apache2 (2.4/JESSIE)

- Objet : installation et configuration d'apache2
- Niveau requis :
  débutant, avisé
- Commentaires : Apprendre à configurer un serveur web sur son réseau local.
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !
- Suivi :
  à-tester
    - Création par Winproof le 16/10/2016
    - Testé par Winproof novembre 2016 sur Jessie
- Commentaires sur le forum : Lien vers le forum concernant ce tuto
- Référence : http://www.apache.org/

Introduction

Ce tuto est une maj de celui d'hypathie (https://debian-facile.org/doc:reseau:apache2:tp01),merci a elle , pour Apache 2.4

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-VM-server”.

Ensuite, on configure le serveur ssh, 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 welcome page used to...
```

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.

```
ls /etc/apache2/
```

retour de la commande

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

4 Fichiers:

apache2.conf: configuration générale du serveur (droits par défaut, emplacement et format des logs,PID, etc...) ainsi que des liens (commande “Include”) vers d'autres fichiers (configuration spécifique, sites et modules)
ports.conf: ports sur lesquels le serveur écoute
envvars: variables d'environnement par defaut

magic: sert au module mime_magic

Les modules sont des éléments permettant d'ajouter des fonctions à apache, comme le php, le ssl, etc...

Sur d'autres distributions linux, la configuration est en grande partie incluse dans le fichier httpd.conf. Sous Debian, il a été décidé d'extraire une partie de ces éléments de configuration pour les avoir sous forme de fichiers séparés (contenus dans le répertoire “conf.d” sous apache 2.2, et dans “conf-available” sous apache 2.4.)
Par exemple, la configuration des options de sécurité se trouve dans /etc/apache2/conf-available/security.conf, celle definisant l'encodage optionel dans /etc/apache2/conf-available/charset.conf, etc..

6 Répertoires:

Les 3 répertoires “*.available” qui contiennent les éléments disponibles (respectivement, la configuration, les sites et les modules) et les 3 répertoires “*.enabled” qui contiennent les éléments activés sur le serveur.

Pour la procédure d'activation, voir plus bas

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

Ce répertoire contient les fichiers (un par site) définissant (chemin des fichiers, droits, etc...) les différents sites disponibles.

Apache utilise des “Virtualhost” (voir plus bas) pour définir la configuration d'un site, y compris pour le site par défaut.
Donc dans la suite de ce document, le fichier de configuration d'un site (/etc/apache2/sites-availables/xxx.conf) sera appelé **fichier vhost** ou juste **vhost**

Puisque après l'installation, il a été possible d'afficher une page web d'accueil, c'est que ce répertoire contient un fichier vhost décrivant le site par défaut:

```
ls /etc/apache2/sites-available/
```

retour de la commande

```
000-default.conf
default-ssl.conf
```

vu que l'on a affiché la page `http://192.168.x.x/` et pas `https://192.168.x.x` (version SSL du site), c'est donc le vhost 000-default.conf (version classique du site par défaut) qui a été utilisé par apache pour afficher la page.

regardons son contenu:

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

retour de la commande

```
<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 www.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
only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
</VirtualHost>

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

On voit la ligne **<DocumentRoot /var/www/html>**. C'est donc là que le serveur apache va aller

chercher les fichiers du site par défaut (fichier index.html, index.php, etc...) et afficher ce qu'il trouve.

Vérifions cela.

```
ls /var/www/html/
```

retour de la commande

```
index.html
```

Si vous regardez le contenu du fichier index.html (je ne met pas le contenu de ce fichier ici, il est trop grand) vous verrez une longue page html, avec le code correspondant a ce que vous affiche apache quand vous visitez la page http://192.168.x.x/.

Donc le vhost “000-default.conf” défini le site par défaut qu'apache affiche quand vous essayez d’accéder au serveur.

Mais ce n'est pas tout!

Si la page d'accueil s'affiche sur le navigateur, c'est que le site par défaut d'apache2 est activé.
En effet, si vous regardez cette fois dans le répertoire “sites-enabled”, vous y trouverez un fichier “000-default.conf”.

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

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

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

retour de la commande

```
000-default.conf
```

Ce fichier est un lien symbolique : il pointe vers “/etc/apache2/sites-available/00-default.conf”

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

retour de la commande

```
total 0
lrwxrwxrwx 1 root root 35 oct.  17 13:44 000-default.conf -> ../sites-
available/000-default.conf
```

L'activation d'un site sous apache se fait donc par la création d'un lien symbolique dans “/etc/apache2/sites-enabled”, lien pointant vers le fichier correspondant dans “/etc/apache2/sites-availables/”.

