Administration Linux - Apache

Sommaire

*Un compte-rendu au format texte (**UTF-8**) devra être rédigé et envoyé à l'adresse*
tvaira@free.fr
La convention de nommage pour les compte-rendus est la suivante :
admin-linux-apache-nom.txt

Introduction

Mise en situation

Vous devez disposer d'un PC possédant un système d'exploitation Linux ou Windows et du logiciel de virtualisation *VirtualBox*. Le système invité sera une installation du **serveur Ubuntu 12.10**.

Apache HTTP Server

Le logiciel libre *Apache HTTP Server* (Apache) est un **serveur HTTP** créé et maintenu au sein de la fondation Apache. C'est le serveur HTTP le plus populaire du *World Wide Web*.

La version 2 d'Apache propose entre autres le support de plusieurs plates-formes (Windows, Linux et UNIX, Solaris, BSD, MAC OS X), le support de processus légers UNIX (*threads*), une nouvelle API et le support IPv6. [Source : http ://fr.wikipedia.org/]

Installation

Deux versions du logiciel Apache (1.3 et 2.2) sont disponibles. On installera la version plus récente.

L'installation du paquet `apache2` entraîne l'installation par défaut de `apache2-mpm-worker`, une version particulière de ce serveur web.

```
# apt-get install apache2
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
  apache2-mpm-worker apache2-utils apache2.2-bin apache2.2-common libapr1 libaprutil1 libaprutil1-
      dbd-sqlite3 libaprutil1-ldap ssl-cer
...
```

Le paquet apache2 ne contient rien, il sert simplement à s'assurer qu'une des versions de Apache 2 est effectivement installée.

Les différentes versions d'Apache 2 se distingue par la politique qu'ils emploient pour gérer le traitement parallèle d'un grand nombre de requêtes. On connaît trois modes de fonctionnement qui changent notamment les performances du serveur HTTP : les modes **Prefork**, **Worker** et **Event**.

- Historiquement, Apache fonctionne en **Prefork**, ce qui signifie qu'un processus père démarre préalablement des processus enfants qui traiteront chacun un certain nombre de requêtes clients. Cependant, sous Linux, la multiplication des processus provoque une augmentation de consommation de ressources (mémoire, descripteurs de fichiers).
- En mode **Worker**, Apache lance des *threads* (des processus légers) qui géreront les demandes entrantes. La différence est qu'il s'agit d'un mode plus préemptif dans lequel le processus père prépare les ressources pour ses *threads*. Certains modules développés par des tiers, ou des librairies utilisées par ces modules, peuvent parfois ne pas être prévus pour fonctionner dans un environnement multi-thread, et dans ce cas peuvent provoquer des problèmes si on les utilise en conjonction avec le mode Worker.
- Depuis la version 2.4, le module **Event** est disponible en production. C'est un fonctionnement derivé du mode Worker à ceci près que les *threads* ne desservent pas seulement une connexion client mais peuvent réaliser plusieurs tâches indépendamment de la connexion. Plus simplement, le *thread* dessert une requête et non pas une connexion.

Une fois l'installation terminée, on peut vérifier l'état du serveur HTTP Apache :

```
# /etc/init.d/apache2 status
Apache2 is running (pid 4060).

# apachectl status

# service apache2 status
```

Configuration

La configuration d'Apache est située dans le répertoire `/etc/apache2/` :

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

Comme toujours sous Linux, les fichiers de configuration sont de simples fichiers "texte" ASCII éditables avec un éditeur de texte comme `vim`, `emacs`, `nano`, ...

La plupart de ces fichiers sont plus ou moins spécifiques à Debian/Ubuntu et permettent de séparer la configuration en plusieurs parties :

- `httpd.conf` est le fichier utilisé par Apache 1, il est conservé vide dans Apache 2 pour assurer la rétrocompatibilité. On ne l'utilisera pas ;
- `apache2.conf` est le fichier principal de configuration et il contient les directives de configuration. Il se charge aussi d'inclure les autres fichiers de configuration ;
- `ports.conf` contient la directive `Listen` qui spécifie les adresses et les ports d'écoutes ;
- `envvars` est utilisé pour définir des variables d'environnement propres à Apache, `magic` est lui utilisé pour déterminer le type de contenu d'un document en regardant les quelques premiers octets de ce contenu ;
- `conf.d` est un répertoire qui contient plusieurs fichiers qui seront analysés par apache. Par exemple, le fichier `charset` permettra de spécifier l'encodage à utiliser par défaut pour tous les fichiers ;
- `mods-available` contient la liste des modules d'apache disponibles ;
- `mods-enabled` celle des modules activés ;
- `sites-available` contient la liste des `vhosts` (*Virtual Host*) disponibles ;
- `sites-enabled` celle des `vhosts` (*Virtual Host*) activés.

