Apache

Plan

Présentation

Protocole HTTP

Langage HTML

Architecture du logiciel et Modules

Installation

Configuration

Sécurisation des répertoires

Authentification et Contrôle d'accès.

Hôtes Virtuels

Administration du serveur

- logs, sauvegarde, monitoring de service

Performances

Sécurisation des échanges avec SSL

Présentation

Le serveur Apache est un des nombreux projets gérés par la *Fondation Apache*.

Utilisé par près de 50% des serveurs web dans le monde, c'est le logiciel OpenSource par excellence. Il faut quand même noté que sa popularité est sur le déclin et qu'il souffre de la concurrence de nginx notamment.

Son architecture modulaire, permet de lui ajouter facilement des fonctionnalités.

La version stable actuelle est la 2.4

Votre nouveau livre de chevet : https://httpd.apache.org/docs/2.4/

Protocole HTTP

Le protocole **H**yper**T**ext **T**ransfer **P**rotocol est un protocole de niveau applicatif, s'appuyant sur TCP/IP.

A l'origine conçu pour le transfert de texte, il est, dans ces versions actuelles capables de supporter différents types de données.

C'est au navigateur client de déterminer au moyen du « MIME » Type des données reçues par le serveur, la façon de les interpréter et de les afficher.

Le protocole HTTP, dont la version actuelle est la 1.1, propose un ensemble de primitives (GET, POST, HEAD...) et de codes de retour (le fameux 404).

Architecture du logiciel et Modules

Un serveur apache "de base" propose la fourniture de pages web classiques.

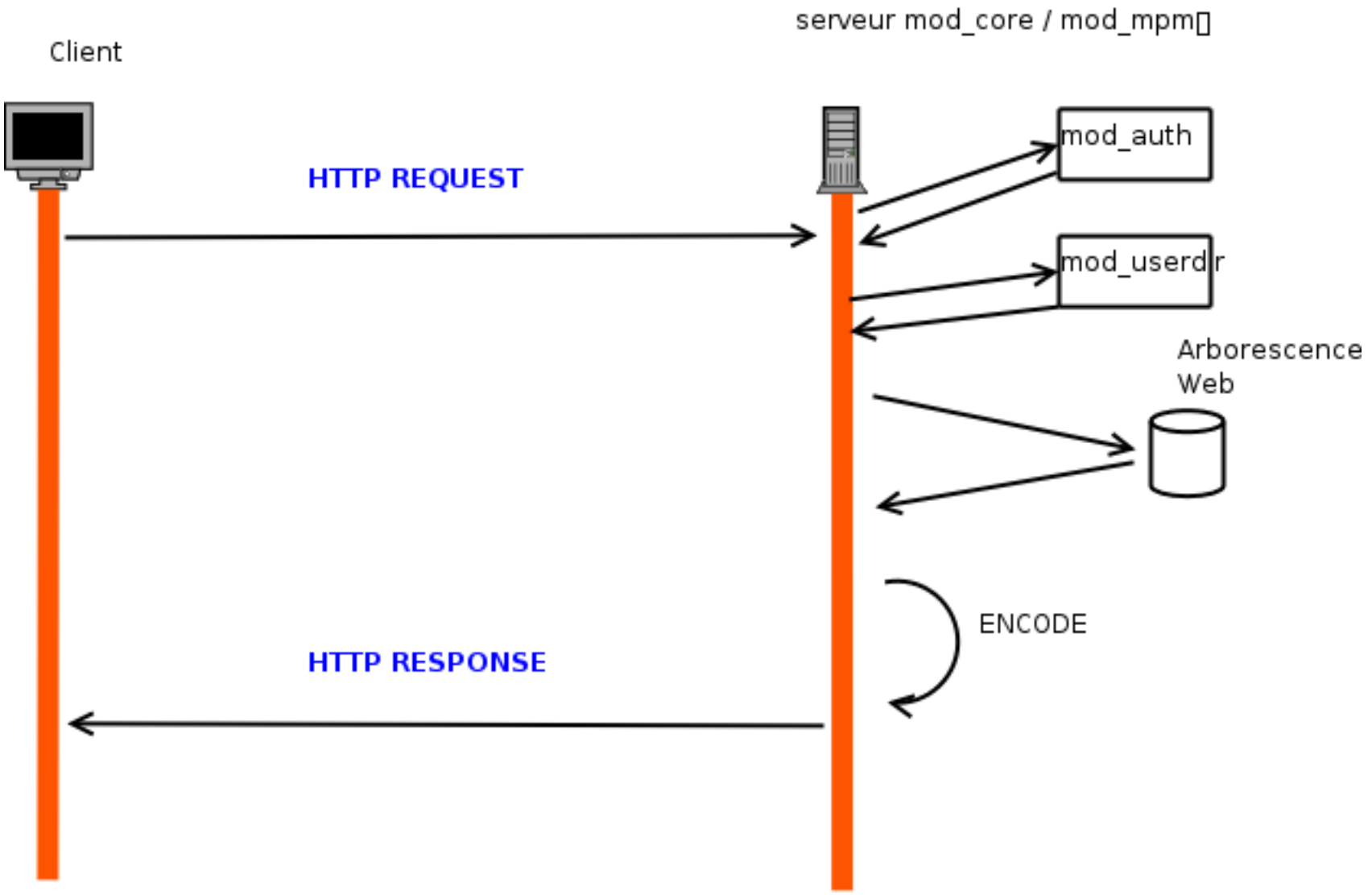
Les fonctions additionnelles sont apportées par des modules complémentaires.

