TP Service sur Réseaux N°5 - MMI 2

L'objectif de ce TP est d'installer un serveur web. Après avoir installé Linux, c'est la deuxième brique du fameux LAMP (Linux / Apache / MySQL / PHP). Apache est l'un des serveurs web les plus utilisé à travers le monde[1]. Ce TP s'appuie sur la version courante d'apache, à savoir la 2.4 en janvier 2015.

Introduction

Ouvrez VirtualBox.

Démarrez votre VM.

Une fois la VM démarrée, ouvrez le menu et lancez Konsole.

Installation d'apache

Exécutez la commande suivante afin d'installer `apache` :

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

Une fois l'installation terminée, vous pouvez vérifier l'état du serveur par la commande suivante :

```
sudo service apache2 status
```

Configuration d'apache

La configuration d'apache se situe dans le répertoire `/etc/apache2` :

```
/etc/apache2/
|-- apache2.conf
|       `-- ports.conf
|-- mods-enabled
|-- sites-enabled
`-- conf-enabled
```

- `apache2.conf` est le fichier principal de configuration, il se charge également d'inclure les autres fichiers de configuration

- `ports.conf` est chargé depuis le fichier de configuration principal `apache2.conf`

Les fichiers de configurations situés dans les répertoires `mods-enabled/`, `conf-enabled/` and `sites-enabled/` contiennent des fragments de configuration permettant de respectivement gérer les modules, la configuration globale ou les hôtes virtuels.

Ils sont activés en créant des liens symboliques depuis le répertoire correspondant `*-available/` ou bien par les utilitaires `a2enmod`, `a2dismod`, `a2ensite`, `a2dissite`, `a2enconf` et `a2disconf`.

Vous pouvez alors accéder à la page par défaut depuis firefox :

`http://localhost`

Modules

Apache est un serveur modulaire et la plupart des fonctionnalités sont implémentées dans des modules externes que le programme charge pendant son initialisation. La configuration par défaut n'active que les modules les plus courants et les plus utiles.

La liste complète des modules standards d'Apache est disponible sur le site :

http://httpd.apache.org/docs/2.4/mod/index.html

1 http://w3techs.com/technologies/market/web_server

La commande `a2enmod <module>` permet d'activer un nouveau module tandis que `a2dismod <module>` désactive un module.

Virtual Hosts

Un Virtual Host est la possibilité offerte par apache d'offrir plusieurs sites web différents sur une même plateforme physique.

Apache distingue deux types de virtual hosts :

- – IP-based : c'est à dire ceux qui utilisent l'adresse IP afin de déterminer quel le virtual host correspondant. Cette méthode nécessite une adresse IP différente pour chaque site.
- – Named-based : le serveur apache va utiliser le contenu du header HTTP émis par le client afin de déterminer quel virtual host il va servir.

  Généralement, c'est la deuxième solution qui est utilisée, compte tenu de la rareté des IP publiques.

Exercice 1 : création de www.labo.fr

En premier lieu, désactivez le site par défaut grâce à `a2dissite`.

Chaque hôte virtuel est ensuite décrit par un fichier placé dans le répertoire `/etc/apache2/sites-available/`. Ainsi, la mise en place du domaine `www.labo.fr` se résume à créer le fichier ci-dessous puis à l'activer avec la commande `a2ensite`.

```
sudo nano /etc/apache2/sites-available/www-labo-fr.conf
```

contenant ceci :

```
<VirtualHost *:80>
        ServerName www.labo.fr
        DocumentRoot /var/www/www.labo.fr
</VirtualHost>
```

`ServerName` définit le nom utilisé pour le vhost. `DocumentRoot` définit le dossier racine dans lequel seront stockés les fichiers du site.

On va ensuite créer une page d'accueil personnalisée.

```
sudo mkdir -p /var/www/www.labo.fr
sudo nano /var/www/www.labo.fr/index.html
```

contenant ceci :

```
<html><body><h1>Bienvenue sur www.labo.fr</h1>
<p>En construction ...</p>
</body></html>
```

Il faut maintenant activer le vhost :

```
sudo a2ensite www-labo-fr
```

Pour terminer, il faut demander à Apache de recharger les fichiers de configuration :

```
sudo apache2ctl graceful
```

Exercice 2 : création de test.labo.fr

Créez un site `test.labo.fr`, avec une page par défaut contenant « En construction ».

Configuration avancée des Virtual hosts

Comme on vient de le voir, les balises **<VirtualHost>** et **</VirtualHost>** permettent de créer un conteneur soulignant les caractéristiques d'un hôte virtuel. Le conteneur **VirtualHost** accepte la plupart des directives de configuration.