Pour activer/désactiver un site, il suffit donc de créer/supprimer le lien symbolique qui relie "/etc/apache2/sites-enabled/xxx.conf" avec "/etc/apache2/sites-available/xxx.conf"

Cela se fait avec les utilitaires fournis par apache:

- `a2ensite xxx.conf` : (apache2 enable site) : crée le lien symbolique /etc/apache2/sites-enabled/xxx.conf et donc active le site xxx.conf,
- `a2dissite xxx.conf` : (apache2 disable site) : supprime le lien symbolique, et donc désactive le site xxx.conf.

Il faut ensuite recharger (reload) apache pour qu'il prenne en compte la modification.

Activer un fichier de configuration ou un module fonctionne de la même manière, avec création d'un lien symbolique dans le dossier *.enabled correspondant.

les utilitaires a utiliser sont `a2enconf`, `a2disconf`, `a2enmod` et `a2dismod`

Quand vous utilisez les utilitaires `a2*` , apache vous dit "To activate the new configuration, you need to run: `service apache2 reload`"

Vu que nous somme sous Debian JESSIE, et donc sous systemd, mieux vaut prendre l'habitude d'utiliser `systemctl reload apache2.service` plutôt.

Voila un petit résumé de la façon (en gros) donc apache vous sert un site, mettons le site www.debian.com:

1) vous tapez dans votre navigateur l'adresse http://www.debian.com

2) votre navigateur demande au DNS l'adresse IP du domaine debian.com, puis envoi une requête HTTP a l'adresse IP correspondante.

3) apache reçoit la requête, vérifie si le site demandé existe (lien vers le vhost dans /etc/apache2/sites-enable/),si oui il passe au traitement du fichier correspondant,sinon il vous affiche le site par défaut (ou une erreur 404 si pas de site par défaut).

C'est a ce niveau qu'intervient la notion de VirtualHost (voir plus bas)

4) apache traite le fichier /etc/apache2/sites-availables/debian.com.conf, vérifie les droits, l'existence des fichiers/dossiers indiqués:
si quelque chose bloque, vous aurez une erreur. (4xx si le pb vient de vous, 5xx si c'est le serveur qui plante)

5) enfin si tout est ok, apache vous affiche le site demandé.

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

Configuration d'un site web

Quand vous travaillez sur un site apache, je ne saurait trop vous conseiller de désactiver provisoirement le cache de votre navigateur (ou le mettre a zéro), cela pourra vous éviter des résultats bizarres liés au fait d'avoir oublié de vider le cache)

Création du site web dans /var/www/

Pour qu'un site dans /var/www/ fonctionne, il faut ajouter une ligne dans votre fichier hosts (ou dans le DNS pour un site en prod) pour faire correspondre l'ip de votre serveur apache avec votre site. par exemple :

```
192.168.x.x     www.monsite.com
```

Avec bien entendu le vhost qui va bien
en cas de problèmes d'accès, voir la section "Précisions sur le fonctionnement des virtualhosts" plus bas

Donc pour commencer, allez donc ajouter dans votre fichier hosts (/etc/hosts sous debian, C:\Windows\System32\drivers\etc\hosts sous windows 7) la ligne qui va bien.

Dans beaucoup de tuto apache2 pour debian, il est indiqué de créer son site dans **/var/www/html/** et de créer le vhost.
Mais si vous créez votre site dans /var/www/html, il s'agit alors d'un sous-domaine du site par défaut, et pas d'un vhost!.

Ce qui a pour conséquence qu'en activant le vhost, votre site est bien accessible par http://192.168.x.x/monsite.com, ce qui semble indiquer que tout fonctionne, or c'est faux!
en effet, désactivez le vhost, et vous verrez que votre site http://192.168.x.x/monsite.com est toujours fonctionnel.
car en fait, avoir mis le site sous /var/www/html/ fait qu'il est servi par le vhost par defaut (000-default.conf) et pas par le vhost monsite.com.conf. Du coup toutes les éventuelles directives de sécurité que vous auriez mis dans le vhost monsite.com.conf ne sont pas prise en compte, puisque le vhost lui-même n'est pas utilisé.

