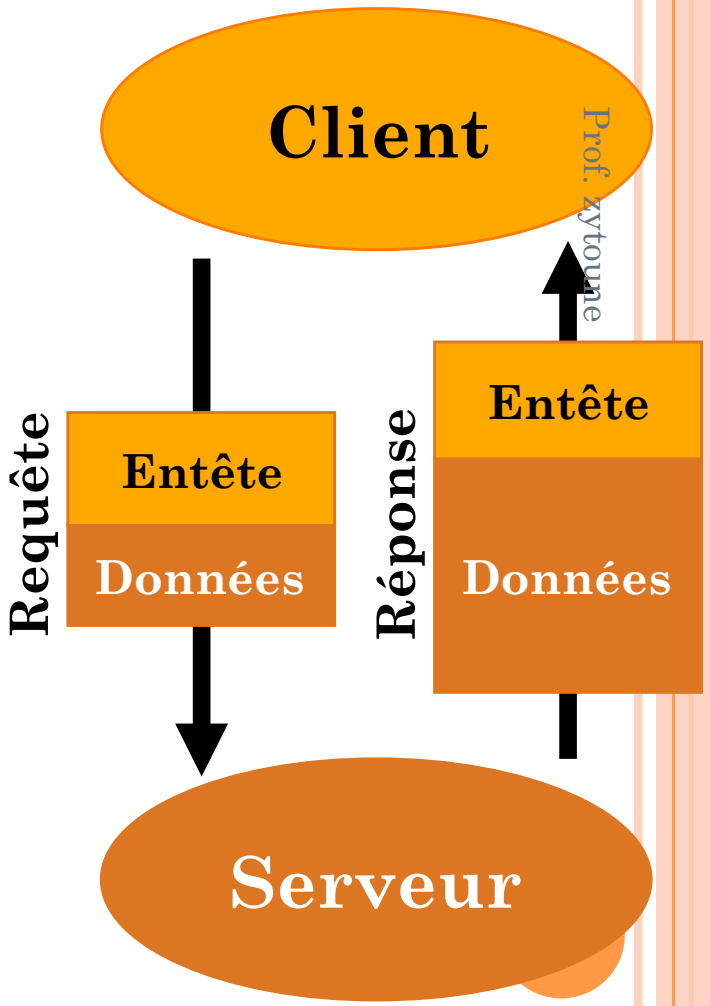
Plan

- Introduction: Présentation HTTP;
- Installation et configuration d'apache;
- VirtualHosts;
- Aliasing;
- Limitation d'accès.

Protocole HTTP – Présentation

- HTTP : *HyperText Transfer Protocol*
- Versions : 0.9, 1.0, 1.1
- Architecture Client-Serveur
- RFC 1945, 2616, 822
- Messages composés de :
  - Entêtes
  - Données
- Messages du type :
  - Requête (demande)
  - Réponse



Protocole HTTP – Requête

- Schéma :
  - Titre de la requête
    - METHODE (`GET`, `HEAD`, `POST`…)
    - RESSOURCE
    - VERSION du protocole
  - Options d'entête supplémentaires
    - Hôte virtuel à interroger
    - Identité du navigateur (`User-Agent`)
    - Types MIME supportés par le navigateur
    - Cookies (`Cookie`)
    - etc…
  - « Ligne blanche »

Exemples de requêtes HTTP

- **GET http://www.yahoo.com HTTP/1.1**
  Host: *www.yahoo.com*
  User-Agent : *Mozilla/5.0 (Linux i686)*
  Accept : *text/html, application/xml, image/jpeg*
- **HEAD http://www.yahoo.com HTTP/1.0**
- **POST /somepage.php HTTP/1.1**
  Host: *example.com*
  Content-Type: *application/x-www-form-urlencoded*
  Content-Length: *19*

  name=Duval&sexe=masculin

Protocole HTTP – Réponse

- Schéma :
  - Titre de la réponse
    - VERSION du protocole
    - CODE d'erreur
    - LIBELLE correspondant au code d'erreur
  - Options d'entête supplémentaires
    - Type MIME du contenu
    - Date/heure du serveur
    - Taille du message
    - Instructions de mise en cache
    - Dépôt de cookies
    - etc…
  - « Ligne blanche »
  - Corps du messages (données à transmettre