L'ensemble de ces fichiers définissent une **configuration par défaut** qui rend le serveur HTTP Apache fonctionnel dès le départ.

```
// Port d'écoute du serveur ?
# cat /etc/apache2/ports.conf | grep -i "listen"
Listen 80

// Racine des documents web du serveur ?
# cat /etc/apache2/sites-enabled/000-default | grep -i "root"
DocumentRoot /var/www
```

Conclusion : par défaut, le serveur web écoute sur le **port 80** (`Listen`) et renvoie les pages web depuis le répertoire `/var/www/` (`DocumentRoot`). Pour l'instant, il est accessible à partir de `localhost` (using `127.0.1.1` for `ServerName`).

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/2.2/mod/index.html

La commande `a2enmod <module>` permet d'activer un nouveau module tandis que `a2dismod <module>` désactive un module. Ces deux programmes ne font rien d'autre que de créer ou supprimer des liens symboliques dans `/etc/apache2/mods-enabled/` pointant vers des fichiers de `/etc/apache2/mods-available/`.

Apache 2.2 intègre en standard le **module SSL** nécessaire au support du HTTP sécurisé (**HTTPS**). Il faut juste l'activer avec `a2enmod ssl` puis placer les directives de configuration nécessaires dans la configuration. Un exemple de configuration est fourni dans `/usr/share/doc/apache2.2-common/examples/httpdssl.conf.gz`.

Lire : http ://httpd.apache.org/docs/2.2/mod/mod_ssl.html

Pour Apache 1.3, le support de SSL nécessite l'installation du paquet `libapache-mod-ssl`. Les instructions dans le fichier `/usr/share/doc/libapache-mod-ssl/README.Debian` détaillent la configuration de ce module.

Test

```
// Existe-t-il une page d'accueil ?
# ll /var/www/
total 12
drwxr-xr-x 2 root root 4096 mars 6 17:35 ./
drwxr-xr-x 14 root root 4096 mars 6 17:18 ../
-rw-r--r-- 1 root root  73 mars  6 17:35 index.html
```

On modifie la page d'accueil existante du serveur web :

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

```
<html><body><h1>It works!</h1>
<p>En construction ...</p>
</body></html>
```

Il faut installer `lynx`, le navigateur en mode console :

```
# apt-get install lynx
```

Puis, on accède à la page :

```
# lynx http://127.0.0.1/
```

Question 1. Installer le serveur HTTP Apache 2, vérifier sa configuration par défaut et tester en accédant à sa page d'accueil.

Documentation en ligne : http ://httpd.apache.org/docs/2.2/fr/

Manipulations

Séquence 1 : *Virtual Hosts*

Un **hôte virtuel** est une identité (supplémentaire) assumée par le serveur web. Le Serveur HTTP Apache 2 est capable de gérer simultanément plusieurs arborescences Web grâce à la notion *Virtual Hosts* (hôtes virtuels).

Apache distingue deux types d'hôtes virtuels :
- ceux qui se basent sur l'adresse IP : cette méthode nécessite une adresse IP différente pour chaque site ;
- ceux qui reposent sur le nom DNS du serveur web : celle-ci n'emploie qu'une adresse IP et différencie les sites par le nom d'hôte communiqué par le client HTTP (ce qui ne fonctionne qu'avec la version 1.1 du protocole HTTP, employée par tous les navigateurs web actuels).

La rareté des adresses IPv4 fait en général privilégier cette deuxième méthode. Elle est cependant impossible si chacun des hôtes virtuels a besoin de HTTPS.

Apache fournit quelques commandes utilitaires :
- `a2ensite` : activer un hôte virtuel
- `a2dissite` : désactiver un hôte virtuel

La configuration par défaut d'Apache 2 a déjà activé les hôtes virtuels basés sur le nom grâce à la directive `NameVirtualHost` du fichier `/etc/apache2/sites-enabled/000-default`. Ce fichier décrit en outre un hôte virtuel par défaut qui sera employé si aucun hôte virtuel correspondant n'existe.

On commence par désactiver la configuration par défaut :

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

```
...
#NameVirtualHost *:80
...
```

```
# a2dissite default
```

Il faut renseigner le fichier `/etc/hosts` afin d'assurer la résolution de nom (Nom -> Adresse Ip) des hôtes virtuels que l'on va créer. Ceci n'est nécessaire que si votre serveur DNS (serveur bind) n'est pas installé et/ou configuré pour vos domaines ou encore, que les noms DNS attribués aux hôtes virtuels sont purement fictifs ou ne vous appartiennent pas.

