OFPPT مكتب التكوين المهني وإنعاش الشغل

Office de la Formation Professionnelle et de la Promotion du Travail

Complexe de Formation dans les Métiers des Nouvelles Technologies de l'Information, de l'Offshoring et de l'Electronique -Oujda

Module : Administration d'un Réseau sous Linux

Installation et configuration du serveur Apache sous Centos

Formatrice : ZITI Iham

Sommaire

1. Introduction :

Apache est le principal serveur web du monde de l'Open Source. À l'origine, c'était la continuation du serveur libre développé par le NCSA (National Center for Supercomputing Applications) à l'Université de l'Illinois. Lorsque le projet officiel a été abandonné en 1994, une équipe de développeurs volontaires a continué à fournir du code sous forme de nombreux correctifs, ce qui explique la genèse du nom a patchy server, c'est-à-dire « serveur rafistolé ».

D'après les statistiques de Netcraft, un peu moins de la moitié des sites Web du monde tournent sur un serveur Apache.

2. Installation

Apache est disponible dans les référentiels de logiciels par défaut de CentOS, ce qui signifie que vous pouvez l'installer avec le gestionnaire de paquets yum. Le nom du paquet est httpd

Commencer par lancer la mise à jour du paquet

```
#yum  update  httpd
```

Une fois les packages mis à jour, installer le package Apache:

```
#yum install httpd
```

L'installation du paquet crée un utilisateur système apache et un groupe système apache correspondant.

```
[root@ntic named]# tail  -n 2 /etc/passwd
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
ldap:x:55:55:OpenLDAP server:/var/lib/ldap:/sbin/nologin
[root@ntic named]#
```

3. Lancement du serveur

Sous Red Hat et CentOS, Apache est préconfiguré pour afficher une page statique par défaut. Il suffit d'activer et de lancer le service.

```
#systemctl enable httpd
#systemctl start httpd
```

Pour vérifier si le serveur est lancé utiliser la commande :

```
#systemctl status httpd
```

```
[root@ntic named]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: dis
bled)
   Active: active (running) since mar. 2019-10-15 19:09:24 CEST; 27s ago
     Docs: man:httpd(8)
           man:apachectl(8)
 Main PID: 11905 (httpd)
   Status: "Total requests: 0; Current requests/sec: 0; Current traffic:   0 B/sec"
    Tasks: 6
   CGroup: /system.slice/httpd.service
           ├─11905 /usr/sbin/httpd -DFOREGROUND
           ├─11917 /usr/sbin/httpd -DFOREGROUND
           ├─11919 /usr/sbin/httpd -DFOREGROUND
           ├─11920 /usr/sbin/httpd -DFOREGROUND
           ├─11921 /usr/sbin/httpd -DFOREGROUND
           └─11922 /usr/sbin/httpd -DFOREGROUND

oct. 15 19:08:23 ntic systemd[1]: Starting The Apache HTTP Server...
oct. 15 19:08:54 ntic httpd[11905]: AH00558: httpd: Could not reliably determine ...ag
```

4. Configuration

Le fichier de configuration du Serveur HTTP Apache est /etc/httpd/conf/httpd.conf. Dans le fichier httpd.conf figurent de nombreux commentaires qui rendent sont contenu très explicite. La configuration par défaut fonctionne dans la plupart des situations ; cependant, il est important de bien connaître certaines des options de configuration les plus importantes.

Les distributions modernes ont donc tendance à répartir la configuration d'Apache sur une série de fichiers *.conf répartis dans les répertoires /etc/httpd/conf.d et /etc/httpd/conf.modules.d, rattachés au fichier principal /etc/httpd/conf/httpd.conf par la directive Include.

4.1 Directives du fichier

Les principales directives du fichier de configuration sont :

La directive **ServerRoot** permet de définir le répertoire dans lequel le serveur est installé.

```
ServerRoot "/etc/httpd"
```

La directive **Listen** permet à Apache d'écouter sur des adresses ou des ports spécifiques. Notons que cette directive est requise. Si elle est absente du fichier de configuration, Apache refuse de démarrer.

```
Listen 80
```

La directive **Include** permet l'inclusion d'autres fichiers de configuration dans le fichier de configuration principal du serveur.