On distingue les modules statiques, activés lors de la compilation, des modules dynamiques, chargés lors du démarrage du serveur (Dynamic Shared Object) par l'intermédiaire d'une directive de configuration *LoadModule*.

On distingue plusieurs types de modules:

- Multi Processing Module
- Modules de base
- Extensions
- Experimental, externe.

Architecture fonctionnelle



Les principaux modules

mpm-prefork gestion des requêtes par processus « lourds »

mpm-worker gestion des requêtes par threads.

mod_access propose des mécanismes de contrôle d'accès, basé sur les caractéristiques réseaux des clients.

mod_cgi permet le support des scripts CGI.

mod_userdir permet de mettre en place des dossiers web personnels.

mod_rewrite propose des fonctions de réécritures d'URL. Ce module est notamment utilisé pour rediriger des requètes.

mod_auth_ldap est un module expérimental permettant d'authentifier les utilisateurs dont les comptes sont stockés dans une base LDAP.

Installation

Sur CentOS / RedHat, l’installation se fait simplement :

```
# yum install httpd
# systemctl enable httpd
# systemctl start httpd
# firewall-cmd --add-service http --permanent
# firewall-cmd --add-service https --permanent
# firewall-cmd --add-service http
# firewall-cmd --add-service https
```

Installation

L'installation du paquet httpd apporte les changements suivants:

- création d'un utilisateur et d'un groupe dédié au service (apache)
- création et activation d’un "unit file" systemd responsable de la gestion du service
- création d'une configuration minimale dans /etc/httpd
- création du répertoire de log /var/log/httpd
- création du répertoire de stockage des pages web /var/www/html
- copie de la documentation dans /usr/share/doc/httpd
- copie des fichiers de bibliothèques dans /usr/lib
- copie des exécutables dans /usr/bin et /usr/sbin

Premier pas

Une fois l'installation achevée, un site web minimal est installé dans */var/www/html*, et accessible par l'intermédiaire de l'url http://localhost

Premier pas

La commande */usr/sbin/apachectl* permet de contrôler l'état du service apache.

Ses paramètres sont les suivants:

start	démarrage du service
stop	arrêt du service
restart	arrêt et démarrage du service
configtest	vérification de la configuration (syntaxe)
graceful	relance du service (relecture de la configuration) sans couper les connexions en cours.

Gestion du service

- On va garder la commande apachectl pour :
  - Faire un redémarrage "graceful" du serveur
  - Vérifier la validité de nos modifications de configuration :

```
apachectl configtest && apachectl graceful
```

- Pour arrêter / démarrer complètement le service, on passera par systemd :

```
systemctl  start | stop | restart httpd
```

Premier pas (un peu plus loin)

La commande */usr/sbin/httpd* constitue l'exécutable serveur.

Il ne doit pas être appelé directement pour lancer le serveur.