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

```
# vim /etc/apache2/sites-available/www.intra.net
```

```
<VirtualHost www.intra.net:80>
# ServerName définit le nom utilisé pour le vhost. Mettez le nom de l'hôte du domaine
ServerName www.intra.net
# ServerAlias définit les autres sous domaines pour lesquels le serveur répondra
ServerAlias intra.net *.intra.net
# ServerAdmin vous permet de spécifier un email à utiliser en cas de problème, sur une
    page d'erreur 404 par exemple
# DocumentRoot définit le dossier racine dans lequel seront stockés les fichiers du site
DocumentRoot /srv/www/www.intra.net
</VirtualHost>
```

On va ensuite créer la racine web du serveur et une page d'accueil personnalisée :

```
# mkdir -p /srv/www/www.intra.net/

# cp /var/www/index.html /srv/www/www.intra.net/

# vim /srv/www/www.intra.net/index.html
```

```
<html><body><h1>Bienvenue sur www.intranet.net</h1>
<p>En construction ...</p>
</body></html>
```

Il faut maintenant activer le `vhost` :

```
# a2ensite www.intra.net
Enabling site www.intra.net.
To activate the new configuration, you need to run:
  service apache2 reload
```

La commande `a2ensite` a créé un lien symbolique du fichier de `sites-available/` vers `sites-enabled/` :

```
# ll /etc/apache2/sites-enabled/
total 8
drwxr-xr-x 2 root root 4096 mars 6 18:36 ./
drwxr-xr-x 7 root root 4096 mars 6 17:18 ../
lrwxrwxrwx 1 root root 32 mars  6 18:28 www.intra.net -> ../sites-available/www.intra.net
```

Si votre serveur DNS n'est pas fonctionnel, on assure alors la résolution de noms en local :

```
# vim /etc/hosts
```

```
...
127.0.1.1       www.intra.net
```

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

```
# service apache2 reload
```

On peut tester l'accès au `vhost` avec `lynx` :

```
# lynx http://www.intra.net/
```


Question 2. Créer un hôte virtuel supplémentaire `www.monsite.net` sur le port 8080.

Question 3. Activer à nouveau le `vhost` par défaut. Tester maintenant les trois accès possibles.

```
# lynx http://www.monsite.net:8080/
# lynx http://www.intra.net/
# lynx http://127.0.0.1/
```

Séquence 2 : Configuration 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
</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`. 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 paramétrée 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 desactivée).

Les directives `Allow from` (autoriser en provenance de) et `Deny from` (refuser en provenance de), qui s'appliquent à un répertoire et à toute l'arborescence qui en est issue, paramètrent les restrictions d'accès.

La directive `Order` contrôle simplement l'ordre dans lequel les directives `Allow` et `Deny` sont analysées.

La directive `Allow` spécifie le client pouvant accéder à un répertoire donné. Le client peut être `all`, un nom de domaine, une adresse IP, une adresse IP partielle, une paire réseau/masque réseau, etc.

La directive `Deny` fonctionne selon le même principe que `Allow`, sauf que cette fois-ci, l'accès est refusé à un client donné.

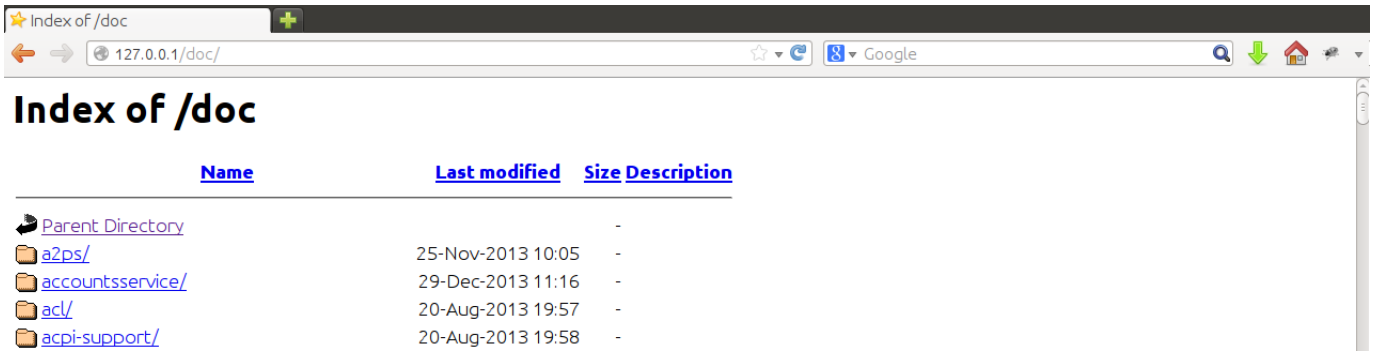
```
<Directory /var/www/lib>
    Order Deny,Allow
    Deny from All
</Directory>
```

La directive `AccessFileName` nomme le fichier que le serveur doit utiliser pour les informations de contrôle d'accès dans chaque répertoire. La valeur par défaut est `.htaccess`.