Exemple de réponse HTTP

```
HTTP/1.1 302 Found
Date: Sun, 05 Nov 2006 10:47:52 GMT
Content-Length: 30
Content-Type: text/html;
  charset=iso-8859-1

<html><body>test</body></html>
```

Exemple de réponse HTTP

```
HTTP/1.0 200 OK
Date: Thu, 14 May 2009 10:08:21 GMT
Server: Apache
Last-Modified: Fri, 21 Apr 2006 01:04:14 GMT
Accept-Ranges: bytes
Content-Length: 318
Content-Type: image/x-icon
Age: 37
Connection: close
```

```
54a8422f61023d4c4545ef445ac4555b732242a5478754a8422f61023
d4c4545ef445ac4555b732242a5478754a8422f61023d4c4545ef445a
c4555b732242a5478754a8422f61023d4c4545ef445ac4555b732242a
5478754a8422f61023d4c4545ef445ac4555b732242a5478754a8422f
61023d4c4545ef445ac4555b732242a5478754a8422f61023d4c4545e
f445ac4555b732242a5478754a8422f61023d4c4545ef445ac4555b73
2242a5478754a8422f61023d4c4545ef445ac4555b732242a54787
```

Protocole HTTP – Méthodes

- **GET** : obtenir une ressource
- **HEAD** : obtenir des informations sur une ressource
- **POST** : envoi de données par formulaire
  - **PUT** : dépôt d'un fichier sur le serveur
  - **DELETE** : suppression d'un fichier sur le serveur
  - **TRACE** : retourne les données envoyés dans la requête
  - **OPTIONS** : permet d'obtenir des informations sur les options de communication d'une ressource
  - **CONNECT** : permet d'utiliser un proxy

Protocole HTTP – Codes d'erreur

- Familles de codes d'erreur
  - **1xx : Information**
    - `100` – *Continue :* Attente de la suite de la requête
  - **2xx : Succès**
    - `200` – *OK :* Requête traitée avec succès
    - `202` – *Accepted :* Requête traitée mais sans garantie de résultat
  - **3xx : Redirection**
    - `301` – *Moved Permanently :* Document déplacé de façon permanente
    - `302` – *Moved Temporarily :* Document déplacé de façon temporaire
  - **4xx : Erreur imputable au client**
    - `403` – *Forbidden :* Refus de traitement de la requête
    - `404` – *Not Found :* Document non trouvé
  - **5xx : Erreur du serveur**
    - `500` – *Internal Server Error :* Erreur interne du serveur
    - `505` – *HTTP Version not supported :* Version HTTP non gérée par le serveur

Protocole HTTP – Versions

- HTTP 0.9
  - Très simple
  - Pas d'entête
  - Pas de type MIME
  - Une seule méthode : GET
- HTTP 1.0 (mai 1996)
  - Entêtes
  - Gestion des types MIME
  - Multiples méthodes
  - Gestion des hôtes virtuels (`Host:`)
- HTTP 1.1 (juin 1999)
  - Entête `Host:` obligatoire
  - Gestion du cache
  - Connexions persistantes (`Connexion: keep-alive`)
  - Négociation du contenu (types MIME, langue…)

