

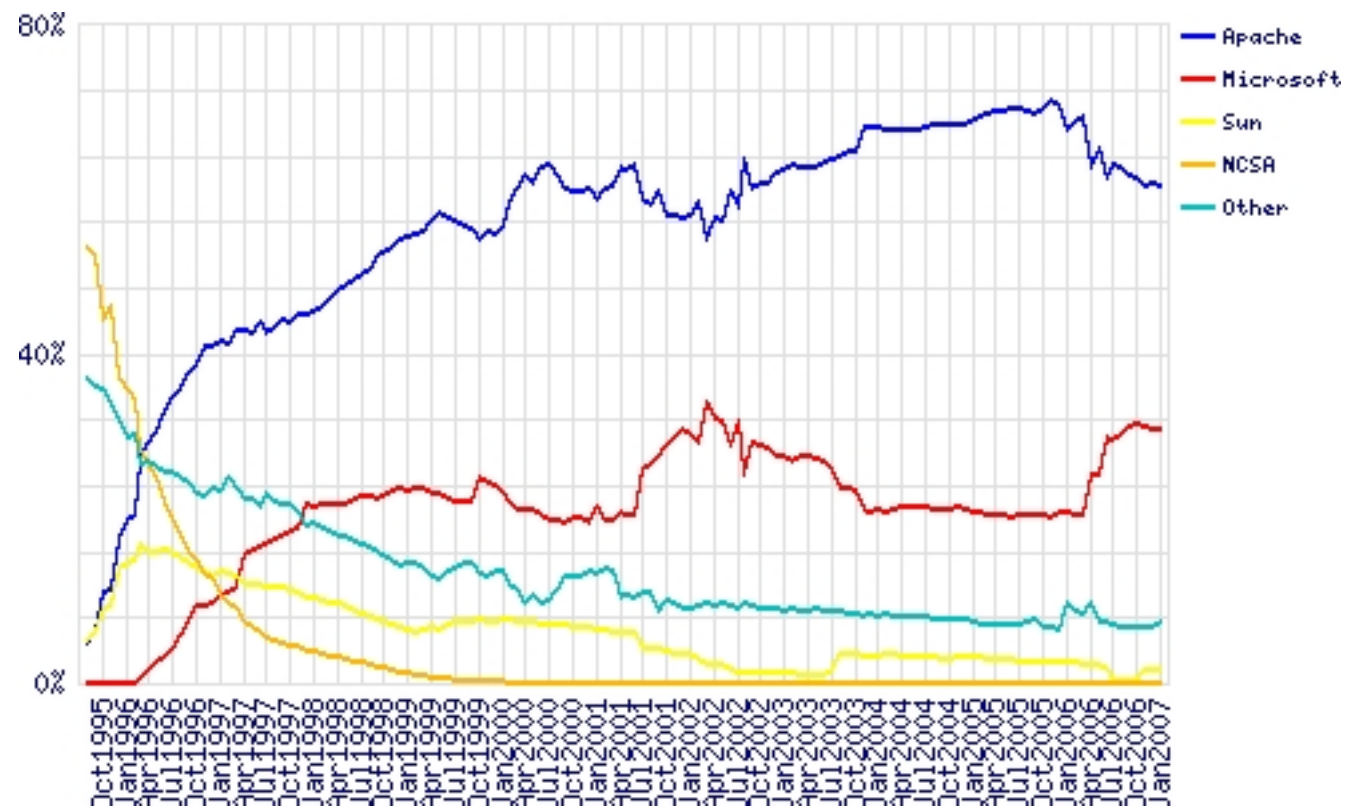
# Configuration du serveur Web Apache

# Configuration du serveur Web Apache

Pourquoi Apache ?

## Quelques chiffres

- ▶ <http://news.netcraft.com>, surveille l'utilisation des systèmes d'exploitation et serveurs web des principales sociétés.



