Apache

Service réseau

Table des matières

[Introduction 2](#_Toc59450943)

[Le modèle client-serveur 2](#_Toc59450944)

[Demande de page en html statique 2](#_Toc59450945)

[Demande de page en html dynamique (côté client) 2](#_Toc59450946)

[Demande de page en html dynamique (côté serveur) 3](#_Toc59450947)

[Demande de page : modèle complet 3](#_Toc59450948)

[Fonctionnalités 4](#_Toc59450949)

[Architecture 5](#_Toc59450950)

[Vue d'ensemble 5](#_Toc59450951)

[Le noyau 5](#_Toc59450952)

[Les modules 6](#_Toc59450953)

[Les filtres 6](#_Toc59450954)

[Installation 7](#_Toc59450955)

[Par les binaires 7](#_Toc59450956)

[Architecture sous CentOS 8 7](#_Toc59450957)

[Configuration 8](#_Toc59450958)

[Structure de l'httpd.conf 8](#_Toc59450959)

[Contexte des directives 8](#_Toc59450960)

[Environnement principal 9](#_Toc59450961)

[La directive ServerName 9](#_Toc59450962)

[La directive ServerRoot 9](#_Toc59450963)

[La directive DocumentRoot 9](#_Toc59450964)

[La directive ServerAdmin 9](#_Toc59450965)

[La directive Listen 9](#_Toc59450966)

[La directive ErrorDocument 9](#_Toc59450967)

[La directive ServerTokens 10](#_Toc59450968)

[Contrôler Apache 10](#_Toc59450969)

[L'arbre des processus 10](#_Toc59450970)

[Les directives User et Group 11](#_Toc59450971)

[Les sites persos 11](#_Toc59450972)

[Les redirections simples 12](#_Toc59450973)

[Les index de répertoires 13](#_Toc59450974)

[Recherche d'une page d'accueil 13](#_Toc59450975)

[Hébergement virtuel 14](#_Toc59450976)

[Introduction 14](#_Toc59450977)

[Principe 14](#_Toc59450978)

[Explication par un exemple 14](#_Toc59450979)

[Création d'alias d'interface 16](#_Toc59450980)

[Les fichiers journaux 16](#_Toc59450981)

# Introduction

Le serveur Apache tient son nom d'une des plus fières tribus indiennes dont la vigueur et la faculté d'adaptation n'était plus à prouver

Apache est :

* Open Source et gratuité.
* Multi-plateformes (Unix- Unix Like - Linux - Windows – MacOs).
* C'est le serveur web 'open source' le plus répandu.
* Modulable, fiable, performant, sécurisé et extensible.
* Supporte les protocoles : HTTP / HTTPS (le plus souvent), POP3, FTP ...

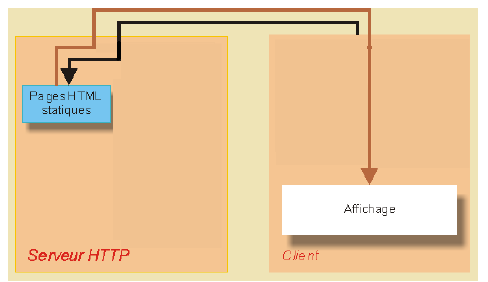
# Le modèle client-serveur

Il existe plusieurs modes d'échange entre le serveur et le client.

1. Demande d'une page en html 'statique'
2. Demande d'une page en html 'dynamique' (côté serveur)
3. Demande d'une page en html 'dynamique' (côté client)

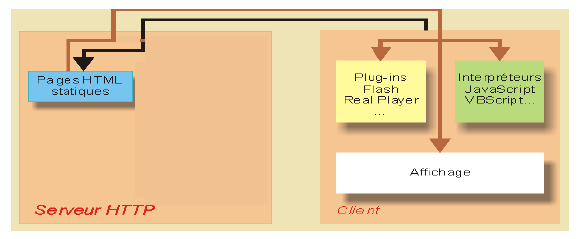
## Demande de page en html statique

Ici, tout le contenu est défini dans la page HTML demandée. Le serveur l'envoie tel quel au client qui n'a plus qu'à l'interpréter et l'afficher.