Le fichier `.htaccess` contient des directives de configuration d'Apache, prises en compte à chaque fois qu'une requête concerne un élément du répertoire où est il stocké. Sa portée embrasse également les fichiers de toute l'arborescence qui en est issue. La plupart des directives qu'on peut placer dans un bloc `Directory` peuvent également se trouver dans un fichier `.htaccess`.

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. Par défaut, un alias pour un répertoire `icons` est déjà configuré. Un répertoire `icons` est accessible par le serveur Web, mais le répertoire ne figure pas dans `DocumentRoot`.

```
Alias /doc/ "/usr/share/doc/"
<Directory "/usr/share/doc/">
   Options Indexes MultiViews FollowSymLinks
   AllowOverride None
   Order deny,allow
   Deny from all
   Allow from 127.0.0.0/255.0.0.0 ::1/128
</Directory>
```



Mais :

Question 4. Dans votre domaine `www.intra.net`, créer un dossier `public`.

Question 5. Configurer votre domaine `www.intra.net` 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. Désactiver les autres options.

Séquence 3 : Authentification

Il est parfois nécessaire de restreindre l'accès à une partie d'un site. Les utilisateurs légitimes doivent alors fournir un identifiant et un mot de passe pour accéder à son contenu (http ://httpd.apache.org/docs/2.2/fr/howto/auth.html).

Exemple : Fichier .htaccess requérant une authentification

```
Require valid-user
AuthName "Répertoire privé"
AuthType Basic
AuthUserFile /var/www/.htpasswd
```

Le fichier `.htpasswd` contiendra la liste des utilisateurs et leur mots de passe. On le manipule avec la commande `htpasswd`. Pour ajouter un utilisateur ou changer un mot de passe, on exécutera la commande suivante :

```
# htpasswd /var/www/.htpasswd <utilisateur>
New password:
Re-type new password:
Adding password for user utilisateur
```

Ce système d'authentification (`Basic`) a une sécurité très faible puisque les mots de passe circulent sans protection (ils sont uniquement codés en base64 ce qui est un simple encodage et non pas un procédé de chiffrement). Il faut noter que les documents protégés par ce mécanisme circulent également de manière non chiffrée. Si la sécurité vous importe, faites appel à **SSL** pour chiffrer toute la connexion HTTP.

Question 6. Créer un dossier `private` à la racine de votre domaine `www.intra.net`. Ajouter à la racine de ce dossier le fichier `index.html` ci-dessous.

```
<html>
<body>
   <h1>Bienvenue dans la zone privée !</h1>
</body>
</html>
```

Question 7. Ajouter à la racine de votre domaine `www.intra.net` le fichier `erreur.html` ci-dessous.

```
<html>
<body>
   <h1>Erreur authentification !</h1>
</body>
</html>
```

Question 8. Mettre en place une authentification pour l'accès à cette zone privée. Si l'utilisateur n'est pas authentifié, le serveur devra afficher la page `erreur.html`.

La directive ErrorDocument permet d'indiquer le document que le serveur renvoie au client en cas d'erreur. Liste des codes HTTP : http ://fr.wikipedia.org/wiki/Liste_des_codes_HTTP

```
ErrorDocument 404 /notfound.html
```

Séquence 4 : Journalisation

Apache permet la journalisation (*log*) des erreurs et des accès. La journalisation des erreurs se configure avec les directives `ErrorLog` et `LogLevel`. Les réglages par défaut sont fixés dans le fichier `/etc/apache2/apache2.conf` :

```
# cat /etc/apache2/apache2.conf | grep ErrorLog
// # ErrorLog: The location of the error log file.
// # If you do not specify an ErrorLog directive within a <VirtualHost>
ErrorLog ${APACHE_LOG_DIR}/error.log

# cat /etc/apache2/apache2.conf | grep LogLevel
// # LogLevel: Control the number of messages logged to the error_log.
LogLevel warn

# cat /etc/apache2/envvars | grep "APACHE_LOG_DIR"
export APACHE_LOG_DIR=/var/log/apache2$SUFFIX
```

L'ensemble des erreurs seront donc journalisées dans le fichier : `/var/log/apache2/error.log`.