# Fonctionnalités

- ▶ Configuration assez simple
- ▶ Limitation possible des accès aux répertoires
  - ▶ Par l'administrateur
  - ▶ Par les utilisateurs sur les répertoires dont ils ont les auteurs
- ▶ Accès sécurisé en fonction des adresses IP
- ▶ Chargement de modules pour ajouter de nouvelles fonctionnalités (php, mysql, ssl, ... )
- ▶ Possibilité de réécrire les adresses web à la volée
  - ▶ Au lieu d'afficher dans l'url  
<http://www.mondomaine.com/toto.php?id=14>, on va faire afficher  
<http://www.mondomaine.com/toto14.html>
- ▶ Hébergements de plusieurs sites web sur un même serveur (via les VirtualHosts)
- ▶ Mise en place d'authentification http
- ▶ Système de log personnalisable

# Configuration du serveur Web Apache

## Les paramètres de base

# Les paramètres de base (1)

- ▶ ServerType standalone : 2 valeurs possibles
  - ▶ standalone pour un serveur autonome
  - ▶ inetd si c'est le gestionnaire de service inetd (ou xinetd) qui réceptionne les requêtes http et les envoie serveur Web. Dans ce cas, il ne faut pas oublier de configurer le service http dans inetd (ou xinetd)
- ▶ ServerRoot /var/lib/apache : définit le répertoire d'installation du serveur
- ▶ DocumentRoot /var/www : définit le répertoire dans lequel le site web est stocké. Généralement c'est /var/www ou /var/www/html
- ▶ Timeout 300 : temps d'attente maximal du serveur d'une réponse d'un programme de traitement externe (parseur PHP, script CGI, ...) A expiration de ce temps, le serveur envoie une erreur au client et au programme externe lui ordonnant d'arrêter son exécution.
- ▶ KeepAlive on : autorise les connexions persistantes. Dès qu'un client s'est connecté, si la connexion est persistante, il va pouvoir envoyer plusieurs requêtes à la fois. Elles seront ainsi traitées plus rapidement

## Les paramètres de base (2)

- ▶ MaxKeepAliveRequests 100 : indique le nombre maximum de requête par connexion. 0 indique une quantité infini !
  - ▶ Si le serveur est très sollicité, il faudra dans un premier temps diminuer cette quantité puis si les problèmes persistes, mettre KeepAlive à Off. Les connexions persistantes peuvent empêcher d'autres utilisateurs d'obtenir leur réponse.
- ▶ KeepAliveTimeout 15: valeur d'attente de la requête suivante venant d'un même client avant d'envoyer un timeout au client
- ▶ MinSpareServers 5 et MaxSpareServers 10 : cela sert à l'autorégulation de la charge du serveur. Si le nombre de processus enfant du serveur dans l'état idle (ne traitant aucune requête) est inférieur à MinSpareServers, le serveur en crée un. Si ce nombre est supérieur à MaxSpareServers, alors il en supprime un
  - ▶ La modification de ces paramètres est à faire uniquement sur des serveurs très chargés, devant répondre à de nombreuses requêtes

## Les paramètres de base (3)

- ▶ StartServers 5 : nombre de serveur à lancer au démarrage
- ▶ MaxClients 150 : indique le nombre maximum de serveurs pouvant fonctionner simultanément
  - ▶ Le nombre maximum est de 256. Pour pouvoir dépasser cette limite, il faut recompiler le serveur après modification de HARD\_SERVER\_LIMIT dans httpd.h
- ▶ ExtendedStatus on : retourner des informations détaillées sur l'activité et les performances du serveur
  - ▶ Fonctionne que si le module mod\_status est chargé et si une directive `<Location /server-status> SetHandler server-status </Location>`
- ▶ Port 80 : port d'écoute du serveur
- ▶ User www et Group www : détermine l'utilisateur et le groupe utilisés par le serveur. **Ne jamais mettre root**

## Les paramètres de base (4)

- ▶ ServerAdmin email : email de l'admin
- ▶ ServerName non.domain.fr : nom et domaine du serveur
- ▶ DocumentRoot /var/www : répertoire utilisé pour le stockage des pages HTML du site racine
- ▶ HostnameLookups Off : indique si le serveur enregistre le nom (on) ou bien l'IP (off) du client qui se connecte
- ▶ CacheNegotiatedDocs : autorise ou pas les proxies à mettre les documents en cache. Pour refuser cette mise en cache, commentez cette ligne (avec un # en début de ligne)
- ▶ DefaultType text/plain : définit le type Mime des documents transmis par le serveur. Dans le cas de diffusion majoritairement d'images ou streaming, mettre application/octet-stream pour éviter que les clients affichent des caractères étranges u%\*d\$gge,s&dg£fæ
- ▶ ServerSignature on : retourne la version du serveur en signature
- ▶ Alias /icons /usr/share/apache/icons : permet de faire des alias