Il accepte les options suivantes:

-V option de compilation

-l modules compilés en statique

-t vérification de la configuration

-M liste des modules statiques et dynamiques chargés.

-k start | stop | restart | graceful

...

Vérification du service

- Il suffit
  - De s'assurer de la présence de processus httpd en cours de fonctionnement

```
# ps -ef|grep http
root      1506     1  0 16:40 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache    1507  1506  0 16:40 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache    1508  1506  0 16:40 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache    1509  1506  0 16:40 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache    1510  1506  0 16:40 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache    1511  1506  0 16:40 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
```

  - D'avoir des processus à l'écoute sur les ports 80 et/ou 443

```
 ss -taupen|grep http
tcp    LISTEN     0      128      :::80                   :::*
users:(("httpd",pid=1511,fd=4),("httpd",pid=1510,fd=4),("httpd",pid=1509,fd=4),
("httpd",pid=1508,fd=4),("httpd",pid=1507,fd=4),("httpd",pid=1506,fd=4)) ino:19863
sk:ffff88003b4c1980 v6only:0 <->
```

Vérification du service

- Il suffit
  - d’interroger le système :

```
# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   Active: active (running) since ven. 2018-11-23 10:08:31 CET; 3h 0min ago
     Docs: man:httpd(8)
           man:apachectl(8)
  Process: 11343 ExecReload=/usr/sbin/httpd $OPTIONS -k graceful (code=exited, status=0/SUCCESS)
 Main PID: 9879 (httpd)
   Status: "Total requests: 0; Current requests/sec: 0; Current traffic:   0 B/sec"
   CGroup: /system.slice/httpd.service
           ├─ 9879 /usr/sbin/httpd -DFOREGROUND
           ├─11344 /usr/sbin/httpd -DFOREGROUND
           ├─11345 /usr/sbin/httpd -DFOREGROUND
           ├─11346 /usr/sbin/httpd -DFOREGROUND
           ├─11347 /usr/sbin/httpd -DFOREGROUND
           ├─11348 /usr/sbin/httpd -DFOREGROUND
           └─11349 /usr/sbin/httpd -DFOREGROUND

nov. 23 12:07:39 centos httpd[10861]: AH00558: httpd: Could not reliably determine the server's fully
qualified domain name,...message
nov. 23 12:07:39 centos systemd[1]: Reloaded The Apache HTTP Server.
nov. 23 12:20:04 centos httpd[11015]: AH00558: http
```

Généralité sur la configuration

Syntaxe:

Vi(m) reconnaît la syntaxe des fichiers de configuration et propose une *colorisation* et des fonctionnalités de mise en page adéquate.

Une phrase commençant par un # n'est pas prise en compte.

La configuration est regroupée en *directive*, associée à un *module.*

Ces directives ne sont pas sensibles à la casse.

Une directive et sa valeur sera héritée par les contextes de niveau inférieur, sauf en cas de redéfinition dans le sous-contexte.

Elles sont valables dans certains *contextes*:

ServerConfig

VirtualHost

Directory

Location

File

...

Fichiers de configuration

Le principal fichier de configuration est */etc/httpd/conf/httpd.conf*

Il fait appel à tous les fichiers *.conf du sous-répertoire /etc/httpd/conf.d via une directive « *Include* »

C'est dans le répertoire /etc/httpd/conf.d/ que nous allons concentrer nos efforts de configuration, afin de ne pas toucher aux fichiers d'origine, susceptibles d'être modifiés / remplacés lors d'un yum update

Toute modification de la conf doit être validée, et le serveur doit être relancé.

```
apachectl configtest && apachectl graceful # dans le cas d'une modification de configuration triviale

apachectl configtest && systemctl restart httpd # dans le cas d'une modification des paramètres réseau, de mpm, nécessitant un redémarrage complet
```

httpd.conf

Il centralise la configuration du serveur proprement dit.