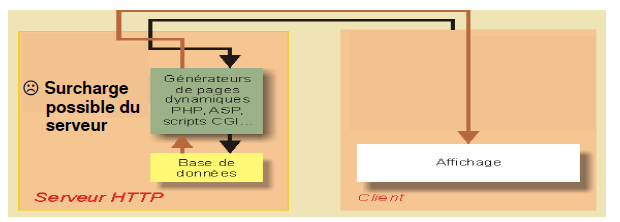
## Demande de page en html dynamique (côté client)

|  |  |
| --- | --- |
| Avantages :   * Pas de surcharge sur le serveur. * Pages animées.   Inconvénients :   * Incompatibilité des navigateurs. * Failles de sécurité possibles côté client | Exemples :   * Contrôler la validité de données avant de les envoyer au serveur. * Effectuer un traitement local pour affiche un résultat. * Animations quelconques. |

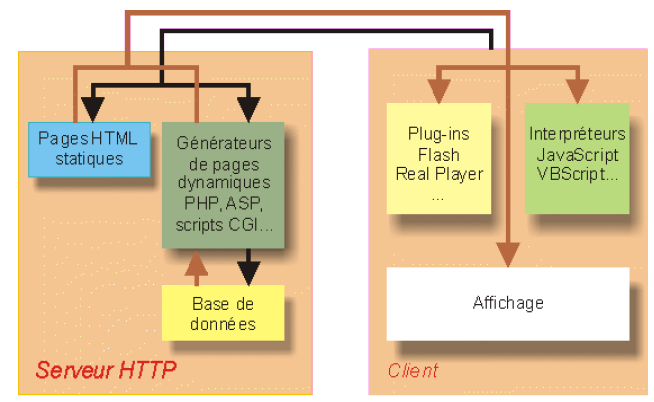


## Demande de page en html dynamique (côté serveur)

|  |  |
| --- | --- |
| * Les scripts CGI[[1]](#footnote-2) construisent totalement un flux HTML au moment de leur appel. On utilise de plus en plus souvent le C (compilé) ou le PERL (interprété).   OU   * PHP (libre) ou ASP (Microsoft) qui sont des langages spécialisés et puissants. | Exemples :   * Demande d'un résultat calculé par le serveur et dont les données sont fournies par le client. * Demande de données qui se trouvent dans une base de données hébergée côté serveur. |



## Demande de page : modèle complet



# Fonctionnalités

Conformité aux standards :

* Entièrement conforme aux standards HTTP/1.1 – RFC 2616).

Scalabilité :

* Permet d'héberger un grand nombre de sites web sur une même machine sans voir les performances diminuer de manière cruciale.
* Il sera aussi performant à 1000 sites qu'à 10.

Objets dynamiques partagés (shared objects) :

* Modules pouvant être compilés séparément du noyau et activés au démarrage d'Apache.

Personnalisation :

* Programmation (en C ou en Perl) de modules personnels via l'API[[2]](#footnote-3) d'Apache.
* L'API est utilisée par les développeurs. C'est une fonction d'apache.

Programmation :

* Permet la programmation côté serveur (PHP, Perl, servlets Java, Java, Server Page, Active Server Pages, CGI, FastCGI, Server-Side Includes)

Serveur mandataire :

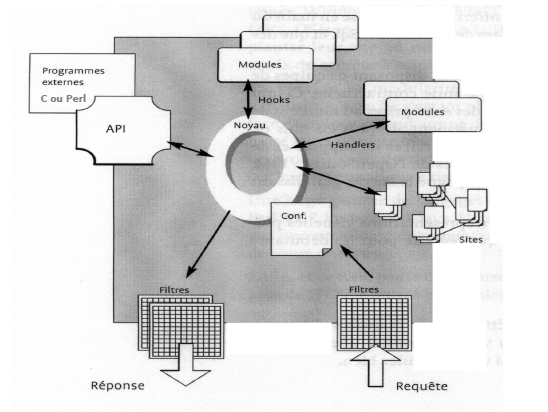
* Apache peut être configuré en proxy serveur à l'aide de son module mod\_proxy.
* Le serveur mandataire n'est pas la fonction 1ère d'apache.

Sécurité :

* Par différentes méthodes d'authentification de l'utilisateur (fichier plats ou bases de données dbm-mysql-...)
* Par l'autorisation du chiffrement sur Internet via l'échange de certificats numériques (SSL)

# Architecture

## Vue d'ensemble



* Le noyau peut être mis à jour via un API.
* Autour du noyau, on peut avoir divers modules.
* Apache contient un fichier de conf.
* Il utilise des filtres (modules) en entrée et en sortie.
* Il peut décompresser une requête et compresser une réponse (grâce aux filtres).