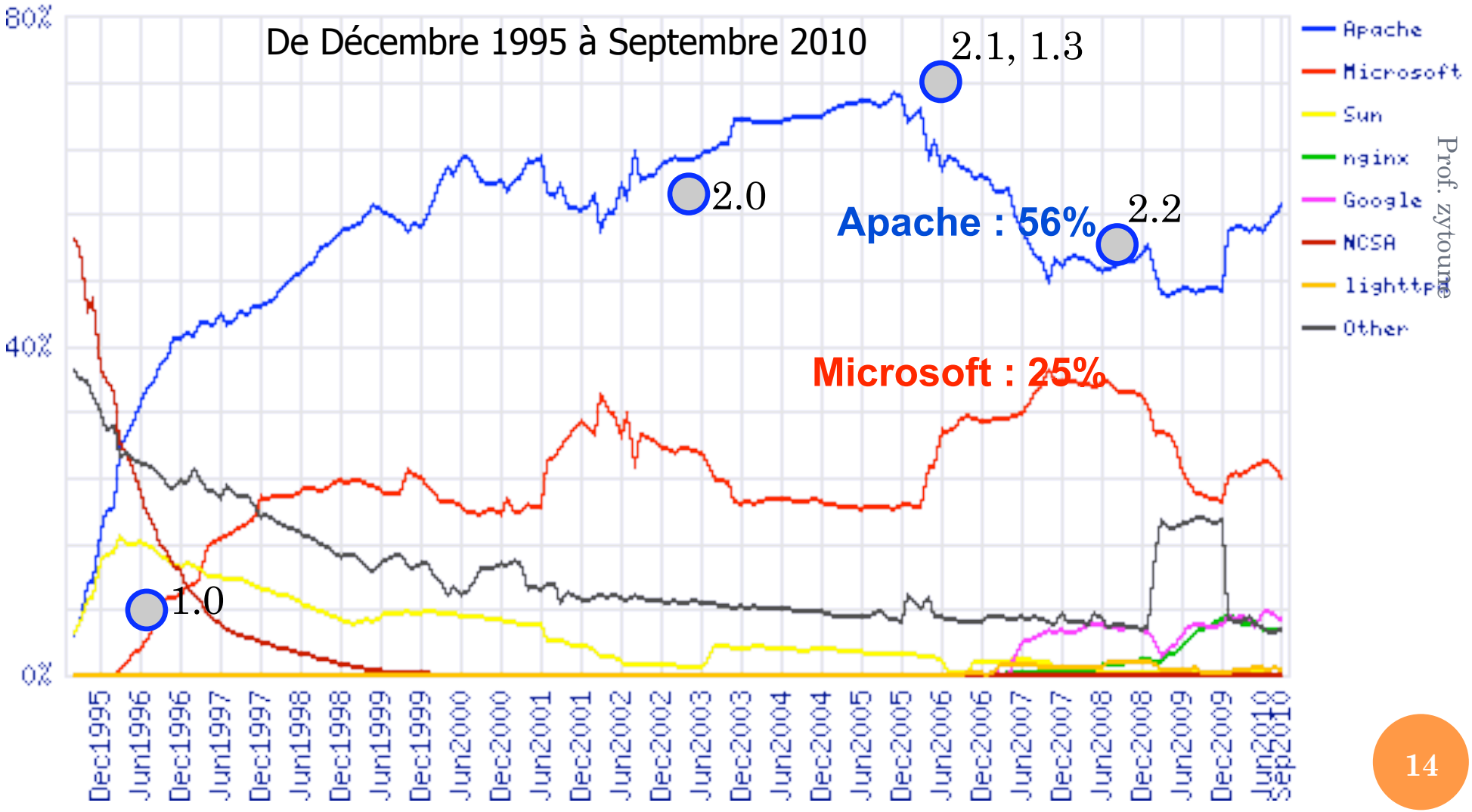
URL – Rappels

- URL : *Uniform Ressource Locator*
- Identifiant unique de toute ressource sur Internet
- RFC 1738
- Composée des éléments :
  - Protocole
  - Machine
  - Nom de domaine
  - TLD (Top Level Domain)
  - Chemin
  - Nom de la ressource
  - Paramètres (liste de couples clé/valeur)
- Exemples :
  - `ftp://noemie.siteweb.com/images/logo.jpg`
  - `http://www.site.fr/appli/script.do?param=valeur`

Généralités – Serveur HTTP

- logiciel servant les requêtes clientes
- conforme au protocoles HTTP 1.1 et 1.0
- journalisation des requêtes pour analyse statistiques
- écoute des ports standards 80 (HTTP) et 443 (HTTPS)
- types de contenus retournés :
  - statique (fichier du système de fichiers)
  - dynamique (généré par un programme ou un script appelé par le serveur HTTP)
- authentification des utilisateurs