Il définit:

- les répertoires dédiés au service.
- les paramètres réseau
- les limites de ressources utilisables
- le compte sous l'identité duquel le service fonctionne
- les fichiers de logs
- des réglages par défaut (qui pourront éventuellement être modifiés au cas par cas)

/etc/httpd/conf/httpd.conf

Les contextes, autre que global, sont définis par des *balises*

Les principaux contextes sont:

Directory(Match) *nom ou expression régulière*

Files(Match) *nom* ou *expression régulière.*

Location(Match) *nom* ou *expression régulière.*

Voir la ligne 171 du fichier, qui permet d'appliquer une directive de contrôle d'accès sur les tous les fichiers .ht* de l'arborescence web.

Paramètres réseau

Il définit simplement l'adresse et le numéro de port sur le(s)quel(s) le service sera à l'écoute.

```
Listen 127.0.0.1:80
Listen 192.168.10.2:8080
Listen :443
```

Les modules activés par défaut

Les modules sont chargés depuis les fichiers /etc/httpd/conf.modules.d/*.conf via la directive *LoadModule*

Il est conseillé de désactiver les modules non utilisés, pour des raisons de performance et de sécurité.

Par exemple, si aucun mécanisme d'authentification http n'est prévu, il est souhaitable de désactiver les modules mod_auth*

De la même manière, les autoindex, mime, negotiation et status ne sont pas indispensables.

Exemple: choix du module mpm

```
# cat /etc/httpd/conf.modules.d/00-mpm.conf
# Select the MPM module which should be used by uncommenting exactly
# one of the following LoadModule lines:

# prefork MPM: Implements a non-threaded, pre-forking web server
# See: http://httpd.apache.org/docs/2.4/mod/prefork.html
LoadModule mpm_prefork_module modules/mod_mpm_prefork.so

# worker MPM: Multi-Processing Module implementing a hybrid
# multi-threaded multi-process web server
# See: http://httpd.apache.org/docs/2.4/mod/worker.html
#
#LoadModule mpm_worker_module modules/mod_mpm_worker.so
```

Améliorations possibles

Positionner les options à *None*. Désactiver au moins *multiviews* et *FollowSymlink.*

Désactiver le support des cgi sauf si c'est nécessaire.

Rajouter des directives plus permissives aux différentes branches de l'arborescence, selon le principe « du plus restrictif au plus permissif ».

Changer le nom des fichiers de logs, de manière à avoir des logs par hôte virtuel.

Positionner la directive *ServerSignature* à *Off*.

Ajouter la directive *ServerToken*, avec la valeur *Prod*

Performances

Choix d'un MPM:

- Worker: plusieurs processus fils avec chacun plusieurs threads. Le plus performant
- Prefork: plusieurs processus fils avec chacun 1 thread. Équivalent au précédent en terme de vitesse, mais plus gourmand en mémoire. Compatible avec le module php.

Désactivez les requêtes DNS (HostnameLookups Off)

Évitez les contrôles d'accès basés sur des noms de domaines ou de machines.

N'utilisez pas de fichier htaccess (AllowOverride None)

Directives

- *timeout* (délai max entre 2 opérations)
- *KeepAliveTimeOut (*délais après lequel une connexion inactive est terminée)
- *MaxRequestsPerChild (limite de la durée de vie d'un processus)*

Définition des contextes

```
<Directory />
    Option ...
    Order allow,deny
    ...
</Directory>
```

Les contextes sont définis par des *balises*, comme dans l'exemple ci-dessus

Les différents contextes sont:

- Contexte global (sans balise)
- Directory(Match) *nom ou expression régulière*
- Files(Match) *nom* ou *expression régulière.*
- Location(Match) *nom* ou *expression régulière.*

A ces contextes sont appliqués les directives de configuration

Directives importantes

DocumentRoot

ServerName / ServerAlias

ServerAdmin

ErrorLog

CustomLog

ServerSignature

ServerToken

Listen

Alias

Options

Allow

Options

La directive *Options* modifie le traitement d'un contexte donné par le serveur.

Elle s'applique aux contextes S*erverConfig, VirtualHost, Directory* et aux fichiers *htaccess.*

Elle peut prendre les arguments suivants:

ALL, None

ExecCGI

FollowSymlinks

Includes(NOEXEC)

Indexes

Multiviews

SymLinksIfOwnerMatch

Si les arguments sont précédés du signe + ou -, ils sont ajouter ou retrancher aux options déjà existantes. Sinon ils les remplacent.