Si vous voulez vraiment faire des test sans vhost ni toucher au fichier host, ou si vous n'avez qu'un seul site 😎, c'est une solution.
donc pour y accéder, il suffit d'utiliser http://192.168.x.x/monsite.com.

le problème de cette configuration est que le répertoire du site monsite.com étant sous /var/www/html/, le site monsite.com hérite des autorisations du site par défaut.
on peut contourner ce problème, en définissant dans le vhost par défaut une directive Alias:

```
Alias /monsite.com /var/www/monsite.com
```

–> dit a apache que pour un accès à http://192.168.x.x/monsite.com c'est le repertoire /var/www/monsite.com qui doit étre servi, et pas le repertoire /var/www/html/monsite.com.
cela permet, en définissant une section <Directory> dans le vhost 000-default.conf, de donner a son site d'autres directives de sécurité que celle définies pour le site par défaut.
mais c'est pas franchement propre comme façon de faire  .

- On crée un dossier dans /var/www/ :

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

```
mkdir /var/www/monsite.com/
```

- On crée sa première page index.html

```
vi /var/www/monsite.com/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 le vhost de ce site :

```
vi /etc/apache2/sites-available/monsite.com.conf
```

Sous apache 2.4, le fichier vhost d'un site (donc le fichier dans “/etc/apache2/sites-available”) doit avoir l'extension “.conf”

Contenant:

monsite.com

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

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

Pour un vrai serveur apache, donc accessible sur le net, les options “ServerName” et “ServerAlias” permettrons d’accéder a ce site en tapant, au choix, **http://monsite.com** ou **http://www.monsite.com**
A condition que votre DNS soit correctement configuré.

- 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 :

```
systemctl reload apache2.service
```

- Vérification :

Dans le navigateur : **http://www.monsite.com**

```
Bravo !

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

Ça ne fonctionne pas? vérifiez votre fichier hosts !

Bon, il y a encore un petit problème d'encodage

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-available/charset.conf :

```
vi /etc/apache2/conf-available/charset.conf
```

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

```
AddDefaultCharset UTF-8
```

Et on enregistre !

- Corriger le fichier /etc/apache2/envvars :

```
vi /etc/apache2/envvars
```

On dé-commente la ligne `. /etc/default/locale`

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

Et on enregistre !

- Faire prendre en compte les modifications à apache2 :

```
systemctl reload apache2.service
```

- 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 !
```

Travailler en PHP

Créer la page de test

Nous allons modifier la page de notre site monsite.com, en y ajoutant du code php.

Premièrement, il faut modifier l'extension du fichier index.html pour indiquer a apache qu'il contient du php:

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

Ensuite, ajoutons le code php:

```
vi /var/www/monsite.com/index.php
```

index.php

```
<html>
 <body>
  <h1>Bravo !</h1>
  <p>La mise en place d'un Virtualhost est réussie !</p>
  <?php
  echo "La date du jour est " . date("d/m/Y") . "!\n";
  ?>
 </body>
</html>
```

Et enfin recharger apache :

```
systemctl reload apache2.service
```

- Tester en tapant dans le navigateur :

http://www.monsite.com

```
Bravo !

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

Ben??? Rien n'a changé? En plus le problème d'encodage est revenu?

Bravo, finement observé, ça prouve que vous suivez bien!

En fait, dans une installation basique d'apache, le module php n'est pas activé, en fait il n'est même pas disponible (vous pouvez vérifier en listant le contenu du répertoire “mods-available”, pas de php)

Donc apache ignore le code php dans notre fichier (et se plante a nouveau sur l'encodage, me demandez pas pourquoi, aucune idée)

On va donc installer php

Installer libapache2-mod-php5

Ce paquet casse le MPM worker[1] et engendre l'installation du MPM prefork[2].

(c'est hypathie qui le dit,moi pas savoir, moi bêtement recopier)

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

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

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

retour de la commande

```
/etc/apache2/mods-available/php5.conf
/etc/apache2/mods-available/php5.load
```

Normalement l'activation du module php s'est faite automatiquement, pour le vérifier il suffit de regarder dans le répertoire “mods-enabled”

```
ls /etc/apache2/mods-enabled/php*
```

retour de la commande

```
/etc/apache2/mods-enabled/php5.conf
/etc/apache2/mods-enabled/php5.load
```

Ca roule!

Apparement certains modules apache n'ont qu'un seul fichier (.conf ou .load) alors que d'autres en ont 2, comme php5. là non plus, aucune idée du pourquoi (flemme de chercher)

s'il fallait l'activer manuellement:

```
a2enmod php5
```

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

Donc on relance apache

```
systemctl reload apache2.service
```

pas sûr que ce soit nécessaire,peut-être que apt-get a fait le nécessaire,mais j'ai la flemme de recommencer
que le prochain qui test le tuto veuille bien effacer ce passage si il ne sert a rien.

en attendant, dans le doute je reboote!!!