Généralité – Les serveurs du marché



Source : http://news.netcraft.com/

Apache

- Dénomination : Apache HTTP Server
- Développeurs : Apache Software Foundation
- Site : http://httpd.apache.org
- Créé en 1995 sur la base du NCSA HTTPd daemon (sur lequel ont été inventés les CGI)
- Dernière version : 2.2.16
- Serveur HTTP open source
- Multi-plateforme (Unix, Windows, NetWare…)
- Très répandu
- Performant

ASF : Apache Software Foundation

- Association à but non lucratif américaine
- Créée en 1999
- Développement de logiciels libres
- Communauté de développeurs décentralisée
- Développement collaboratif basé sur le consensus
- Offre la protection juridique pour :
  - la marque Apache
  - les logiciels développés

Licence Apache

- Licence logicielle gratuite
- Écrite par l'ASF
- S'applique à tous les logiciels qu'elle publie
- Dernière version : 2.0
- Compatible GPL
- Elle permet :
  - Réutilisation du code dans des projets libres ou commerciaux
  - Modification du code
  - Inclusion de code protégé
- Lien : http://www.apache.org/licenses/

INSTALLATION

- Un serveur "LAMP" est un exemple de serveur web.
- Linux: le système d'exploitation constituant la base du système.
- Apache: le serveur HTTP qui gère la communication avec le client.
- MySQL: le système de gestion de base de données.
- PHP: le langage de script utilisé pour générer les pages dynamiques.
- Sous linux Ubuntu:
  - ***apt-get install lamp-server***
  - Permet d'installer Apache, mySQL et PHP;

INSTALLATION

- Installation directe des paquets nécessaires:
  - apache2, mysql-server, php5, php5-mysql
- En ligne de commande :
  - ***sudo apt-get install apache2 mysql-server php5 php5-mysql***
- Description des paquets :
  - Le paquet apache2 installe le serveur Apache 2.
  - Le paquet mysql-server installe le serveur MySQL.
  - Le paquet php5 installe le langage PHP 5 mais aussi, grâce aux dépendances, le module d'intégration dans Apache (paquet libapache2-mod-php5).
  - Le paquet php5-mysql installe les mécanismes de communication entre PHP 5 et MySQL.

CONFIGURATION D'APACHE

Toutes les configurations se font dans le répértoire: /etc/apache2.

La structure de configuration est éclatée sur plusieurs fichiers;

/etc/apache2/apache2.conf	Fichier de configuration de base, ne pas toucher
/etc/apache2/ports.conf	Pour écouter sur des ports autre que 80 (ex: 443 pour HTTPS)
/etc/apache2/envvars	Environnement Apache (ex: ORACLE_HOME, TNS_ADMIN, etc.)
/etc/apache2/conf.d	Configuration globales des applications web
/etc/apache2/mods-available	Modules disponibles (activation et configuration)
/etc/apache2/mods-enabled	Modules activés (activation et configuration)
/etc/apache2/sites-available	Sites disponibles/configurés
/etc/apache2/sites-enabled	Sites activés

Configuration d'apache

- Paramètres généraux se trouvant apache2.conf:
  - User www-data: fixe l'utilisateur qui peut posséder des scripts et des données sensibles;
  - Group www-data: fixe le groupe qui peut posséder des scripts et données sensibles. Si on veut ajouter un utilisateur à posséder des scripts CGI on ajoute l'utilisateur au groupe www-data par adduser;
  - AccessFileName .htaccess: fixe le nom du fichier (par défaut .htaccess) à trouver dans un répertoire pour que l'accès de ce répertoire soit protégé, en imposant à l'utilisateur une authentification par nom et mot de passe. Ces comptes sont spécifiques à Apache et n'interfèrent pas avec les comptes Linux.