## Le noyau

* Contient toutes les fonctionnalités de base du serveur (ex. directive Listen).
* Il distribue le travail aux modules ou aux programmes externes.
* Les documents sont extraits du ou des sites dont il a la charge.
* Il reçoit et comprend une requête.
* Il prépare et envoie une réponse.
* Il connaît ses ressources disponibles (modules disponibles, liste et localisation des sites à gérer ...) grâce à son fichier de configuration (httpd.conf).
* Pour chaque module qu'il va solliciter, il dispose d'un jeu de directives.

## Les modules

Ils étendent les fonctionnalités du noyau.

Ils peuvent être :

* Standards :
  + Maintenus par l'ASF (ex. mod\_auth\_basic)
  + Font partie de la distribution d'Apache (http://httpd.apache.org/docs/2.4/fr/mod)
* Tiers :
  + Maintenus par des tiers (ex. mod\_evasive)
  + Disponibles sur la toile.

Ils peuvent être installés :

* Lors de la compilation (ou de l'installation) de la distribution d'Apache (modules standards)
* Lors d'une compilation séparée (modules tiers) via l'utilitaire *apxs*# path\_to/apxs -cia module.c (-cia : compile, installe et active)

Le choix des modules à utiliser se fait dans le fichier de configuration principal. Deux directives sont disponibles pour gérer les modules :

|  |  |  |  |
| --- | --- | --- | --- |
| LoadModule : | qui permet de charger un module au démarrage | | |
| <IfModule> : | qui permet d'activer certaines parties du fichier de configuration si un module a été chargé. | | |
|  |  | | |
| httpd -l | Pour voir la liste des modules compilés dans le cœur d'Apache. Ce n'est pas la liste des modules chargés dynamiquement via la directive LoadModule.  Donc, il liste les modules static et les composantes du noyau. | | |
|  | Compiled in modules:  core.c http\_core.c mod\_so.c | | core.c et http\_core.c doivent toujours être présents  mod\_so.c permet le bon fonctionnement de la directive LoadModule |
|  |  | |  |
| httpd -L | grep -i userdir | | Pour connaître le module correspondant à chaque directive (ici la directive 'UserDir'). | |

## Les filtres

Les filtres sont des modules standards ou tiers que vous choisissez d'incorporer ou non à Apache.

* Soit à l'entrée, sur le chemin des demandes entrantes
* Soit en sortie, sur le chemin des réponses sortantes

Par exemple un module de compression des données qui :

* En entrée décompresse des requêtes compressées par un browser
* En sortie compresse l'envoi de tous les documents du site

# Installation

On peut installer Apache par 2 manières.

* Par les sources
* Par les binaires

## Par les binaires

Via la commande d'installation automatique de la distribution.

Ce sera la retenue pour la suite du cours malgré que cette installation soit moins souple que par compilation des sources (ciblage des modules plus difficile) ... par contre, elle offre un outil de désinstallation.

CentOS 8 : dnf install httpd -y (Apache version 2.4.37)

Nous retiendrons cette distribution car :

* La version d'Apache offerte est de la dernière branche 2.x
* De plus, le protocole HTTP 2 est correctement supportée à partir d'Apache 2.4.23.

## Architecture sous CentOS 8

/etc/httpd/ → dossier contenant l'ensemble des fichiers de configuration.

* conf/httpd.conf → fichier principal de configuration.
* conf.d → dossier contenant les fichiers secondaires de configuration.
* conf.modules.d → dossier contenant les fichiers de lancement des modules.

/var/www/ → dossier contenant les données du site par défaut.

* ~~cgi-bin: dossier (vide) contenant les scripts~~. Il n'est plus trop utilisé.
* html : dossier (vide) contenant les pages du site par défaut.

/var/log/httpd/ → dossier contenant les journaux

* access\_log → journal des accès aux pages traitées par le serveur.
* error\_log → journal des erreurs.