Hôtes Virtuels

Le serveur Apache est capable de gérer plusieurs arborescence web, en fonction de différents critères:

- Hôtes Virtuels basés sur l'adresse ip ou le numéro de port.
- Hôtes Virtuels basés sur le nom.

Cette dernière méthode est généralement utilisée. Elle ne nécessite qu'une configuration DNS adéquate.

La mise en place d'une telle architecture se fait de la manière suivante:

```
<VirtualHost *:80>
    ServerName   www.ex1.org
    ServerAlias  ex1
    DocumentRoot /var/www/ex1.org/
</VirtualHost>
```

```
<VirtualHost *:80>
    ServerName   www.ex2.org
    ServerAlias  ex2
    DocumentRoot /var/www/ex2.org/
</VirtualHost>
```

Organisation des hôtes virtuels

Chaque hôte virtuel et site dispose de son propre fichier de configuration dans /etc/httpd/conf.d

Les directives minimales à inclure dans une configuration d'hôte virtuel sont:

- ServerName
- ServerAdmin
- DocumentRoot
- CustomLog
- ErrorLog

Contrôle d'accès - ancienne version.

```
<Directory />
    Order Allow,Deny
    Allow from localhost
    Allow from 192.168.10.0/24
    Allow from example.org
    Deny from all
</Directory>
```

Les directives de contrôle d'accès sont proposées par le module *mod_access*.

Leur contexte d'application est le *répertoire*, ou un fichier .htaccess.

Order Allow,Deny:

Seuls les hôtes ou les domaines explicitement spécifiées par une directive «allow from» sont autorisés.

Order Deny,Allow:

Tous les hôtes ou les domaines ne correspondant pas à la directive «Deny from» sont autorisés.

Contrôle d’accès, nouvelle version

```
<RequireAll>
    Require all granted
    Require not ip 10.252.46.165
</RequireAll>
```

Fichier .htaccess

Les fichiers .htacess permettent de décentraliser la configuration liée à un répertoire dans un fichier.

Ce fichier étant lu à chaque requête, il n'est pas nécessaire de redémarrer le serveur après modification.

En revanche, il est pénalisant en terme de performance.

Il est donc préférable de limiter son utilisation à la délégation d'administration. (par exemple, site personnel).

La directive *AllowOverride* permet d'autoriser ou non l'utilisation de fichier htaccess, et de spécifier quelle catégorie de directives il est possible d'y inclure. Ses argument sont:

- All, None
- AuthConfig
- FileInfo
- Indexes
- Limit, LimitExcept
- Options

Affichage du contenu d'un répertoire

Lorsqu'une requête pointe sur un répertoire, le serveur recherche dans ce répertoire un fichier correspondant à la directive *DirectoryIndex*.

Par défaut, *DirectoryIndex* est égal à *index.html*.

S'il ce fichier n'est pas présent:

- Si l'option *Indexes* est activée, le serveur affiche le contenu du répertoire.
- Si l'option *MultiViews* est activée, il affiche le fichier «le plus ressemblant» à la requète.
- Si cette option n'est pas activée, un message d'erreur est affichée.

Il est déconseillé d'activer l'option ci-dessus, sauf dans de rares cas.

Pages Personnelles

Les pages personnelles mettent à la disposition des utilisateurs du système une arborescence web.

La directive *UserDir* spécifie quel répertoire du dossier personnel de l'utilisateur sera le conteneur des pages web.

Cette directive est fournie par le module *mod_userdir*.

Ex, si UserDir = public_html, alors l'utilisateur *tom* devra stocker ses pages web dans le dossier /home/tom/public_html.

Bien évidemment, l'utilisateur sous lequel tourne le serveur apache doit avoir les droits en lecture sur ce dossier.

Les pages seront accessibles via l'adresse http://serveur/~user

Authentification

Le module *mod_auth* fournit des directives permettant d'authentifier les utilisateurs.

Les comptes sont stockés dans des fichiers textes et sont gérés par l'intermédiaire de la commande *htpasswd*.