Configuration d'apache

- ServerRoot /etc/apache2:
  - Il s'agit du répertoire où le serveur trouvera son répertoire de configuration. On trouve dans /etc/apache2, un lien vers /var/log/httpd/access_log, le fichier-journal des accès aux ressources, réussis ou non (le consulter)
- PidFile /var/run/httpd.pid
  - C'est le fichier où le serveur en exécution stocke son premier numéro de processus (PID), ce qui peut être utile à d'autres processus.
- ErrorLog /var/log/apache2/error.log
  - C'est le fichier qui contient l'historique des erreurs qui se sont produites (exemple : script cgi qui n'a pas marché...).

Configuration d'apache: Paramètres spécifiques à chaque serveur

- Les paramètres (en général) spécifiques à chaque serveur (qui se trouvent dans sites-enabled sont (liste non exhaustive) :
  - DocumentRoot /var/www/html
    - fixe la racine du serveur Web, c'est-à-dire le répertoire de base où sont cherchées par défaut les pages html, lorsque l'URL ne comporte pas de chemin de répertoire
  - DirectoryIndex index.html index.php index.htm…
    - Il est courant d'omettre le nom du fichier de la page d'accueil d'un site ou de l'un de ses sous-répertoires. Pour ne pas retourner systématiquement une erreur 404 signalant une adresse erronée, le serveur possède une liste standard de noms de fichiers qu'il s'efforce de trouver dans le répertoire. Cette liste ordonnée est indiquée par la clause Directory Index
  - ServerAdmin webmaster@localhost
    - S'il a un problème, le serveur écrit un message à cette adresse
  - CustomLog /var/log/apache2/access.log combined
    - Définit le fichier qui contient l'historique des connections, des clients, des dates, de l'origine (referer site) de la connexion, ainsi que le format pour mémoriser ces informations (ici le format combined).

Contrôle des accès à un répertoire

- Chaque répertoire auquel Apache accède peut être configuré, et root peut permettre certaines fonctionnalités d'apache pour ces répertoires, et en interdire d'autres.
- Cela permet, en fonction des besoin et de la confiance accordée à chaque webmaster, de gérer les problèmes de sécurité.
- En général, root cherche à donner tout juste les permissions qui sont requises en fonction des besoins. Le paramétrage d'un répertoire se précise dans un conteneur noté :
  - <Directory /chemin/vers/le/répertoire/> </Directory>

Exemple de conteneur

- NameVirtualHost *
- <VirtualHost *>
- DocumentRoot /home/monRepertoire/ # racine du site
  - <Directory /> # droits du répertoire racine
    - Options FollowSymLinks
    - AllowOverride None
  - </Directory>
  - <Directory /home/monRepertoire/> # droits sur l'ensemble du site
    - Options Indexes FollowSymLinks MultiViews
    - AllowOverride None # interdit les .htaccess
    - Order allow,deny # donne l'ordre des permissions
    - allow from all # autorise tous les clients
    - # avec la directive suivante, il faut mettre la
    - # page d'acueil dans /home/monRepertoire/apache2-default/
    - </Directory>
  - etc...
- </VirtualHost>

Les options

- Les principales options d'un répertoire peuvent être les suivantes :
  - None : Désactive toutes les options.
  - All : Active toutes les options SAUF Multiviews.
  - Indexes : Permet aux utilisateurs d'avoir des indexes généré par le serveur. C'est à dire si l'index du répertoire (index.htm le + souvent) est manquant, cela autorise le serveur a lister le contenu du répertoire (dangereux suivant les fichiers contenu dans ce répertoire).
  - FollowSymLinks : Autorise à suivre les liens symboliques.
  - ExecCGI : Autorise à exécuter des scripts CGI dans ce répertoire.
  - Includes : Autorise des fichiers include coté serveur SSI(Server Side Includes).
  - IncludesNOEXEC : Permet les includes mais empêche la commande EXEC (qui permet d'executer du code).
  - Multiviews : Autorise les vue multiples suivant un contexte. Par exemple permet d'afficher les pages dans un langage suivant la configuration du langage du client.
  - SymLinksIfOwnerMatch : Autorise à suivre les liens seulement si l'user ID du fichier (ou répertoire) sur lequel le lien pointe est le même que celui du lien.