/usr/lib64/httpd/modules → dossier contenant le binaire des modules (.so)

/usr/sbin/httpd → le daemon Apache

systemctl start httpd.service → lancement d'Apache (port d'écoute par défaut: 80)

systemctl enable httpd.service → sera lancé au démarrage du système

# Configuration

## Structure de l'httpd.conf

Il est construit en 3 sections, la section environnement global, une pour la configuration du site et une pour la configuration des hôtes virtuel.

Section 1 : Environnement global 1

* Les directives contrôlent le comportement global des processus Apache.
* On y retrouvera les directives : ServerRoot, Listen, LoadModule, User, Group etc.

Section 2 : Configuration du site "principal" ou "par défaut"

* Les directives fournissent les paramètres du site par défaut, celui qui répond aux requêtes qui ne sont pas gérées par un hôte virtuel. Ces directives servent également à fournir les valeurs par défaut pour tous les hôtes virtuels.
* On y retrouvera les directives : ServerAdmin, ServerName, DocumentRoot

Section 3 : Configuration des "hôtes virtuels"

* Directives qui sont propres à hôte virtuel. Le processus central leur attribue les requêtes envoyées à des adresses ou à des noms de sites particuliers.
* On y retrouvera les conteneurs VirtualHost.

Il est préférable de ne pas modifier le httpd.conf. A la place, on ira créer un fichier .conf dans le dossier /etc/httpd/conf.d

Cela permet de ne pas bloquer les mises à jour éventuelles du fichier principal et simplifiera grandement les migrations.

## Contexte des directives

L'aide officielle d'Apache classe les directives dans 5 catégories

|  |  |  |
| --- | --- | --- |
| **s** | server config | (Section 1 ou 2) |
| **v** | virtual host | (Section 3) |
| **d** | directory |  |
| **h** | .htacces |  |
| **p** | proxi |  |

Contexte général du serveur (server config - s)

* Agit sur tout le serveur.
* ex.
  + StartServers (uniquement dans ce contexte)
  + ServerName (dans ce contexte mais peut être redéfinie pour chaque hôte virtuel)

Contexte hôte virtuel (virtualhost – v)

* De nombreuses directives d'hôte virtuel (définies dans le conteneur <VirtualHost>) surchargent celles qui sont générales au serveur.
* ex.
  + ServerName
  + DocumentRoot

# Environnement principal

## La directive ServerName

Apache doit toujours pouvoir déterminer le nom d'hôte de la machine sur laquelle il tourne car il l'utilise pour créer des URL d'autoréférence.

On aura toujours un dossier /var/www/html/ où se trouve le site maitre/principal avec une page index.html

Exemple : ServerName [www.mysite.be](http://www.mysite.be/)

## La directive ServerRoot

Répertoire dans lequel les fichiers de configuration, les logs et les modules sont gardés. Ce nom de dossier servira à préfixer tout chemin relatif rencontré dans l'httpd.conf.

Exemple : ServerRoot "/etc/httpd"

## La directive DocumentRoot

Dossier dans lequel les documents du site par défaut sont déposés.

C'est donc ce dossier qui contient les fichiers qu'Apache fournit lorsqu'il reçoit des requêtes avec l'URL /.

Exemple : DocumentRoot "/var/www/html"

☞ Le changement de nom du DocumentRoot doit aussi être effectué dans Son conteneur <Directory ...>, qui regroupe toutes les directives s'appliquant à DocumentRoot et ses sous-répertoires.

## La directive ServerAdmin

Adresse mail du webmaster qui pourrait s'afficher sur certains documents construits par le serveur et renvoyés au client en cas d'incidents.

Valable si la directive ServerSignature est à l'état on permettant alors l'ajout de cette information en bas de page.

Exemple:   
ServerAdmin [webmaster@mysite.be](mailto:webmaster@mysite.be)  
ServerSignature on

## La directive Listen

Pour définir les adresses IP et les numéros de ports sur lesquels Apache attend et reçoit les connexions des clients.

Exemples :

|  |  |  |
| --- | --- | --- |
| Listen 80 |  | Apache écoute sur toutes les interfaces sur le port 80. |
| Listen 10.0.0.7:80 |  | Apache écoute sur le port 80 sur l'interface d'IP 10.0.0.7. |
| Listen 80  Listen 8080 |  | Apache écoute sur le port 80 et 8080 sur toutes les interfaces. |
| Listen 192.168.1.1:80  Listen 216.180.25.168:443 |  | Apache répond aux requêtes http sur l'interface interne et aux requêtes https (connexions SSL) sur l'interface publique. |

## La directive ErrorDocument

Pour remplacer les pages d'erreur standards envoyées au client en cas de problème.