IncludeOptional fonctionne comme **Include**, au détail près que la directive ne produira pas une erreur au cas où le métacaractère * ne correspond à aucun fichier.

Le chemin est relatif par rapport à l'emplacement spécifié dans la directive ServerRoot.

```
Include conf.modules.d/*.conf
...
IncludeOptional conf.d/*.conf
```

Apache ne tourne pas en tant que root, mais en tant qu'utilisateur spécial défini par les directives **User** et **Group**

Plus précisément, il est lancé par root pour ensuite changer de propriétaire.

```
User apache
Group apache
```

L'adresse mail de l'administrateur, définie par la directive **ServerAdmin,** apparaîtra sur certaines pages générées par le serveur, notamment les pages d'erreur.

```
ServerAdmin root@localhost
```

La directive **DocumentRoot** permet de définir le répertoire à partir duquel Apache va servir des fichiers.

```
DocumentRoot "/var/www/html"
```

Les balises **<Directory>** et **</Directory>** permettent de regrouper un ensemble de directives qui ne s'appliquent qu'au répertoire précisé, à ses sous-répertoires, et aux fichiers situés dans ces sous-répertoires.

```
<Directory />
  AllowOverride none
  Require all denied
</Directory>
...
<Directory "/var/www">
  AllowOverride None
  Require all granted
</Directory>

<Directory "/var/www/html">
  Options Indexes FollowSymLinks
  AllowOverride None
  Require all granted
</Directory>
```

- **AllowOverride :** ne peut être utilisée que dans les sections <Directory>. À partir du moment où elle est définie à none, les fichiers .htaccess sont totalement ignorés.

- **Require** permet de contrôler l'accès au système de fichiers du serveur. Require all denied bloque l'accès pour tous les utilisateurs, Require all granted autorise tout le monde.
- **Options** permet d'activer ou de désactiver une série d'options (ou de comportements) pour un répertoire donné. Ici par exemple, l'option **Indexes** affiche la liste des fichiers d'un répertoire en cas d'absence de fichier index.html. **FollowSymlinks** permet de suivre les liens symboliques.

Les balises **<IfModule> et </IfModule>** contiennent des directives qui ne s'appliquent qu'en fonction de la présence ou de l'absence d'un module spécifique. La directive DirectoryIndex spécifie le fichier à envoyer par Apache lorsqu'une URL se termine par / et concerne un répertoire entier.

```
<IfModule dir_module>
  DirectoryIndex index.html
</IfModule>
```

Les balises **<Files> et </Files>** contiennent des directives qui s'appliquent aux fichiers précisés.Ici par exemple, on interdit l'accès à tous les fichiers dont le nom commence par .ht, notamment .htaccess.

```
<Files ".ht*">
  Require all denied
</Files>
```

La directive **ErrorLog** définit le chemin vers le journal des erreurs. La verbosité de ce journal est contrôlée par la directive **LogLevel.**

```
ErrorLog "logs/error_log"
LogLevel warn
```

La directive **CustomLog** permet de contrôler la journalisation des requêtes destinées au serveur. Elle définit le nom et le format du fichier journal.

```
CustomLog "logs/access_log" combined
```

La directive **Timeout** définit la durée, exprimée en secondes, pendant laquelle le serveur attend des réceptions et des émissions pendant les communications. La valeur de Timeout est paramétrée sur 300 secondes par défaut, ce qui est approprié pour la plupart des situations.

```
Timeout 300
```

La directive **MaxClients** fixe une limite au nombre total de processus serveur ou de clients connectés simultanément qui peuvent s'exécuter en même temps. L'objectif principal de cette directive est d'éviter qu'un Serveur HTTP Apache surchargé n'entraîne le plantage de votre système d'exploitation. Pour des serveurs très solicités, cette valeur devrait être élevée. La valeur par défaut du serveur est 150.

```
MaxClients  150
```