# Archivage des erreurs

- ▶ ErrorLog /var/log/apache/error.log : chemin du fichier utilisé pour stocker les messages d'erreurs du serveur
- ▶ LogLevel warn : définit le niveau d'enregistrement des erreurs. Les valeurs possibles sont :
  - ▶ emerg : enregistre seulement les erreurs qui rendent le serveur inutilisable
  - ▶ alert : emerg + erreurs nécessitant une intervention
  - ▶ crit : "emerg" + "alert" + erreurs critiques (accès réseau impossible par exemple)
  - ▶ error : "emerg" + "alert" + "crit" + erreurs dans les pages, les scripts
  - ▶ warn : "emerg" + "alert" + "crit" + "error" + erreurs non bloquantes (pages mal codées, scripts comportant des erreurs non bloquantes...)
  - ▶ notice : normal mais information significative
  - ▶ info : "emerg" + "alert" + "crit" + "error" + "warn" + toutes les informations générées du type « serveur sur-chargé »
  - ▶ debug : enregistre TOUT ce qui peut se passer sur le serveur
- ▶ Mettre toujours au moins le niveau error

# Alias sur des répertoires

- ▶ Supposons que l'on désire accéder au répertoire <http://www.monsite.com/docs>
- ▶ Si nous avons DocumentRoot /home/www, alors le serveur va aller lire /home/www/docs
- ▶ Si docs est en réalité le répertoire /var/web, nous avons 2 solutions :
  - ▶ Soit on fait un lien symbolique (`ln -s /var/web /home/www/docs`) et on s'assure que l'option FollowSymLinks ou SymLinksIfOwnerMatch est présente
  - ▶ Soit on utilise les alias :
    - ▶ Alias /docs /var/web
    - ▶ Dans ce cas, <http://www.monsite.com/docs/files.html> va lire le répertoire /var/web/files.html

# Configuration du serveur Web Apache

## La directive Directory

# La directive Directory

- ▶ Permet de définir les règles sur des répertoires

```
<Directory "/var/lib/apache/htdocs">  
  Options Indexes FollowSymlinks Multiviews  
  AllowOverride None  
  Order allow,deny  
  allow from all  
</Directory>
```
- ▶ Dans l'exemple, avec les options, nous définissons les informations suivantes :
  - ▶ indexes : autorise le serveur à fabriquer une liste des fichiers et répertoires disponibles
  - ▶ FollowSymlinks : autorise le serveur à suivre des liens symboliques
  - ▶ Multiviews : permet par exemple d'afficher des pages en fonction de la langue du client
- ▶ AllowOverride None : personne n'est autorisé à redéfinir les droits d'accès à ce répertoire

# Options dans Directory

- ▶ Liste des options possibles dans la directive Directory
  - ▶ None : désactive toutes les options
  - ▶ All : active toutes les options, sauf multiviews
  - ▶ Indexes : le serveur gère automatiquement un index du répertoire si le index.htm (ou index.html, ...) est manquant; **option dangereuse en fonction du contenu du répertoire**
    - ▶ Ne modifie pas le chemin d'accès défini dans Directory
    - ▶ Option ignoré dans une section <Location>
  - ▶ FollowSymLinks : autorise à suivre les liens symboliques
  - ▶ SymLinksIfOwnerMatch : autorise à suivre les liens seulement si le user id du fichier sur lequel le lien pointe est le même que celui du lien
  - ▶ ExecCGI : possibilité d'exécution de scripts CGI à partir de ce répertoire
  - ▶ Multiviews : autorise les vues multiples en fonction du contexte. Par exemple, en fonction de la langue du client

## Options dans Directory (2)

- ▶ Includes : autorise l'inclusion de document dans des pages HTML avec la commande `<!-- #include toto.shtml-->`, ou bien l'exécution de commande comme `ls`
  - ▶ Exemple :

```
<html>
<body>
<!--#exec cmd="ls" --><BR/>
La date locale <!--#echo var="DATE_LOCAL" -->
</body>
</html>
```
  - ▶ Pour plus d'info, voir <http://httpd.apache.org/docs/howto/ssi.html>
- ▶ IncludesNOEXEC : permet les includes mais empêche l'exécution de commande