Exemples :

ErrorDocument 403 "Vous n'êtes pas autorisé à lire cette page !  
Ici on affiche simplement un texte adapté à l'erreur.

ErrorDocument 401 /missing.html  
Ici on affiche une page html sensée se trouver à la racine du site web.

ErrorDocument 500 <http://www.bidon.com/erreur.html>  
Ici on affiche une page html extérieure au site.

☞ ErrorDocument 401 nécessite toujours une URL interne.

## La directive ServerTokens

Permet de contrôler le contenu de l'en-tête Server inclus dans la réponse envoyée au client.  
Elle sert donc à la sécurité d'Apache.  
On travaille avec le ServerTokens Prod.

Exemples:

**ServerTokens Prod[uctOnly]**

* Le serveur renvoie (par ex.): Server: Apache

**ServerTokens Major**

* Le serveur renvoie (par ex.): Server: Apache/2

**ServerTokens Minor**

* Le serveur renvoie (par ex.): Server: Apache/2.0

**ServerTokens Min[imal]**

* Le serveur renvoie (par ex.): Server: Apache/2.0.41

**ServerTokens OS**

* Le serveur renvoie (par ex.) : Server: Apache/2.0.41 (Unix)

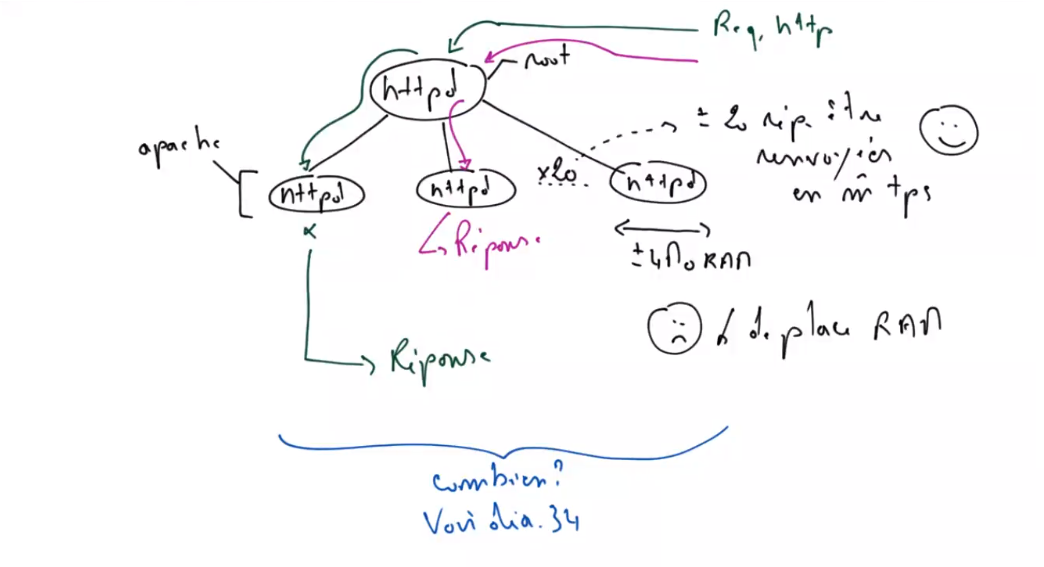
**ServerTokens Full** (valeur par défaut)

* Le serveur renvoie (par ex.) : Server: Apache/2.0.41 (Unix) PHP/4.2.2

# Contrôler Apache

## L'arbre des processus

Apache lance plusieurs daemons en parallèle ; ceux-ci se trouvent en permanence à l'écoute du réseau afin de pouvoir répondre rapidement à un grand nombre de requêtes simultanées.



Le processus principal est celui qui "orchestre" les autres instances. Il peut y avoir jusqu'à maximum 256 instances (fils).  
Chaque instance sont attachées à un utilisateur : apache.

Inconvénient : une instance peut aller jusqu'à 4Mb en ram

## Les directives User et Group

Utilisateur et groupe Linux sous lequel s'exécuteront les processus fils d'Apache chargés de répondre aux requêtes des clients.

Le processus maître doit être lancé sous le compte root pour pouvoir changer le user et le group de ses processus fils.