La directive **LogLevel** définit le niveau de détail avec lequel les messages d'erreur devraient être enregistrés dans les journaux d'erreurs. Les valeurs possibles de LogLevel sont (du niveau le moins détaillé au niveau le plus détaillé) emerg, alert, crit, error, warn, notice, info ou debug. La valeur par défaut donnée à LogLevel est warn.

```
LogLevel   warn
```

La directive **KeepAliveTimeout** définit la durée exprimée en secondes pendant laquelle le serveur attend après avoir servi une requête, avant d'interrompre la connexion. Une fois que le serveur reçoit une requête, c'est la directive Timeout qui s'applique à sa place. Par défaut, la valeur donnée à la directive KeepAliveTimeout est de 15 secondes.

```
KeepAliveTimeout 15
```

La directive **KeepAlive** définit si votre serveur autorisera plus d'une requête par connexion ; cette directive peut servir à empêcher un client particulier d'utiliser une trop grande quantité des ressources dont le serveur est doté.

Par défaut, la valeur de Keepalive est réglée sur off. Si la valeur de Keepalive est on et que le serveur devient très occupé, il peut générer rapidement le nombre maximum de processus enfants. Dans ce cas, le serveur sera considérablement ralenti. Si la directive Keepalive est activée, il est recommandé de donner à KeepAliveTimeout une valeur basse

```
KeepAlive off
```

La directive **MaxKeepAliveRequests** définit le nombre maximum de requêtes autorisées par connexion persistante. L'organisation Apache Project recommande l'utilisation d'un paramétrage élevé, ce qui entraîne une amélioration des performances du serveur. Par défaut, la valeur de MaxKeepAliveRequests paramétrée sur 100 est approprié pour la plupart des situations.

```
MaxKeepAliveRequests  100
```

La directive **AddDefaultCharset** paramètre le jeu de caractères par défaut pour les pages de texte. Lorsqu'elle est désactivée **(AddDefaultCharset off),** Apache prend en compte l'encodage spécifié dans la balise **<meta>** des fichiers HTML à envoyer au navigateur.

```
meta http-equiv="Content-Type" content="text/html; charset=utf-8"
```

Ici en revanche, Apache utilise d'emblée le jeu de caractères spécifié en ignorant la balise **<meta>.**

```
AddDefaultCharset UTF-8
```

Tester la nouvelle configuration.

```
# apachectl configtest
```

Prendre en compte les modifications.

```
sudo systemctl reload httpd
```

5. Héberger plusieurs sites sur un serveur local

Le principe des hôtes virtuels (*Virtual Hosts*) consiste à faire fonctionner un ou plusieurs sites Web sur une même machine. L'utilisateur final ne perçoit pas qu'en fait il s'agit d'un même serveur physique.

Voici l'exemple de mon site de développement : **ntic.local**

5.1 Création du répertoire du site

Il faut commencer par la création du répertoire pour le site

```
#mkdir -p  /var/www/html/ntic.local
```

Modifier les droits :

```
#chown -R apache:apache ntic.local
#chmod -R 755 ntic.local
```

Créez ensuite un exemple de page index.html

```
#vim /var/www/html/ntic.lcal/index.html
```

Appuyez sur i pour passer en mode INSERT et ajoutez l'exemple HTML suivant au fichier:

```
<html>
  <head>
    <title>Welcome to ntic.local</title>
  </head>
  <body>
    <h1>l'Exemple de virtual host fonctionne </h1>
  </body>
</html>
```

5.2 Configuration httpd.conf

D'abord, s'assurer que dans le fichier **httpd.conf** on ait une ligne de ce style'assurer que dans le fichier **httpd.conf** on ait une ligne de ce style :

```
Include /etc/httpd/conf.d /*.conf
```

Grâce à cette ligne, on inclura tous les **.conf** de **/etc/httpd/conf.d.** Tous les "virtual hosts" vont être des fichiers du style **xxx.conf** dans **/etc/httpd/conf.d**

5.3 Création des Virtual Hosts

Créer un fichier **/etc/httpd/conf.d/nomdomaine.conf**. Ce fichier définira le site affiché par défaut, c'est-à-dire lorsqu'on invoque l'adresse IP ou le nom d'hôte de la machine.

```
#touch   /etc/httpd/conf.d/ntic.local.conf
```