Les niveaux disponibles par ordre de criticité décroissante pour la directive `LogLevel` sont : `emerg`, `alert`, `crit`, `error`, `warn`, `notice`, `info`, `debug`. Lorsqu'un niveau particulier est spécifié, les messages de tous les autres niveaux de criticité supérieure seront aussi enregistrés. Voir : http ://httpd.apache.org/docs/2.2/fr/mod/core.html#loglevel

Apache fournit plusieurs directives pour personnaliser la journalisation des accès. Trois directives sont fournies : `TransferLog` pour créer un fichier journal, `LogFormat` pour définir un format personnalisé, et `CustomLog` pour définir un fichier journal et le format en une seule étape.

```
# cat /etc/apache2/apache2.conf | grep LogFormat
LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" vhost_combined
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

# cat /etc/apache2/sites-available/default | grep CustomLog
CustomLog ${APACHE_LOG_DIR}/access.log combined
```

Ici, `vhost_combined`, `combined`, `common`, et `agent` sont les noms donnés aux formats définis avec la directive `LogFormat` et utilisables avec la directive `CustomLog`. Les formats disponibles sont : http ://httpd.apache.org/docs/2.2/fr/mod/mod_log_config.html#formats

Actuellement, le serveur Apache est configuré pour n'utiliser qu'un seul fichier de *log* pour tous les hôtes virtuels (ce qu'on pourrait changer en intégrant des directives `CustomLog` dans les définitions des hôtes virtuels).

```
# cat /etc/apache2/conf.d/other-vhosts-access-log
// # Define an access log for VirtualHosts that don't define their own logfile
CustomLog ${APACHE_LOG_DIR}/other_vhosts_access.log vhost_combined
```

Question 9. Configurer votre serveur Apache pour qu'il journalise l'ensemble des messages d'informations sauf ceux de débogage.

Question 10. Configurer vos deux hôtes virtuels pour qu'ils assurent une journalisation des accès dans des fichiers de *log* séparés : `/var/log/apache2/www-intra-net.log` et `/var/log/apache2/www-monsite-net.log`.

Séquence 5 : Moteur de réécriture d'URL

Apache fournit un moteur de réécriture à base de règles permettant de réécrire les URLs des requêtes à la volée (http ://httpd.apache.org/docs/2.2/fr/mod/mod_rewrite.html). Il accepte un nombre illimité de règles, ainsi q'un nombre illimité de conditions attachées à chaque règle, fournissant ainsi un mécanisme de manipulation d'URL vraiment souple et puissant. Les manipulations d'URL peuvent dépendre de nombreux tests, des variables du serveur, des variables d'environnement, des en-têtes HTTP ou de l'horodatage.

Il faut commencer par activer le module `mod_rewrite` et redémarrer le serveur Apache :

```
# a2enmod rewrite

# service apache2 restart
```

Vous trouverez de nombreux exemples d'utilisation courante (et moins courante) dans la documentation spécifique à la réécriture.

Exemple : supposons qu'on a récemment renommé la page `default.html` en `index.html` et que l'on désire que les accès à l'ancienne URL restent compatibles. Cependant, on veut que les utilisateurs de l'ancienne URL ne puissent pas reconnaître que les pages ont été renommées. Pour cela, on utilise les directives :

- `RewriteEngine` qui active ou désactive l'exécution du moteur de réécriture.
- `RewriteRule` qui définit les règles pour le moteur de réécriture en utilisant des expressions rationnelles compatible perl.

```
# cat /etc/apache2/sites-available/www.intra.net
```

```
<VirtualHost www.intra.net:80>
ServerName www.intra.net
ServerAlias intra.net
DocumentRoot /srv/www/www.intra.net
<Directory "/srv/www/www.intra.net">
   ...
```

```
    RewriteEngine On
    RewriteRule ^default\.html$ index.html
</Directory>
</VirtualHost>
```

Question 11. Mettre en oeuvre une réécriture d'URL qui permet d'accéder aux pages HTML du site sans avoir à préciser l'extension `.html`.

Question 12. En utilisant le moteur de réécriture d'URL, assurez-vous que toutes les requêtes sur l'hôte virtuel `intra.net` soient redirigées vers `www.intra.net`.

Question 13. Expliquer en détails la règle de réécriture ci-dessous. Donner un exemple d'utilisation de cette règle.

```
RewriteRule ^page-([0-9]+)\.html$ /page.php?id=$1 [L]
```

Séquence 6 : HTTP sécurisé

Apache 2.2 intègre en standard le **module SSL** nécessaire au support du HTTP sécurisé (**HTTPS**). Il faut juste l'activer avec `a2enmod ssl` puis placer les directives de configuration nécessaires dans la configuration. Un exemple de configuration est fourni dans `/usr/share/doc/apache2.2-common/examples/httpdssl.conf.gz`.

Lire : http ://httpd.apache.org/docs/2.2/mod/mod_ssl.html

Pour Apache 1.3, le support de SSL nécessite l'installation du paquet `libapache-mod-ssl`. Les instructions dans le fichier `/usr/share/doc/libapache-mod-ssl/README.Debian` détaillent la configuration du module.