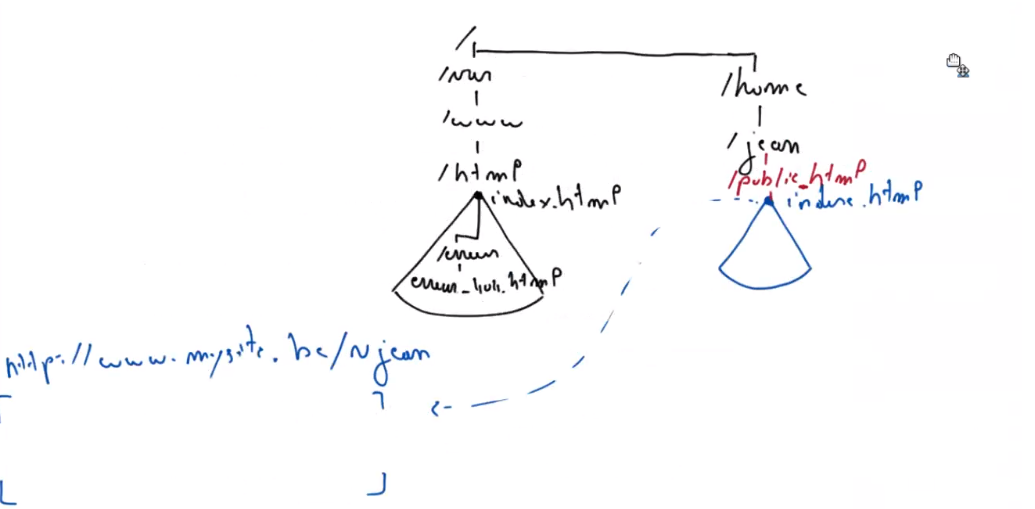
L'user "apache" est créé à son installation  
Si apache n'a pas le droit de lecture sur un html ou quelque chose d'autres, il ne saura pas le lire et le renvoyer

Exemples :  
User apache  
Group apache

# Les sites persos

Il est possible de permettre aux utilisateurs disposant d'un compte sur le serveur de posséder leur propre site.

* Pratique très courante parmi les FAI qui proposent l'hébergement de pages web de leurs clients.
* Cette fonctionnalité est fournie par le module standard mod\_userdir. LoadModule userdir\_module (dans 00-base.conf)...
* Elle est activée par la directive Userdir.
* Le site sera accessible vial'URL: http://ip\_serveur/~login\_utilisateur Exemple: <http://www.mysite.be/~jean>



Autres formes de la directive UserDir :

* UserDir disabled <user1 user2 ...>
  + Désactive la gestion des sites perso pour la liste des utilisateurs indiquée.
* UserDir disabled
  + Désactive la gestion de tous les sites persos.
  + Souvent utilisée avant une directive UserDir enabled.
* UserDir enabled <user1 user2 ...>
  + Active la gestion des sites perso pour la liste des utilisateurs indiquée.

Exemple :   
UserDir disabled  
UserDir enabled jean louis

# Les redirections simples

La directive standard Alias permet d'accéder facilement à des documents HTML en dehors de l'arborescence DocumentRoot.

- Cette fonctionnalité est fournie par le module standard mod\_alias.  
 LoadModule alias\_module ...

- Soit accéder à la page HTML /usr/share/doc/HTML/index.html via le site www.mysite.be par redirection:

Alias /CentOS "/usr/share/doc/HTML/"  
 <Directory "/usr/share/doc/HTML">  
 Require all granted  
 </ Directory>

Cette page sera accessible via l'URL: <http://www.mysite.be/CentOS>

Si on n'a pas d'alias, il essayera d'aller chercher un dossier CentOS dans le /var/www/html avec une page index.html.

Pour aller chercher une page en dehors du site web, il faut utiliser les alias : le module alias est chargé par défaut

Require all granted : On lui dit qu'on a le droit d'y aller

L'Alias ne fonctionne pas sans le Require all granted

# Les index de répertoires

## Recherche d'une page d'accueil

Apache est capable de retrouver la page index.html sans pour autant que celle-ci soit indiquée dans l'URL.

Exemple : <http://www.mysite.be/cours> équivaut à: <http://www.mysite.be/cours/index.html>

* C'est le module standard mod\_dir qui recherche et fournit une page (index.html par défaut) susceptible de se trouver dans le répertoire indiqué dans l'URL.
* Si la directive DirectoryIndex est présente, elle permettra d'ajouter des références à d'autres pages.