# La directive AllowOverride

- ▶ AllowOverride : définit les types de redéfinitions autorisées par le fichier de contrôle d'accès
  - ▶ AllowOverride All : autorise tous les types de redéfinition de contrôle d'accès
  - ▶ AllowOverride AuthConfig : active les directives d'autorisations (AuthDBMGroupFile, AuthDBMUserFile, AuthGroupFile, AuthName, AuthDigestRealmSeed, AuthType, AuthUserFile, Require, ...)
  - ▶ AllowOverride FileInfo : active les directives de contrôle des types de documents (AddEncoding, AddLanguage, AddType, DefaultType, ErrorDocument, LanguagePriority, etc.)
  - ▶ Indexes : autorise les directives de création d'index d'un répertoire (AddDescription, AddIcon, AddIconByEncoding, AddIconByType, DefaultIcon, DirectoryIndex, FancyIndexing, HeaderName, IndexIgnore, IndexOptions, ReadmeName, etc.)
  - ▶ Limit : autorise les contrôles sur les accès des machines (Allow, Deny and Order)
  - ▶ Options : autorise des contrôles spécifiques sur les fonctionnalités Options
  - ▶ None : ne permet pas de redéfinir les contrôles d'accès

# Configuration du serveur Web Apache

Autres directives

## La directive <Files>

- ▶ Permet de contrôler les accès sur un fichier, de la même manière que le fait <Directory> sur les répertoires
- ▶ Généralement utilisé sur un ensemble de fichier particulier
  - ▶ *Exemple* : Interdit à tout client d'accéder aux fichiers commençant par .ht

```
<Files ~ "^\.ht">  
    Order allow,deny  
    Deny from all  
</Files>
```

- ▶ Syntaxe : <Files *Filename*>...</File>
  - ▶ *Filename* peut contenir un nom de fichier mais aussi une chaîne de caractère utilisant ? pour remplacer un unique caractère ou \* pour remplacer n'importe quelle séquence de caractères
  - ▶ L'utilisation d'expression régulière est possible mais on ajoute alors le caractère ~ comme dans l'exemple
- ▶ Plus d'info sur les expressions régulières utilisées dans Apache : <http://www.perl.com/doc/manual/html/pod/perlre.html> (ce sont les mêmes qu'en perl)

# Configuration du serveur Web Apache

Les contrôles d'accès

# Contrôle sur les clients

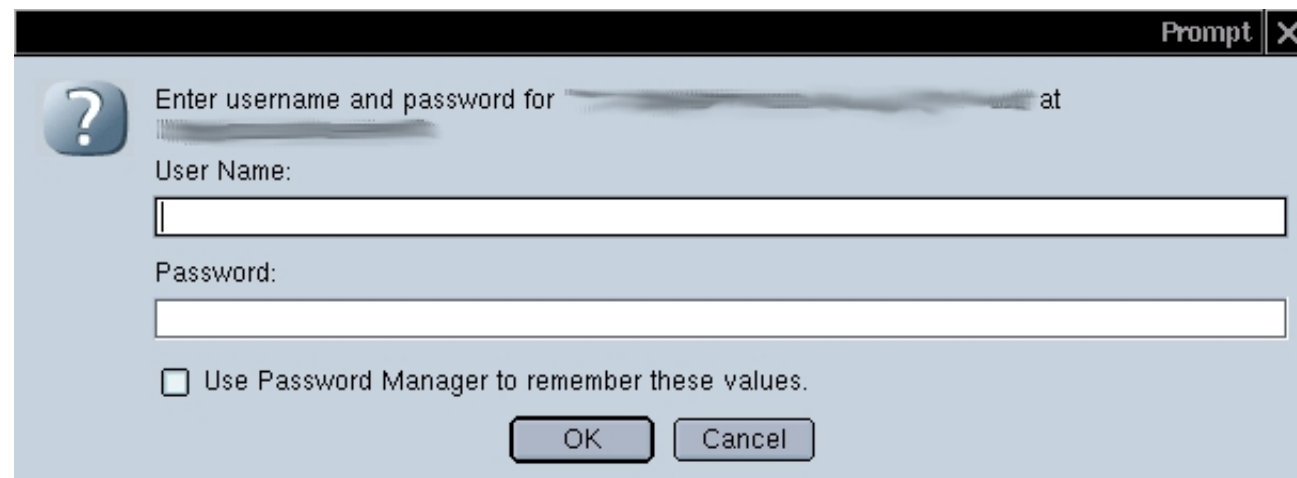
- ▶ Il est possible d'autoriser ou pas certain accès en fonction de l'adresse IP ou de l'adresse réseau du client
  - ▶ *Exemple :*

```
Order Allow, Deny
Allow from 192.168.0.0/24, 89.45.47.14
Deny from All
```
  - ▶ Order : détermine l'ordre d'applications des règles allow et deny
  - ▶ Fonctionnement général :
    - ▶ Order Deny, Allow
      - ▶ La directive Deny est évaluée en premier. Par défaut, les accès sont autorisés. Un client est autorisé s'il ne vérifie pas la directive Deny ou s'il vérifie une directive Allow
    - ▶ Order Allow, Deny
      - ▶ La directive Allow est évaluée en premier. Par défaut, les accès sont refusés. Un client est refusé s'il ne vérifie pas la directive Allow ou s'il vérifie une directive Deny
  - ▶ Dans l'exemple : les clients d'adresse réseau 192.168.0.0/24 et 89.45.47.14 sont autorisés et tous les autres sont refusés
- ▶ Cette directive est autorisée dans <Directory> et .htaccess si le module mod\_access est chargé