Exemple 1:

```
<VirtualHost *:80>
     ServerAdmin admin@ntic.local
     ServerName  ntic.local
     ServerAlias www.ntic.local
     DocumentRoot  /var/www/html/ntic.lcal/
     ErrorLog /var/log/httpd/ntic/error.log
     CustomLog /var/log/httpd/ntic/access.log combined
</VirtualHost>
```

- **<VirtualHost *:80> :** Adresse IP de la machine serveur ou *, suivie du port 80 qui est le port http
- **ServerAlias www.ntic.local** : Nom alternatif du serveur
- **DocumentRoot /var/www/html/ntic.lcal/** : le chemin de l'accès au fichier index.html
- **ErrorLog /var/log/httpd/error.log** : permet de définir le nom du fichier dans lequel le serveur va journaliser toutes les erreurs qu'il rencontre

- **CustomLog /var/log/httpd/access.log combined** : permet de contrôler la journalisation des requêtes destinées au serveur.

Exemple 2 :

```
<VirtualHost *:80>
        ServerName ntic.local
        ServerAlias www.ntic.local
        ServerAdmin admin@ntic.local
        ErrorLog /var/log/httpd/ntic/error.log
        TransferLog /var/log/httpd/ntic _log
        DocumentRoot "var/www/html/ntic.lcal"
        <Directory "/var/www/html/ntic.lcal/">
                Options Indexes FollowSymLinks
                AllowOverride All
                Require all granted
        </Directory>
</VirtualHost>
```

- **Require all denied** : Tout refuser
- **Require all granted** : Tout accepter
- **Require host example.org** : Accepter example.org
- **Require ip 1.2.3.4** : Accepter 1.2.3.4
- **Require not ip 1.2.3.4** : Refuser 1.2.3.4

6. Ecouter sur plusieurs ports

Si on souhaite ajouter un vhost qui écoute sur un autre port, il suffit d'ajouter une directive listen en plus dans le fichier de configuration du vhost :

```
Listen 8080
<VirtualHost *:8080>
  #Le contenu du vhost
</VirtualHost>
```

7. Test de configuration

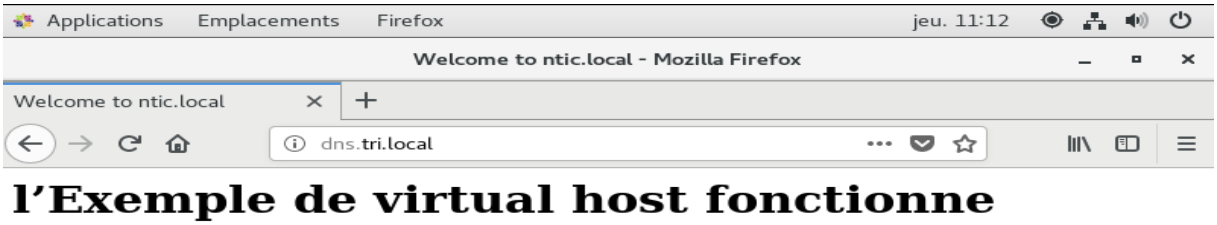
Tester la nouvelle configuration.

```
# apachectl configtest
```

Prendre en compte les modifications.

```
sudo systemctl reload httpd
```

Vérifier l'accès via un navigateur



8. Sécuriser Apache2 avec SSL

8.1 Le protocole SSL

SSL est un protocole qui a été développé par la société Netscape. Ce protocole permet à deux machines de communiquer de manière sécurisée. Les informations échangées entre les deux machines sont de ce fait inviolables.

Le protocole SSL se traduit par la combinaison de deux protocoles bien distincts (*Handshake* et *Record*) qui permettent la négociation entre les deux machines et le chiffrement des données échangées.

8.2 Les Certificats

Un certificat permet de fournir diverses informations concernant l'identité de son détenteur (la personne qui publie les données). Ce certificat s'accompagne d'une **clé publique** qui est indispensable pour que la communication entre les machines soit chiffrée.