<IfModule dir\_module>  
 DirectoryIndex index.html index.htm index.php  
</IfModule>

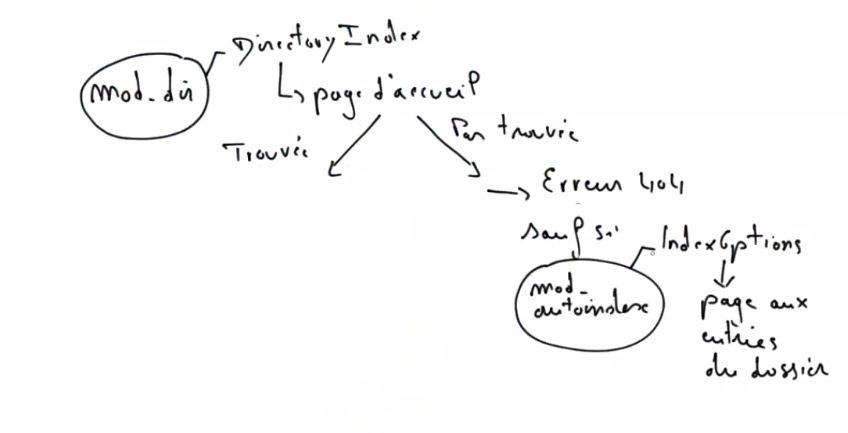
* Si aucune page d'accueil n'est trouvée, le module mod\_autoindex (piloté par sa directive IndexOptions) crée un index des fichiers du répertoire concerné.

Exemples :  
IndexOptions None (indexation classique)  
IndexOptions FancyIndexing (indexation au look plus agréable)  
IndexOptions FancyIndexing VersionSort (idem + tri des entrées contenant des numéros de versions)

* Pour supprimer l'indexation sur un dossier particulier

<Directory directory\_name>  
Options -Indexes  
</Directory>

Avec un Options -Indexes on a une erreur 403

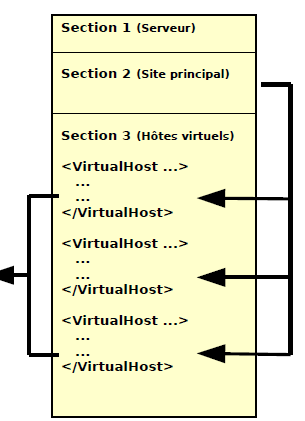


# Hébergement virtuel

## Introduction

* Un seul service httpd pour gérer plusieurs sites web.
* Chaque Ip du serveur peut être dédicacée à la gestion d’un ou plusieurs sites web (HTTP/1.1).

## Principe

On ajoutera dans la Section 3 du httpd.conf les directives propres à chaque hôte virtuel.   
Chaque hôte virtuel hérite des directives du maitres (site principal).  
La directive 'Listen' n'est pas autorisée en VirtualHost.

# Section 2

ServerName [www.mysite.be](http://www.mysite.be/)  
DocumentRoot /var/www/html

...

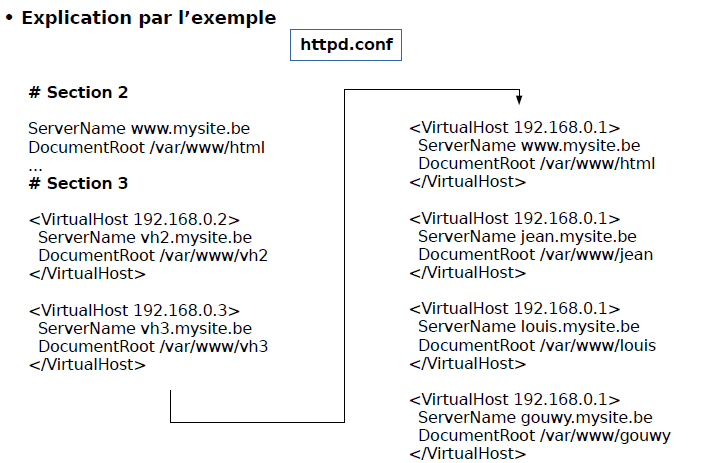
# Section 3

<VirtualHost 192.168.0.1>  
 ServerName [www.mysite.be](http://www.mysite.be/)  
 DocumentRoot /var/www/html  
</VirtualHost>