Il est fortement conseillé de stocker les fichiers de comptes en dehors de l'arborescence accessible au serveur web, ou de les protéger à l'aide de la directive suivantes:

```
<FileMatch "^\.ht">
    Order Allow,Deny
</FileMatch>
```

Création du fichier de mot de passe:

```
htpasswd -c /etc/httpd/passwd foo
```

Modification d'un compte existant ou création d'un nouveau compte:

```
htpasswd /etc/httpd/passwd bar
```

Directive d'authentification

Ces directives s'appliquent dans n'importe quel contexte.

Pour fonctionner par l'intermédiaire d'un fichier htaccess, la directive *AllowOverride* doit avoir pour valeur au moins « *AuthConfig* » doit être activée.

AuthName correspond au message qui sera affiché dans la boîte de dialogue de connexion

AuthType Basic spécifie une authentification sur les fichiers htpasswds

Require permet de définir les critères d'autorisation

Satisfy any ou *all* permet de coupler l'authentification avec d'autre directives de contrôles d'accès.

AuthUserFile définit le fichier de comptes.

AuthGroupFile définit le fichier de groupe

```
AuthType Basic
AuthName "Password Required"
AuthUserFile /etc/httpd/passwd
Require valid-user
```

Fichiers de logs

Les accès et les erreurs sont enregistrés dans des fichiers journaux.

Le nom, et le format de ces logs sont contrôlés par les directives suivantes:

LogFormat *format nom*

%l %u %t %h %r %s %b .....

CustomLog *fichier nomFormat*

ErrorLog *fichier nomFormat | syslog:facility*

LogLevel *emerg alert crit error warn notice info debug*

Génération de statistiques d'accès

Le logiciel webalizer permet de générer des statistiques à partir des logs d'Apache.

Il permet de rassembler les informations suivantes:

- nb de hits, de visites
- trafic en ko
- sites et requêtes référents
- ...

Il est fourni par la paquet du même nom.

Il installe une tâche cron qui va générer les statistiques quotidiennement, en fonction de la configuration du fichier /etc/webalizer/webalizer.conf

Administration du serveur

Gestion du serveur

- systemctl start | stop | restart httpd
- apachectl graceful | configtest

Suivi du serveur

- ps -ef | grep httpd
- ss -tapen | grep httpd
- tail /var/log/httpd/*
- monit

Sécurisation des échanges avec SSL

Le module *mod_ssl*, s'appuyant sur la librairie de cryptographie *openSSL* propose différents outils afin de:

- authentifier les parties communicantes (dans notre cas, en général seul le serveur est authentifié).
- garantir la confidentialité des échanges, en chiffrant les communications.
- garantir leur intégrité et l'identité des parties communiquantes en signant numériquement les échanges.

Sa mise en œuvre est désormais obligatoire *de fait*, les moteurs de recherche commençant à pénaliser les sites n’offrant qu’un accès "en clair".

Intégration avec le serveur Apache

Activation du module

Acquisition d'un certificat auprès d'une autorité commerciale reconnue

Ou création d'un certificat privé, dans le cas d'un intranet.

Dans ce dernier cas, on pourra utiliser des utilitaires de gestion de certificats pour faciliter la tâche.

Le but est d'obtenir au minimum le certificat public de l'autorité de certification, ainsi qu'un certificat pour le serveur Apache.

Ce dernier ne devra pas être chiffré, afin de ne pas avoir à rentrer de "passphrase" lors du démarrage du serveur.

À noter le projet https://letsencrypt.org/ qui propose gratuitement des certificats ssl gratuits reconnus par la majorité des navigateurs modernes.

Rappel, Création d'une CA et d'un certificat serveur

Création de l'Autorité de Certification:

```
/usr/lib/openssl/misc/CA.pl -newca
```

Création d'une demande de certificat:

```
/usr/lib/openssl/misc/CA.pl -newreq
```

- Attention, le *CommonName* demandé doit correspondre au *nom DNS* du serveur.

Signature de la demande et génération du certificat:

```
/usr/lib/openssl/misc/CA.pl -sign
```

- On obtient les fichiers suivants:
  - newreq.pem – requète de certificat
  - newkey.pem – clé privée associée au certificat.
  - newcert.pem – certificat serveur

Installation du certificat

Déchiffrer la clé associée au certificat, afin qu'aucune "passphrase" ne soit demandé au démarrage du serveur

```
openssl rsa -in newkey.pem -out key.pem
```

Copier la clé, le certificat serveur et le certificat de l'Autorité dans */etc/apache2/ssl*

Configuration du serveur

Un serveur Apache écoute en mode sécurisé sur le port 443 par défaut.