Donner les droits

- Avec Order allow,deny, on peut permettre un accès à tous sauf quelques-uns.
- Par exemple,
  - Order allow,deny
  - allow from all # autorise tous les clients
  - deny from 192.168.0.67 # interdit l'accès par une IP
    - permet à tous d'accéder sauf l'hôte 192.168.0.67.
- Avec Order deny, allow, on peut permettre l'accès seulement par un sous-réseau.
- Par exemple,
  - Order deny,allow
  - Deny from all
  - Allow from 192.168.0
  - Allow from .mydomain.com
    - permet l'accès seulement à partir du réseau local 192.168.0 et du domaine mydomain.com

Directive AllowOverride

- La directive AllowOverride permet au webmaster de redéfinir par lui-même certains droits ou certaines options spécifiquement dans certains répertoires.
- Pour cela, le webmaster crée dans un répertoire un fichier .htaccess dans lequel il définit les options et les droits qu'il souhaite. Par exemple, si root a mis dans les permissions d'un répertoire
  - AllowOverride Options Limit
  - le webmaster peut mettre les droits suivants dans un fichier .htaccess d'un répertoire contenant des fichiers de l'intranet de son entreprise :
  - Options ExecCGI
  - Order deny,allow
  - Deny from all
  - Allow from 192.168.0

Types de directives pour AllowOverride

- None : n'autorise aucun contrôle par le webmaster au niveau du .htaccess. Apache ne lis pas le fichier .htaccess et laisse les permissions "Linux" de ce répertoire et les droits donnés par root dans la balise <Directory> dans la configuration d'apache.
- All : toutes les permissions et options peuvent être gérés dans par le webmaster dans le .htaccess d'un répertoire.
- Limit : Active la directive d'autorisation order, allow, deny dans le .htaccess.
- Options : Active la directive Options dans le .htaccess.
- AuthConfig : permet au webmaster de configurer dans le .htaccess les directives d'autentification pour les sites sécurisés (AuthDBMGroupFile, AuthDBMUserFile, AuthGroupFile AuthName, AuthType, AuthUserFile, Require).
- FileInfo : Active les directives d'autorisations AddEncoding, AddLanguage, AddType, DefaultType, ErrorDocument, LanguagePriority.
- Indexes : permet de définir dans .htaccess des directives comme *DirectoryIndex*.

VIRTUAL HOSTS

- Une machine peut en général avoir plusieurs noms d'hôte.
- One peut en déclarer plusieurs pour la même adresse IP dans les DNS, et une machine peut aussi avoir plusieurs adresses IP si elle a plusieurs interfaces réseaux.
- Si une machine a plusieurs noms d'hôte, on peut alors mettre plusieurs sites HTTP sur le même serveur.
- Pour cela, on crée plusieurs fichiers de configuration différents dans le répertoire /etc/apache2/sites-enabled
- Dans la déclaration du virtual host, on peut mettre :
  - NameVirtualHost *
  - <VirtualHost *>
  - ServerName mon_nom_d_hote ...
  - suivi de la déclaration du répertoire racine du site.

Configuration du serveur

- # Apache doit écouter sur le port 80
- Listen 80
- # Toutes les adresses IP doivent répondre aux requêtes sur les serveurs virtuels
- NameVirtualHost *:80
- <VirtualHost *:80>
  - DocumentRoot /www/example.com
  - ServerName www.example1.com
  - # Autres directives ici
- </VirtualHost>
- <VirtualHost *:80>
  - DocumentRoot /www/example.org
  - ServerName www.example2.org
- # Autres directives ici
- </VirtualHost>

Configuration du serveur

- Les astérisques correspondent à toutes les adresses,
- Comme www.example.com se trouve en premier dans le fichier de configuration, il a la plus grande priorité et peut être vu comme serveur par défaut ou primaire ;
- ce qui signifie que toute requête reçue ne correspondant à aucune des directives ServerName sera servie par ce premier VirtualHost.