## Les htaccess ?

- ▶ Il est possible de redéfinir les règles d'accès à un répertoire en plaçant dans celui-ci un fichier texte appelé fichier de contrôle d'accès dont le nom est défini par la directive `AccessFileName`
  - ▶ *Exemple* : `AccessFileName .htaccess` définit le nom `.htaccess` pour le fichier de contrôle d'accès.
- ▶ Les règles d'accès qu'il est possible de redéfinir sont gérées par la directive `AllowOverride`. Pour interdire les redéfinitions, il suffit de mettre `AllowOverride none`
- ▶ Si la redéfinition des droits est autorisée, le serveur va vérifier l'existence du fichier de contrôle d'accès dans tous les répertoires du chemin correspondant aux fichiers demandés
  - ▶ *Exemple* : `AccessFileName .htaccess` et si la page demandée est dans `/home/www/doc/index.html`, le serveur va vérifier le répertoire `/.htaccess`, `/home/.htaccess`, `/home/www/.htaccess`, puis `/home/www/doc/.htaccess`

# Authentication (1)



► Exemple d'authentification http

AuthType Basic

AuthName "Password Required"

AuthUserFile /www/passwords/password.file

Require valid-user

► Possibilité de mettre ces lignes soit dans <Directory>, soit dans un fichier de contrôle d'accès (si AllowOverride AuthConfig est défini)

## Authentification (2)

- ▶ L'authentification basic (`AuthType Basic`) nécessite la création d'un fichier de mot de passe
- ▶ Création du mot de passe
  - ▶ `htpasswd -c /www/passwords/password.file username`
  - ▶ `-c` quand on crée le fichier de mot de passe, sinon on ne met plus cette option
  - ▶ Définit le fichier de mot de passe à utiliser dans la configuration : `AuthUserFile /www/passwords/password.file`
- ▶ Il est possible de donner l'accès qu'à certaines personnes ou à toutes personnes avec la directive `Require`
  - ▶ `Require user toto tata tutu`
  - ▶ `Require valid-user`
- ▶ Il est possible de définir des groupes de personnes dans un fichier texte :
  - ▶ Format du fichier :

```
authors: rich daniel allan
```
  - ▶ `AuthGroupFile /www/passwords/groups.file`
  - ▶ `Require group authors`
  - ▶ Dans ce dernier cas, il faut alors avoir un login et mot de passe correcte et appartenir au bon groupe

## Authentification (3)

- ▶ L'authentification basique n'est pas considéré comme sécurisé ! Le login et mot de passe transitent régulièrement sur le réseau, à chaque requête
- ▶ Autres solutions :
  - ▶ Utilisation de md5 pour envoyer le mot de passe
    - ▶ Création des mots de passe, obligation de donner le nom de l'authentification qui utilisera ce fichier

```
htdigest -c /var/www/digest realm username
```
    - ▶ Exemple de directive pour utiliser cette méthode d'authentification

```
AuthType Digest
AuthName "realm"
AuthDigestFile /usr/local/apache/passwd/digest
AuthDigestGroupFile /usr/local/apache/passwd/digest.groups
Require group admins
```
  - ▶ Utilisation de bases de données DB ou DBM : plus d'info sur <http://httpd.apache.org/docs/howto/auth.html>

## Authentication (4)

- ▶ Possibilité de faire des combinaisons de l'authentification avec le contrôle d'accès en fonction des IP

```
<Directory /usr/local/apache/htdocs/sekrit>  
AuthType Basic  
AuthName intranet  
AuthUserFile /www/passwd/users  
AuthGroupFile /www/passwd/groups  
Require group customers  
Order allow,deny  
Allow from internal.com  
Satisfy any  
</Directory>
```

- ▶ Il suffit de mettre toutes les directives et d'ajouter à la fin la directive Satisfy
  - ▶ Soit Satisfy any pour signaler que l'un des 2 contrôles est suffisant
  - ▶ Soit Satisfy all pour imposer la validation des 2 contrôles