Il faut commencer par activer le module `mod_ssl` et redémarrer le serveur Apache :

```
# a2enmod a2enmod ssl

# service apache2 restart
```

SSL est un protocole initialement proposé par la société Netscape Inc. Il est aujourd'hui adopté par l'ensemble de la communauté informatique pour l'authentification et le chiffrement des données entre clients et serveurs. Un nouveau standard basé sur SSL, TLS (*Transport Layer Security*) a vu le jour et est aujourd'hui normalisé par l'IETF (*Internet Engineering Task Force*). Initialement proposé pour sécuriser les connexions Web, SSL est utilisé aujourd'hui par bien d'autres services réseau grâce à sa simplicité de mise en oeuvre.

SSL offre des fonctions fondamentales nécessaires à la communication sécurisé sur Internet et sur tout réseau TCP/IP :

– L'authentification SSL du serveur permet de garantir son identité à chacun des clients utilisant ses services. Cet authentification s'appuye en particulier sur des techniques de chiffrement à clé publique. La confirmation de l'identité d'un serveur est très importante. Notamment, si vous devez envoyer votre numéro de carte de crédit sur le réseau pour réaliser un achat électronique, il faut que vous soyez certain de l'identité du site de commerce électronique destinataire.

- L'authentification SSL du client permet au serveur de valider l'identité du client. Cette authentification mutuelle est également très importante si le serveur Web de votre banque doit vous faire parvenir des informations confidentielles relatives à vos comptes bancaires.
- Une connexion SSL permet de chiffrer l'ensemble des données échangées entre un client et un serveur, ce qui apporte un haut niveau de confidentialité. La confidentialité est importante pour les deux parties dans la plupart des transactions privées. En complément de ce chiffrement, des mécanismes de vérification d'intégrité détectent automatiquement l'altération des données lors du transfert.

Le **certificat** est un ensemble d'informations utilisé par la couche SSL (située entre la couche Application et la couche Transport) pour réaliser l'authentification d'un service, d'une machine ou d'un utilisateur. Le certificat contient la clé publique de son détenteur et des informations sur son identité. Le certificat est signé électroniquement par une Autorité de Certification (CA) qui atteste son authenticité.

La vérification du certificat peut être effectuée par tout service qui possède la clé publique de l'autorité de certification.

OpenSSL est une implémentation libre et gratuite du protocole SSL. Elle est intégrée de manière systématique dans toutes les bonnes distributions Linux. OpenSSL se matérialise par un ensemble de bibliothèques nécessaires au développement et au fonctionnement des protocoles HTTPS (HTTP sur SSL), IMAPS (IMAP sur SSL)... Un jeu de commandes est également fourni pour générer de manière aléatoire les couples de clés privée/publique et pour manipuler les certificats utilisés dans les Infrastructures à gestion de clés (IGC).

L'installation d'Apache a créé un hôte virtuel par défaut utilisant SSL :

```
# ls -l /etc/apache2/sites-available/ | grep -i ssl
-rw-r--r-- 1 root root 7251 juil. 16 2012 default-ssl

# cat /etc/apache2/sites-available/default-ssl | grep VirtualHost
<VirtualHost _default_:443>
</VirtualHost>
```

La directive `SSLEngine` permet d'activer l'utilisation du protocole SSL/TLS :

```
# cat /etc/apache2/sites-available/default-ssl | grep SSL
// #   SSL Engine Switch:
// #   Enable/Disable SSL for this virtual host.
SSLEngine on
// #   SSLCertificateFile directive is needed.
SSLCertificateFile  /etc/ssl/certs/ssl-cert-snakeoil.pem
SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
...
```

La directive `SSLCertificateFile` indique le chemin du certificat (`/etc/ssl/certs/ssl-cert-snakeoil.pem`) et la directive `SSLCertificateKeyFile` indique le chemin de la clé privée (`/etc/ssl/private/ssl-cert-snakeoil.key`).

Lors de l'installation d'Apache, le paquet `ssl-cer` a été auto-installé et un certificat a été créé (`/etc/ssl/certs/ssl-cert-snakeoil.pem`).

Ce certificat est **auto-signé** : le nom de l'AC signataire du certificat (voir le champ `Issuer`) est le même que le nom du titulaire du certificat (voir le champ `Subject`) :

```
# openssl x509 -in /etc/ssl/certs/ssl-cert-snakeoil.pem -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 16262385324728032614 (0xe1af994a80f09d66)
```

```
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: CN=server-tv
        Validity
            Not Before: Mar 6 16:18:29 2014 GMT
            Not After : Mar 3 16:18:29 2024 GMT
        Subject: CN=server-tv
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    ...
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
    Signature Algorithm: sha1WithRSAEncryption
         ...
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
```