CONFIGURATION DU SERVEUR

- On peut remplacer * par l'adresse IP du système. Dans ce cas, l'argument de VirtualHost doit correspondre à l'argument de NameVirtualHost :
  - NameVirtualHost 172.20.30.40
  - <VirtualHost 172.20.30.40>
  - # etc ...
- En général, il est commode d'utiliser * sur les systèmes dont l'adresse IP n'est pas constante - par exemple, pour des serveurs dont l'adresse IP est attribuée dynamiquement par le FAI, et où le DNS est géré au moyen d'un DNS dynamique quelconque.
- Comme * signifie n'importe quelle adresse, cette configuration fonctionne sans devoir être modifiée quand l'adresse IP du système est modifiée.
- Cette configuration est en pratique utilisée dans la plupart des cas pour les serveurs virtuels par nom. En fait, le seul cas où cette configuration ne fonctionne pas est lorsque différents contenus doivent être servis en fonction de l'adresse IP et du port contactés par le client.

Serveurs virtuels par nom sur plus d'une seule adresse IP

- Le serveur a deux adresses IP. Sur l'une (172.20.30.40), le serveur "principal" server.domain.com doit répondre, et sur l'autre (172.20.30.50), deux serveurs virtuels (ou plus) répondront.
- Configuration du serveur
  - **Listen 80**
  - **# Serveur "principal" sur 172.20.30.40**
  - **ServerName server.domain.com**
  - **DocumentRoot /www/mainserver**
  - **# l'autre adresse**
  - **NameVirtualHost 172.20.30.50**
  - **<VirtualHost 172.20.30.50>**
  - **DocumentRoot /www/example.com**
  - **ServerName www.example.com**
  - **# D'autres directives ici ...**
  - **</VirtualHost>**
  - **<VirtualHost 172.20.30.50>**
  - **DocumentRoot /www/example.org**
  - **ServerName www.example.org**
  - **# D'autres directives ici ...**
  - **</VirtualHost>**
- Toute requête arrivant sur une autre adresse que 172.20.30.50 sera servie par le serveur principal. Les requêtes vers 172.20.30.50 avec un nom de serveur inconnu, ou sans en-tête Host:, seront servies par www.example.com.

Ajout d'un site

- Dans /etc/apache2/sites-available, créer un fichier "myself.lan"
- Activer le VirtualHost :
  - sudo a2ensite myself.lan
  - redémarrer Apache,
- Un VirtualHost (site) ajouté peut ensuite être retiré comme suit : commande
  - sudo a2dissite nom_du_site
  - ou en retirant le lien symbolique dans /etc/apache2 / sites-enabled, redémarrer Apache.

Créer un alias

- Un alias permet de stocker les pages d'un utilisateur dans son répertoire "home" et les faire servir par Apache avec une URL simplifiée.
- Exemple : on suppose un utilisateur "myself" et on souhaite faire pointer l'URL "http://localhost/mien/" vers "/home/myself/htdocs/ ».
  - il faut s'assurer que le groupe de sécurité du serveur Apache ("www-data") puisse accéder au répertoire "/home/myself/htdocs", en vérifiant les droits et en autorisant au moins la lecture.
  - Il suffit alors d'ajouter le fichier "mien" suivant dans /etc/apache2/conf.
  - # Users's pages
  - Alias /mien/ "/home/myself/htdocs/"
  - <Directory "/home/myself/htdocs/">
    - Options Indexes MultiViews
    - AllowOverride All
    - Order allow,deny
    - Allow from all
  - </Directory>

Fichiers htaccess

- Les fichiers .htaccess peuvent être utilisés dans n'importe quel répertoire virtuel ou sous-répertoire.
- Les principales raisons d'utilisation des fichiers .htaccess sont :
  - Gérer l'accès à certains fichiers.
  - Ajouter un mime-type.
  - Protéger l'accès à un répertoire par un mot de passe.
  - Protéger l'accès à un fichier par un mot de passe.
  - Définir des pages d'erreurs personnalisées.