Plus qu'a vider a nouveau le cache du navigateur, recharger la page, et là :

```
Bravo !

La mise en place d'un Virtualhost est réussie !
La date du jour est 18/10/2016!
```

Youpie, ca fonctionne!

merveilleux, on vient de passer 10 mn pour apprendre comment insérer une date dans un site web

Voila, tout le bordel nécessaire pour faire un zoli site sur des chatons en html et php est installé, amusez-vous!

Le tuto d'origine continu au delà, avec 2 paragraphes concernant la sécurité et la création d'un site en SSL avec génération de certificats locaux, mais les informations qu'on y trouve ne sont absolument plus a jour.
entre autre, la syntaxe des directives de sécurité a bien changée sous apache 2.4.
vous pouvez quand même le consulter (https://debian-facile.org/doc:reseau:apache2), mais je vous conseille trés fortement de tout vérifier avec la doc officielle apache! (http://httpd.apache.org/docs/current/)

Précisions sur le fonctionnement des virtualhosts (ou comment héberger plusieurs sites)

Quelques explications sur les virtualhosts. Ça peut vous éviter de vous arracher les cheveux comme moi avant d'arriver a comprendre pourquoi apache vous sert le site par défaut sur certaines adresses, et votre beau site tout neuf sur d'autres .

Généralités:

Les virtualhosts apache, c'est une méthode pour héberger plusieurs sites web sur la même machine physique. Je ne traiterais que des virtualhosts par nom, les virtualhosts par IP faisant intervenir des notions de réseau plus poussées.

En gros, quand vous tentez de vous connecter a un site web (par exemple www.debian.com), le client (vous) utilise le DNS (ou le fichier hosts en cas de test sur une machine virtuelle) pour obtenir l'IP correspondant au domaine debian.com , puis il envoi une requête HTTP sur le port 80 de l'IP correspondante.

Cette requête contient (entre autre) un champ “Host” (dans l’en-tête de la requête) correspondant au nom du serveur demandé.
ce champ est rempli a partir de l'URL que vous avez tapé, par exemple:

URL http://www.debian.com --> champ host rempli avec www.debian.com

URL http://192.168.x.x --> champ host rempli avec 192.168.x.x

https://fr.wikipedia.org/wiki/Hypertext_Transfer_Protocol#/media/File:Requ%C3%AAte_HTTP.png
https://openclassrooms.com/courses/les-requetes-http

c'est la valeur de ce champ qu'apache va utiliser pour savoir quel site servir. Apache va parcourir ses différents fichiers vhost pour rechercher les directives “ServerName” et “ServerAlias” et servir le répertoire correspondant (défini par “DocumentRoot” dans le vhost).

Apache lit ses fichiers vhost dans l'ordre alphabétique, et sert le premier si aucune correspondance n'a été trouvée dans les autres vhost. C'est pour cela que le vhost par défaut est nommé 000-default.conf, pour qu'il soit en premier dans l'ordre alphabétique.
cela veut dire que si vous désactivez le vhost par defaut, le suivant dans l'ordre alphabétique deviendra le site par défaut.

Si vous tentez de désactiver le vhost par défaut (via a2dissite 000-default.conf) sans avoir un autre vhost, apache continuera a vous servir le contenu de /var/www/html, c'est un comportement propre a apache 2.4, mis en place pour des raisons de sécurité.
si vous voulez vraiment désactiver le vhost par defaut (et donc avoir une erreur 404 au lieu de la page “it's work”) il faut supprimer le dossier /var/www/html

Du coup, si vous avez besoin d’héberger plusieurs sites, c'est simple:
dans votre DNS, faite pointer chaque site sur l'IP de votre serveur.
puis au niveau apache, un fichier vhost par site pour indiquer a apache quel dossier servir pour chaque site.

Pour aller plus loin

Voila, vous avez les bases pour utiliser apache en mode “test” sur une VM.

Si vous voulez passer votre site en prod, il y a beaucoup plus de choses a régler.

renseignez-vous sur le SSL, la configuration d'un firewall, et la sécurisation d'un serveur apache.

Je vous conseille de lire le tuto d'hypathie, même si il n'est plus entièrement a jour, il est pleins de bon conseils, et trouver ce qui a changé en version apache 2.4 ne pourra que vous aider a comprendre le fonctionnement du bestiau
https://debian-facile.org/doc:reseau:apache2

Bon courage!

1)

http://httpd.apache.org/docs/2.2/mod/worker.html

2)

http://httpd.apache.org/docs/2.2/mod/prefork.html