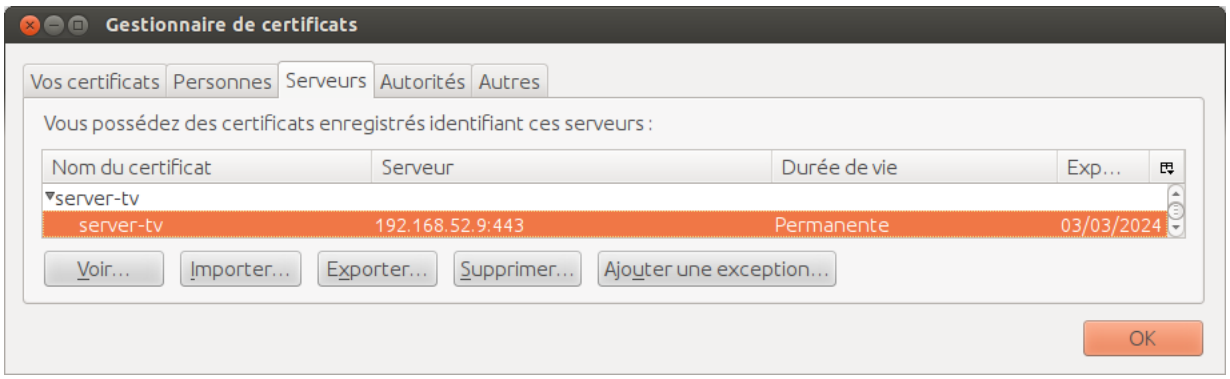
Il suffit maintenant d'activer l'hôte virtuel utilisant SSL, de recharger la configuration d'Apache et de tester l'accès HTTPS :

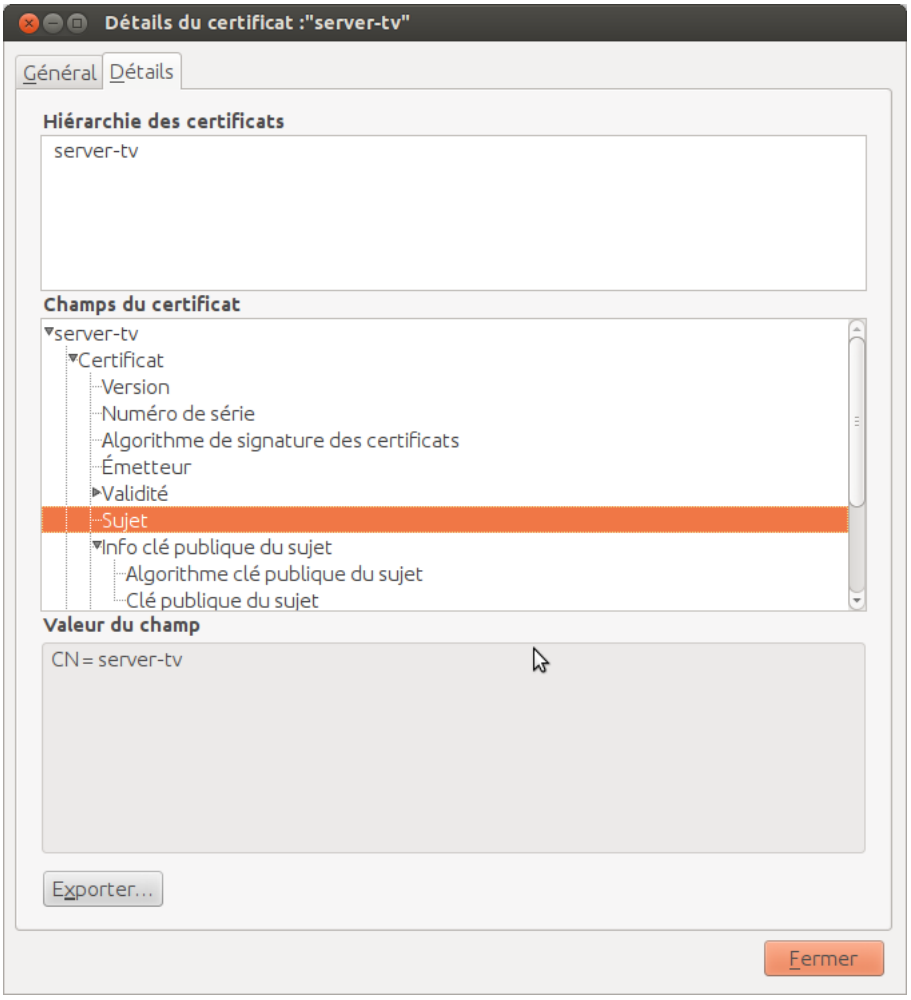
```
# a2ensite default-ssl

# service apache2 reload

# lynx https://127.0.0.1/
```

À partir du navigateur `firefox`, il est possible d'afficher les certificats enregistrés :