Principe des fichiers htaccess

- Le fichier .htaccess est placé dans le répertoire dans lequel il doit agir.
- Il agit ainsi sur les permissions du répertoire qui le contient et de tous ses sous-répertoires.
- Vous pouvez placer un autre fichier .htaccess dans un sous-répertoire d'un répertoire déjà contrôlé par un fichier .htaccess.
- Le fichier .htaccess du répertoire parent reste en « activité » tant que les fonctionnalités n'ont pas été réécrites.

EMPÊCHER L'ACCÈS À DES RESSOURCES

- Un fichier .htaccess est composé de deux sections :
  - Une première section contient les chemins vers les fichiers contenant les définitions de groupes et d'utilisateurs :
    - AuthUserFile /repertoire/de/votre/fichier/.FichierDeMotDePasse
    - AuthGroupFile /repertoire/de/votre/fichier/.FichierDeGroupe
    - AuthName "Accès protégé »
    - AuthType Basic
  - **AuthUserFile** définit le chemin d'accès absolu vers le fichier de mot de passe.
  - **AuthGroupFile** définit le chemin d'accès absolu vers le fichier de groupe.
  - **AuthName** entraîne l'affichage dans le navigateur Internet de : « Tapez votre nom d'utilisateur et votre mot de passe. Domaine: "Accès protégé »
  - **AuthType** Basic précise qu'il faut utiliser AuthUserFile pour l'authentification.

Empêcher l'accès à des ressources

- Une seconde section contient la définition des conditions d'accès :
  - Require valid-user
  - {instruction d'accès à satisfaire }
- require valid-user précise que l'on autorise uniquement les personnes identifiées.
- Il est également possible de préciser explicitement le nom des personnes autorisées à s'identifier :
  - require user {username}

Protéger un répertoire par un mot de passe

- Il s'agit d'une des applications les plus utiles du fichier .htaccess car elle permet de définir de façon sûre (à l'aide d'un login et d'un mot de passe) les droits d'accès à des fichiers par certains utilisateurs.
- La syntaxe est la suivante :
  - AuthUserFile {emplacement du fichier de mot de passe}
  - AuthGroupFile {emplacement du fichier de groupe}
  - AuthName "Accès protégé"
  - AuthType Basic
  - Require valid-user

Protéger un répertoire par un mot de passe

- AuthUserFile: permet de définir l'emplacement du fichier contenant les logins et les mots de passe des utilisateurs autorisés à accéder à une ressource donnée.
- AuthGroupFile : permet de définir l'emplacement du fichier contenant les groupes d'utilisateurs autorisés à s'identifier. Il est possible d'outrepasser cette déclaration en déclarant le fichier suivant : /dev/null.
- Voici un exemple de fichier .htaccess :
  - ErrorDocument 403 http://www.commentcamarche.net/accesrefuse.php3
  - AuthUserFile /repertoire/de/votre/fichier/.FichierDeMotDePasse
  - AuthGroupFile /dev/null
  - AuthName "Accès sécurisé au site CCM"
  - AuthType Basic
  - Require valid-user

Création des mots de passe

- Apache fournit un outil permettant de générer facilement des mots de passe cryptés (aussi bien sous Windows que sous Unix), il s'agit de l'utilitaire htpasswd accessible dans le sous-répertoire bin d'Apache.
- La syntaxe de cet utilitaire est la suivante :
  - Pour créer un nouveau fichier de mots de passe :
    - *htpasswd -c {chemin du fichier de mot de passe} utilisateur*
  - Pour ajouter un nouvel utilisateur/mot de passe à un fichier existant :
    - *htpasswd {chemin du fichier de mot de passe} utilisateur*

TP

- On veut créer et héberger deux sites web sur un serveur apache: www.exemple1.com et www1.exemple1.com.
- Le contenu des deux sites va être limité à une page d'accueil index.html élémentaire.
- Créer et héberger ces deux sites.
- Vérifier le fonctionnement.