# Configuration du serveur Web Apache

## Les Virtual Host

# Définition

- ▶ Les Virtual Host permettent de faire correspondre un même serveur Web à plusieurs adresses différentes
  - ▶ Exemple : [www.mondomaine.com](http://www.mondomaine.com) et [intranet.mondomaine.com](http://intranet.mondomaine.com) peuvent être hébergés sur un même machine
- ▶ 2 types de Virtual Hosts
  - ▶ Name-based Virtual Hosts
    - ▶ On utilise le nom envoyé par le client dans l'URL pour différencier les serveurs virtuels
  - ▶ IP-based Virtual Hosts
    - ▶ on utilise des adresses IP différentes pour chaque domaine

## IP-based ou Name-based ?

- ▶ Limites des name-based
  - ▶ Ne sont compatibles qu'avec des clients supportant HTTP/1.1
  - ▶ On ne peut pas utiliser avec des serveurs SSL de part la nature de SSL
  - ▶ Certains systèmes et réseaux implémentent des optimisations de la bande passante qui ne peuvent différencier les clients sans que ce soit des IP différents
- ▶ Si l'un de ces 3 cas risquent d'arriver sur votre réseau, il est préférable de faire des virtual host basé sur les Ips
  - ▶ Dans ce cas, obligation pour le serveur d'avoir plusieurs interfaces réseaux et donc plusieurs adresses IP ou bien d'utiliser la technique des alias IP

# Les name-based virtual host

- ▶ Déclarer l'adresse IP du serveur avec la directive NameVirtualHost
- ▶ Créer un bloc <VirtualHost> pour chaque hôte que vous voulez servir
- ▶ Dans un bloc, au minimum, il doit y avoir
  - ▶ Une directive ServerName dont le nom désigne celui que vous voulez servir
  - ▶ Une directive DocumentRoot qui désigne la racine du site que vous servez
- ▶ Si vous faites des Virtual hosts sur un serveur Web existant, qui sert déjà une adresse, vous devez faire un bloc <VirtualHost> dont les directives ServerName et DocumentRoot porteront les mêmes paramètres que les directives globales
- ▶ **Attention** : La DNS doit correctement être configurée pour retourner la même adresse IP à plusieurs domaines

## Les IP-based virtual host

- ▶ **Le serveur doit avoir des adresses IP différentes pour chaque host virtuel**
- ▶ 2 solutions :
  - ▶ Un démon http par IP
    - ▶ A utiliser dans le cas où l'on veut restreindre les accès avec les directives User, Group, Listen et ServerRoot
  - ▶ Un unique démon qui utilise des hosts virtuels
- ▶ La configuration se fait de la même manière que pour les name-based virtual host

# Configuration du serveur Ftp Proftpd

# Configuration du serveur Ftp Proftpd

Rappel sur FTP

# FTP : Introduction

- ▶ Service défini au départ par la RFC 959 (1985)
- ▶ Avant l'apparition de HTTP, protocole de transfert de fichiers le plus utilisé.
- ▶ Supporté par la plupart des navigateurs Web ainsi que par des clients spécialisés.
- ▶ Serveur FTP « classique »: wu-ftp (Université de Washington)
- ▶ Ce cours se base sur proftpd, beaucoup plus puissant et configurable.

# FTP : avantages et inconvénients

- ▶ Au lieu de FTP, on peut aussi utiliser HTTP pour transférer des fichiers.
- ▶ Avantages de FTP :
  - ▶ Possibilité de reprise d'un transfert interrompu.
  - ▶ « Upload » plus simple à configurer.
  - ▶ Connexions séparées pour « contrôle » et « données » .
  - ▶ ...
- ▶ Inconvénients
  - ▶ Gestion d'un service supplémentaire sur les serveurs
  - ▶ Certains anciens clients FTP ont des problèmes avec les « firewall »
  - ▶ Risque supplémentaire pour la sécurité
  - ▶ ...

# Configuration du serveur FTP Proftpd

Pourquoi Proftpd ?