Il est aussi possible de créer son certificat SSL auto signé :

```
// Création du certificat SSL auto signé :
# openssl req -x509 -nodes -days 365 -newkey rsa:1024 -out /etc/apache2/server.crt -keyout /etc/
    apache2/server.key
Generating a 1024 bit RSA private key
......++++++
.........++++++
writing new private key to '/etc/apache2/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) [AU]:FR
State or Province Name (full name) [Some-State]:FRANCE
Locality Name (eg, city) []:SARRIANS
Organization Name (eg, company) [Internet Widgits Pty Ltd]:mon-serveur
Organizational Unit Name (eg, section) []:mon-serveur
Common Name (e.g. server FQDN or YOUR name) []:intra.net
Email Address []:admin@intra.net

// On modifie les droits sur le fichier de clé privée :
# chmod o-rw /etc/apache2/server.key

// Vérification :
# openssl x509 -in /etc/apache2/server.crt -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 11374047905899830327 (0x9dd8bc4ee16f5c37)
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=FR, ST=FRANCE, L=SARRIANS, O=mon-serveur, OU=mon-serveur, CN=intra.net/
            emailAddress=admin@intra.net
        Validity
            Not Before: Mar 18 17:30:48 2014 GMT
            Not After : Mar 18 17:30:48 2015 GMT
        Subject: C=FR, ST=FRANCE, L=SARRIANS, O=mon-serveur, OU=mon-serveur, CN=intra.net/
            emailAddress=admin@intra.net
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
                Modulus:
                    ...
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                39:EC:7C:95:52:DD:23:C0:4F:CB:2D:61:DE:65:67:64:2E:45:A4:A9
            X509v3 Authority Key Identifier:
                keyid:39:EC:7C:95:52:DD:23:C0:4F:CB:2D:61:DE:65:67:64:2E:45:A4:A9

            X509v3 Basic Constraints:
                CA:TRUE
    Signature Algorithm: sha1WithRSAEncryption
        ...
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
```

Question 14. Créer votre certificat auto-signé. Puis à partir de l'hôte virtuel SSL fourni par Apache (`/etc/apache2/sites-available/default-ssl`), créer votre hôte virtuel SSL pour le site `https://www.intra.net`.

Question 15. En utilisant la directive `Redirect` dans la configuration de l'hôte virtuel accessible sur le port 80, assurez-vous que les accès vers le site `http://www.intra.net/` soit redirigés vers `https://www.intra.net/`.

Séquence 7 : Module PHP

Il faut installer le paquetage `libapache2-mod-php5` pour activer le support PHP dans Apache.

```
# apt-get install libapache2-mod-php5
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
  apache2-mpm-prefork libxml2 php5-cli php5-common sgml-base xml-core
Paquets suggérés :
  php-pear sgml-base-doc debhelper
Les paquets suivants seront ENLEVÉS :
  apache2-mpm-worker
Les NOUVEAUX paquets suivants seront installés :
  apache2-mpm-prefork libapache2-mod-php5 libxml2 php5-cli php5-common sgml-base xml-core
...
```

Cela entraîne la suppression de `apache2-mpm-worker` et l'installation de `apache2-mpm-prefork`, tout en préservant la configuration du serveur précdemment installé. En effet, PHP ne fonctionne qu'avec cette version du serveur web.

On peut constater que le module PHP5 a été installé et activé :

```
# ll /etc/apache2/mods-enabled/php*
lrwxrwxrwx 1 root root 27 mars 19 08:55 /etc/apache2/mods-enabled/php5.conf -> ../mods-available/
    php5.conf
lrwxrwxrwx 1 root root 27 mars 19 08:55 /etc/apache2/mods-enabled/php5.load -> ../mods-available/
    php5.load

# cat /etc/apache2/mods-enabled/php5.load
LoadModule php5_module /usr/lib/apache2/modules/libphp5.so

# ll /usr/lib/apache2/modules/libphp5.so
-rw-r--r-- 1 root root 9040032 mars 1 00:39 /usr/lib/apache2/modules/libphp5.so
```

Pour vérifier que le moteur PHP est fonctionnel, on va créer un script `info.php` à la racine du site `www.monsite.php` :

```
# vim /srv/www/www.monsite.net/info.php
```

```
<?php
phpinfo();
?>
```

Et on teste :

```
# lynx http://www.monsite.net:8080/info.php
```

Question 16. Installer le module PHP5. Puis, identifier la version de PHP actuellement installée à partir de l'exécution de la fonction `phpinfo()`.

Question 17. Faire fonctionner le script `page.php` fourni ci-dessous avec la configuration de réécriture d'URL réalisée à la séquence 5.

```php
<?php
// Script : page.php

// Récupère le paramètre id passé dans l'URL
echo "<h1>Bienvenue sur la page n°".$_REQUEST["id"]."<br />";
echo "<p>En construction ...</p>";

// Affiche la version de PHP : la constante PHP_VERSION contient la version courante de PHP
echo "<p>Version de PHP : ".PHP_VERSION."</p>";
?>
```