Les balises **<Directory /path/to/directory>** et **</Directory>** créent un conteneur utilisé pour entourer un groupe de directives de configuration devant uniquement s'appliquer à ce répertoire et à ses sous-répertoires. Toute directive applicable à un répertoire peut être utilisée à l'intérieur de balises Directory.

```
<Directory /var/www>
      Options Includes FollowSymlinks
      AllowOverride All
      DirectoryIndex index.php index.html index.htm
      Require ip 10.40.0.0/16
</Directory>
```

La directive **Options** contrôle les fonctionnalités spécifiques du serveur qui sont disponibles dans un répertoire particulier. Elle est suivie d'une liste d'options à activer. L'option `None` désactive toutes les options. Inversement, l'option `All` les active toutes (sauf `MultiViews` voir la doc pour plus d'info). Voici les options existantes :

- – `ExecCGI` indique qu'il est possible d'exécuter des scripts CGI.
- – `FollowSymlinks` indique au serveur qu'il doit suivre les liens symboliques et donc effectuer la requête sur le fichier réel qui en est la cible.
- – `SymlinksIfOwnerMatch` a le même rôle mais impose la restriction supplémentaire de ne suivre le lien que si le fichier pointé appartient au même propriétaire.
- – `Includes` active les inclusions côté serveur SSI (Server Side Includes). Il s'agit de directives directement intégrées dans les pages HTML et exécutées à la volée à chaque requête.
- – `Indexes` autorise le serveur à retourner le contenu du dossier si la requête HTTP pointe sur un répertoire dépourvu de fichier d'index (tous les fichiers de la directive DirectoryIndex ayant été tentés en vain).
- – `MultiViews` active la négociation de contenu, ce qui permet notamment au serveur de renvoyer la page web correspondant à la langue annoncée par le navigateur web.

  La directive **AllowOverride** définit si des Options peuvent être annulées par les instructions présente dans un fichier `.htaccess`. Par défaut, aussi bien le répertoire racine que le répertoire `DocumentRoot` sont paramétrés pour ne permettre aucune annulation via `.htaccess`.

  La directive **DirectoryIndex** précise la liste des fichiers à essayer pour répondre à une requête sur un répertoire (une URL se terminant par `/`). Le premier fichier existant est appelé pour générer la réponse. S'il ne trouve aucun des fichiers et que `Options Indexes` est paramétré pour ce répertoire, le serveur génère et renvoie une liste au format HTML, des sous-répertoires et fichiers contenus dans le répertoire

  (à moins que la fonctionnalité de listage des répertoires ne soit désactivée).

La directive **Require** est utilisée afin de vérifier si l'utilisateur se voit accorder ou refuser l'accès à une ressource. `mod_authz_core` met à disposition les fournisseurs d'autorisation génériques suivants :

Require `all granted`

L'accès est autorisé sans restriction.

Require `all denied`

L'accès est systématiquement refusé.

Require `env env-var [env-var] ...`

L'accès n'est autorisé que si l'une au moins des variables d'environnement spécifiées est définie.

Require `method http-method [http-method] ...`

L'accès n'est autorisé que pour les méthodes HTTP spécifiées.

Require `expr expression`

L'accès est autorisé si expression est évalué à vrai. La syntaxe de l'expression est décrite dans la documentation de ap_expr : http://httpd.apache.org/docs/2.4/expr.html.

Voici quelques exemples de syntaxes autorisées par `mod_authz_user`, `mod_authz_host` et `mod_authz_groupfile` :

Require `user identifiant_utilisateur [identifiant utilisateur] ...`

Seuls les utilisateurs spécifiés auront accès à la ressource.

Require `group nom_groupe [nom groupe] ...`

Seuls les utilisateurs appartenant aux groupes spécifiés auront accès à la ressource.

Require `valid-user`

Tous les utilisateurs valides auront accès à la ressource.

Require `ip 192.168.1.104 192.168.1.205`

Une adresse IP complète

Require `ip 10 172.20 192.168.2`

Une adresse IP partielle

Require `ip 10.1.0.0/255.255.0.0`

Une paire réseau/masque de sous-réseau

Require `ip 10.1.0.0/16`

Une spécification CIDR réseau/nnn :

Require `host .net example.edu`

Un nom de domaine, même partiel

Require `local`

Le serveur local

Attention 	Les directives **Allow**, **Deny**, et **Order** fournies par le module `mod_access_compat` sont obsolètes, et sont appelées à disparaître dans les versions futures. Il est donc déconseillé de les utiliser, et de se fier aux tutoriels qui recommandent leur utilisation

La directive `Alias` permet d'accéder aux répertoires se trouvant en dehors du répertoire `DocumentRoot`. Toute URL se terminant par un alias sera automatiquement convertie en chemin d'accès vers l'alias.

```
Alias /doc/ "/usr/share/doc/"
<Directory "/usr/share/doc/">
      Options Indexes MultiViews FollowSymLinks
</Directory>
```

Exercice 3

Dans votre domaine `www.labo.fr`, créez un dossier public. Désactivez toutes les options.

Exercice 4

Configurez votre domaine `www.labo.fr` pour qu'il n'accepte que des requêtes de l'adresse IP votre serveur, qu'il recherche la page default.html par défaut sinon celui-ci retournera le contenu du dossier (sous-répertoires et fichiers) au format HTML.

Exercice 5

Configurez votre site web afin que `www.labo.fr/test` soit un alias renvoyé vers le répertoire `/home/utilisateur/test`.