De même, afin de garantir l'authenticité du certificat, ce dernier est signé numériquement par le biais d'une **clé privée** provenant soit d'un organisme officiel (Société spécialisée dans la certification) soit par le détenteur du Certificat lui-même. Dans ce dernier cas, on parlera de certificat auto-signé.

Dans la plupart des cas, l'obtention d'un Certificat certifié par une autorité officielle ayant un prix assez élevé, les webmasters auront tendance à vouloir signer eux-mêmes leur certificat. Ce faisant, il est à noter que dans ce cas, le certificat ne sera pas reconnu par les navigateurs web comme étant certifié.

CA Cert permet d'obtenir des certificats gratuits. Il vous faudra néanmoins installer le certificat racine dans votre navigateur.

8.3 Activation du module SSL

Pour que le protocole SSL puisse fonctionner avec le Serveur HTTP Apache2, il faut installer le module **SSL** avec la commande :

```
#yum install mod_ssl
```

8.4 Création du certificat

On peut créer son certificat SSL auto signé en installant le paquet **openssl**.

```
openssl req -x509 -nodes -days 365 -newkey rsa:1024 -out /etc/httpd/server.crt -keyout /etc/httpd/server.key
```

Explications :

- **-x509 -nodes** donne le type de certificat voulu
- **-days 365** indique la durée de validité (en jours) de votre certificat
- **-newkey rsa:1024** demande une clé RSA de 1024 bits - d'après la doc apache, il est déconseillé de créer une clé plus grosse pour des histoires de compatibilité
- **-out /etc/httpd/server.crt** est le chemin de votre certificat
- **-keyout /etc/httpd/server.key** est le chemin de la clé privée

Répondez alors aux questions posées :

```
Country Name (2 letter code) [GB]:
```
Entrez **FR** pour indiquer que vous êtes en France et validez par la touche « Entrée »

```
State or Province Name (full name) [Some-State]:
```
Entrez **Maroc** et validez par la touche « Entrée »

```
Locality Name (eg, city) []:
```
Indiquez ici le nom de votre ville et validez par la touche « Entrée »

```
Organization Name (eg, company; recommended) []:
```
Indiquez le nom de votre organisation, de votre société. (*exemple :* **ofppt**) et validez par la touche « Entrée ». Si vous n'avez pas de société, vous pouvez mettre un nom fictif, le nom de notre site Web par exemple.

```
Organizational Unit Name (eg, section) []:
```

Indiquez ici le nom de la section de votre organisation, de votre société. Si vous n'en avez pas, mettez la même chose que pour la question précédente.

```
Common Name (eg, YOUR name) []:
```

Ici, il convient de faire particulièrement attention à ce que vous allez entrer. Vous devez indiquer le *nom de domaine* que vous désirez sécuriser.

Email Address []:

Ici, il s'agit d'indiquer l'adresse E-mail de l'administrateur. Nous terminons bien entendu en validant par la touche « Entrée ».

```
[root@ntic conf.d]# openssl req -x509 -nodes -days 365 -newkey rsa:1024 -out /etc/httpd/server.crt -
Generating a 1024 bit RSA private key
...........++++++
.................++++++
writing new private key to '/etc/httpd/server.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:MO
State or Province Name (full name) []:Oujda
Locality Name (eg, city) [Default City]:Oujda
Organization Name (eg, company) [Default Company Ltd]:ofppt
Organizational Unit Name (eg, section) []:NTIC
Common Name (eg, your name or your server's hostname) []:TRI
Email Address []:admin@tri.local
[root@ntic conf.d]#
```

8.5 Configuration Apache2

Par défaut, Apache2 est configuré pour écouter sur le port 80. Il s'agit là de la configuration usuelle d'un Serveur Web. Cependant, le protocole SSL a besoin d'un port spécifique pour pouvoir fonctionner. Il s'agit du **port 443**. Il faut vérifier dans le fichier « /etc/httpd/conf.d/ssl.conf » que la directive de configuration nommée **Listen** écouter sur le port 443

Pour sécuriser cet Hôte Virtuel, nous allons donc devoir modifier ce fichier en y ajoutant un hôte virtuel accessible sur le **port 443**, ce dernier contenant des directives particulières qui sont les suivantes :