## Similaire à Apache

- ▶ S'inspire d'Apache pour le modèle de développement, les principes de conception (flexibilité, ...), le format des fichiers de configuration, ...
- ▶ Grande quantité de directives de configuration.
- ▶ L'architecture supporte des modules externes (comme pour Apache) qui permettent d'ajouter des fonctionnalités (PAM, LDAP, ...)
- ▶ Informations complémentaires :
  - ▶ Site du projet : <http://www.proftpd.org/>
  - ▶ Liste des directives : <http://www.proftpd.org/docs/configuration.html>

# Configuration du serveur FTP Proftpd

## Configuration

# Configuration de base

- ▶ Sur les distributions Debian et RedHat, lors de l'installation de proftpd
  - ▶ l'installation peut créer un répertoire ftp anonyme (`/home/ftp`)
  - ▶ Par défaut, chaque utilisateur peut de se logger dans son répertoire `home`
- ▶ Par convention, les fichiers publics sont placés dans `/home/ftp/pub` si l'accès anonyme est autorisé
- ▶ Certaines distributions permettent aux utilisateurs anonymes d' « uploader » des fichiers généralement dans le répertoire `/home/ftp/pub/incoming`

## La sécurité sur proftpd

- ▶ Le serveur crée un sous-processus par connexion, avec un UID spécifié par le fichier de configuration.
- ▶ Après avoir créé et initialisé ce sous-processus, le serveur peut effectuer un « chroot » sur un répertoire spécifié.
- ▶ Les directives « Directory » et « Limit » peuvent entre autres être utilisées pour limiter l'accès à des fichiers ou à des répertoires.
- ▶ Les informations sur les utilisateurs et les groupes peuvent venir de plusieurs sources : fichiers /etc/passwd et /etc/group (par défaut) mais aussi d'autres fichiers de même structure ou bien de serveur LDAP, de bases de données SQL, etc...
- ▶ Les permissions de ProFTPD ne se substituent pas à celles du système !

# Le fichier de configuration

- ▶ Généralement, ce fichier est le fichier suivant : `/etc/proftpd.conf`
- ▶ Comme celui d'Apache, le fichier de configuration contient des directives globales (hors de tout contexte particulier), et des directives à l'intérieur d'un « contexte », par exemple dans un élément `<Directory>`
- ▶ Les directives globales s'appliquent au serveur dans son ensemble, les autres au contexte dans lequel elles apparaissent
- ▶ Les principaux contextes définis sont « Directory » (répertoire), « Limit » (commandes), « Anonymous » (accès anonyme) et « VirtualHost » (serveur virtuel)

# Listes des utilisateurs et des groupes

- ▶ Les commandes AllowGroup, AllowUser, DenyGroup et DenyUser peuvent prendre en paramètre une liste de groupes ou d'utilisateurs
- ▶ Une telle liste est constituée d'une suite de nom d'utilisateurs ou de groupes, séparés par des virgules
- ▶ On peut exclure explicitement un utilisateur ou un groupe en précédant son nom dans la liste par le caractère « ! »

# Directives globales

- ▶ `ServerAdmin` *admin\_email\_address* : définit l'adresse e-mail de l'administrateur du serveur.
- ▶ `ServerName` *name* : définit le nom du serveur (peut être redéfini par Virtual Host).
- ▶ `ServerType` *type* : « type » ne peut prendre que deux valeurs : *standalone* si le serveur est lancé au démarrage du système, et *inetd* si le serveur est lancé par *inetd*.
- ▶ `DefaultRoot` *répertoire liste\_de\_groupes* : établit le répertoire racine pour les membres des groupes spécifiés. Un utilisateur ne peut pas « remonter » en dessus de son répertoire racine.
  - ▶ Le caractère « ~ » est remplacé par le répertoire « home » de l'utilisateur.
- ▶ Il y a encore beaucoup d'autres directives. Consultez la liste pour plus de détails.

# Les contextes

- ▶ `<Directory chemin >...</Directory>` : précise que les directives incluses s'appliquent au répertoire de nom « *chemin* » (avec expansion des métacaractères)
- ▶ `<Anonymous chemin> ... </Anonymous>` : : précise que les directives incluses s'appliquent au répertoire de nom « *chemin* », qui sera le répertoire racine pour les accès anonymes
- ▶ `<Limit commande ...> </Limit>` : précise que les restrictions ou autorisations incluses s'appliquent à la commande « *commande* »
- ▶ Si plusieurs contextes s'appliquent à un fichier donné, ils s'appliquent par « précision » décroissante.

