https://192.168.0.13/pegaseous.com/.

Il est normal que le navigateur demande une acception pour accéder au site.

1)
N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !
2)
http://httpd.apache.org/docs/2.2/mod/worker.html
3)
http://httpd.apache.org/docs/2.2/mod/prefork.html
4)
http://fr.wikipedia.org/wiki/Common_Gateway_Interface
5)

“De même, soyez méfiant en jouant avec la directive UserDir”
6)

par exemple gandi.net, tustico (peu cher mais reconnu par les navigateur) ou startSSL qui est gratuit (peu fiable bien bien pour essayer).
7)

certaines en fournissent gratitement voir
http://www.planet-libre.org/index.php?post_id=16507&go=external

From:
http://debian-facile.org/ - **Documentation - Wiki**

Permanent link:
http://debian-facile.org/doc:reseau:apache2:tp01

Last update: **01/11/2019 08:43**