1. Directive **SSLEngine** :
   Cette directive permet d'activer le moteur SSL au sein d'un hôte virtuel, Elle peut prendre deux arguments –> **on/off**
2. Directive **SSLCertificateFile** :
   Cette directive définit le certificat authentifiant le Serveur auprès des clients.
   L'argument est le chemin d'accès au certificat. En ce qui nous concerne, le certificat se trouve dans le répertoire **/etc/apache2/**
3. Directive **SSLCertificateKeyFile** :
   Cette directive définit la clé privée du Serveur utilisée pour signer l'échange de clé entre le client et le serveur. Elle prend en argument le chemin d'accès à la clé (fichier). Dans notre cas, la clé se trouve dans le répertoire **/etc/apache2/**.

Enfin, afin que les clients puissent continuer d'accéder au site Web en tapant une url de type **http** et non **https**, nous allons modifier l'hôte virtuel accessible sur le **port 80** en remplaçant la directive **DocumentRoot** par une directive de redirection.

Voici donc le contenu de notre fichier une fois modifié :

```
<VirtualHost *:80>
   ServerName ntic.local/
   Redirect / https://ntic.local/
</VirtualHost>

<VirtualHost 192.168.0.2:443>
        ServerName ntic.local
        DocumentRoot /var/www/ntic

        SSLEngine on
        SSLCertificateFile /etc/httpd/server.crt
        SSLCertificateKeyFile /etc/httpd/server.key
</VirtualHost>
```

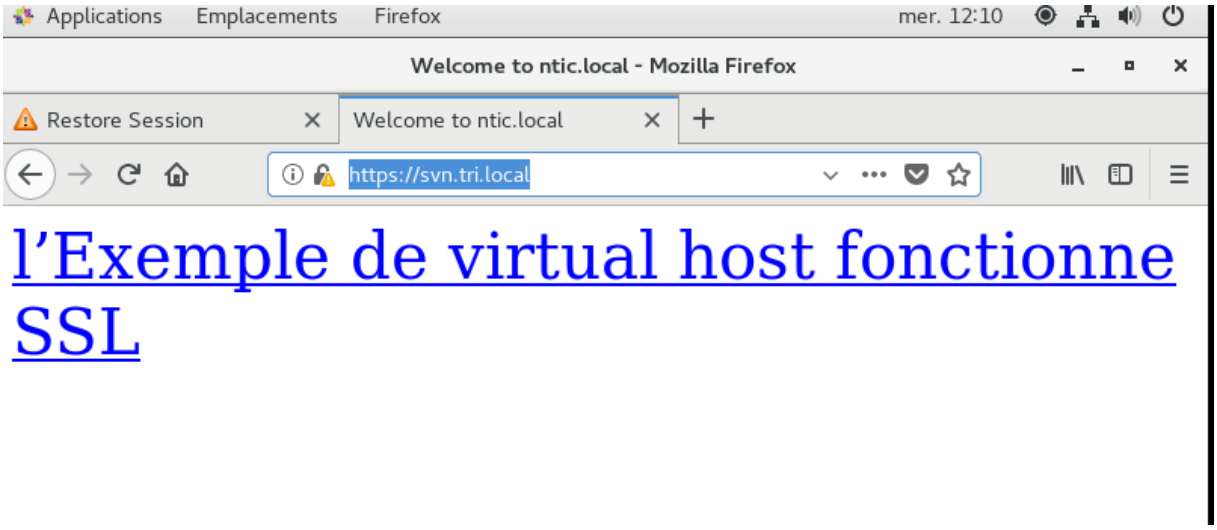
8.6 Relance du serveur HTTP Apache2

Afin que les modifications que nous venons d'effectuer soient prises en compte, nous devons demander au **Serveur Http Apache2** de relire ses fichiers de configuration.

Pour ce faire, il suffit de taper la commande suivante dans un terminal :

```
#systemct restart httpd.service
```

8.7 Test



9. Référence

https://www.microlinux.fr/apache-centos-7/

http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-fr-4/s1-apache-config.html

https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-centos-7

https://www.linuxtricks.fr/wiki/virtual-hosts-avec-apache-vhosts