<VirtualHost 192.168.0.2>  
 ServerName vh2.mysite.be  
 DocumentRoot /var/www/vh2  
</VirtualHost>

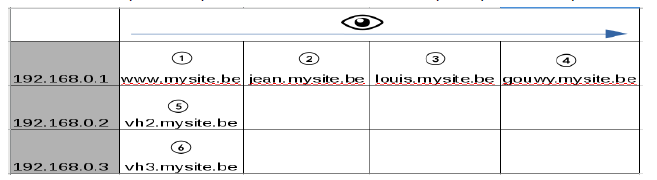
## Explication par un exemple

On écrira cette config dans /etc/httpd/conf.d/virtual.conf et non dans httpd.conf



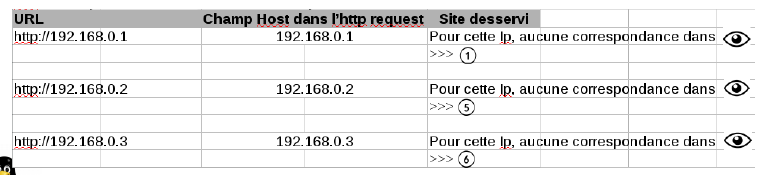
A la lecture de l’httpd.conf, Apache crée une table contenant la liste des hôtes virtuels (servername) déclarés pour chaque Ip.

Lorsque Apache reçoit une requête sur une Ip se trouvant dans la table, il recherche le ServerName qui correspond à l'en-tête Host de la requête pour cette Ip…





Si aucune correspondance n'est trouvée, c'est le premier hôte virtuel de la liste qui sera sélectionné. Ainsi, pour qu'il soit toujours accessible, on indique souvent le site maître en tant que premier de liste dans l'entrée correspondant à son Ip.



Lorsque Apache reçoit une requête sur une Ip sur laquelle il écoute mais qu'aucun site virtuel n'est attaché à celle-ci, alors c'est le site principal défini en section 2 de l'httpd.conf qui sera desservi.



Le sens du tableau ne changera pas tant qu'on ne le change pas dans le fichier virtual.conf

## Création d'alias d'interface

La création d'un alias sur une interface *iface* se fait par la création d'un fichier :

/etc/sysconfig/network-scripts/ifcfg-iface:0

IPADDR=nouvelle\_ip  
NETMASK=nouveau\_masque  
ONBOOT=yes  
NAME=iface:0  
DEVICE=iface:0

Pour créer d'autres alias sur cette même interface, il suffit de créer d'autres fichiers en incrémentant le numéro d'alias (…**:1**, …**:2**, etc ) et en adaptant les valeurs des variables NAME et DEVICE en conséquence.

# Les fichiers journaux

Pour véritablement gérer un serveur web, il est nécessaire de disposer d'un retour d'informations à propos de l'activité et des performances du serveur, ainsi que de tout problème qui pourrait survenir.

**• Le journal des erreurs** (Directives ErrorLog & LogLevel)

Directive ErrorLog : Le nom et la localisation du journal.

Exemples

ErrorLog logs/errors-logs Enregistrement vers un fichier particulier.

ErrorLog "|/usr/local/bin/erreurs\_httpd" Traitement de l'erreur par un binaire.

ErrorLog syslog:user Traitement de l'erreur par le daemon syslogd.

Directive LogLevel : Indique quels sont les messages à écrire dans le fichier journal.

Exemple LogLevel warn

Tableau des criticités en ordre croissant

emerg Urgences - le serveur est inutilisable.

alert Des mesures doivent être prises immédiatement.

crit Conditions critiques (accès réseau impossible par ex.).

error Erreurs dans les pages, les scripts.

warn Avertissements (pages mal codées, erreurs non bloquantes dans un script...

notice Evénement important mais normal.

info Informations.

debug Enregistre TOUT ce qui se passe sur le serveur.

☞ Lorsqu'un niveau particulier est spécifie, les messages de tous les autres niveaux de criticité supérieure seront aussi enregistrés (ex. le niveau crit enregistre en plus les messages de niveau alert et emerg ).

1. CGI : Common Gateway Interface [↑](#footnote-ref-2)
2. API : application programming interface [↑](#footnote-ref-3)