# Commandes du contexte Limit

- ▶ Dans la directive <Limit *commande* ...>, l'argument commande peut prendre comme valeur soit l'une des commandes définies par le protocole ftp, soit un groupe de commandes
- ▶ Les commandes FTP :
  - ▶ MKD / XMKD (MaKe Directory) : créer un nouveau répertoire
  - ▶ RNFR (ReName FRom), RNTD (ReName TO) : renommer un répertoire
  - ▶ DELE (DELEte) Effacer un fichier
  - ▶ RMD / XRMD (ReMove Directory) Supprimer un répertoire
  - ▶ RETR (RETRieve) Transférer un fichier depuis le serveur vers le client
  - ▶ STOR (STORe) Transférer un fichier du client vers le serveur
- ▶ Groupe de commandes :
  - ▶ READ : toute lecture de fichiers à l'exception de la lecture d'un répertoire (RETR, SITE, SIZE, STAT)
  - ▶ WRITE : toute création, modification ou effacement de fichiers ou de répertoires (APPE, DELE, MKD, RMD, RNTD, STOR, XMKD, XRMD)
  - ▶ DIRS : listage des répertoires (CDUP, CWD, LIST, MDTM, NLST, PWD, RNFR, XCUP, XCWD, XPWD)
  - ▶ ALL : toutes les opérations

## La directive <Limit LOGIN>

- ▶ Permet de limiter les accès aux serveurs via les commandes Allow from et Deny from
- ▶ Ne fonctionne pas exactement de la même manière que pour Apache avec la directive Order
  - ▶ Si Order n'est pas spécifié, par défaut, c'est Order Allow,Deny qui est appliqué
  - ▶ Avec Order Allow, Deny les directives Allow sont évaluées en premier
    - ▶ Si une directive Allow est vérifiées, l'accès est accordé
    - ▶ Sinon, soit une directive Deny est vérifié et l'accès est refusé sinon il est accordé
- ▶ Possibilité de définir un contexte LOGIN globale mais aussi dans les contextes <Anonymous>, <Directory> et <VirtualHost>

# Les autorisations

- ▶ Allow from *addr\_ip ...* : comme dans la configuration d'Apache, permet l'accès aux adresses IP listées (qui peuvent être des adresses de réseaux)
- ▶ AllowAll : permet tout accès
- ▶ AllowGroup *liste\_de\_groupes* : permet l'accès par groupe selon le contenu de la liste de groupes
- ▶ AllowUser *liste\_utilisateurs* : permet l'accès par utilisateur selon le contenu de la liste d'utilisateurs
- ▶ AllowOverwrite « on » ou « off » : selon que le paramètre soit « on » ou « off », autorise ou interdit l'écrasement des fichiers existants par les utilisateurs
- ▶ PathAllowFilter *expression\_régulière* : rejette tous les « uploads » dont le nom de fichier ne correspond pas à *expression\_régulière* .

# Les interdictions

- ▶ Deny from *addr\_ip ...* : interdit l'accès aux adresses IP listées (qui peuvent être des adresses de réseaux)
- ▶ DenyAll : interdit tout accès
- ▶ DenyGroup *liste\_de\_groupes* : interdit l'accès par groupe selon le contenu de la liste de groupes
- ▶ DenyUser *liste\_utilisateurs* : interdit l'accès par utilisateur selon le contenu de la liste d'utilisateurs
- ▶ PathDenyFilter *expression\_régulière* : rejette tous les « uploads » dont le nom de fichier correspond à *expression\_régulière*

# Les logs

- ▶ Par défaut, ProFTPD ajoute des informations dans un fichier de log à chaque transfert de fichier.
  - ▶ sous Debian, ce fichier est `/var/log/xferlog`
  - ▶ sous Mandrake, plusieurs fichiers se trouvent dans `/var/log/proftpd/`
- ▶ Il est possible d'obtenir des logs plus détaillés en utilisant la directive `ExtendedLog`.

# Les Virtual Hosts

- ▶ ProFTPD permet de configurer plusieurs serveurs virtuels par machine physique
- ▶ La directive `<VirtualHost adresse> .. </VirtualHost>` définit un contexte de serveur virtuel. Les directives qu'elle contient ne s'appliqueront qu'à ce serveur virtuel
- ▶ L'adresse spécifie l'interface auquel le serveur virtuel va être attaché
- ▶ Deux serveurs virtuels peuvent partager la même adresse s'ils utilisent des ports différents (directive `Port numéro_de_port`)
- ▶ Utilité: par exemple pour des providers qui veulent héberger les services ftp de plusieurs clients de manière indépendante sur la même machine

# Les modules

- ▶ ProFTP a une architecture modulaire et peut inclure de nombreuses fonctionnalités supplémentaires dont par exemple:
  - ▶ Accès à un annuaire LDAP
  - ▶ Stockage des informations d'authentification dans une base SQL (mod\_sqlpw)
  - ▶ Rapport upload/download en nombre de fichier ou en octet (mod\_ratio)
  - ▶ ...