Il faut avant tout activer le module mod_ssl

Configurer les directives ssl

Configurer les hôtes virtuels pour qu'ils écoutent soit sur le port 80, soit sur le port 443

Les directives ssl:

- SSLEngine On | Off
- SSLCertificateFile
- SSLCertificateKeyFile
- SSLCACertificateFile
- SSLVerifyClient none, optional, require

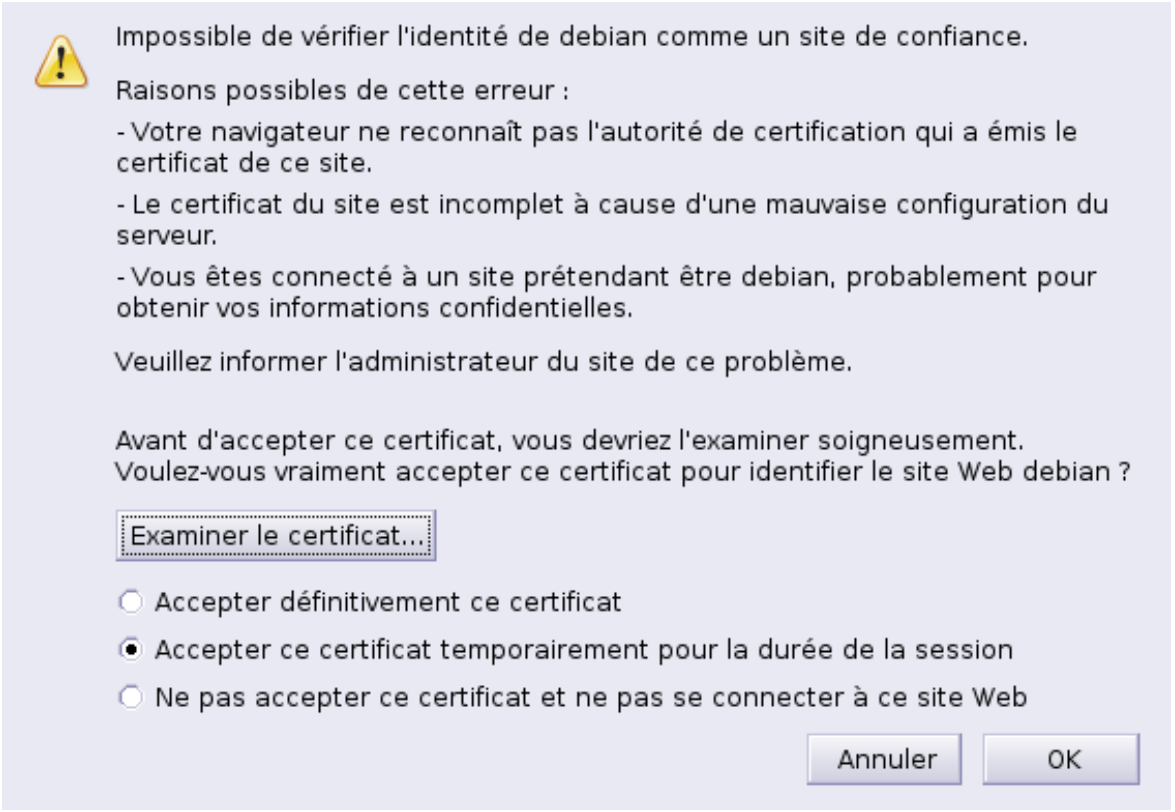
Remarque, firefox n'accepte que les certificat client au format pkc12:

```
sudo openssl pkcs12 -export -in newcert.pem -inkey newkey.pem -out cred.p12
```

openSSL

Remarques:

Un certificat serveur dont l'Autorité de Certification n'est pas reconnu par le navigateur donne le message suivant:



openSSL

Remarques:

Un certificat dont le *CommonName* ne correspond pas au nom du serveur appelé donne le message suivant: