Apache

Administrer un serveur Web

Version 20.8

Auteur : F. Micaux

formation@actilis.net

Table des matières

1- Du navigateur au Serveur d'application....4
1.1- De la question DNS au serveur d'application....5
1.2- Rôle des composants en amont des serveurs web....14
1.3- Le protocole HTTP....16

2- Le DNS....19
2.1- Configuration de la résolution de noms....20
2.2- Organisation du DNS....22
2.3- Mécanisme de résolution des noms : récursivité....24
2.4- Resource Records....28
2.5- Les zones....35
2.6- Zone de type master....37
2.7- Zone de type slave....40
2.8- Les enregistrements....43

3- Apache HTTP Server : Un peu d'histoire....52
3.1- Introduction à Apache HTTP server....53
3.2- Evolution des versions....54
3.3- Premier lancement du service....58

4- Apache HTTP Server : principe de configuration....60
4.1- Les bases de la configuration d'Apache....61
4.2- La documentation et les modules....66
4.3- Configuration réseau....70
4.4- De l'URL au contenu à servir....74
4.5- Contextes de répertoire....75
4.6- Le répertoire pointé par DocumentRoot....77
4.7- Le module autoindex....79
4.8- Les alias et les redirections....81
4.9- Pages de gestion des erreurs....84
4.10- Options dans les contextes de répertoire....85
4.11- Délégation de pouvoir d'administration....87
4.12- Les logs du serveur apache....90
4.13- Gestion des connexions réseau....98

5- MPM et gestion de la charge....100
5.1- MPM : Modèles de gestion de la charge....101
5.2- Contrôler la charge du serveur....105
5.3- Outils de Benchmark : Ab....107

6- Apache HTTP Server : les bases de la sécurité....110
6.1- Le contrôle d'accès historique....111
6.2- Le contrôle d'accès avec mod_authz_host....115
6.3- Compléments sur Require....116
6.4- L'authentification des utilisateurs....117

7- Apache HTTP Server : les serveurs virtuels....123
7.1- Notion de serveur virtuel....124
7.2- Déclaration d'un hôte virtuel....125
7.3- HTTPS et le module mod_ssl....133
7.4- Certificat....134
7.5- Mise en œuvre d'un hôte virtuel HTTPS....136
7.6- Compléments sur les certificats....138

8- Apache HTTP Server : Contenus dynamiques....142
8.1- CGI....143

9- PHP-FPM....149
9.1- HTTPD & PHP : plusieurs approches....150
9.2- Configuration de PHP-FPM....153
9.3- Définition de pools PHP-FPM....155
9.4- Architecture distribuée avec PHP-FPM....158
9.5- Introduction à TOMCAT....159
9.6- Connecter Apache et TOMCAT : le connecteur AJP....163
9.7- Mise en œuvre avec mod_proxy_ajp....164

Notes personnelles

9.8- Mise en œuvre avec mod_jk....165

10- Annexe : Configuration réseau sous RedHat Linux....169
10.1- Noms des interfaces réseau....170
10.2- Route et table de routage....174
10.3- Configurer le réseau....177
10.4- Fichiers de configuration des interfaces réseau....181
10.5- Analyse et diagnostic réseau : ping....182

11- Annexes concernant Apache....185
11.1- Compilation du serveur Apache....186

12- Annexe : services liés au mail....194
12.1- Acteurs de la messagerie....195
12.2- Les protocoles de la messagerie : SMTP....197
12.3- Les protocoles de messagerie : POP3....199
12.4- Les protocoles de messagerie : IMAP4....201
12.5- Le Serveur SMTP Postfix....203
12.6- Postfix et les tables....204
12.7- Commandes d'administration de Postfix....205
12.8- Postfix et les logs....206
12.9- Configuration de base de Postfix....207
12.10- Démarrage et maintenance de Postfix....211
12.11- Dovecot : un serveur POP / IMAP....213

13- Annexe : Serveur FTP....215
13.1- Le Serveur VSFTPD....216

Notes personnelles

1- Du navigateur au Serveur d'application

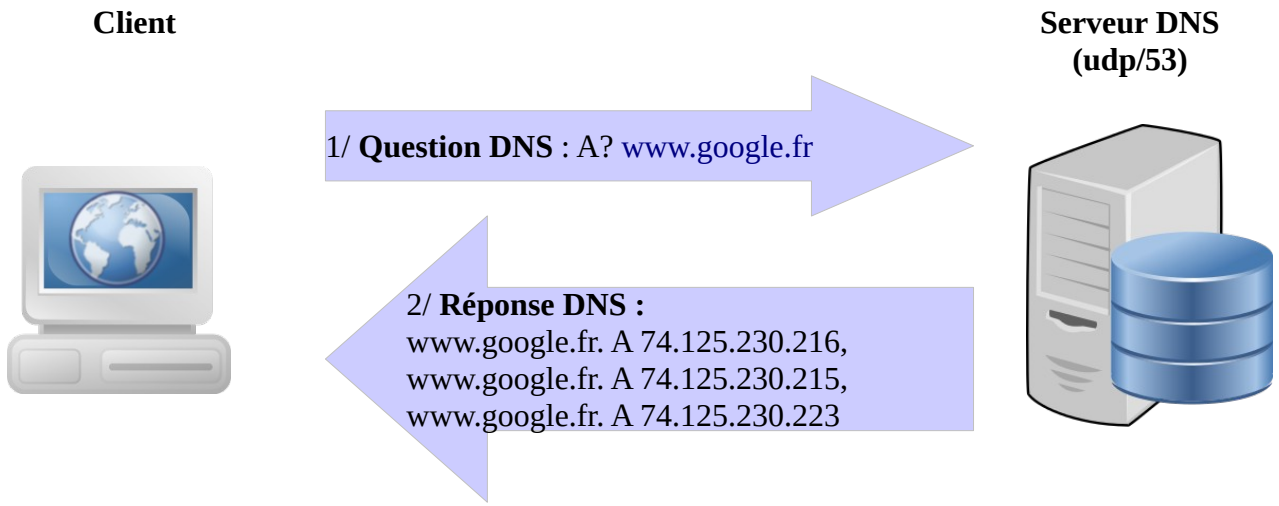
Notes personnelles

1.1- De la question DNS au serveur d'application

1.1.1- Requête DNS

La première requête réalisée par un navigateur web est une **requête DNS**.

Elle est adressée à un des serveurs DNS configurés sur le poste client (/etc/resolv.conf)



Le client ouvre une connexion vers **la première adresse IP** retournée par le serveur DNS.

Notes personnelles

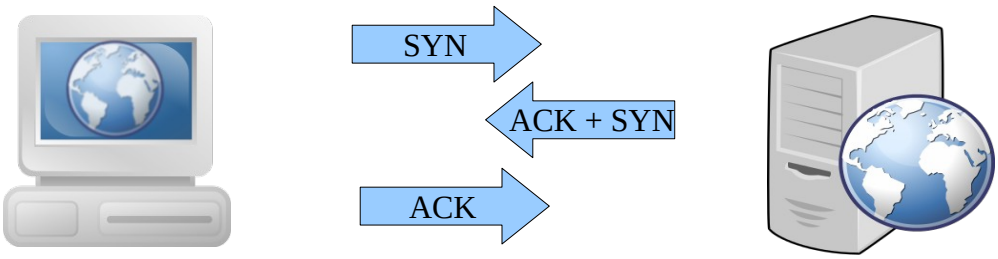
1.1.2- Connexion TCP

Elle se passe en trois temps : Handshake TCP, échanges de données, et Déconnexion.

1.1.2.1- Le Handshake

Établir une session à durée indéterminée, et synchroniser les numéros de séquence.

Durant cette phase, les communicants synchronisent leurs numéros de séquence.



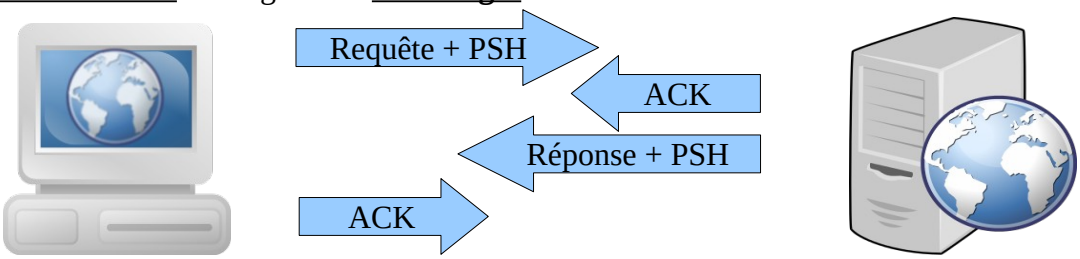
Après cette phase, la session est dite établie.

Le dialogue qui suit est une série d'échanges.

Notes personnelles

1.1.2.2- La session TCP

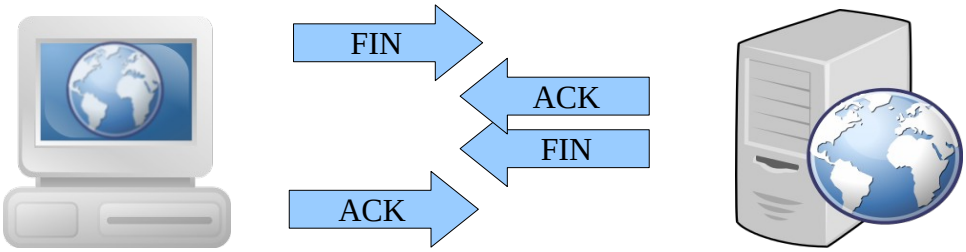
La session est établie : il s'agit alors **d'échanges** entre les 2 connectés.



Les échanges doivent alors respecter un vocabulaire, celui du protocole utilisé.

1.1.2.3- La déconnexion TCP

La déconnexion peut être à l'initiative du client ou du serveur, peu importe.



La connexion est rompue de part et d'autre : côté serveur, cela libère un worker (un slot).

Notes personnelles

1.1.3- Session TCP et connexion persistante

HTTP prévoit au départ que la **connexion TCP soit fermée** après chaque requête HTTP.

Avec HTTP/1.1, le client **peut** demander qu'elle **soit conservée** pour émettre une nouvelle requête.
(Sous réserve que le serveur l'accepte)

Cela se fait grâce à l'en-tête **Connection: keep-alive**
Plusieurs requêtes peuvent alors être transmises dans la même connexion TCP,

Réglages : Le serveur doit **l'autoriser**, et régler certains détails :

timeout entre deux requêtes (au delà duquel le serveur déconnecte le client)

et **nombre de requêtes maximum** (au delà duquel le serveur déconnecte le client)

Directives de configuration à ce sujet :

Apache HTTPD : Keepalive, KeepAliveTimeOut, MaxKeepAliveRequests

Nginx : keepalive_disable, keepalive_timeout, keepalive_requests

Tomcat et dérivés : restrictedUserAgents, maxKeepAliveRequests

Notes personnelles

1.1.4- Requête HTTP

Requête = Une **méthode** HTTP, une **URI**, et une **version** du protocole HTTP.
+ **des en-têtes** … et se termine par une ligne vide

Exemple de requête : http://172.17.1.102/ (émise par **Firefox**)

```
GET / HTTP/1.1
Host: 172.17.1.102
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:16.0) Gecko/20100101 Firefox/16.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fr,fr-fr;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
```

La même, émise par "**wget**" en direct et via un proxy.

En direct :

```
GET / HTTP/1.0
User-Agent: Wget/1.12 (linux-gnu)
Accept: */*
Host: 172.17.1.102
Connection: Keep-Alive
```

Via un serveur proxy-cache (côté client) :

```
GET / HTTP/1.1
User-Agent: Wget/1.12 (linux-gnu)
Accept: */*
Host: 172.17.1.102
Via: 1.0 proxy.formation (squid/3.1.10)
X-Forwarded-For: 172.17.1.101
Cache-Control: max-age=259200
Connection: keep-alive
```

En jaune, les en-têtes ajoutées par le proxy-cache

Notes personnelles

1.1.5- Réponse HTTP

Soit la requête ci-dessous :

```
$ wget --quiet -S --no-http-keep-alive -O /tmp/contenu.txt http://172.17.1.102/
```

La réponse : un code retour, des en-têtes, terminés par une ligne vide, puis le contenu demandé.

```
HTTP/1.1 200 OK
Server: nc -l
Content-Length: 10
Connection: close
Content-Type: text/plain

HELLO, WORLD
```

Les en-têtes reçues : code retour et en-têtes (le contenu a été enregistré dans le fichier /tmp/contenu.txt)

```
  HTTP/1.1 200 OK
  Server: nc -l
  Content-Length: 10
  Connection: close
  Content-Type: text/plain
$
```

Ici, **le contenu renvoyé est plus long que les 10 octets annoncés**[1] dans Content-Length.

```
$ cat /tmp/contenu.txt
HELLO, WOR
```

1 Wget coupe la connexion et s'arrête à 10 octets reçus, d'autres navigateurs peuvent avoir un comportement différent

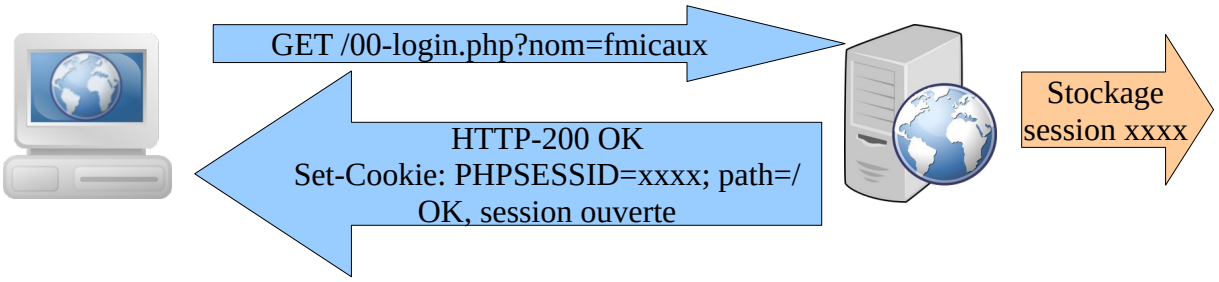
Notes personnelles

1.1.6- Notion de session

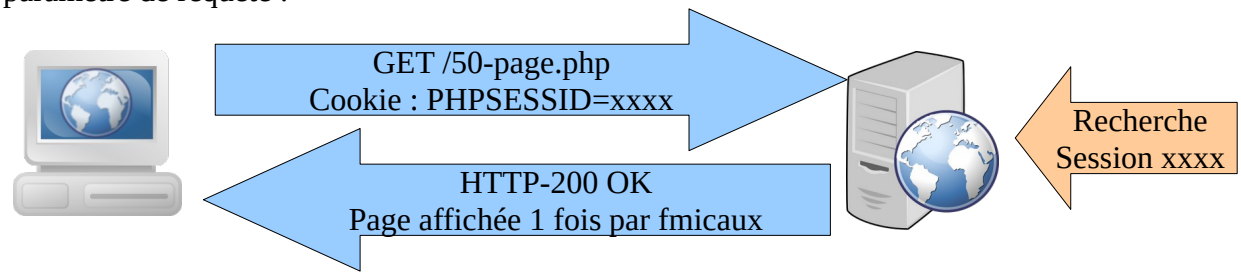
Côté serveur, un **contexte applicatif** est souvent maintenu.

L'ouverture d'une session engendre la création d'un **identifiant de session**.

L'identifiant de **session** est **transmis par le serveur** au client par un en-tête (Set-Cookie)



Lors des requêtes suivantes, l'identifiant de session est **cité par le client** au serveur via un en-tête ou paramètre de requête :



Notes personnelles

Exemple :

Document : **00-login.php**

```php
<?php
    session_start();

    echo "OK:Session ouverte\n";
    $_SESSION['num']=1;
    $_SESSION['nom']=$_GET["nom"];

?>
```

La fonction **session_start()** (PHP)

Crée une session si elle n'existait pas déjà.

Recharge les variables de session si une session existe., on accède aux variables par **$_SESSION['nom-variable']**

Document : **50-page.php**

```php
<?php
    session_start();

    if ((!isset($_SESSION['num'])) || (empty($_SESSION['num']))) {
        echo "KO:Pas connecté.\n";
    } else {
        echo "OK:Page appelée " .  $_SESSION['num']++ . " fois par " . $_SESSION['nom']. "\n";
    }
?>
```

Document : **99-logout.php**

La fonction **session_destroy** (PHP) détruit la session.

Il faut y penser à la déconnexion, car chaque session est stockée dans un fichier du serveur.

```php
<?php
    session_start();
    echo "Bye Bye " .$_SESSION['nom']. "\n";

    session_destroy();
    echo "OK:Session fermée\n";
?>
```


Notes personnelles

Solliciter des pages...

```
$ requete () { wget --quiet -O - --save-cookies cookies.txt --load-cookies cookies.txt --keep-
session-cookies http://192.168.156.100/$1  ; }
```

```
$ requete 00-login.php?nom=fmicaux
OK:Session ouverte
$ requete 50-page.php
OK:Page appelée 1 fois par fmicaux
$ requete 50-page.php
OK:Page appelée 2 fois par fmicaux
$ requete 99-logout.php
Bye bye fmicaux
OK:Session fermée


$ cat cookies.txt
# HTTP cookie file.
# Generated by Wget on 2012-12-10 18:34:18.
# Edit at your own risk.

192.168.156.100 FALSE   /       FALSE   0       PHPSESSID       aek6mklse4hhu4rqm8tdu95a27


$ requete 00-login.php?nom=fmicaux
OK:Session ouverte
$ requete 50-page.php
OK:Page appelée 1 fois par fmicaux
$ rm cookies.txt
$ requete 50-page.php
KO:Pas connecté.
```

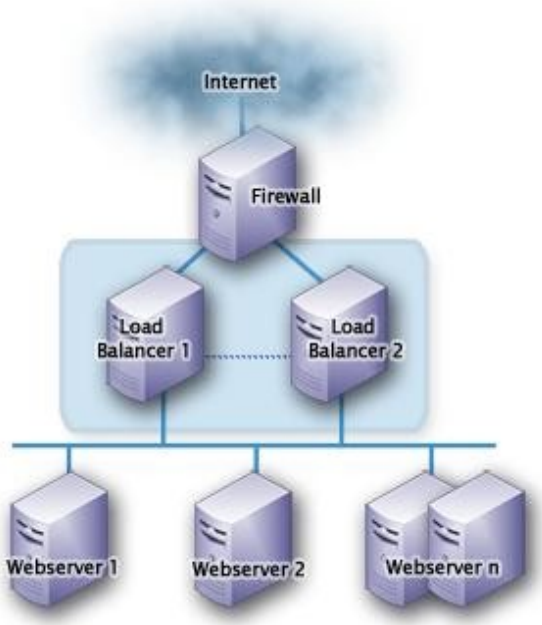

Notes personnelles

1.2- Rôle des composants en amont des serveurs web

1.2.1- Load Balancing et Haute disponibilité

Load Balancing : répartition de charge.

Objectif : offrir les meilleures performances possibles.



Le load balancer oriente vers un pool de serveurs web, dont il connait la charge de chacun des nœuds.

Souvent, pour éviter le Single Point Of Failure, on double le Load Balancer.

Il répartit les clients (les requêtes) sur N serveurs d'application (serveurs Réels), et peut (doit) mettre en œuvre un système de détection de pannes des serveurs réels (tests de vie).

Si un nœud est HS, le répartiteur le sort du pool... et si possible émet une alerte.

Logiciels : LVS, HAProxy, NGINX, ...

Notes personnelles

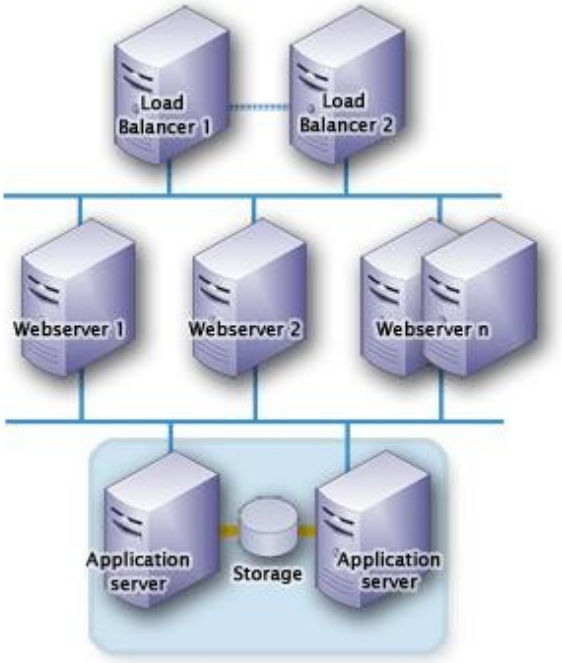
1.2.2- Applications critiques : haute disponibilité totale

Le Load Balancer est doublé et les clients contactent une **VIP**.

VIP : Adresse IP virtuelle mise en œuvre par un système comme **KeepAlived**.

Un dispositif veille à la santé de chaque Load Balancer.

Si un LB est HS, c'est l'autre qui prend la **VIP**.



Comme dans une solution de load-balancing, les requêtes sont distribuées sur N serveurs du pool, et il y a détection des nœuds défectueux (tests de vie) et retrait du pool en cas défaillance.

Les nœuds accèdent à un pool de serveurs d'application (dispositif type Reverse Proxy, HAProxy, NGINX).

Un Cluster de stockage propose une haute disponibilité au niveau stockage : redondance des chemins d'accès aux disques (**Multipathing**, ...).

La haute disponibilité doit inclure stockage physique... assurant la redondance des données (SAN, DRBD, etc...)

Notes personnelles

1.3- Le protocole HTTP

1.3.1- Principe du protocole HTTP

Le protocole HTTP est défini par un ensemble de RFC, et prévoit un dialogue **à l'initiative du client** :
Il se connecte au serveur et émet une requête à laquelle il recevra une réponse.

Une requête HTTP émise par le client est une suite d'octets / de lignes :

appel à une **méthode HTTP** et ses arguments :
l'URL demandée + la version du protocole (HTTP/1.0, HTTP/1.1...)

accompagnée d'**en-têtes HTTP** :
Hôte demandé : **Host:**
Informations sur le navigateur : **User-Agent:**
Souhaits concernant le contenu attendu :
Accept-language: ,
Accept-content:,
Accept-encoding:,
Accept-charset:

Volonté de conserver la connexion (HTTP/1.1 seulement) : **KeepAlive:**

une requête est terminée par une ligne vide.

Notes personnelles

Une réponse HTTP retournée par le serveur est une suite d'octets formant des lignes :

Un état / code retour selon la forme PROTOCOLE CODE-RETOUR STATUT
exemple : **HTTP/1.0 200 OK**

Une série d'**en-têtes HTTP** :
Date:
Server:

Des informations sur le contenu qui suit après la dernière en-tête:
Last-Modified:
Etag:
Accept-Ranges:
Content-Length:
Content-Type:

Informations sur la possibilité de conserver la connexion TCP (HTTP/1.1 seulement) :
Connection: Keep-alive
KeepAlive: timeout=XX, max=YYY

une ligne vide signalant la fin des en-têtes

le contenu demandé si il est disponible.

Notes personnelles

1.3.2- Méthodes HTTP

Les méthodes HTTP (**RFC 2616**) implémentées par Apache HTTP Server 2.2 sont les suivantes :

HEAD,
GET,
PUT,
POST,
DELETE,
CONNECT,
OPTIONS,
TRACE

Le serveur implémente aussi une méthode **PATCH**, ne faisant l'objet d'aucune RFC.

D'autres méthodes, concernant le protocole WebDav (**RFC 2518**), extension du protocole HTTP : **PROPFIND**, **PROPPATCH**, **MKCOL**, **COPY**, **MOVE**, **LOCK**, **UNLOCK**.

D'autres méthodes, concernant le protocole WebDav Versionning (**RFC 3253**) :
VERSION_CONTROL, **CHECKOUT**, **UNCHECKOUT**, **CHECKIN**, **UPDATE**, **LABEL**, **REPORT**, **MKWORKSPACE**, **MKACTIVITY**, **BASELINE_CONTROL**, **MERGE**, **INVALID**

Notes personnelles

2- Le DNS

Notes personnelles

2.1- Configuration de la résolution de noms

2.1.1- Bibliothèque nss et fichier /etc/nsswitch.conf

Aujourd'hui, sur presque tous les Unix, on utilise une bibliothèque (libnss) pour toutes les résolutions de noms (machines ou pas).

Les fonctions *gethostbyname()* et similaires **choisissent les méthodes**, en fonction de ce qu'on indique dans le fichier **/etc/nsswitch.conf**.

Extrait de fichier /etc/nsswitch.conf

```
passwd:     files nisplus
shadow:     files nisplus
group:      files nisplus
#hosts:     db files nisplus nis dns
hosts:      files dns
```

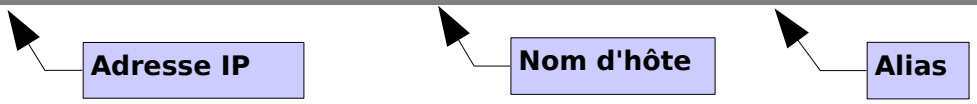
Quel que soit le service, la méthode **files** se rapporte toujours au fichier du nom du service dans **/etc**.
Pour le service « **hosts** », c'est le fichier **/etc/hosts**.
La méthode « **dns** » fait appel au résolveur DNS, et donc au fichier **/etc/resolv.conf**.

Notes personnelles

2.1.2- Le fichier /etc/hosts

Le fichier **/etc/hosts** associe des adresses à des noms de machines

```
# File : /etc/hosts  -- readable pour tout le monde.
127.0.0.1               localhost.localdomain localhost
```



2.1.3- Le fichier /etc/resolv.conf

Le « **client DNS** », qui sert à faire un appel à un serveur DNS est configuré par **/etc/resolv.conf**
On peut en indiquer jusqu'à 3 par le mot clé **nameserver**
On peut définir un "nom de domaine par défaut". (grâce à "search" ou "domain")
Des options peuvent être présentées par le mot "options"

```
search actilis.net
nameserver 192.168.0.200
options attempts:2 timeout:3
```

Astuce : Penser à "**PEERDNS=no**" dans les fichiers /etc/sysconfig/network-scripts/ifcfg-*
Sinon, le démarrage d'une interface peut modifier le contenu de /etc/resolv.conf.

Notes personnelles

2.2- Organisation du DNS

2.2.1- Hiérarchie et zones

Le DNS est un système **organisé de manière hiérarchique**.
C'est un arbre dont la **racine** est notée "**.**".
Les différentes branches de cet arbre sont reliées par des **nœuds** eux aussi notés "**.**".

Son organisation s'apparente à celle d'un système d'une arborescence de répertoires / fichiers.
www.actilis.net. => / net / actilis / www

FQDN (*Fully Qualified Domain Name*) :

On devrait toujours indiquer le "**.**" final pour les noms pleinement qualifiés (FQDN).
Ce "**.**" final est optionnel en saisie, il est systématiquement ajouté par nos outils clients.

TLD : en premier niveau après la racine, se trouvent les **TLD** (*Top Level Domains*);

Les **GTLD** (*Generic Top Level Domain*) :
Les plus connus sont .**COM**, .**ORG**, .**NET**, (environ 20 domaines génériques)

Les **CCTLD** (*Country Code Top Level Domain*) :
Les domaines nationaux : .**FR**, **.DZ** ...(environ 260 domaines nationaux)
Les domaines des DOM/TOM **.PF**, **.MQ**, **.GP**, **.RE**,
Les domaines régionaux : **.BZH**...

Notes personnelles

2.2.2- Délégation et déclaration d'une zone

Chaque zone (nom de domaine) est une délégation faite par un supérieur hiérarchique.
En France on trouve des prix autour de 10€/an pour un .fr, .com, .net, .org...

En principe, on respecte les étapes suivantes :

(a) Décrire la zone : présenter une section "zone", lui donner un nom, lui affecter des propriétés
la première propriété est son type , qui engendre à son tour d'autres propriétés
en fonction du type, on présente un fichier de données par la directive "file",

(b) Fournir la base de données : ce sont les enregistrements, les réponses aux questions
alimenter la base de données en la peuplant de Resource Records.

(c) Enregistrer la zone auprès d'un registrar : pour les domaines "Internet".
Ce sont des sociétés qui payent une cotisation pour être autorisées à vendre des noms,
et à les déclarer au niveau des TLD, les ajouter dans le "whois", en quelque sorte...

On déclare le nom de domaine, s'il est disponible, et on le renouvelle annuellement
On doit fournir des informations au "registrar" (identité, Siret, adresses, ... n° de CB..)

(**d**) **Le registrar déclare** le nom de zone auprès des GTLD (l'Afnic pour un .fr, par exemple)
On doit préciser au registrar les adresses des serveurs de noms gérant la zone,
donc les adresses des serveurs "NS" paramétrés en (a) et enregistrés en (b).

Notes personnelles

2.3- Mécanisme de résolution des noms : récursivité

Pour toute question DNS (exemple : "A www.actilis.net. ?"), soit le serveur DNS interrogé connaît la réponse, soit il doit la chercher... parfois **récursivement** dans toute la hiérarchie **depuis la racine**.

2.3.1- Les root-servers

Pour trouver l'adresse IP de www.actilis.net, le DNS doit trouver un **root-server** .

C'est un serveur qui peut répondre aux requêtes du domaine "**.**", la racine.

Un root-server ne sait pas tout, il peut juste orienter vers le serveur qui connaît la **suite de la réponse**.

Notes personnelles

Il y a 13[2] installations au monde, opérées par des entités indépendantes :

```
$ host -t NS .
. name server D.ROOT-SERVERS.NET.
. name server E.ROOT-SERVERS.NET.
. name server F.ROOT-SERVERS.NET.
. name server G.ROOT-SERVERS.NET.
. name server H.ROOT-SERVERS.NET.
. name server I.ROOT-SERVERS.NET.
. name server J.ROOT-SERVERS.NET.
. name server K.ROOT-SERVERS.NET.
. name server L.ROOT-SERVERS.NET.
. name server M.ROOT-SERVERS.NET.
. name server A.ROOT-SERVERS.NET.
. name server B.ROOT-SERVERS.NET.
. name server C.ROOT-SERVERS.NET.
```

La question est :
qui est NS sur “.” ?

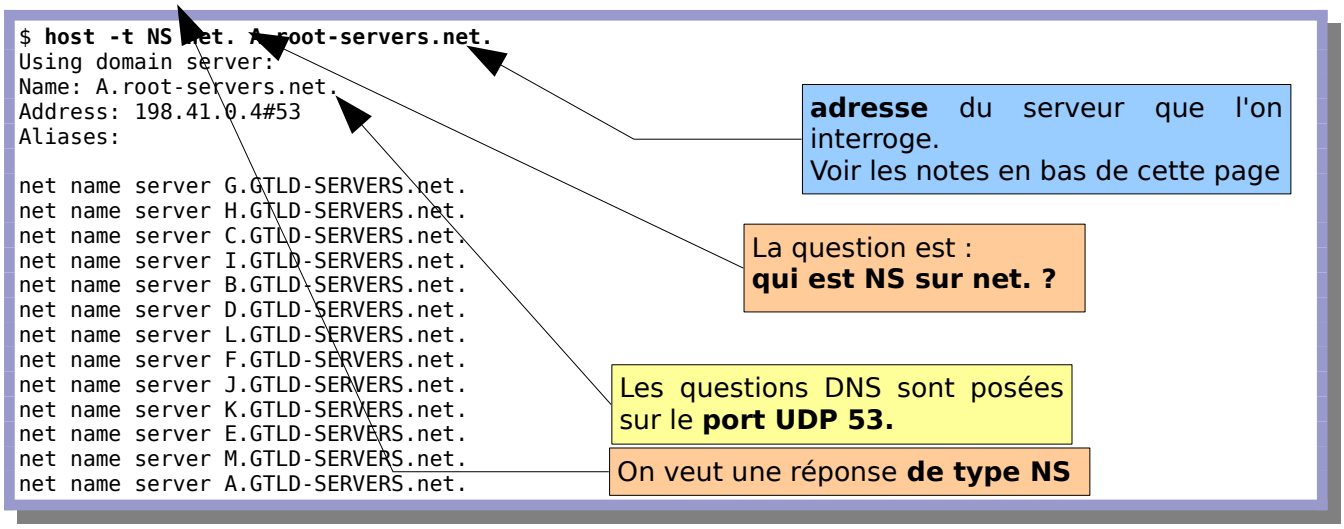
Il y a plusieurs réponses possibles

2 Il y a en fait un peu plus de 180 serveurs, mais 13 entités techniques. **A visiter** : http://root-servers.org/

Notes personnelles

2.3.2- Les GTLD-servers

Pour trouver un NS pour le domaine **net.** On demande (en théorie[3]) à un serveur gérant la zone « **.** ».



3 en théorie, car il est aujourd'hui difficile pour l'internaute lambda de contacter directement les serveurs racines... Seuls les DNS des FAI peuvent le faire, on s'appuie donc sur eux car on ne peut plus (2003) contacter directement [a-m].root-servers.net, suite à une attaque massive (Distributed DOS) contre les root-servers (le 22 octobre 2002) qui avait provoqué une "panne d'Internet".

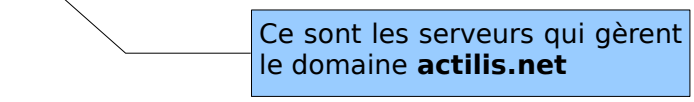
Notes personnelles

2.3.3- Les délégations de noms de domaines

Il suffit ensuite de contacter un des serveurs gérant la zone **net** pour savoir où trouver de l'information sur **actilis.net.** et ainsi de suite (on demande à un gestionnaire du TLD[4] ".net" s'il connaît "actilis.net.").

Fin de la résolution de nom :

```
~$ host -t NS actilis.net. A.GTLD-SERVERS.net
Using domain server:
Name: A.GTLD-SERVERS.net
Address: 192.5.6.30#53
Aliases:

actilis.net name server dns1.actilis.net.
actilis.net name server dns2.actilis.net.
```



4 Certains GTLD servers sont encore accessibles directement, ce n'est pas (plus) le cas pour ceux de ".net" et ".fr".

Notes personnelles

2.4- Resource Records

Les enregistrement sont appelés des **Resource Records** (RR).

Ce sont les composants d'une zone,
Ils définissent les réponses aux questions qu'on nous posera.

Il y a plusieurs types d'enregistrements, pas forcément limités à une association "nom - adresse IP".

2.4.1- Enregistrements A & AAAA

Une requête de type **A** (*Address*) (ou **AAAA**) établit une correspondance entre

un Nom (la question) et Adresse IP (la réponse).

Une réponse **A** est une adresse **IPv4** :

```
~$ host -t A www.actilis.net
www.actilis.net has address 88.190.216.21
```

Une réponse **AAAA** est une adresse **IPv6** :

```
$ host -t AAAA www.free.fr
www.free.fr has IPv6 address 2a01:e0c:1::1
```


Notes personnelles

2.4.2- Enregistrement SOA : "Start Of Authority"

L'enregistrement **SOA** existe une seule fois par domaine, et souvent cité le premier.

```
... SOA MNAME MAILADDR (SERIAL REFR RETR EXP NEGTTL)
```

Il donne des informations sur un domaine :

le nom du serveur principal pour un domaine (celui à contacter par les Slaves) : **MNAME**
l'adresse mail de contact pour les problèmes concernant le domaine : **MAILADDR**
le numéro de version de la base des données de la zone : **SERIAL**
les paramètres de rafraîchissement des slaves : **REFRESH**, **RETRY**, **EXPIRE**
le délai de mémorisation des questions sans réponse pour les serveurs cache : **NEGTTL**

Paramètres d'un enregistrement SOA :
MNAME : le nom du serveur autoritaire pour le domaine
MAILADDR : adresse mail à contacter en cas de problème, le "@" étant remplacé par un "."
SERIAL : Numéro de série que l'on l'incrémente à chaque mise à jour.

Codé sur 10 digits, il peut exprimer une date + un numéro de modification dans la journée
Ce n'est absolument pas une obligation, 1, 2, 3, ... sont aussi de bons numéros de série...

REFRESH : Temps au bout duquel un secondaire doit se relancer une tentative de transfert.
RETRY : Temps de ré-essai si échec d'un transfert précédent.
EXPIRE : Délai d'abandon des tentatives de transfert par les secondaires.
NEGTTL : Durée de vie d'une réponse de type NXDOMAIN ("je ne connais pas") transmise à un cache. Maximum 3h. Ce champs est aussi appelé **MINTTL**.

Notes personnelles

2.4.3- Enregistrement NS

Cet enregistrement **NS** (*Name Server*) déclare un serveur DNS pour un domaine donné,
Si une réponse est fournie par un de ces serveurs, elle est dite "autoritative".
Par opposition à la même réponse qui serait fournie par un cache intermédiaire.

Il peut (doit) y en avoir plusieurs.

```
$ host -t NS actilis.net
actilis.net name server dns2.actilis.net.
actilis.net name server dns1.actilis.net.
```

L'option "**-C**" de la commande "**host**" demande à chaque **NS** d'un domaine son enregistrement **SOA**.

```
# host -C actilis.net
Nameserver 212.83.186.142:
      actilis.net has SOA record dns1.actilis.net. admin.actilis.net. 2016030802 3600 1800 1209600
600
Nameserver 212.83.186.49:
      actilis.net has SOA record dns1.actilis.net. admin.actilis.net. 2016030802 3600 1800 1209600
600
```

Cela permet par exemple de vérifier si les secondaires sont "à jour" (même numéro de série).

Notes personnelles

2.4.4- Enregistrement MX

L'enregistrement de type **MX** (*Mail eXchanger)* indique quels sont les serveurs SMTP responsables du mail entrant pour un domaine.

Il y en a souvent plusieurs, et chacun indique la priorité du serveur qu'il mentionne. On doit contacter en premier celui de plus petite priorité.

Une réponse de type MX est composée de deux éléments : la **priorité** et le **nom du serveur**.

```
$ host -t MX actilis.net
actilis.net mail is handled by 10 mx01.actilis.net.
actilis.net mail is handled by 10 mx02.actilis.net.
```

2.4.5- Enregistrement TXT

Il sert à publier des informations générales textuelles sur un équipement ou un nom.

On l'utilise en particulier pour implémenter **SPF** (**Sender Policy Framework**).

Il avait au départ été proposé le type **SPF**, mais c'est l'enregistrement **TXT** qu'utilise **OpenSPF**[5].

```
$ host -t TXT actilis.net
actilis.net descriptive text "v=spf1 mx ip4:88.190.224.166/32 ip4:88.190.224.163/32
ip4:88.190.224.162/32 88.190.244.22/32 ip4:88.190.244.61/32 ip4:88.190.244.62/32 -all"
```

5 http://www.openspf.org/ vise à refuser un e-mail s'il n'est pas émis par un serveurs habilité à le faire pour un domaine.

Notes personnelles

2.4.6- Autres types d'enregistrements

2.4.6.1- Enregistrement SRV

Les entrées **SRV** (RFC 2782) servent à "déclarer des services quelconques".

Les enregistrements **SRV** permettent de spécifier **un protocole** , un poids (utilisé lorsque plusieurs enregistrements de priorité identique existent) et **un port** pour chaque enregistrement, en complément de la priorité et du nom de serveur.

Le format d'un enregistrement SRV est spécifique, et permet par exemple de placer un service web sur 2 machines nominales de priorité 10 (une rapide recevant 90% des requêtes et une lente n'en recevant que 10), un service étant associé à un port "pas forcément standard", tout en prévoyant un serveur de backup (priorité de 20) si les serveurs nominaux sont indisponibles :

```
_web._tcp.www.actilis.net. SRV 10 90 80   srvwebrapide.actilis.net.
_web._tcp.www.actilis.net. SRV 10 10 8080 srvweblent.actilis.net.
_web._tcp.www.actilis.net. SRV 20 5 8000  srv-backup.actilis.net.
```

Windows ainsi que certains protocoles récents (SIP, XMPP,..) utilisent ces enregistrements.

Pour l'instant, il existe encore (trop) peu de programmes clients qui gèrent les entrées SRV.

Voir la page http://www.bortzmeyer.org/2782.html

Notes personnelles

2.4.6.2- Enregistrement LOC

Ils renseignent sur la position géographique d'un équipement.

LOC (RFC 1876) remplace GPOS, dont la RFC était erronée (confusion Latitude / Longitude).

Il indique aussi la taille des éléments décrits (diamètre de la sphère dans laquelle l'élément peut être contenu, entre 1cm et 90.000 km), ainsi qu'un degré de précision horizontale ou verticale sur la position donnée (latitude, longitude, altitude).

```
... LOC ( d1 [m1 [s1]] {"N"|"S"} d2 [m2 [s2]] {"E"|"W"}
          alt["m"] [size["m"] [hp["m"] [vp["m"]]]] )
```

Comme pour l'enregistrement SOA, les valeurs de cet enregistrement doivent être fournies **entre parenthèses**, même si celles-ci n'apparaissent dans la formulation de la réponse.

Exemple :

```
$ host -t LOC kervam.actilis.net
kervam.actilis.net location 47 43 01.7 N  3 25 18.6 W 32.00m 20m 10m 10m
```


Notes personnelles

2.4.7- Enregistrement PTR

On le trouve dans les zones "**reverse**".
C'est l'inverse de l'enregistrement A.

L'enregistrement **PTR** (*PoinTeR*) associe
un nom pleinement qualifié (réponse) à une adresse (question).

Ce type d'enregistrement constitue la réponse à une question posée pour une résolution "inverse" (reverse), et se trouvera uniquement dans une zone "reverse".

```
$ host -t PTR 78.212.154.196
196.154.212.78.in-addr.arpa domain name pointer kervam.actilis.net.
```

Notez le sens dans lequel est exprimée l'adresse IP :

Dans la question, elle est citée normalement
Dans la réponse, elle est à l'envers, et suivie par le nom "**in-addr.arpa**".

La question peut être une adresse IPv6.

```
# host -t AAAA srv-gw.actilis.net
srv-gw.actilis.net has IPv6 address 2001:bc8:35f1:501::1
# host -t PTR 2001:bc8:35f1:501::1
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.0.5.0.1.f.5.3.8.c.b.0.1.0.0.2.ip6.arpa domain name pointer srv-
gw.actilis.net.
```


Notes personnelles

2.5- Les zones

Un serveur DNS gère des zones (les bases de données correspondant à un nom de domaine).

On appelle **zone** un nom de domaine et les enregistrements qu'il contient.

Déclarer une zone, c'est annoncer qu'on connaît un nom de domaine,
et se donner un moyen de trouver les réponses aux questions qui le concernent.

2.5.1- Le sens des zones

On distingue deux sens de zone :
des zones de sens "**direct**" (correspondance Nom vers Adresse)
des zones de sens "**reverse**" (correspondance Adresse vers Nom).

C'est le **nom de la zone** qui **définit si elle est** de sens **direct ou reverse**.

```
zone "actilis.net." {
   type master;
   file "generic-master";
};

zone "0.168.192.in-addr.arpa" {
   type master;
   file "actilis.reverse" ;
};
```

Nous déclarons que nous gérons la zone actilis.net.
Notez le "." à la fin.

Ce nom est spécifique, il introduit une zone reverse pour un réseau dont l'adresse est :
- **codée à l'envers** (192.168.0 => 0.168.192),
- suivie du nom de domaine **in-addr.arpa**

Notes personnelles

2.5.2- Les types de zone

Une zone a forcément un **type**, **spécifié obligatoirement** dans la configuration de celle-ci.

Master: : gère intégralement la zone, aussi appelé DNS primaire

Slave : réplica d'un maître, par recopie périodique, DNS secondaire
Lorsque l'on déclare une zone slave, il faudra spécifier quels sont les maîtres

Forward : renvoie les requêtes à un (ou des) forwarder(s).
On peut déclarer une zone de type forward sans forwarders,
sert à inhiber "juste pour cette zone" le forwarding déclaré de manière globale.

On trouve aussi d'autres types de zone, moins fréquents :

Hint ("recursor") : résolutions par les racines, et mise en cache des réponses.
Ce type permet de spécifier une liste de root-servers.
Si aucune zone de type "hint" n'est spécifiée, une liste "built-in" (*lib/dns/rootdns.c*) est utilisée.
Au démarrage, Bind tente de contacter au moins 1 des serveurs de la liste,
But : mettre à jour la liste des root-servers "actualisée" qu'il garde en mémoire.

Delegation-Only : utilisé sur les serveurs gérant les zones dites d'infrastructure (les TLD),

Stub : réplica d'un maître, mais ne prend que les enregistrements SOA & NS.

Notes personnelles

2.6- Zone de type master

Pour une zone de sens **direct**, le nom de la zone est le **nom de domaine**.
Pour une zone **reverse**, c'est l'adresse du **réseau à l'envers**, suivi de **.in-addr.arpa**.

Tout le reste est identique, que la zone soit de type **direct** ou **reverse**.

Une zone de type "**master**" doit spécifier un fichier de zone grâce à la directive **file**.
Si le fichier précisé par **file** n'est pas pleinement qualifié :
Il est cherché dans le répertoire défini par **directory** (voir dans la section "**options**").
Souvent, c'est "/var/named".

2.6.1.1- Liste (non exhaustive) de paramètres courants dans une zone de type master

```
zone zone_name [class] {

type master;
[ allow-query { address_match_list }; ]
[ allow-query-on { address_match_list }; ]
[ allow-transfer { address_match_list }; ]
[ allow-update { address_match_list }; ]
…
[ also-notify { ip_addr [port ip_port] ; [ ip_addr [port ip_port] ; ... ] }; ]
…
[ file string ; ]
…
};
```


Notes personnelles

2.6.1.2- Exemple de déclaration de zone master de type direct

```
zone "dom0.fr." {
        type master;
        file "dom0.fr.hosts";
        allow-update    { 192.168.0.199; };
        allow-transfer { 192.168.0.201; };
        allow-query     { 192.168.0.0/24; };
};
```

Allow-update : machines autorisées à nous soumettre des mises à jour.

Allow-transfer : machines autorisées à transférer la zone (pour les DNS secondaire).

Allow-query : machines autorisées à nous poser des questions.

On a cité des directives **de contrôle d'accès,** qui seront détaillées plus loin :

allow-transfer (sans t) : machines qui peuvent nous solliciter pour un transfert (listage) de zone,

allow-update : machines qui pourront nous soumettre des mises à jour dynamiques

allow-query : celle-ci l'emporte sur celle déclarée dans la section options.

Notes personnelles

2.6.1.3- Exemple de déclaration de zone master de type reverse

Pour une zone **reverse**, souvenons nous que le nom est spécifique

```
zone "0.168.192.in-addr.arpa" {
        type master;
        file "dom0.fr.reverse";
        allow-update { 192.168.0.199; };
};
```

Il s'agit ici d'une zone reverse pouvant résoudre les adresses en noms d'hôtes sur le réseau 192.168.0.

Notes personnelles

2.7- Zone de type slave

Pour un domaine classique, dans le sens **forward**, le nom de la zone est là aussi le nom de domaine.

2.7.1.1- Déclaration

On spécifie comment joindre le ou les maîtres (directive **masters**), on peut indiquer un fichier de zone grâce à la directive **file**.

Si aucun fichier n'est précisé, Bind travaille en mémoire; si le fichier précisé par **file** n'est pas pleinement qualifié, il est cherché dans le répertoire défini par l'option **directory**.

2.7.1.2- Liste (non exhaustive) de paramètres d'une zone slave

```
zone zone_name [class] {
   type slave;
   [ allow-notify { address_match_list }; ]
   [ allow-query { address_match_list }; ]
   [ allow-query-on { address_match_list }; ]
   [ allow-transfer { address_match_list }; ]
   [ allow-update-forwarding { address_match_list }; ]
   …
   [ file string ; ]
   …
};
```


Notes personnelles

2.7.1.3- Exemple de déclaration d'une zone slave

Pour faire "au plus simple", la description doit indiquer les adresses des maîtres.

```
zone "dom1.fr." {
        type slave;
        masters { 192.168.0.10; };
};
```

Il faut que le maitre nous autorise à le seconder (**Allow-transfer** de son coté).

Notes personnelles

2.7.1.4- La section de type "masters"

Il ne faut pas la confondre avec la directive "**masters**" d'une section zone. Cependant, la section "masters" permet de déclarer des listes nommées de maîtres.

Lorsque l'on est **slave** ou **stub** sur un grand nombre de zone, et qu'il y a toujours la même liste de serveurs maîtres pour ces zones, on peut créer une section de type masters, et la citer dans les clauses **masters** des zones.

Cela facilitera le changement de l'un des maîtres...

Exemple :

```
masters les_primaires {
   192.168.0.1 ; 192.168.0.2;
}

zone "dom1.fr." {
        type slave;
        masters { les_primaires; };
};
```


Notes personnelles

2.8- Les enregistrements

La zone est l'ensemble "description" + "base de données" (un fichier texte appelé "**fichier de zone**") formée de l'ensemble des enregistrements, que l'on appelle des "**Resources Records**" (**RR**).

2.8.1- Format général d'un RR

Chaque RR fait l'objet d'une ligne dans le "**fichier de zone**" :

```
<domain> <opt_ttl> <opt_class> <type> <resource_record_data>
```

domain : nom de domaine
Lorsqu'il est vide, alors l'enregistrement concerne le même nom que le précédent,
Dans certains cas, on le remplace par le signe "@", que Bind interprète comme "l'origine".

opt_ttl : durée de vie dans le cache de l'information (en secondes) (optionnel)
opt_class : toujours IN (Internet), c'est la valeur par défaut (optionnel).

type : **SOA** (Start of Authority), **NS** (Name Server), **MX** (Mail Exchanger), **A** (Address Ipv4), **AAAA** (Address Ipv6), **CNAME** (Canonical Name : alias de nom de machine), **PTR** : (PoinTeR : Reverse Mapping) , **HINFO** (informations sur le matériel et logiciels de l'hôte), ...

resource_record_data : l'information à enregistrer (la réponse à la question)

Notes personnelles

2.8.2- Enregistrement spéciaux, mots clés

Dans les fichiers de zone, mots clés peuvent apparaître et ne sont pas des enregistrements :

2.8.2.1- $ORIGIN nom.

C'est ce qui doit être ajouté pour "compléter" chaque nom non terminé par un "."

Par défaut, "$ORIGIN" a la valeur du nom de la zone

Il peut apparaître plusieurs fois, concernant à chaque fois les lignes qui le suivent

2.8.2.2- $INCLUDE fichier[origine]

Inclut un sous-fichier de zone à l'endroit où est cité l'ordre "include"

l'origine peut donc être spécifiée spécialement pour une inclusion

2.8.2.3- $TTL nombre

On peut préciser un TTL pour chaque enregistrement (champ opt_ttl)

Sans lui, c'est la valeur de l'enregistrement spécial "$TTL" qui compte

Une tolérance permet (pour l'instant) d'oublier "$TTL", mais cela ne durera pas

Notes personnelles

2.8.2.4- $GENERATE intervalle éléments de réponse

On peut exprimer des listes en utilisant le mot clé **$GENERATE** :

```
$GENERATE 200-210 $ PTR m$.bzh.
```

Est équivalent à déclarer les enregistrements 200 à 210 sous la forme :

```
200 PTR m200.bzh.  … jusqu'à 210 PTR m210.bzh.
```

Cela peut être utilisé pour générer des noms pour les zones reverse... c'est spécifique à Bind 9.

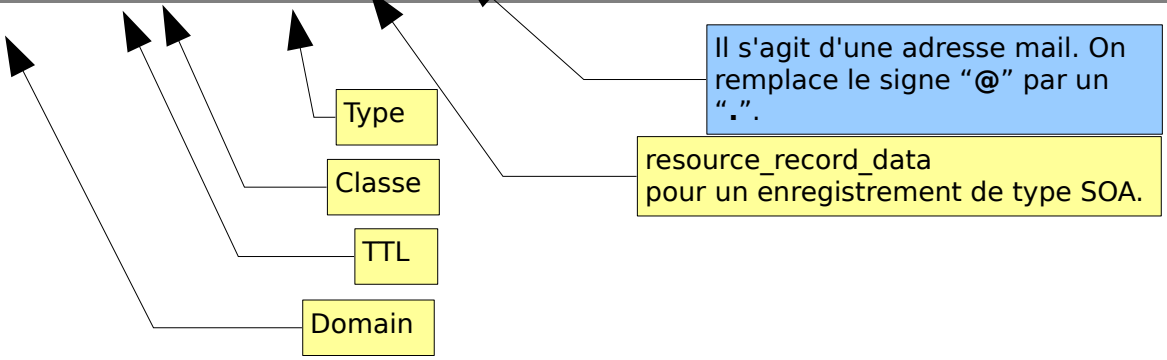
Notes personnelles

2.8.3- Saisir les différents enregistrements

2.8.3.1- L'enregistrement SOA

La réponse est composée de plusieurs éléments dont 5 encadrés par des parenthèses :

```
;<domain>  <ttl>  <class>  <type>  <resource_record_data>
;                                  NS   MAIL     SERIAL REFRE RETR EXPIR   MIN
lo.bzh.    38400  IN       SOA     ns1 root.ns1 ( 1    3600  600  86400   10800 )
```



Attention : les noms "ns1" et "root.ns1" ne sont pas terminés par un "." (non FQDN).

Pour Bind, c'est un "problème" et il leur ajoutera le nom de la zone dans laquelle ils se trouvent,
Pour nous, cela les rend génériques.

Notes personnelles

2.8.3.2- L'enregistrement MX

La réponse est composée de deux éléments :

```
;<domain>  <opt_ttl>  <opt_class>  <type>  <resource_record_data>
;                                          PRI MAILSRV
lo.bzh.    38400      IN          MX      10 srvmail1
                                  MX      20 srvmail2
```

Type

C'est le nom du serveur de mail pour le domaine, précédé de sa priorité.

resource_record_data pour un enregistrement de type MX : 2 éléments de réponse.

On cite deux enregistrements du même type pour une même question "MX". Les deux réponses seront fournies lorsque la question sera posée.

Grâce à la priorité, un serveur de mail cherchant à nous contacter saura qu'il faut d'abord utiliser le serveur "srvmail1", puis, s'il est indisponible, le serveur "srvmail2".

2.8.3.3- L'enregistrement NS

Un seul élément dans la réponse :

```
;<domain>  <opt_ttl>  <opt_class>  <type>  <resource_record_data>
;                                          SRV
lo.bzh.    38400      IN          NS      srvdns1
                                  NS      srvdns2
```

On cite ici 2 enregistrements NS, ce n'est absolument pas obligatoire.

Notes personnelles

2.8.3.4- Enregistrements A et CNAME, enregistrements tournants

Ils associent un nom à une adresse (A) ou un autre nom (CNAME)

```
;<domain>          <opt_ttl> <opt_class>  <type>   <resource_record_data>
;
srvdns1.lo.bzh.     38400        IN        A       192.168.0.1
srvdns2.lo.bzh.                            A       192.168.0.2

srvmail.lo.bzh.                            A       192.168.0.3
srvimap.lo.bzh.                            CNAME   srvmail
```

L'enregistrement srvdns2 n'a pas les propriétés optionnelles TTL et IN. Il hérite donc du TTL de la zone, défini par $TTL, et est de classe IN.

Notre serveur IMAP est le même que notre serveur "mail", on peut donc utiliser un alias (CNAME).

Important :
Les noms dont on parle dans les réponses aux enregistrements NS et MX doivent obligatoirement faire l'objet d'enregistrements **A**.

On ne peut pas utiliser d'alias (**CNAME**) pour ces deux éléments.

Notes personnelles

Plusieurs réponses à une question de type A :
Nous avons 4 serveurs web, nous les mettons de cette manière en répartition de charge derrière le nom "www".

```
;<domain>          <opt_ttl> <opt_class>  <type>   <resource_record_data>
;
www.lo.bzh.          38400        IN        A       192.168.0.10
                                            A       192.168.0.11
                                            A       192.168.0.12
                                            A       192.168.0.13
```

Il y a plusieurs réponses du même type pour un enregistrement, celles-ci peuvent être servies dans un ordre défini par la directive **rrset-order** (dans le déclaration de la zone ou les options)

```
rrset-order { class ANY type ANY name "*" order cyclic ; };
```

Au lieu de "**cyclic**", on pourrait utiliser "**fixed**" ou "**random**".

Dans Bind 9, on peut directement passer plusieurs réponses pour un enregistrement A.
Dans Bind 8, on devait passer par un **CNAME** pour la déclaration d'enregistrements tournants.

Notes personnelles

2.8.4- Simplification des fichiers de zone

2.8.4.1- Omettre des champs

On peut omettre le champ Classe (**IN**), préciser les TTL par l'option **$TTL**, et ne pas les préciser pour chaque enregistrement.

2.8.4.2- Utiliser l'origine ($ORIGIN)

On peut spécifier des sous-domaines dans une même zone en redéfinissant l'**origine courante**. Cela évite de déclarer plusieurs zones.

On utilise aussi le $ORIGIN pour éviter d'avoir à qualifier pleinement les noms. Dans le cas là, pour les endroits où il faut "parler de l'origine", on utilise le caractère "@".

Le signe « @ » **remplace** dans tout fichier de zone **le nom de l'origine courante**.
En début de zone, elle vaut le nom de la zone tel que défini dans named.conf.

```
$ORIGIN @
;
@       SOA     srvdns1 formation.actilis.net. ( 5 10800 3600 432000 38400 )
```

On note "@" au lieu de "lo.bzh." et tout nom non terminé par un "." sera complèté par l'origine.

On note aussi "srvdns1" pour profiter de cela, mais on fait attention à l'adresse mail "formation@actilis.net" qui doit être exprimée "formation.actilis.net." avec le "." final, sous peine d'être interprétée formation@actilis.net.lo.bzh...

Notes personnelles

La suite, ce sont des enregistrements de sous-domaines.

```
$ORIGIN ports.lo.bzh.
peche         CNAME     keroman
voile         CNAME     kernevel
              TXT       "A71, bien abrité, mais beaucoup de passage"

$ORIGIN plages.lo.bzh.
stole         TXT       "familiale, calme et agreable le soir, mouillage peu connu"
              GPOS      47.705022 -3.426619 5
perello       TXT       "idem, tres belle, mais caillou en entrant sur babord"
kaolins       TXT       "bronzage integral, abrite du nordet mais pas du ressac"
fortbloque    TXT       "planche, surf, kite"
loch          TXT       "rouleaux cassants, coucher de soleil"
```

A partir d'ici, on redéfinit l'origine, les noms peche et voile sont maintenant relatifs à ports.lo.bzh.

Qui a dit que le DNS servait uniquement à gérer des adresses IP ? Ici, grâce aux enregistrements de type GPOS (il faudrait lui préférer LOC), et TXT (du texte libre), on constitue une petite base de données.

Notes personnelles

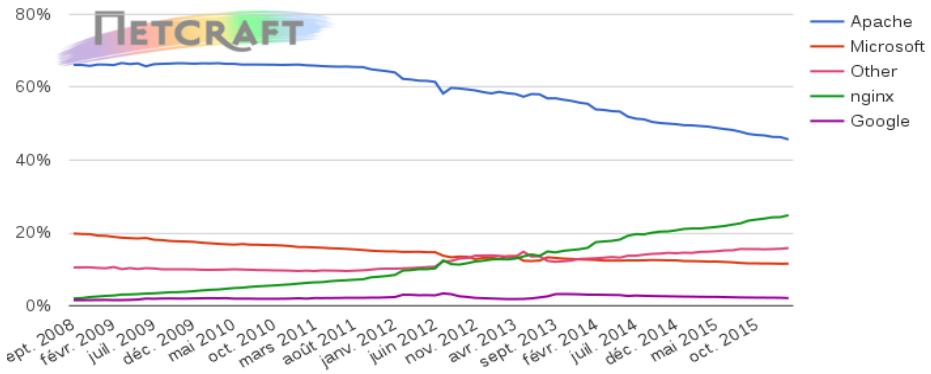
3- Apache HTTP Server : Un peu d'histoire

Notes personnelles

3.1- Introduction à Apache HTTP server

Apache HTTP Server, souvent appelé **Apache**, est un logiciel serveur HTTP produit par la fondation "Apache Software Foundation" (www.apache.org).

C'est le serveur HTTP le plus populaire (netcraft).



Graphique tiré du Site Netcraft (Top million busiest sites)

Developer	January 2016	Percent	February 2016	Percent	Change
Apache	463,092	46.31%	456,483	45.65%	-0.66
nginx	243,340	24.33%	248,627	24.86%	0.53
Microsoft	115,358	11.54%	115,447	11.54%	0.01
Google	21,824	2.18%	21,008	2.10%	-0.08

C'est un Logiciel Libre avec un type spécifique de licence, nommée " licence Apache ".
http://www.apache.org/licenses/LICENSE-2.0

Apache HTTP Server est devenu le serveur web le plus populaire à partir de 1996, avant la sortie de sa version 1.3 (sortie en juin 1998).

Notes personnelles

3.2- Evolution des versions

3.2.1- De 1.3 à 2.0...

Threads sur Unix : Sur les systèmes Unix, Apache peut s'exécuter selon un modèle hybride multi-processus et multi-threads, en employant les threads selon la norme POSIX. Ceci peut **dans certains cas** améliorer les performances.

Nouveau système de construction : entièrement ré-écrit et repose enfin sur **autoconf** et **libtool**. Cela rend le système de configuration plus semblable aux autres applications.

Support multiprotocole : une infrastructure afin de servir de multiples protocoles (autres que HTTP).

Meilleur support des plates-formes autres qu'Unix : 2.0 est plus rapide et stable que 1.3 sur les plates-formes non Unix telles que BeOS, OS/2, et Windows. **L'introduction des modules multi-traitements (MPMs)** : permettant ainsi d'éviter les couches d'émulation POSIX souvent boguées et peu performantes.

Nouvelle API Apache : L'API pour les modules de la version 2.0 a changé de manière importante. Beaucoup de problèmes d'ordonnancement des modules existants dans la version 1.3 ont disparu. En version 2.0, le chargement des modules s'effectue par une unique fonction afin de permettre une plus grande flexibilité.

Autres améliorations : Support Ipv6, Filtering, Simplification de la configuration...

Notes personnelles

3.2.2- De 2.0 à 2.2...

Refonte des systèmes de contrôle d'accès et d'authentification (Authn/Authz) et ajout de nouveaux modules simplifiant la configuration.

Mise en cache : arrivé à maturité

Proxy : nouveaux modules pour la répartition de charge (mod_proxy_balancer) et pour la prise en charge d'AJP 1.3 utilisé pour Tomcat (mod_proxy_ajp).

La Configuration par défaut : a changé et été simplifiée.

Arrêt en douceur : possible par un signal "graceful-stop".

Filtrage en sortie : mod_filter permet la mise en place de filtres de contenus de manière intéressante.

Autres améliorations : Support des gros fichiers (> 2 Go), MPM "event", amélioration de la gestion des connections keep-alive.

Notes personnelles

3.2.3- De 2.2 à 2.4...

Le **choix du MPM** peut s'effectuer **à l'exécution**, et le **MPM Event** est stable.

La directive **LogLevel** peut maintenant être définie par module et par répertoire.

Les sections <**If**>, <**ElseIf**> et <**Else**> permettent de définir une configuration en fonction de critères liés à la requête.

Un nouvel interpréteur d'expressions permet de spécifier des conditions complexes via des directives à syntaxe commune comme **SetEnvIfExpr**, **RewriteCond**, **Header**, **<If>**, etc..

KeepAliveTimeout peut maintenant être exprimé en millisecondes
NameVirtualHost est obsolète et n'est plus utile.

La nouvelle directive **AllowOverrideList** permet de contrôler de manière plus précise la liste des directives autorisées dans les fichiers **.htaccess**.

La directive **Define** permet de définir des variables dans les fichiers de configuration, améliorant ainsi la clarté si la même valeur est utilisée en plusieurs points de la configuration.

La version 2.4.x tend à utiliser moins de mémoire que la version 2.2.x.

Un nombre important de nouveaux modules apparaît, mais par défaut, la plupart sont inhibés.

Notes personnelles

3.2.4- Quelle version choisir : 2.2 ou 2.4 ?

Actuellement, les versions disponibles sont les suivantes

La version **1.3** : sortie juin 1998, fin de vie en février 2010, (v1.3.42) : **12 ANS**

La version **2.0** : sortie en mai 2002, fin de vie en juillet 2013 (2.0.65). : **11 ANS**

La version **2.2** : sortie fin 2005… (11 ans à fin 2016)
Package httpd-2.2.3 est fourni avec RHEL5 (⇒ mars 2017),
Package httpd-2.2.16 avec RHEL6 (⇒ novembre 2020),
Package apache-2.2.22 avec Debian 7 (⇒ mai 2018).

La version **2.4** : sortie début 2012.
Package httpd-2.4.6 avec RHEL 7 (⇒ juin 2024),
Package apache-2.4.7 avec Debian 8 (⇒ mai 2020).

3.2.5- Les sites d'Apache.org

Le site principal de la fondation Apache : http://www.apache.org/.

Le projet HTTPD (le serveur web) : http://httpd.apache.org/

La documentation : http://httpd.apache.org/docs/

Notes personnelles

3.3- Premier lancement du service

Les fichiers exécutables sont dans le répertoire **PREFIX/bin** et on trouve notamment :

httpd : l'exécutable du serveur,
apachectl : un script de pilotage de **httpd**.

3.3.1- Le binaire httpd

L'exécutable **httpd** accepte des options permettant de s'informer sur :

la version (**-v**), affiche aussi la date de compilation,
les options de compilation (**-V**), permet notamment d'identifier le fichier de configuration,
les modules supportés (**-l**), (ce sont les modules intégrés en statique)
les options de configuration disponibles (**-L**) (appartenant aux modules statiques)

Il dispose même d'une option de validation du fichier de configuration (**-t**).

Sans oublier

l'option (**-f**) qui permet de spécifier un fichier de configuration différent,
l'option (**-D**) qui permet de spécifier un "ServerRoot" différent (=Prefix sinon)
l'option (**-D PARAM**) permet de positionner un paramètre.
l'option (**-h**) qui affiche la liste des options supportées.

C'est le script **apachectl** que nous allons utiliser pour lancer Apache.

Notes personnelles

3.3.2- Le script apachectl

```
# cd /opt/httpd-2.4/bin
# ./apachectl
Usage :
apachectl (start|stop|restart|fullstatus|status|graceful|configtest|help)
```

Ce script accepte plusieurs arguments différents (un à la fois) et sert à :

démarrer **(start)** le serveur,
arrêter **(stop)** le serveur,
redémarrer (**restart**),

relire le fichier de configuration **(graceful)** : cette option démarre le serveur s'il ne l'était pas.

tester le fichier de configuration **(configtest),**

donner l'état du serveur
de manière sommaire **(status),**
ou de manière complète (**fullstatus**)

Notes personnelles

4- Apache HTTP Server : principe de configuration

Notes personnelles

4.1- Les bases de la configuration d'Apache

4.1.1- Les fichiers de configuration

Le point d'entrée de la configuration standard d'apache HTTPD est "**httpd.conf**" ou "**apache2.conf**" :

Sur le package de RedHat, c'est **/etc/httpd/conf/httpd.conf**.
Sous Debian, c'est **/etc/apache2/apache2.conf**.

La configuration réside en partie seulement dans ce fichier principal…

Des directives **Include** et **IncludeOptional** permettent de réaliser des inclusions de sous-fichiers...
Générer automatiquement certains morceaux de configuration
Diviser les différentes directives en catégories thématiques.
Ranger correctement les descriptions de sites virtuels, de répertoires

On doit donc recenser les fichiers "*.conf" pointés par des directives **Include** ou **IncludeOptional**.
Sous RedHat, il y en a dans "**conf.d**" et "**conf.modules.d**",
Sous Debian, c'est dans "**mods-enabled**", "**conf-enabled**" et "**sites-enabled**"

ServerRoot : la racine du serveur.

C'est à ce répertoire que sont relatifs les chemins d'accès non-pleinement qualifiés (logs, modules, sous-fichiers de configuration…)

Notes personnelles

4.1.2- Principes de configuration

Règles générales :

Une seule directive par ligne
Les **commentaires** sont introduits par le caractère "**#**"
Des lignes peuvent continuer sur celle du dessous : il faut les terminer par le caractère "\"

Sections / Contextes : http://httpd.apache.org/docs/2.4/sections.html
La configuration est découpée en **contextes** : **server-config**, **directory**, **virtual-host**, ***.htaccess***[6]

On définit des **directives, dont les contextes limitent la portée.**

Certaines directives ne sont admises que dans certains contexte.
L'utilisation d'une directive hors contexte de validité engendre une erreur
"...not allowed here…"

Chaque directive est applicable dans un des 4 contextes suivants :
"**S**erver-Config" : Configuration du Serveur,
"**V**irtualHost" : Configuration d'un Serveur Virtuel,
"**D**irectory" : Configuration relative à un répertoire,
"**H**taccess" : Dans un fichier .htaccess

Il existe plusieurs formes de contextes de type répertoire: **<Directory… >**, **<Location… >**, **<Files… >**

6 ".htaccess" n'est pas un contexte déclaré dans le fichier de configuration, mais côté contenu web... voir plus loin.

Notes personnelles

4.1.3- Le contextes conditionnels

On déclare un sous-contexte par des balises **<TypeContexte>** … **</TypeContexte>**.

Tout contexte peut contenir des sous-contextes
qui peuvent être imbriqués,
dont le contenu n'est applicable qu'à la condition qu'ils énoncent.

Il existe <**IfDefine**>, <**IfModule**>**,** <**IfVersion**>**,** et <**If**> (v2.4).

```
<IfDefine SSL> … </IfDefine>
```

Prise en compte si le paramètre « **SSL** » a été défini.
Au lancement, grâce à l'option **-D** de l'exécutable **httpd**.

```
<IfModule mod_name.c> … </IfModule>
```

Prise en compte si le module « mod_name » est chargé.

Ici, on choisit d'utiliser la directive **DirectoryIndex** si le module qui la fournit est chargé.

```
<IfModule mod_dir.c>
    DirectoryIndex index.html
</IfModule>
```

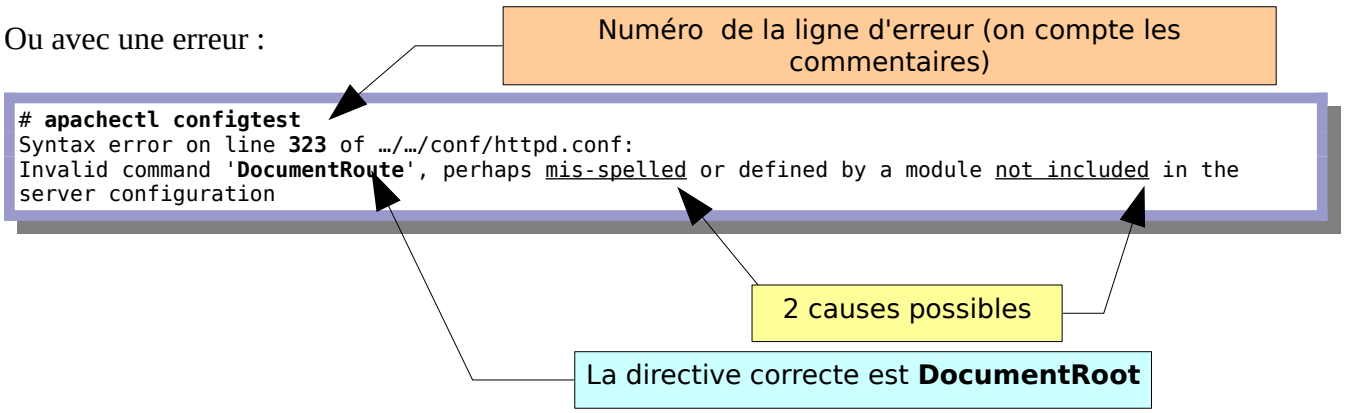
Nom du module tel qu'il apparait dans la directive "**LoadModule**" qui le charge.

Notes personnelles

4.1.4- Tester la configuration : apachectl configtest

```
# apachectl configtest
Syntax OK
```

Ou avec une erreur :



En cas d'erreur dans la configuration, rien n'est indiqué dans le fichier **error_log**.

Celui-ci concerne les requêtes effectuées par les utilisateurs.

Notes personnelles

4.1.5- Les fichiers de journalisation

Ils se trouvent dans le sous répertoire **/var/log/httpd** (RedHat) ou **/var/log/apache2** (Debian).

Il y en a deux par défaut :

Le fichier **access_log** (ou access.log) : trace de chaque hit.

Défini en principe par une directive **CustomLog**
On pourra spécifier le format des lignes
C'est ce fichier qui intéresse les outils de statistiques

Le fichier **error_log** (ou error.log) : trace des messages d'erreur.

Défini par la directive **ErrorLog**
On pourra préciser le niveau de sévérité à partir duquel on trace.
C'est ce fichier qui intéresse l'administrateur…

Chaque site virtuel pourra disposer de son propre fichier de trace des hits et d'erreur.

Notes personnelles

4.2- La documentation et les modules

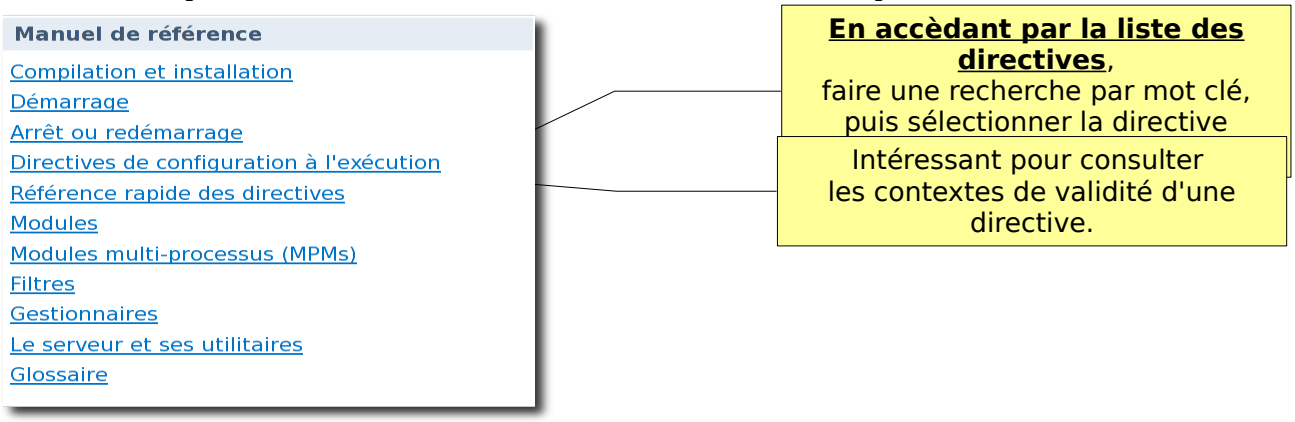
4.2.1- Qu'est-ce qu'un module ?

Apache HTTP-server repose sur des modules qui apportent des fonctionnalités :
des **modules statiques**, intégrés dans l'exécutable
des **modules dynamiques** que l'on peut charger (DSO = Dynamic Shared Object).

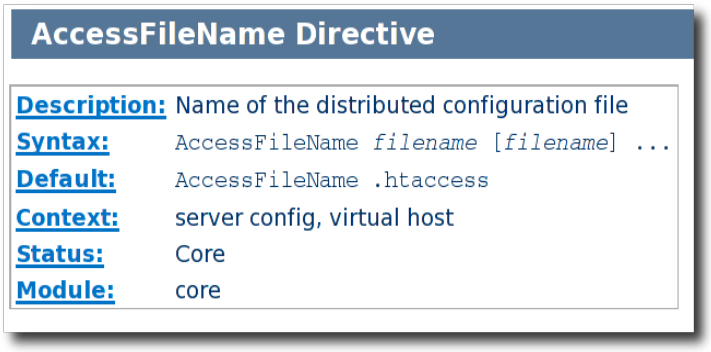
Chaque module apporte une ou plusieurs fonctionnalités, paramétrables au travers de directives.
Pour utiliser les directives d'un module, celui-ci doit être **actif**.

4.2.2- Documentation : http://httpd.apache.org/docs/

Le site officiel présente une documentation très bien faite et très complète



Notes personnelles



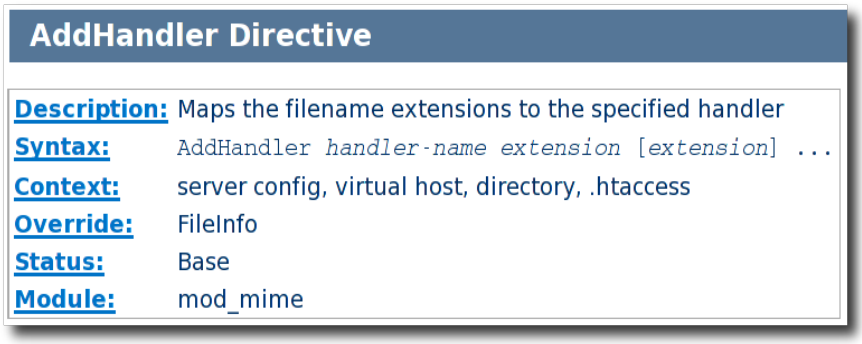
AccessFileName Directive

Description:	Name of the distributed configuration file
Syntax:	`AccessFileName filename [filename] ...`
Default:	`AccessFileName .htaccess`
Context:	server config, virtual host
Status:	Core
Module:	core

La directive **AccessFileName** n'est valide que dans les contextes Server Config et VirtualHost.

Elle n'est pas admise dans les fichiers **.htaccess**.

Elle fait partie du core (coeur) d'Apache et vaut " .htaccess " par défaut.



AddHandler Directive

Description:	Maps the filename extensions to the specified handler
Syntax:	`AddHandler handler-name extension [extension] ...`
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime

La directive **AddHandler** est admise dans tous les contextes...

...y compris **.htaccess**, si la liste "**Override**" mentionne " FileInfo ".

Elle est fournie par le module **mime** et n'a pas de valeur par défaut.

À propos de la ligne Status :

Core = toujours disponible,
Base = fournie par un module qui est en principe compilé,
MPM = fournie par un Multi-Processing Module, donc disponibilité dépendante du MPM choisi.
Extension = fournie par un module non compilé en standard.

Notes personnelles

4.2.3- Chargement des modules DSO

4.2.3.1- Les modules statiques

Ces modules sont ceux apportant les fonctions de base (du coeur, « core ») du serveur, notamment sa capacité à démarrer et charger d'autres modules.

(Apache 2.2)

```
# ./httpd -l
Compiled in modules:
  core.c
  prefork.c
  http_core.c
  mod_so.c
```

Apache 2.2 : le "MPM" choisi est visible en tant que "module statique" : ici "**prefork**"

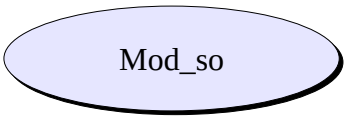
(Apache 2.4) : le MPM n'est plus dans la liste, on peut le charger dynamiquement.

```
# httpd -l
Compiled in modules:
  core.c
  mod_so.c
  http_core.c
```

Le module "**mod_so**" apparaît si l'exécutable accepte les modules dynamiques.

Notes personnelles

4.2.3.2- Les modules dynamiques



Ils sont en principe stockés dans "/usr/lib64/httpd/modules/" (Redhat)
ou dans "/usr/lib/apache2/modules" (Debian)

(ici un vieil exemple, sous Apache 1.3)

```
# pwd
/usr/local/apache/modules
# ls *.so
mod_access.so   mod_autoindex.so  mod_include.so      mod_status.so
mod_actions.so  mod_cgi.so        mod_log_config.so   mod_userdir.so
mod_alias.so    mod_dir.so        mod_mime.so
mod_asis.so     mod_env.so        mod_negotiation.so
mod_auth.so     mod_imap.so       mod_setenvif.so
```

Répertoire des modules

Aujourd'hui, il y en standard **près de 100 modules** avec Apache 2.4.

Chargement d'un module : LoadModule (*mod_so*) : charge un module dynamique

4.2.3.3- Lesquels faut il charger ?

Cela dépend des directives utilisées...
et c'est un bon exercice pour apprendre à chercher dans la documentation des directives.

Sans aucun module chargé, le fichier de configuration standard comporte des "erreurs".

Si des modules essentiels ne sont pas chargés, Apache ne sait pas servir de contenu.

Notes personnelles

4.3- Configuration réseau

Par défaut, Apache écoute sur toutes les adresses IP de la machine, mais seulement sur le port 80.

4.3.1- Adresses IP et port d'écoute

On utilise désormais uniquement la directive **Listen**. Que l'on peut répéter plusieurs fois :

Exemples :

```
Listen 8000
Listen 8001
```

Toutes les adresses IP du serveur,
sur les ports 8000 et 8001

“socket” qualifiée : Adresse:port
Apache écoute alors “seulement sur cette socket”
Adresses Ipv6 [encadrées par des crochets]

```
Listen 192.168.0.1:80
Listen [2001:db8::a00:20ff:fea7:ccea]:80
```

4.3.2- Paramètre concernant le protocole

Depuis Apache 2.1.5, la directive **Listen** accepte un argument optionnel : **le protocole**.
Si non précisé : c'est "**http**" sur le port 80, et "**https**" sur le port 443.

But : déterminer le protocole appliquer sur les ports non-standards et les optimisations liées.

Notes personnelles

4.3.3- Identité du serveur

Le processus père, ne sert qu'à piloter les autres, et fonctionne en "root"
Il nécessite le privilège **cap_net_bind_service** pour écouter sur un port privilégié.
Il ne répond à aucune requête.

Les directives **User** et **Group** permettent de spécifier l'identité Unix des autres processus.

Par défaut :
Dans le package RedHat, les processus ont **l'identité** de l'utilisateur unix Apache, et le groupe Apache.
Dans le package Debian, c'est "www-data".

Sécurité & permissions :
Les processus du serveur HTTPD doivent **pouvoir lire les fichiers de contenu** qu'ils servent.
Pour exécuter du code via CGI, il devront disposer du droit d'exécution.

```
# UID et GID des serveurs
User            apache
Group           apache
```

Identité des processus qui répondront aux requêtes des clients. L'utilisateur doit exister.

Tous les sites fonctionnent forcément sous la même identité, sauf :
avec le MPM "**itk**"
avec les MPM non threadés, sous Solaris 10 uniquement (expérimental) (**VHostUser**).

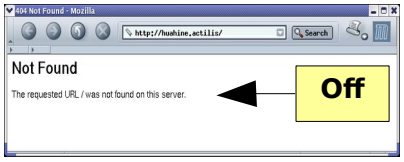
Notes personnelles

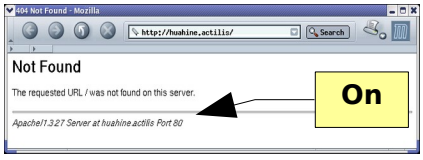
4.3.4- Informations concernant le serveur

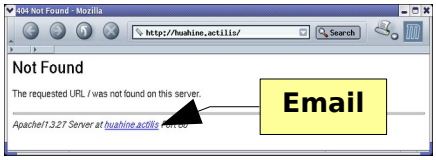
ServerName : nom du site

```
ServerName      www.actilis.net
```

ServerSignature : Signatures dans les pieds de page sur les documents générés par le serveur (les messages d'erreur et les pages générées à la volée par certains modules).







ServerAdmin : Adresse mail de l'administrateur du site

```
ServerAdmin     fmicaux@actilis.net
```

Adresse mail destinataire dans la référence (**ServerSignature Email**) .

Notes personnelles

ServerTokens : La signature du serveur dans les en-têtes dépend de la valeur de **ServerTokens**.

```
ServerTokens     { Prod[uctOnly] | Major | Minor | Min[imal] | OS | Full }
```

Prod[uctOnly]

```
Server: Apache
```

Major

```
Server: Apache/2
```

Minor

```
Server: Apache/2.0
```

Min[imal]

```
Server: Apache/2.0.41
```

OS

```
Server: Apache/2.0.41 (Unix)
```

Full (ou non précisé)

```
Server: Apache/2.0.41 (Unix) PHP/4.2.2 MyMod/1.2
```

À partir de Apache 2.0.44, cette directive contrôle aussi ce que montre la **ServerSignature**.

Notes personnelles

4.4- De l'URL au contenu à servir

4.4.1- DocumentRoot, Alias, Location...

Peut être situé dans le "**DocumentRoot**" : c'est un répertoire sur le système.
Peut être pointé par un "**Alias**" : un répertoire (ou fichier) local, pas forcément fils de DocumentRoot.
Peut être défini par un contexte"<**Location**>"
Et engendrer un traitement par un module associé à un **Handler** : status, info, mon_module...

Peut subir une ré-écriture de l'URL, une redirection, un traitement de type reverse-proxy...

Cas où l'URL désigne un "fichier" : http://serveur/fichier.html
Si le serveur ne le trouve pas : Erreur (404).

Cas où l'URL désigne un "répertoire" : http://serveur/rep/ ou http://serveur/
Le serveur renverra un des documents proposés par "**DirectoryIndex**" (le premier trouvé).

S'il n'en trouve aucun: Apache **peut, s'il y est autorisé, produire** un index (listing) du répertoire...

ou **renvoyer une erreur** "HTTP-403" (Forbidden),

qui **peut être détournée (ErrorDocument)** et engendrer l'affichage d'un autre contenu.

Location + Handler ou Files + Handler : http://…/server-status, http://…./*.php
Apache peut déclencher un traitement lorsque l'URL correspond à un motif ou un chemin
Exécuter du code… pour produire un contenu dynamique.

Notes personnelles

4.5- Contextes de répertoire

On peut appliquer certaines directives à contextes **de répertoires**, **de fichiers**, ou **d'URL**.

Toutes les directives valides dans ces "**contextes**" peuvent être (re)définies :

dans le **fichier de configuration**, par **des déclarations de contextes de type répertoire**

dans un **fichier spécifique du répertoire concerné** (**.htaccess** par exemple)

Exemple : la directive **DirectoryIndex**[7], qui peut varier d'un répertoire à l'autre.

4.5.1- Portée des directives

Le principe est celui de l'**héritage** de père en fils
décrire des options pour toute une arborescence en le faisant sur le répertoire de base.
créer des exceptions à partir d'un sous-répertoire en surchargeant des directives héritées.

Pour un "contexte de type répertoire", on peut à tout moment
activer une option à laquelle le père n'était pas sujet (surcharger),
désactiver une option à laquelle le père était sujet (alléger),
la descendance du sous-répertoire **hérite** toujours de son père.

Doc : http://httpd.apache.org/docs/2.4/configuring.html#scope

7 DirectoryIndex définit le nom du fichier d'index d'un répertoire (index.html, index.php, default.htm, ...)

Notes personnelles

4.5.2- Déclaration de contextes

Ces syntaxes prennent en paramètre un répertoire, un fichier, un morceau d'URL, sauf ".htaccess".

En cas de conflit, ce que dit "N+1" l'emporte sur ce que dit "N".

"1" - Selon des répertoires (nommés en dur) :

```
<Directory>...</Directory>
```

"2" - Dans chaque répertoire concerné, par le fichier ".htaccess": **L'emporte sur "1"**
".htaccess" est un fichier présent du côté contenu (donc DocumentRoot) et contenant des directives de configurations propre au répertoire où il se trouve.

"3" - Selon des répertoires nommés par des expressions régulières : **L'emporte sur "1"et "2"** .

```
<DirectoryMatch>...</DirectoryMatch> et <Directory ~ NOM>...</Directory>
```

"4" - Selon les noms des fichiers : **L'emporte sur "1"à "3"**.

```
<Files>... </Files>, et <FilesMatch>...</FilesMatch>
```

"5" - Selon un morceau d' l'URL : **L'emporte sur "1" à "4"**.

```
<Location>...</Location>, et <LocationMatch>...</LocationMatch>
```


Notes personnelles

4.6- Le répertoire pointé par DocumentRoot

4.6.1- Localisation des contenus servis

La directive **DocumentRoot** définit la racine des pages qui sont servies.

```
DocumentRoot    /home/www/www.actilis
```

Cela concerne le site par défaut, mais aussi chaque site virtuel qui pourra être déclaré avec sa propre directive **DocumentRoot**.

Ne pas confondre **DocumentRoot** (racine des documents d'un site) et **ServerRoot** (racine du serveur).

4.6.2- Choix du document à servir

Mod_dir

La directive **DirectoryIndex** définit des noms de fichier à servir...

```
DirectoryIndex  index.html index.php index.htm /erreurs/noindex.html
```

DirectorySlash : active ou désactive la gestion du "/" automatique à la fin des URLs. (> 2.0.51)

```
DirectorySlash  On
```


Notes personnelles

4.6.3- Parcours de répertoire

Lorsqu'une requête désigne un répertoire (et non pas un fichier), alors :

Mod_dir

Si le répertoire contient un des fichiers préconisés par la directive **DirectoryIndex**,

alors celui-ci est servi.

Mod_autoindex

Sinon

Si l'option Indexes est positionnée : le module **mod_autoindex** génère un contenu.

Sinon, Apache renverra une erreur de type HTTP-403 ("**Forbidden**".)

Notes personnelles

4.7- Le module autoindex

Il génère un listage du contenu d'un répertoire.

Le module **autoindex** permet
de lister l'arborescence du serveur,
de trier les documents selon plusieurs critères...

Son affichage est personnalisable :

le mode dit « FancyIndexing » engendre des pages contenant
des possibilités de tri
les détails sur les entrées du répertoire.

On l'active grâce à l'option "**Indexes**".

La directive **IndexOptions** sert à paramétrer le mode d'affichage, les ordres de tris, etc...

```
IndexOptions FanxyIndexing
```


Notes personnelles

Autres directives :

AddAlt, **AddAltByEncoding**, **AddAltByType**
Libellé texte au lieu d'une icône pour matérialiser le type de fichier.

AddDescription
Champ de description du fichier (le dernier à droite)

AddIcon, **AddIconByEncoding**, **AddIconByType**, **DefaultIcon**
Gestion des icônes en fonction des types de fichier

HeaderName et **ReadmeName** : Noms des fichiers d'entête ou de bas de page
Peuvent être des documents HTML (voir **IndexOptions SuppressHTMLPreamble**)

IndexIgnore : Nom des fichiers à cacher.
On peut constituer une liste par répétition de la directive.

IndexOrderDefault { Name | Date | Size | Description } :
Dans le cas où **FancyIndexing** est actif, détermine l'ordre par défaut.

IndexOptions : paramètrage fin du mode de listage des répertoires
Propriétés des répertoires : casse, répertoires en premier, champs affichés, largeur des colonnes, etc...

Notes personnelles

4.8- Les alias et les redirections

4.8.1- Alias

Grâce au module **mod_alias** (et sa directive **Alias**) on peut faire correspondre des URL à des répertoires ne se trouvant pas au niveau de **DocumentRoot**. C'est la notion de **répertoire virtuel**.

```
Alias chemin_URL chemin_répertoire
```

Mod_alias

Exemple :
L'URL http://serveur/doc ne pointe pas sur /home/doc, mais sur /opt/apache/htdocs/manual.

```
DocumentRoot /home
Alias /doc /opt/apache/htdocs/manual
```

La directive **ScriptAlias** sert à la même chose et autorise en plus l'exécution des CGI.

```
ScriptAlias chemin_URL chemin_répertoire
```

La directive **Redirect** déclare des redirections, donnant lieu à une nouvelle requête du client.

```
Redirect /chemin_URL http://serveur/service
```

Le module **mod_rewrite** permet aussi de faire des redirections, de manière plus souple et plus puissante, mais plus complexe.

Notes personnelles

4.8.2- La directive Redirect

Le module **mod_alias** permet aussi de faire des redirections, qui donnent lieu à une nouvelle requête de la part du client.

Cela se fait grâce à la directive **Redirect**.

```
Redirect [status] /chemin_URL http://serveur/service
```

On peut spécifier le type de redirection (à la place de [status]) qui peut être :

permanent (code HTTP 301),

temp (code HTTP 302) : c'est la valeur par défaut si aucun type n'est précisé,

seeother (code HTTP 303) : indique que la ressource est remplacée,

gone (code HTTP 410) : indique que la ressource a été définitivement supprimée

D'autres codes peuvent être spécifiés, numériquement :
Entre 300 et 399, le second argument (cible de la redirection) doit être fourni,
Si d'autres codes sont donnés, le second argument doit être omis.

Notes personnelles

4.8.3- Autres directives de redirection

La directive **RedirectPermanent** est équivalente à "Redirect permanent",

La directive **RedirectTemp** est équivalente à "Redirect temp",

La directive **RedirectMatch** (équivalente à Redirect), peut utiliser des expressions rationnelles

Exemple :

```
RedirectMatch (.*)\.gif$ http://www.anotherserver.com$1.jpg
```

On ne l'utilise en principe que pour mettre en œuvre une redirection externe.

Notes concernant les redirections

Il s'agit d'une demande retournée au client lui indiquant d'émettre une autre requête.

Le client est alors conscient de la nouvelle URL, et celle-ci s'affiche dans la barre d'adresse.

Les moteurs de recherche y sont sensibles pour mettre à jour leurs bases.

Notes personnelles

4.9- Pages de gestion des erreurs

La directive **ErrorDocument** permet de gérer les erreurs de manière plus élégante que l'affichage de la page blanche contenant en anglais le libellé de l'erreur HTTP :

Afficher une chaîne de caractères
Afficher une URL locale
Router vers une URL distante

C'est la directive **ErrorDocument** qui permet de paramétrer ce comportement.

```
ErrorDocument N°_ERR CIBLE
```

La CIBLE est un message :

```
ErrorDocument 403 "accès interdit"
```

Ou une URL locale :

```
ErrorDocument 404 /erreurs/missing.html
```

La page doit exister !

Ou une URL distante :

```
ErrorDocument 500 http://www.serveur.com
```


Notes personnelles

Note

Quelques codes HTTP :
401 : authentification
403 : contrôle d'accès
404 : not found
500 : serveur planté
http://fr.wikipedia.org/wiki/Liste_des_codes_HTTP

4.10- Options dans les contextes de répertoire

Des options d'exploration peuvent être définies par la directive **Options**.

La directive **Options** accepte le signe + et le signe – devant le nom de l'option ajoutée (+), ou retirée (-).

4.10.1- Gestion des options

Par défaut : seule l'option FollowSymLinks est active (2.4)… (All sur Version <=2.2).

On initialise au moins une fois la liste en plaçant la directive **Options** dans un contexte de type répertoire de haut niveau.

On peut vider la liste :

```
Options None
```

On peut activer toutes les options (sauf Multiviews)

```
Options All
```

On ajoute ou retire des options (à celles déjà présentes) en plaçant + et – devant les options.

```
Options +Indexes -ExecCGI
```


Notes personnelles

4.10.2- Options disponibles

Indexes
Autoriser le module "mod_autoindex" à travailler.

ExecCGI
Autoriser l'exécution des CGI via le module mod_cgi (même en l'absence de *ScriptAlias*)

Includes
Autoriser les Server Side Includes via le module mod_includes

IncludesNoExec
Idem à Includes, sauf pour les appels #exec cmd et #exec cgi

FollowSymlinks
Autoriser à suivre les liens symboliques

Uniquement dans un contexte **Directory**. Ignoré dans les contextes **Location**.

SymlinksIfOwnerMatch
Seulement si la cible appartient au même utilisateur que le lien

Multiviews
Autoriser les négociations de nom de ressource (mod_negociation) , notamment pour la langue

Notes personnelles

4.11- Délégation de pouvoir d'administration

On peut déléguer une partie de l'administration dans les répertoires des pages servies.
on délègue ainsi des pouvoirs de configuration à des développeurs

4.11.1- Principe

On crée des **fichiers "de configuration"** dans les répertoires **de données** des sites.

Apache en tiendra compte à chaque visite dans le royaume du répertoire contenant ce fichier
(ou un de ses répertoires fils, puisqu'**il y a héritage**).

L'administrateur garde le pouvoir et décide dans la configuration générale,

du nom d'un fichier de délégation qui sera pris en compte s'il est présent : **AccesFileName**

des directives ou classes de directives qu'il pourra contenir : **AllowOverride**

4.11.2- Fichier de délégation de pouvoir

La directive **AccessFileName** définit le nom de fichier de délégation.
Par défaut c'est **.htaccess** et la plupart des administrateurs conservent ce nom.

Le webmaster a le droit de modifier les fichiers **.htaccess** de son site
et se trouve donc en mesure d'affiner les réglages relatifs à son site.

Notes personnelles

4.11.3- Protéger les fichier “.htaccess”

Le fichier lui-même ne devrait pas être téléchargeable.

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

Le téléchargement de tout fichier dont le nom commence par “.ht” est interdit.

4.11.4- Choisir les directives déléguées

La directive **AllowOverride** n'est valide que dans un contexte <**Directory**> .

En terme de sécurité, on préfère tout fermer par défaut, et n'ouvrir que ce qui doit l'être.

```
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>
```

À partir d'un point de l'arborescence, on décide d'offrir des pouvoirs

```
<Directory /home/sites/clients>
    AllowOverride Options Limit
</Directory>
```


Notes personnelles

4.11.5- Classes de directives

All : toutes les catégories ci-dessous
None : aucune (fichiers .htaccess ignorés)

AuthConfig : Authentification (login/password...)
Require, AuthName, AuthType, AuthUserFile, etc.

FileInfo : accès aux types de documents (*mod_mime, mod_negociation*)
AddEncoding, AddLanguage, AddType, DefaultType,ErrorDocument, LanguagePriority, …

Indexes : contrôle de l'indexation des répertoires
DirectoryIndex, et toutes les directives du module autoindex

Limit : contrôle d'accès (*basé sur adresse IP*)
Order, Allow from, Deny from, …

Options : fonctionnalités des répertoires (*mod_dir*, *mod_include*)
Options, XbitHAck, …

Ce que ".htaccess" ne fait pas
le tuning du serveur
le paramétrage réseau
le chargement des modules
la gestion des logs

Notes personnelles

4.12- Les logs du serveur apache

Chaque requête HTTP peut être tracée dans des fichiers ou des bases de données.

L'enregistrement de traces et leur conservation est une obligation et offre quelques possibilités :

générer des statistiques de fréquentation, analyser les parcours des visiteurs

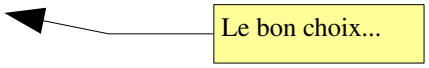
débogger la configuration, et les erreurs sur les sites (erreurs 40X notamment)

traquer les éventuelles intrusions ou tentatives d'intrusion

4.12.1- Logs et résolution de noms

La directive **HostnameLookups** (core) concerne de près traces et performances

Par défaut, on enregistre l'adresse IP du visiteur, sauf si on demande :
Une résolution reverse : **HostnameLookups On**
Une résolution « paranoïd » : **HostnameLookups double**
Par défaut : **HostnameLookups Off**



Pour une résolution « à postériori » des noms d'hôtes, voir la commande "**logresolve**".

Notes personnelles

4.12.2- Le traçage des erreurs

Le niveau de sévérité : celui à partir duquel on génère des **traces d'erreur**

```
LogLevel {debug | info | notice | warn | error | crit | alert | emerg}
```

Description des niveaux

Emerg : Urgences - le système est inutilisable
Alert : Une action doit être prise immédiatement
Crit : Conditions critiques
Error : Cas d'erreur classique
Warn : Avertissements (c'est le niveau de log par défaut)
Notice : Normal mais condition significative
Info : Pour information
Debug : Messages de débogage

La destination :
On peut choisir le chemin et le nom du fichier de log des erreurs.

```
ErrorLog fichier
```

ou

```
ErrorLog syslog:local6[8]
```

Le format des messages de log d'erreur est fixe.

8 Choisir une "facility" disponible... Si "LDAP" a choisi local4, alors pourquoi ne pas choisir "local6" pour "APACHE"

Notes personnelles

4.12.3- Les formats d'enregistrement des hits

Le module **mod_log_config** (compilé par défaut) permet de choisir entre deux approches :

1°/ Définir un nom de format qui sera utilisé par **CustomLog**
1/ On définit un (ou plusieurs) format(s)
2/ On utilise ceux que l'on souhaite en fonction du contexte.

```
# Avec un format prédéfini
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log common
```

On peut paramétrer plusieurs fichiers **CustomLog**.
On peut aussi utiliser **CustomLog** avec un format explicite.

```
# Avec un format explicite
CustomLog logs/volume-par-client_log "%h %b"
```

2°/ Définir le format à utiliser par la prochaine directive **TransferLog**
1 – On définit un (ou plusieurs) format(s) par **LogFormat**
2 – On sélectionne celui que l'on souhaite par **LogFormat NOM**
3 – On déclare un fichier à ce format par **TransferLog**

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\""
TransferLog logs/access_log
```


Notes personnelles

4.12.3.1- Formats standards

Le format le plus fréquent est « **common** », pris en compte par de nombreux outils d'analyse.

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
```

Avec HostnameLookup à « on »

```
montaigu.cert-ist.com - - [22/May/2009:16:10:18 +0200] "GET /formation/contenu/formation-securite-
linux-unix.pdf HTTP/1.1" 302 225
montaigu.cert-ist.com - - [22/May/2009:16:10:18 +0200] "GET /formation/ HTTP/1.1" 200 13080
```

Ce document est volumineux, transfert par morceaux

Avec HostnameLookup à « off »

```
195.101.175.25 - - [22/May/2009:16:16:36 +0200] "GET /Supports/UNIX/UNIX-UTILISATEUR-Mai-2009.pdf
HTTP/1.0" 200 95749 "-" "Mozilla/4.7 [en] (WinNT; I)"
195.101.175.25 - - [22/May/2009:16:16:38 +0200] "GET /Supports/UNIX/UNIX-UTILISATEUR-Mai-2009.pdf
HTTP/1.0" 206 1234 "-" "Mozilla/4.7 [en] (WinNT; I)"
195.101.175.25 - - [22/May/2009:16:16:39 +0200] "GET /Supports/UNIX/UNIX-UTILISATEUR-Mai-2009.pdf
HTTP/1.0" 206 79780 "-" "Mozilla/4.7 [en] (WinNT; I)"
```

C'est vraisemblablement le lecteur PDF du visiteur qui avait téléchargé une première portion, et a ensuite demandé deux fois une suite, en précisant l'offset de départ à chaque fois.

Notes personnelles

Le format « **combined** » permet de connaître les Référents ainsi que les UserAgent.

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
```

Exemple :

```
aneuilly-109-1-19-40.w81-53.abo.wanadoo.fr - - [18/May/2009:17:23:29 +0200] "GET /formation/
HTTP/1.1" 200 12994 "http://www.google.fr/search?q=formation+linux&ie=UTF-8&oe=UTF-8&hl=fr&meta="
"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
```

D'où vient le visiteur :

```
\"%{Referer}i\" ==>  http://www.google.fr/search?q=formation+linux
```

Son navigateur :

```
\"%{User-Agent}i\" : Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
```

Logguer les données POST ?

Par défaut, Apache ne loggue pas les données des requêtes POST. Mais, grâce aux modules suivants...

- Le module **mod_dumpio**.
- Le module **mod_log_forensic**.
- Le module **mod_security**.

Un lien sur le sujet : http://www.loggly.com/ultimate-guide/apache-logging-basics/

Notes personnelles

4.12.4- Marqueurs interprétés dans les formats

Liste des **marqueurs** utilisables dans la description d'un format de trace.

Marqueur	Description
%...a	adresse IP distante
%...A	adresse IP locale
%...B	octets envoyés sans les entêtes HTTP
%...B	octets reçus sans les entêtes HTTP, dans le format CLF (Common Log Format)
%...c	statut de la connexion une fois la réponse envoyée
%...{VAR}e	valeur de la variable d'environnement VAR
%...f	Nom de fichier
%...h	hôte distant
%...H	le protocole de la requête
%...{entête}i	le contenu de l'entête « entête : » dans la requête envoyée au serveur
%...l	nom de login distant si fourni par ident
%...m	méthode de la requête
%...{Foobar}n	la valeur d'une note Foobar d'un autre module
%...{Foobar}o	la valeur de l'en-tête « Foobar : » dans la réponse
%...p	le port canonique du serveur
%...P	PID du processus fils ayant traité la requête
%...q	paramètre d'URL s'il y en a une
%...r	première ligne de la requête
%...s	statut (originel si re-direction interne, %...>s renvoie le dernier statut)
%...t	date et heure au format CLF anglais
%...{format}t	date et heure au format spécifié. Voire strftime(3)
%...T	temps de traitement de la requête
%...u	utilisateur distant déclaré par mod_auth
%...U	URL demandée, sans argument d'interrogation
%...v	contenu du ServerName canonique servant la requête
%...V	Nom du serveur selon la valeur de la directive UseCanonicalName

Notes personnelles

Les «...» situés entre le signe « % » et la lettre peuvent être omis ou être remplacés par une condition pour laquelle il faut tracer.

Cette condition est constituée du code retour HTTP concerné.

On peut introduire plusieurs codes en les séparant par une virgule.

On peut introduire une négation par le signe « ! ».

Notes personnelles

4.12.5- Solutions pour analyser les logs

Apache-Scalp : https://code.google.com/p/apache-scalp/

Libre
Recherche les problèmes liés à la sécurité
S'appuie sur des filtres (expressions régulières)
Tente d'extraire les traces d'attaques potentielles
Ecrit en Python

Une base de filtres : https://github.com/PHPIDS/PHPIDS/blob/master/lib/IDS/default_filter.xml

Exemple : python scalp-0.4.py –l /var/log/httpd/access.log –f filter.xml –o output –html

⇒ La dernière version date de 2008.

WebForensik ⇒ Lorg : https://github.com/jensvoid/lorg

C'est un outil complet visant à faciliter la recherche de vulnérabilité d'une application après une attaque.

C'esdt une commande, écrite en PHP, qui produit des rapports (HTML, JSON, …) et peut consommer différents formats de fichiers de log (access_log).

Voir le Wiki : https://github.com/jensvoid/lorg/wiki

Notes personnelles

4.13- Gestion des connexions réseau

4.13.1- Timeout réseau

Timeout

```
Timeout 30
```

Nombre des **secondes** avant un timeout (envoi et réception)

Lors de la lecture de données en provenance du client, le temps maximum jusqu'à l'arrivée d'un paquet TCP si le tampon est vide.

Lors de l'écriture de données destinées au client, le temps maximum jusqu'à l'arrivée de l'accusé-réception d'un paquet si le tampon d'envoi est plein.

ProxyTimeout
En utilisation Proxy ou reverse Proxy, la directive **ProxyTimeout**, spécifie un temps de réponse maximal du serveur applicatif.

Au bout de ce timeout, le client est informé que le serveur applicatif n'a pas répondu dans les délais (HTTP-504)

Si **ProxyTimeout** n'est pas spécifiée, alors sa valeur par défaut est celle de **Timeout**.

Notes personnelles

4.13.2- Les requêtes persistantes

HTTP 1.1 propose de maintenir la connexion TCP après la réponse du serveur... dans l'attente d'une nouvelle requête.

⇒ Améliore les temps de réponse lors de requêtes successives… mais bloque un worker.

Keepalive

```
KeepAlive On
```

Il faut en plus que le client le demande (Connection : Keepalive)
Réduction des temps de latence, sur certaines pages mais blocage d'un worker

MaxKeepaliveRequests : Limiter le nombre de requêtes... donc libérer un worker

```
MaxKeepAliveRequests 100
```

Nombre maxi de requêtes sur une seule connexion persistante TCP
conserver un nombre assez haut pour les performances
0 pour illimité,

KeepaliveTimeout : Limiter le temps d'attente de la requête suivante

```
KeepAliveTimeout 15
```

La connexion réseau est fermée par le serveur au bout de ce délai.
Si ce paramètre est trop élevé, il peut nuire aux performances.

Notes personnelles

5- MPM et gestion de la charge

Notes personnelles

5.1- MPM : Modèles de gestion de la charge

5.1.1- Les MPMS

C'est le processus père (celui à l'identité de root) qui gère le nombre de ses fils, grâce à un **MPM**.

```
1368 root      /usr/sbin/httpd
1604 apache     \_ /usr/sbin/httpd
1605 apache     \_ /usr/sbin/httpd
1606 apache     \_ /usr/sbin/httpd
1607 apache     \_ /usr/sbin/httpd
1608 apache     \_ /usr/sbin/httpd
```

Sous Unix, chaque fils peut être **non threadé** (MPM-prefork) ou **multi-thread** (MPM-worker, MPM-event),

Dans **PREFORK**, 1 processus fils = 1 worker (slot).
Dans **WORKER**, 1 processus fils = N workers (slots)
Dans **EVENT,** 1 processus fils = N workers (slots)

Un worker ne peut traiter qu'une une requête à la fois.

5.1.2- Apache 2.4 : le MPM Event

Dans HTTP-Server 2.4, le **MPM EVENT** est "production ready"
C'est une approche similaire à Worker (Hybride multi-thread),
des threads sont dédiés à certaines tâches (threads de travail, threads d'écoute, ...)
c'est le même type d'approche que celle de NGINX.

C'est le MPM à utiliser aujourd'hui sur les systèmes modernes
Sa configuration est similaire à celle de Worker,
il ajoute une directive : **AsyncRequestWorkerFactor**

Notes personnelles

5.1.3- Principe de gestion de charge

Le rôle du processus "père" : anticiper les montées ou descente de charge…. "PRÉ"-"fork"...
Augmentation de charge : on doit augmenter le nombre de workers
Baisse de la charge : on peut réduire le nombre de workers

StartServers :
Au départ, un certain nombre de "workers" démarrent : **StartServers (prefork, worker, event)**

MaxRequestWorkers : (anciennement MaxClients)
C'est la limite maximale du nombre de workers.
Cela correspond dans prefork à des processus "fils", et dans worker/event, à des threads

Valeur par défaut :
Dans Prefork : 256 (le plafond est défini par **ServerLimit**)
Dans Worker/Event : 16 (**ServerLimit**) * 25 (**ThreadsPerChild**)

Au delà, les connexions sont mises en file d'attente (voir directive **ListenBacklog**)

MaxConnectionsPerChild :
On peut éliminer un worker au bout de X requêtes/connexions : **MaxConnectionsPerChild**
Avant, cette directive s'appelait **MaxRequestsPerChild**

Valeur par défaut : 0
Historiquement dû à des problèmes de mémoire de certaines librairies.

Notes personnelles

On doit le limiter dès que du code utilisé (PHP, SGBD,...) est douteux (fuite mémoire)

Notes personnelles

5.1.4- Paramètres spécifiques au MPM prefork

Maintenir au moins 10 workers "libres" (spare) en permanence (Si < : un processus child créé)

```
MinSpareServers 10 (prefork)
```

Si plus de 20 workers libres, arrêter les plus vieux (connus grâce à un *scoreboard géré par le père*)

```
MaxSpareServers 20 (prefork)
```

5.1.5- Paramètres spécifiques aux MPM worker & event

ThreadsPerChild : nombre de threads créés par chaque "child"

MinSpareThreads : nombre min de threads disponibles (Si inférieur : 1 processus "child" créé)

MaxSpareThreads : nombre max de threads disponibles (Si supérieur : 1 processus "child" tué)
Doit être supérieur à la somme "MinSpareThreads + ThreadsPerChild"

Pour pouvoir augmenter **MaxRequestWorkers**, on augmente le nombre de threads par processus
et le nombre de processus... en fonction des capacité mémoire du serveur.

Les paramètres **ServerLimit** et **ThreadLimit** sont des limites HARD :
nécessitent un arrêt / redémarrage d'Apache pour prise en compte.
La valeur **ServerLimit** * **ThreadLimit** doit être supérieure à **MaxRequestWorkers**.
La valeur de **ThreadLimit** doit être supérieure à **ThreadsPerChild**

Notes personnelles

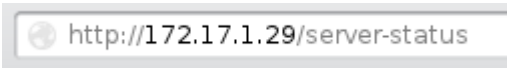
5.2- Contrôler la charge du serveur

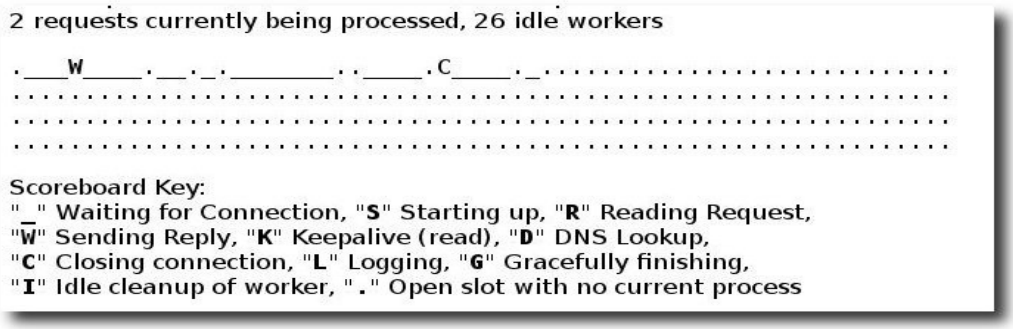
5.2.1- Le module "mod_status"

```
<Location /server-status>
    SetHandler server-status
    Order deny,allow
    Deny from all
    Allow from 172.17.
</Location>
```

Il offre un "Handler" (**server-status**) permettant de connaître l'état du serveur à un instant T.

Ce Handler est activé par l'appel à une URL de cette manière :





Le module gère les paramètres suivant :

"**auto**" : propose un affichage "Computer Readable"
http://mon-serveur/server-status/?auto

"**refresh=N**" : permet de spécifier (en secondes) le délai de rafraichissement automatique
http://mon-serveur/server-status/?refresh=5

Notes personnelles

5.2.2- Limitation de consommation CPU, Mémoire

On peut limiter la consommation de ressources système lors du traitement de requêtes par Apache.

```
RLimitCPU soft [hard]
```

Règle en secondes le temps CPU maximum accordé aux processus servant les requêtes

Si **soft** ou **hard** vaut "max", alors c'est la limite du système qui est prise en compte
Et la valeur "max" ne peut être atteinte que si Apache est lancé en tant que "root"
soft se rapporte à chaque processus,
hard se rapporte à l'ensemble des processus

Ce paramètre est exprimé en **secondes par processus**.
Par défaut : non positionné.

```
RlimitMEM soft [soft]
```

Similaire à RLimitCPU, pour une limite mémoire exprimée en octets par processus.

```
RlimitNPROC soft [hard]
```

Similaire à RLimitCPU, pour une limite du nombre de processus pouvant être lancés par les processus répondant aux requêtes.

Pour aller plus loin, voir aussi : **MaxMemFree** et **ThreadStackSize**

Notes personnelles

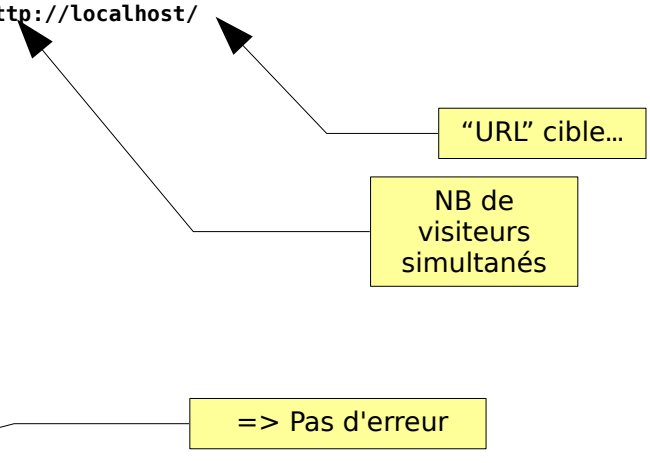
5.3- Outils de Benchmark : Ab

5.3.1- Apache Benchmark : Ab

Ab (Apache Behnchmark) est fourni avec Apache HTTP Server.

Outil de stress de serveur permettant des tests de charge brute contre un serveur HTTP.

```
# /usr/local/apache/bin/ab -n 1000 -c 10 http://localhost/
...
Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
...
Completed 900 requests
Finished 1000 requests
Server Software:        Apache/1.3.X
Server Hostname:        localhost
Server Port:            80

Document Path:          /
Document Length:        1456 bytes

Concurrency Level:      10
Time taken for tests:   2.823 seconds
Complete requests:      1000
Failed requests:        0
Broken pipe errors:     0
Total transferred:      1870869 bytes
HTML transferred:       1457456 bytes
```



Notes personnelles

Ab donne des moyennes en nombre de requêtes traitées à la seconde

```
Requests per second:    354.23 [#/sec] (mean)
Time per request:       28.23 [ms] (mean)
Time per request:       2.82 [ms] (mean, across all concurrent requests)
Transfer rate:          662.72 [Kbytes/sec] received
```

Ab affiche aussi des statistiques sur les temps de traitement :

Nb moyen de requêtes par seconde

```
Connnection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    5   11.3      3    92
Processing:    15   20   22.8     14   142
Waiting:        7   20   22.9     13   142
Total:         15   25   24.5     17   142

Percentage of the requests served within a certain time (ms)
  50%     17
  66%     17
  75%     18
  80%     18
  90%     27
  95%    100
  98%    102
  99%    105
 100%    142 (last request)
```

99% des requêtes sont servies en moins de 105 ms.

Lors d'un test de montée en charge on étudie l'**évolution** du **temps moyen** et du **temps le plus long** pour chaque requête.

Notes personnelles

5.3.2- Httperf & Autobench

Httperf (http://www.hpl.hp.com/research/linux/httperf/) a le même objectif que Ab, mais avec des fonctionalités complémentaires :

Scénario :

énoncer des requêtes différentes et indiquer un scénario de session
on peut dispatcher les requêtes de manière aléatoire sur une période donnée

Autobench (http://www.xenoclast.org/autobench/) est une surcouche de httperf, écrite en perl, pour automatiser la collecte de données avec httpperf.

Graphes :
Une représentation graphique de ces données est proposée par la commande **bench2graph**.

Notes personnelles

6- Apache HTTP Server : les bases de la sécurité

Notes personnelles

6.1- Le contrôle d'accès historique

On peut contrôler l'accès à une ressource en fonction
de l'adresse du client
de la présence d'une en-tête particulière.

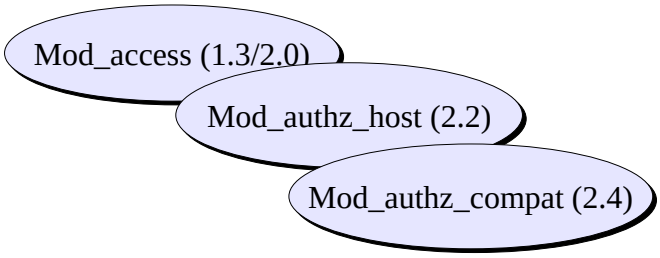
On s'appuie sur l'appartenance du client à une liste blanche (**Allow from**) une liste noire (**Deny from**).

On doit définir un ordre d'analyse des listes blanche puis noire ou noire puis blanche (**Order**).

Lorsque le client ne fait partie d'aucune des deux listes, alors une "décision par défaut" est appliquée.

Double sauf : On peut créer des exceptions, et les deux approches sont possibles :
Interdire tout client, sauf les membres de la liste blanche à moins d'être cité en liste noire
Autoriser tout client, sauf les membres de la liste noire à moins d'être cité en la liste blanche.

On doit donc paramétrer :
le contenu de la liste blanche : **Allow from**
le contenu de la liste noire : **Deny from**
l'ordre d'analyse des listes : **Order**
la Policy : **Order** aussi



En cas d'interdiction, c'est l'erreur **HTTP 403** (Forbidden) qui est renvoyée au client.

Notes personnelles

6.1.1- Gestion des listes d'adresses

Cela s'effectue jusqu'en version 2.2 incluse avec les directives :

Allow from : établir la liste blanche,
Deny from : établir la liste noire,

Les directive **Allow from** et **Deny from** prennent en argument une adresse (ou un début d'adresse) ou des noms de clients à filtrer.

On peut utiliser le mot clé **All** ou spécifier explicitement :

Un nom de domaine **.actilis.net**
L'adresse IP d'une machine
Un début d'adresse IP pour considérer toute adresse commençant par ce début : 192.168
Une adresse de réseau, avec précision du masque réseau

6.1.2- Paramétrage de l'ordre et de la Policy

La directive **Order** a deux valeurs possibles[9] :

Deny,Allow : Interdit si présent en liste noire, à moins d'être en liste blanche
Accès autorisé à tout client présent dans aucune des deux listes.

Allow,Deny : Autorisé si présent en liste blanche, à moins d'être en liste noire
Accès interdit à tout client présent dans aucune des deux listes.

C'est le second mot (Allow ou Deny) de l'argument qui définit la **policy** (le comportement par défaut).

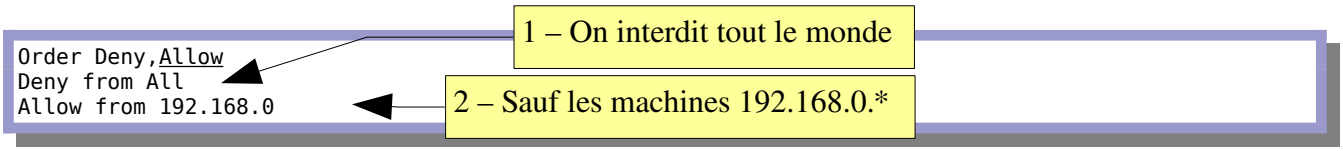
9 Ce sont des mots-clés, et il ne faut rien d'autre qu'une virgule entre le mot Allow et le mot Deny

Notes personnelles

Exemples

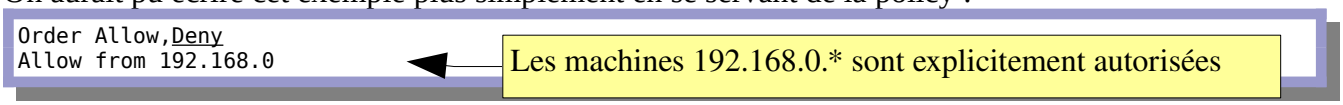
Autoriser seulement le LAN 192.168.0

"interdire toute machine, à moins qu'elle ne vienne du LAN"



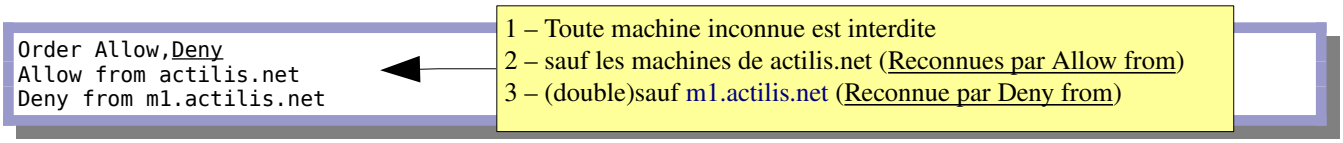
Toute machine qui fait partie de « **All** » est concernée par **Deny from**. La Policy ne joue donc jamais.

On aurait pu écrire cet exemple plus simplement en se servant de la policy :



L'accès est explicitement réservé aux membres du réseau 192.168.0. Il est interdit sinon.

Autoriser seulement les machines de **actilis.net** sauf **m1.actilis.net**.



Remarque : la dernière correspondance l'emporte, à l'opposé d'un firewall, où en principe la première correspondance prend la décision définitive.

Notes personnelles

6.1.3- Test de variables et module setenvif

La directives **Allow from** et **Deny from** acceptent une autre forme d'argument : env=VARIABLE
On peut alors s'appuyer sur la simple présence de variables d'environnement.

Le module **mod_setenvif** permet de définir des variables en fonction des en-têtes présentes dans la requête du client.

Les directives **BrowserMatch** et **SetEnvIf** ainsi que leurs équivalents case-insensitive (**BrowserMatchNoCase** et **SetEnvIfNoCase**) permettent d'écrire ce genre de contrôle d'accès :

```
SetEnvIf ENTETE  EXPRESSION-REGIULIERE VARIABLE
Order Deny,Allow
Deny from env=VARIABLE
```

Exemple...

```
/home/www/.htaccess:          SetEnvIf User-Agent ^Mozilla  nav_mozilla
                                      SetEnvIf User-Agent ^Wget     nav_wget

/home/www/firefox/.htaccess: Order Allow,Deny
                                      Allow from env=nav_mozilla

/home/www/wget/.htaccess:             Order Allow,Deny
                                      Allow from env=nav_wget
```


Notes personnelles

6.2- Le contrôle d'accès avec mod_authz_host

Depuis la version 2.4, la directive **Require** (module **mod_authz_host**) gère le contrôle d'accès.

En cas de conflit avec **mod_auth_compat**, c'est **mod_authz_host** qui l'emporte.

6.2.1- Principe

Mod_authz_host (2.4)

Le module **mod_authz_host** offre des fournisseurs d'autorisation :

ip : filtrage selon l'adresse IP (v4/v6), une partie de l'adresse, un réseau…
host : un nom d'hôte éventuellement partiel (double requete DNS : inverse puis directe)
forward-dns (≥ 2.4.19) : uniquement une requête directe. (fonctionne avec un "dyndns")
local : une adresse locale : 127.0.0.0/8 ou::1 ou adresse IP du serveur = celle du client

On les utilise avec la directive **Require**, dans des contextes de type répertoire (d+h).

```
Require ip 1.2.3.4
Require ip 1.2.3
Require ip 1.2.3.0/255.255.255.0
Require ip 1.2.3/24
Require host  actilis.net .fr
Require host client.actilis.net
Require forward-dns mypc.dyndns.org
```

Attention : lorsque l'on passe par un proxy, c'est bien l'adresse du proxy qui est soumise au contrôle.
Voir le module **mod_remoteip** pour une solution à ce "problème".

Notes personnelles

6.3- Compléments sur Require

6.3.1- Regroupement de directives Require

Le module **mod_authz_core** fournit des balises permettant de regrouper des directives **Require**.

Les 3 blocs peuvent s'imbriquer et sont disponibles dans les contextes Répertoire / htaccess :

<**RequireAll**> : Toutes les directives Require doivent satisfaire,
<**RequireAny**> : L'une d'entre-elles suffit,
<**RequireNone**> : Aucune ne doit satisfaire,

On peut aussi utiliser le mot clé "**not**" pour spécifier une négation.

6.3.2- Compatibilité avec mod_setenvif

On peut bien entendu utiliser **Require** avec un"Fournisseur" défini par **SetEnvIf**.

Exemple complet :

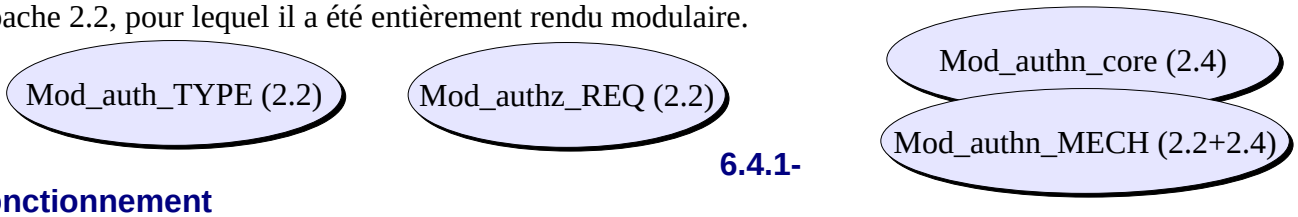
```
SetEnvIf User-Agent ^KnockKnock/2\.0 let_me_in

<RequireAny>
  <RequireAll>
    Require ip 192.168.0
    Require not ip 192.168.0.200
  </RequireAll>
  Require env let_me_in
</RequireAny>
```


Notes personnelles

6.4- L'authentification des utilisateurs

On peut soumettre les utilisateurs à une authentification pour restreindre les accès à un contexte de type répertoire. Le principe simple et universel est défini dans le protocole HTTP.

Il est implémenté par le module **mod_auth** (Apache 1.3/2.0) et par un ensemble de modules dans Apache 2.2, pour lequel il a été entièrement rendu modulaire.



6.4.1- Fonctionnement

Lorsqu'un navigateur entre dans le répertoire le serveur lui retourne une erreur **HTTP-401**.

Le navigateur présente alors une boite de dialogue à l'utilisateur qui doit alors s'authentifier. Les informations sont retournées au serveur qui accepte ou refuse l'accès en fonction de conditions à respecter (nom d'utilisateur, appartenance à un groupe, etc...).

Si le serveur refuse, celui-ci retourne une nouvelle erreur HTTP-401.

On peut mixer contrôle d'accès et authentification, grâce à la directive **Satisfy** :

Satisfy Any : la réussite d'un des deux contrôles est suffisante (Allow ou Require)
Satisfy All : la réussite des deux contrôles est nécessaire (Allow + Require)

Notes personnelles

6.4.2- Principe de Configuration

Dans le contexte de type répertoire où l'on souhaite déclarer l'authentification, on présente :

son **type** (Basic ou Digest), son **nom** (libre)
le **contrôle requis** (Require … …)
les **moyens de valider** le contrôle

6.4.2.1- AuthType : deux types d'authentification

None : il n'y a pas d'authentification

Basic : Le mot de passe n'est pas chiffré, mais encodé (base 64) et donc transmis en clair…

Digest : Serveur et client chiffrent une phrase négociée et c'est la forme chiffrée qui est transmise au serveur. Les résultats de ce chiffrement sont alors comparés.

Form : authentification basée sur un formulaire (et donc une gestion de session) : **Doc**.

6.4.2.2- Authname : le royaume

Le répertoire où se trouve l'authentification ainsi que toute sa descendance s'appelle un royaume, auquel on donne un nom par la directive **AuthName**.

Le nom du royaume apparaitra en titre de la boite de dialogue.

Notes personnelles

6.4.2.3- AuthBasicProvider / AuthDigestProvider : quelle est la base de données

On définit (depuis 2.2 uniquement) la manière dont Apache vérifie les informations requises
fichiers plats,
base de données,
ldap...

Chaque fournisseur fait l'objet d'un module qui l'implémente :

file : **mod_authn_file** : fichier plat ou **dbm** : **mod_authn_dbm** : base à plat

dbd : **mod_authn_dbd** : base de données SQL

ldap : **mod_authnz_ldap** : annuaire LDAP

… il faut ensuite paramétrer le provider... **AuthUserFile**, **AuthLDAP***, **AuthDBD*** …

Notes personnelles

Note

Performances
S'il y a beaucoup d'utilisateurs (milliers), les fichiers plats sont très lents à analyser. Il faut dans ce cas utiliser plutôt le module **mod_authn_dbm** (fourni en standard) :
- AuthDBMUserFile
- AuthDBMGroupFile

Ces deux fichiers sont maintenus grâce à la commande “dbmanage”.

Bases de données : DBD
Le module DBD et ses drivers (apr_dbd_mysql, apr_dbd_pgsql, …)
offrent un accès aux bases de données et supporte **mysql**, **oracle**, **pgsql**, **sqlite2**, **sqlite3**.

On spécifie simplement un driver (**DBDriver**),
ses paramètres (**DBDParams**, **DBDMin**, **DBDMax**, **DBDKeep**, **DBDExpTime**...),
puis la requête pour exécuter le contrôle (AuthDBDUserPWQuery)

6.4.2.4- Require : Le contrôle à effectuer

On requière un contrôle auquel le client est soumis (authentification simple, appartenance à un groupe...) par la directive **Require**. Jusqu'à Apache 2.2, on ne spécifie qu'une seule clause **Require**[10].

Valid-user : C'est le contrôle minimum, inclus dans les autres.
Il ne prend pas d'argument et vérifie que le nom et le mot de passe saisis sont corrects.
Il est implémenté par le module **mod_authz_user**.

User xxx yyy : vérifie Valid-User + vérifie que le login est un de ceux cités (xxx, yyy, ...).
Il est rarement utilisé car on lui préfère la notion de groupe, plus souple.
Il est implémenté par le module **mod_authz_user**.
Exemple : **Require User** fmicaux mghernati

Group xxx yyy : vérifie Valid-User + vérifie que le login est membre d'un groupe.
Pour le provider « DBD », qui s'appuie sur une requête SQL libre, ce contrôle est inutile
Pour LDAP, on utilise AuthLDAPGroupAttribute et AuthLDAPGroupAttributeIsDN
Dans le cas du provider « file », les groupes sont décrits dans un fichier texte
Son nom est défini par la directive **AuthGroupFile** (ou AuthLdapGroup*)
Ce contrôle est implémenté par le module **mod_authz_groupfile**.
Exemple : **Require Group** voile vtt canyon

D'autres contrôles existent et nécessitent d'autres modules (contrôle du propriétaire d'un fichier, contrôle basé sur une requête LDAP, SQL, ...)

10 Cela évolue dans Apache 2.4, qui permettra (**mod_authn_core**) de créer des groupes de contrôles et des exceptions avec possibilité de négation ou de validation d'expressions. Des nouveaux contextes <RequireAll>, <RequireAny>, <RequireNone> apparaissent.

Notes personnelles

Note

Require demande forcément un login et un mot de passe valide. Elle s'appuie donc forcément sur des fichiers de mots de passe (ou de groupe) qu'il est nécessaire d'indiquer dans le royaume.
- Il faut donc penser aux directives **AuthUserFile** et **AuthGroupFile** le cas échéant.
Ces fichiers sont gérés par “htpasswd” (user) et avec un éditeur de textes (group).

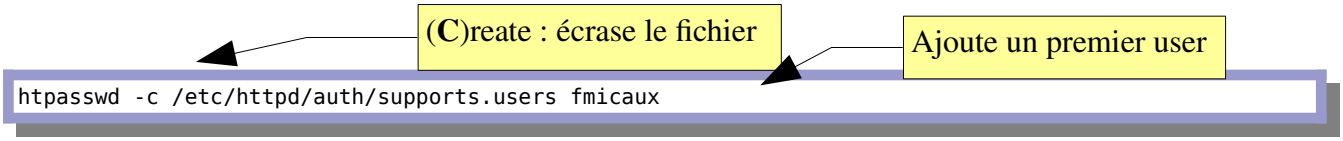
Cumuler authentification et contrôle d’accès.
La directive `Satisfy` (*core*) permet de définir si « une des » (`any`) ou « toutes (les deux)» (`all`) conditions doivent être respectées pour que l’accès soit autorisé.

6.4.3- Gestion des fichiers de mots passe et groupes

6.4.3.1- Le fichier des utilisateurs

Le fichier des utilisateurs doit contenir des couples sous la forme « login:password » Il s'agit d'utilisateurs distincts de ceux du système d'exploitation hôte.

Pour créer ce fichier on utilise la commande **htpasswd** fournie et compilée en même temps qu'Apache, de la manière suivante :



Cette commande crée dans le fichier des utilisateurs *supports.users* l'utilisateur *fmicaux*. La commande demande un mot de passe qu'il faut saisir deux fois.

L'option **-c** est nécessaire **uniquement** à la création du fichier. **Attention**, elel engendre un écrasement du fichier et les données qu'il contenait sont perdues.

Pour **Ajouter** un utilisateur supplémentaire : pas d'option « -c » !

```
htpasswd /etc/httpd/auth/supports.users user2
```


Notes personnelles

6.4.3.2- Le fichier des groupes

Le fichier des groupes est un simple fichier texte dont la syntaxe est :

```
voile: fanfan stef bobby mike kat val … …
karting: fanfan jeff stef kev … …
vtt: fanfan stef jeff pepette … …
canyon: fanfan stef jeff val … …
```

Une ligne par groupe,
Un espace entre deux utilisateurs,
Un utilisateur peut figurer dans plusieurs groupes.

Dans le cas de AuthType Digest :
On utilisera la commande **htdigest** pour manipuler le fichier des login/password.

Notes personnelles

7- Apache HTTP Server : les serveurs virtuels

Notes personnelles

7.1- Notion de serveur virtuel

Il s'agit de mutualiser un serveur, c'est à dire héberger plusieurs sites web répondant à des noms différents avec un seule instance de Apache HTTPD.

C'est ce qu'utilisent les hébergeurs sur leurs serveurs pour plusieurs raisons :
- Mutualisation des ressources
- Économie de coûts de maintenance

Des **serveurs virtuels** peuvent être déclarés de plusieurs manières différentes :

- Plusieurs adresses IP ou ports différents
- Une seule adresse IP, mais des ports différents

- Même adresse IP, même port, sites **distingués par le nom du site**
  - Dans ce cas, on parle d'**hôtes virtuels** basés sur les noms

À part celles concernant le serveur (Tuning, modules, Réseau), toutes les directives sont utilisables dans des hôtes virtuels. La documentation indique "Virtual Host" dans "Context" pour ces directives.

Les hôtes virtuels sont souvent configurés dans des fichiers séparés, introduits dans la configuration par des directives "include". Cela facilite la déclaration et la maintenance des hôtes définis.

Notes personnelles

7.2- Déclaration d'un hôte virtuel

7.2.1- La directive VirtualHost

Ou plutôt la directive **<VirtualHost>**, car il s'agit d'une déclaration de contexte.

```
<VirtualHost adresse[ :port] [adresse[ :port]] ...> … … … </VirtualHost>
```

Chaque contexte VirtualHost décrit **un hôte virtuel**. Toute directive valide dans les contexte « VirtualHost » peut être citée. Cela exclut le tuning de charge, le réseau...etc...

La déclaration commence par spécifier l'adresse IP (éventuellement le port) de hôte virtuel

Si aucun port n'est spécifié, c'est celui mentionné par la plus récente directive **Listen.**

On peut spécifier plusieurs adresses dans une balise **<VirtualHost>**.

Chaque couple adresse:port utilisé doit faire l'objet d'une directive **Listen**.

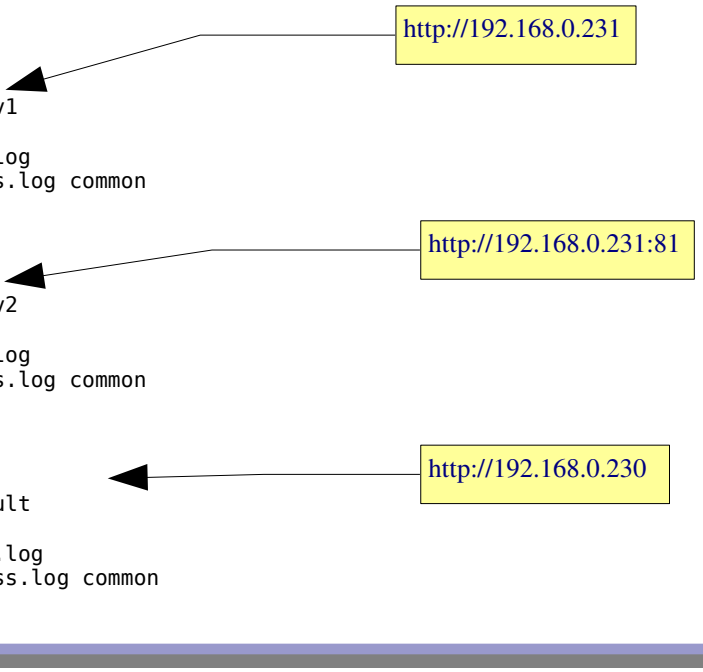
Le nom spécial **_default_** ou *****, caractère générique, correspond à toute adresse IP.

En l'absence de virtualhost **_default_** ou *****, toute requête est traitée par l'hôte « global » (celui décrit dans la configuration à l'extérieur des contextes VirtualHosts).

Notes personnelles

Exemple :

```
Listen 192.168.0.230
Listen 192.168.0.231:80
Listen 192.168.0.231:81

<VirtualHost 192.168.0.231:80>
        DocumentRoot /home/www/sitev1
...
        ErrorLog logs/sitev1-error.log
        CustomLog logs/sitev1-access.log common
...
</VirtualHost>

<VirtualHost 192.168.0.231:81>
        DocumentRoot /home/www/sitev2
...
        ErrorLog logs/sitev2-error.log
        CustomLog logs/sitev2-access.log common
...
</VirtualHost>

<VirtualHost _default_:*>
        DocumentRoot /home/www/default
...
        ErrorLog logs/default-error.log
        CustomLog logs/default-access.log common
...
</VirtualHost>
```



Dans la pratique, on déclare le site virtuel « par défaut » en tout premier.

Notes personnelles

7.2.2- Hôtes virtuels basés sur les noms

Il s'agit d'héberger plusieurs sites virtuels, répondant sur une même paire adresse:port. On les distingue par le nom du site demandé dans la requête (en-tête Host:)

Dans chaque contexte **VirtualHost**, c'est grâce à la directive **ServerName** (ou **ServerAlias**) qu'Apache détermine quel hôte virtuel correspond à celui demandé par le client :

> La directive **ServerName** ne peut être spécifiée qu'une seule fois par VirtualHost.
> On utilisera donc **ServerAlias** si plusieurs noms correspondent au même site.

À propos de NameVirtualHost...
Depuis la version 2.3.11, car la fonctionnalité est active pour chaque paire adresse:port indiquée dans une balise <VirtualHost>. Avant la version 2.3.11, on doit déclarer les couples "adresse:port" sur lesquels la fonctionnalité est gérée.

```
NameVirtualHost {adresse_IP[ :port] | *}
```

Chaque **VirtualHost** doit (avant 2.3.11) être déclaré avec strictement le même libellé que la directive NameVirtualHost qui le concerne. Dans Apache 2.4, NameVirtualHost n'a plus aucun effet.

Exemples :

```
NameVirtualHost srv-web1:80 ==> <VirtualHost srv-web1:80 >... </VirtualHost>
NameVirtualHost *                  ==> <VirtualHost *> …. </VirtualHost>
```


Notes personnelles

Exemples complet

Toutes les adresses IP du serveur sont servies par des hôtes virtuels.

Le serveur « par défaut » (celui en dehors des contextes VirtualHost) ne recevra jamais de requête.

Toute requête ne correspondant à aucun VirtualHost sera dirigée vers le premier déclaré.

```
Listen 80
NameVirtualHost *

<VirtualHost *>
        DocumentRoot /data/www/www.actilis.net/html
        ServerName  www.actilis.net
        ServerAlias     actilis.net
        ServerAdmin webmaster@actilis.net
...
        ErrorLog logs/actilis.net-error.log
        CustomLog logs/ actilis.net-access.log common
...
</VirtualHost>
```


Notes personnelles

Note

Notes sur les problèmes de DNS

Tout nom utilisé pour un site virtuel doit être connu par la machine au moment où Apache démarre.

Apache vérifie au moment où il démarre que chaque nom de VirtualHost pointe bien vers lui.
Dans le cas où Apache ne se connaît pas par le nom, il désactive cet hôte virtuel
Les autres peuvent toutefois continuer à fonctionner.

Un script de création automatique, utilisant un template

```
Appel : ./newsite.sh site1
```

```
#!/bin/bash

# Constantes
REP_SITES=/home/sites
REP_VH=/etc/httpd/conf/vhosts.d

# Tests
[ $# -ne 1 ] && echo "Usage: $0 nom-du-site" && exit 1
SITE=$1

[ -d ${REP_SITES}/${SITE} ] && echo "Site déjà existant dans ${REP_SITES}" && exit 1
[ -f ${REP_VH}/${SITE} ]  && echo "Site déjà déclaré dans ${REP_VH}" && exit 1


# Création du site et de son répertoire de données
sed "s/_SITENAME_/${SITE}/g" ${REP_VH}/00-template > ${REP_VH}/${SITE}.conf
mkdir -p ${REP_SITES}/${SITE}
mkdir -p ${REP_SITES}/logs


# Déclaration dans le DNS
echo -e "update delete ${SITE}.dom0.fr. 60 A 192.168.3.100 \nsend" |nsupdate
echo -e "update add ${SITE}.dom0.fr. 60 A 192.168.3.100 \nsend"  |nsupdate

# Reload d'Apache
apachectl graceful
```

Ce script procède à des contrôles, puis substitutions de SITENAME par le nom de site demandé, puis fait la déclaration DNS.

Notes personnelles

Le template :

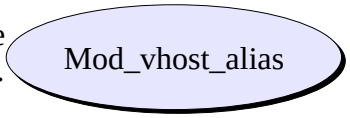
```
<VirtualHost *:80>
  ServerName _SITENAME_.dom0.fr

  DocumentRoot /home/sites/_SITENAME_
  <Directory /home/sites/_SITENAME_>
      # Permissions de base
      Order Allow,Deny
      Allow from All

      # Htaccess
      AllowOverride All

      # Exploration
      DirectoryIndex index.html index.php index.cgi
      Options None
      ErrorDocument 403 "Site _SITENAME_ en maintenance"

  </Directory>

  # Pas de ErrorLog, pour rester sur l'approche "Syslog" héritée
  CustomLog /home/sites/logs/_SITENAME_.access_log combined

</VirtualHost>
```


Notes personnelles

7.2.3- Module mod_vhost_alias

Le but est de simplifier le déploiement d'hôtes virtuels au maximum, tout en évitant le rechargement de la configuration pour déployer de nouveaux hôtes.

Le module **mod_vhost_alias** propose des directives permettant de matérialiser les noms de sites demandés grâce à des marqueurs génériques. On ne déclare qu'une fois pour tous les sites le chemin d'accès génériques.



Créer un hôte virtuel revient alors à :
- créer le répertoire contenant ses données.
- déclarer chaque nouvel hôte du point de vue DNS

Directives :
VirtualDocumentRoot : similaire à DocumentRoot.
VirtualScriptAlias : similaire à ScriptAlias.
VirtualDocumentRootIP et **VirtualScriptAliasIP**, similaires mais s'appuient sur l'adresse IP.

Exemples :

```
UseCanonicalName Off
VirtualDocumentRoot /www/vhosts/%0

http://www.dom1.fr/index.html       ==> /www/vhosts/www.dom1.fr/index.html
```


Notes personnelles

Utilisation des marqueurs :

%0 : tout le nom
%1 : premier mot, %2 le second,
%X+ : du Xième à la fin...
%X.Y : les Y premières lettres du Xième mot...

```
UseCanonicalName Off
VirtualDocumentRoot /www/vhosts/%3+/%2.1/%2

http://www.actilis.net/index.html   ==> /www/vhosts/net/a/actilis/index.html
http://www.actilis.info/index.html  ==> /www/vhosts/info/a/actilis/index.html
http://www.actilis.fr/index.html    ==> /www/vhosts/fr/a/actilis/index.html
http://www.bateau.fr/index.html     ==> /www/vhosts/fr/b/bateau/index.html
http://www.avoile.fr/index.html     ==> /www/vhosts/fr/a/avoile/index.html
http://www.v-l-m.org/index.html             ==> /www/vhosts/org/v/v-l-m/index.html
```


Notes personnelles

Note

Virtual hosting massif et traces

Au niveau des logs, il faut utiliser les marqueurs %V (ou %A) si l'on souhaite dans les traces conserver le nom du site contacté (%V) où l'adresse IP (%A).

Ceci résoud le problème lié à l'utilisation obligatoire d'un seul fichier de log.

7.3- HTTPS et le module mod_ssl

HTTPS : encapsuler du trafic HTTP dans un tunnel SSL & TLS.

Le module **mod_ssl** est l'interface à la bibliothèque OpenSSL permettant à Apache d'effectuer un chiffrement fort s'appuyant sur SSL & TLS.

Documentation : http://httpd.apache.org/docs/2.4/ssl/ssl_howto.html

7.3.1.1- Installation par le package

Le module est **fourni en standard** dans les distributions 2.X d'Apache

Il est installé avec le package "apache2" sous Debian.

Le package s'appelle **mod_ssl** sous CentOS et doit être installé en plus de "httpd".

7.3.1.2- Si on procède par compilation

Lors de la configuration (./configure...), s'assurer que l'option "**--enable-ssl**" est activée.

Pour compiler le module, le package openssl-devel (ou libssl-dev) est nécessaire.

Si les bibliothèques et includes OpenSSL ne sont pas installées dans les répertoires standards, préciser leur emplacement grâce à l'option ""**--with-ssl=**/chemin/alternatif".

À l'installation, le module "**mod_ssl.so**" est déposé dans le répertoire des modules.

Notes personnelles

7.4- Certificat

Pour mettre en œuvre un site **HTTPS**, il faut un **certificat** :

Document dans lequel des informations textuelles sont associées à une clé publique.
Signé par une autorité de confiance qui a pris la peine de vérifier les informations.
Le certificat est inutilisable sans la clé privée qui va avec.

7.4.1- Obtenir un certificat

On peut **le créer soi-même**

Celui-ci peut être auto-signé ou signé par un certificat tiers de confiance,
On peut lui donner une durée de validité arbitraire (1 mois, 1 an, 10 ans…),
Reconnaissance de l'AC par les clients… à prévoir.

On peut **l'acheter auprès d'une AC reconnue**

Plusieurs fournisseurs proposent des prix à partir de 10€ /an,
Beaucoup plus cher pour un certificat valide pour plusieurs noms,
Il faut le renouveler annuellement et installer le nouveau sur le serveur,
Reconnaissance en principe implicite de l'AC par les clients.

Let's Encrypt permet d'**obtenir gratuitement des certificats en principe reconnus**

La durée de validité est de 3 mois,
La procédure de renouvellement est automatisable.

Notes personnelles

7.4.2- Démarche de création d'un certificat

7.4.2.1- Générer une clé privée

À réaliser une fois, la clé privée doit être protégée et stockée en lieu sûr.

7.4.2.2- Générer une CSR

À réaliser pour chaque demande de renouvellement du certificat[11].

7.4.2.3- Signature du certificat

Il peut être auto-signé

C'est un certificat signé par la clé privée du demandeur. C'est le moyen le plus gratuit… mais il faut réserver ces certificats à des tests, ou informer/éduquer les utilisateurs (si on les connaît)…

Ce genre de certificat engendre systématiquement l'affichage d'un message rendant peu crédible le site, effrayant l'utilisateur, ou mieux, qu'il prend l'habitude d'ignorer… jusqu'au jour où il ignorera encore, mais sur un site réellement falsifié.

Il peut être signé par un tiers

C'est un certificat signé par la clé privée d'une autorité de certification, si possible reconnue.

Si on est maître du déploiement des navigateurs, on peut disposer de sa propre CA et faire reconnaître son certificat par les navigateurs.
Dans le cas contraire, on s'appuie sur un tiers déjà reconnu : let's encrypt ?

11 On peut demander la création de la clé privée lors de la génération de la CSR (Certificate Signing Request).

Notes personnelles

7.5- Mise en œuvre d'un hôte virtuel HTTPS

1/ **Module SSL**
On vérifie que le module **mod_ssl** est chargé : **LoadModule ssl_module modules/mod_ssl.so**

2/ **Port d'écoute**
On vérifie que le serveur écoute sur le port tcp/443 : **Listen 443**

3/ **VirtualHost avec SSLEngine**
On déclare un **VirtualHost** pour lequel SSL est activé :

```
SSLEngine On
```

4/ **Renseigner les documents**
On renseigne le certificat du serveur, **ET** la clé privée : **SSLCertificateFile** et **SSLCertificateKeyFile.**

```
SSLCertificateFile    /etc/ssl/certs/webserver.crt
SSLCertificateKeyFile /etc/ssl/private/webserver.key
```

Ces deux éléments peuvent être réunis dans un même fichier, par concaténation. Ce fichier doit être protégé par des permissions Unix fermées.

On utilise alors uniquement **SSLCertificateFile** (plus besoin de CertificateKeyFile)

Notes personnelles

5/ **Créer les documents**

Les répertoires :

```
# mkdir -p /etc/ssl/{private,certs}
```

On peut signer le **certificat** (avec la clé-privée créée au passage) : certificat autosigné

```
# openssl req -x509 -newkey rsa:2048 -keyout /etc/ssl/private/webserver.key \
                  -nodes           -out /etc/ssl/certs/webserver.crt
Generating a 2048 bit RSA private key
.........................................+++
.................+++
writing new private key to '/etc/ssl/private/webserver.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:FR
State or Province Name (full name) []:Bretagne
Locality Name (eg, city) [Default City]:Ploemeur
Organization Name (eg, company) [Default Company Ltd]:ACTILIS
Organizational Unit Name (eg, section) []:Hosting
Common Name (eg, your name or your server's hostname) []:example.actilis.net
Email Address []:admin@actilis.net
```

⇒ Sans l'option "**-x509**", c'est une CSR qui est créée. Il n'y a plus qu'à la transmettre à une CA pour qu'elle nous la signe et nous rende un certificat signé (CRT).

Attention au **CommonName**… c'est le nom du site !

Notes personnelles

7.6- Compléments sur les certificats

Création non interactive d'un certificat autosigné

```
# openssl req -x509 -newkey rsa:2048 -keyout /etc/ssl/private/webserver.key \
                    -nodes           -out /etc/ssl/certs/webserver.crt \
 -subj /countryName=FR/stateOrProvinceName=Bretagne/organizationName=Actilis/CN=example.actilis.net/
```

Il s'agit du même certificat que précédemment, mais créé sans saisie des éléments du sujets.

Vérifications…

```
# openssl x509 -text -noout < /etc/ssl/certs/webserver.crt
```

```
        Subject: C=FR, ST=Bretagne, O=Actilis, CN=example.actilis.net
…
            X509v3 Basic Constraints:
                CA:TRUE
```

On remarque que ce certificat a le "droit" d'en signer d'autres. En principe, ce n'est pas le cas pour un certificat de serveur web ou de client (messagerie, authentification…).

On réserve la contrainte "**CA:TRUE**" aux certificats d'autorité de certification.

Notes personnelles

Un certificat valide seulement pour un serveur web, signé par un tiers (CA)

Cela nécessite des options de la commande "**x509**" que n'admet pas la commande "**req**".

On doit donc partir d'une **CSR** (pas d'option "x509" lors de la commande précédente).

```
# openssl req      -newkey rsa:2048 -keyout /etc/ssl/private/webserver.key \
                   -nodes           -out /etc/ssl/certs/webserver.csr \
-subj /countryName=FR/stateOrProvinceName=Bretagne/organizationName=Actilis/CN=example.actilis.net/
```

On crée ensuite un fichier contenant les contraintes & extensions du certificat

```
cat >> /etc/ssl/ext-serveur
basicConstraints=CA:FALSE
keyUsage=digitalSignature,keyEncipherment,dataEncipherment,keyAgreement
extendedKeyUsage=serverAuth
<CRTL-D>
```

On signe ensuite avec la commande "**x509**", en appliquant les extensions du fichier "ext-serveur"

```
# openssl x509 -req -days 30 -in /etc/ssl/certs/webserver.csr -extfile /etc/ssl/ext-serveur \
               -CA /etc/ssl/certs/CA.crt -CAKey /etc/ssl/private/CA.key  \
               -out /etc/ssl/certs/webserver.crt
Signature ok
subject=/C=FR/ST=Bretagne/O=Actilis/CN=example.actilis.net
Getting Private key
```

Cette fois-ci, une vérification doit montrer que la contrainte CA est à FALSE :

```
            X509v3 Basic Constraints:
                CA:FALSE
```


Notes personnelles

Un certificat valable pour plusieurs sites

On ne peut pas utiliser plusieurs certificats sur un même couple adresse-IP/port.

Sur un serveur hébergeant plusieurs sites HTTPS, il faut donc que tous les noms de sites soient validés par le seul certificat qu'on présentera.

Il faut donc un certificat doté de champs "subject" alternatifs… on ajoute tout simplement au fichier "ext-serveur" :

```
subjectAltName=DNS:exemple.actilis.net,DNS:wwexample.actilis.net
```

Puis on relance la signature :

```
# openssl x509 -req -days 30 -in /etc/ssl/certs/webserver.csr -extfile /etc/ssl/ext-serveur \
                -signkey /etc/ssl/private/webserver.key  -out /etc/ssl/certs/webserver.crt
Signature ok
subject=/C=FR/ST=Bretagne/O=Actilis/CN=example.actilis.net
Getting Private key
```


Notes personnelles

Vérifications…

```
# openssl x509 -text -noout < /etc/ssl/certs/webserver.crt
```

Repérer notamment les blocs suivants…

```
        Subject: C=FR, ST=Bretagne, O=Actilis, CN=example.actilis.net
```

Et...

```
…
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage:
                Digital Signature, Key Encipherment, Data Encipherment, Key Agreement
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 Subject Alternative Name:
                DNS:exemple.actilis.net, DNS:wwexample.actilis.net
…
```


Notes personnelles

8- Apache HTTP Server : Contenus dynamiques

Notes personnelles

8.1- CGI

8.1.1- Principe de CGI

Le client envoie une demande au serveur qui va lui retourner du contenu HTML. Le code HTML peut en fait **être généré juste avant d'être envoyé** au client, par un programme s'exécutant sur le serveur.

Mod_cgi (mod_cgid)

L'interface permettant d'exécuter un programme sur le serveur sur appel d'une URL est appelée **CGI** (**C**ommon **G**ateway **I**nterface). On parle aussi parfois de ***programme CGI***, pour désigner l'exécutable que l'on appelle par l'interface CGI. CGI est l'ancêtre des méthodes de développement de sites dynamiques d'aujourd'hui.

Les programmes exécutables par **CGIs** doivent être placés dans un répertoire autorisant leur exécution. Cela permet à l'administrateur du serveur web de contrôler un peu le code exécuté sur son serveur. Ce répertoire devra donc posséder l'option **ExecCGI**.

8.1.2- Programmer en CGI...

CGI n'est pas un langage, mais **une interface...** dont le rôle est de **définir la façon dont le serveur HTTP (Apache) va passer les différents arguments** au programme dit « CGI ».

Les langages les utilisés pour la réalisation de programmes CGI sont :

interprétés ou compilés à la volée : les scripts Shell,, Perl, PHP (en mode CGI)

compilés à l'avance : C, ...Tout langage produisant un exécutable peut être utilisé.

Notes personnelles

8.1.3- Quelques bases

Un appel CGI se passe plus ou moins comme un lancement de programme sur la ligne de commande du système.

Pour tester le résultat d'un programme CGI, on peut capturer sa sortie standard et l'ouvrir comme un fichier dans le navigateur.

La programmation CGI dépasse largement le cadre de ce cours, mais ces quelques éléments sont importants :

- le contenu envoyé par CGI est celui émis sur la sortie standard du "programme CGI"
- un "programme CGI" doit commencer par émettre des en-têtes, soit au moins :
  - L'en-tête Content-Type: , pour indiquer le type de contenu à suivre
  - Une ligne vide, pour signifier la fin des en-têtes
- la sortie d'erreur n'est pas transmise

Attention, si on fait appel à une "un CGI" alors que le serveur n'est pas paramétré pour exécuter c'est le contenu demandé qui est envoyé, sans interprétation ni exécution !

Notes personnelles

Note

Problème classique avec CGI :

Il faut absolument charger ou activer le module **mod_cgi**. Sinon, c'est le code source qui est envoyé au client !

8.1.3.1- Utilisation des scripts CGI

Tout script **CGI** est un processus "fils" du processus **httpd** qui le lance:

Il a **la même identité** : **ses droits** sont donc ceux de l'utilisateur qui a lancé **httpd**.
Il doit avoir le **droit système d'exécution** sur les scripts **CGI**, sinon : Erreur 403
Un plantage d'un script conduit parfois à une erreur du type **500**… difficile à diagnostiquer.

Lire les fichiers **error_log**, aide à comprendre les problèmes de permission ou d'identité.

Débugger un script CGI est plus facile sans passer par Apache, en jouant sur des redirections... pour vérifier le code généré.

8.1.3.2- Variables et environnement

Les **variables postées** par un formulaire sont reçues sous forme de **variables d'environnement**. On peut **tester un script** sensé être appelé depuis un formulaire en **le lançant en mode ligne de commande en initialisant au préalable les variables** d'environnement nécessaires.

Mod_env

Le module **mod_env** permet spécifier des variables d'environnement pour les scripts CGI exécutés coté serveur. Cela permet de gérer un contexte applicatif coté serveur.

```
<Location /cgi-bin>
      SetEnv PATH /bin-pour-les-cgi
</Location>
```


Notes personnelles

Exemple de script shell appelable par CGI :

Problème de SECURITE (exécution de code arbitraire)

```
#!/bin/sh
echo Content-type: text/html
echo ""
echo "<HEAD><TITLE>Test CGI en Shell</TITLE></HEAD>"
echo "<H1>Essai CGI</H1>"
echo "<H2>Quelques variables d'environnement du serveur</H2>"
echo "<P>"
echo SERVER_SOFTWARE = $SERVER_SOFTWARE         "<BR>"
echo SERVER_NAME = $SERVER_NAME                 "<BR>"
echo SERVER_PROTOCOL = $SERVER_PROTOCOL         "<BR>"
echo SERVER_PORT = $SERVER_PORT                 "<BR>"
echo REQUEST_METHOD = $REQUEST_METHOD           "<BR>"
echo SCRIPT_NAME = "$SCRIPT_NAME"               "<BR>"
echo QUERY_STRING = "$QUERY_STRING"             "<BR>"

echo "<HR>"
echo "<H2>Lancement de la commande donn&eacute;e en argument dans l'URL</H2>"

# Remise en forme, (les espaces ont disparu)
QUERY_STRING=`echo $QUERY_STRING | sed 's/%20/ /g'`

echo "Appel de la commande : <B>$QUERY_STRING</B> <BR><BR>"
echo "<code>"
# On introduit les sauts de ligne, nécessaires en HTML
$QUERY_STRING | sed 's/$/<BR>/'
echo "</code>"
echo "</P>"
```


Notes personnelles

8.1.4- SuEXEC et l'identité des processus

Du point de vue **sécurité**, les serveurs qui exécutent des scripts CGI sont plus vulnérables, par le contenu même du script et les actions qu'il est supposé réaliser.

Il est possible d'utiliser un "wrapper", une enveloppe pour le script, qui permet de modifier son comportement, ses variables d'environnement et de le faire fonctionner **comme s'il avait été invoqué par un autre utilisateur** avec des droits différents.

suEXEC diminue les risques simplement en modifiant les droits accordés aux scripts CGI, et sera lancé à chaque fois qu'une requête faisant appel à un programme CGI est reçue par Apache.

Mise en oeuvre
Il faut recompiler Apache en ajoutant le support de suexec. La fin de l'affichage de ./configure --help donne les informations suivantes :

```
suEXEC options:
 --enable-suexec        enable the suEXEC feature
 --suexec-caller=NAME   set the suEXEC username of the allowed caller [www]
 --suexec-docroot=DIR   set the suEXEC root directory [PREFIX/share/htdocs]
 --suexec-logfile=FILE  set the suEXEC logfile [PREFIX/var/log/suexec_log]
 --suexec-userdir=DIR   set the suEXEC user subdirectory [public_html]
 --suexec-uidmin=UID    set the suEXEC minimal allowed UID [100]
 --suexec-gidmin=GID    set the suEXEC minimal allowed GID [100]
 --suexec-safepath=PATH set the suEXEC safe PATH [/usr/local/bin:/usr/bin:/bin]
 --suexec-umask=UMASK   set the umask for the suEXEC'd script [server's umask]
```


Notes personnelles

Dès que l'on active « enable-suexec », il faut combiner au moins une des autres options liées. Le message d'information dans le cas contraire est amusant.

On peut par exemple utiliser :

```
# ./config.nice --enable-suexec \
--suexec-caller=apache --suexec-logfile=logs/suexec.log \
--suexec-safepath=/usr/local/bin:/usr/bin:/bin \
--suexec-docroot=/home/scripts-valides
```

Dans le cas présent, il faudra respecter les points suivants :

L'identité du serveur Apache (l'appelant) : User Apache

Les commandes utilisées par des scripts doivent uniquement être celles du PATH sécurisé.

L'utilisateur Unix cible (défini dans le VirtualHost par la directive User) devra au minimum posséder un UID>100

Idem pour le groupe si l'on le précise par « Group NOM »

Les scripts utilisés devront forcément être dans un sous-répertoire de /home/scripts-valides

Les informations liées aux erreurs sont dans le fichier suexec.log.

Et il y en aura forcément lors de la première mise en place, mais toutes sont toujours très clairement expliquées (droits, identité, fichiers non exécutables, etc...)

Notes personnelles

9- PHP-FPM

Notes personnelles

9.1- HTTPD & PHP : plusieurs approches

9.1.1- PHP en mode Handler

Historiquement, le branchement PHP-Apache passe par un **DSO** fournissant un **Handler** :

C'est la manière annoncée comme la plus performante historiquement, celle par défaut.

Dans Apache, un Handler (ou un gestionnaire Mime) affecte à PHP le traitement des fichiers

```
AddHandler php5-script .php
# ou
# AddType application/x-httpd-php .php
```

Problèmes :

C'est Apache qui charge un module PHP…

Ce n'est pas compatible avec les MPM multi-thread[12].

Cela ne permet pas d'avoir des identités PHP différentes sur un serveur mutualisé.
User / Group de Apache sont donc les mêmes pour tous les sites servis.

12 Depuis Apache-HTTPD 2.4, "**event**" est surement le bon choix.. et c'est celui par défaut !.

Notes personnelles

9.1.2- PHP en mode CGI

PHP peut être appelé via un mécanisme **CGI** et le module **Action**.

C'est un un exécutable externe, donc compatible avec les MPM Threadés.

On peut instancier des Handlers grâce au module « Action » et « SetHandler ».

```
ScriptAlias  /php-cgi/  /data/www/phpversions.actilis.fr/
Action       PHP4       /php-cgi/php
<Files "info-44.php">
       SetHandler   PHP4
</Files>
```

Problème :

À chaque appel, il faut lancer un exécutable : c'est lent !

Notes personnelles

9.1.3- PHP-FPM : FastCGI Process Monitor

Depuis la version 5.3, PHP propose un « serveur PHP » : **PHP-FPM** : **FastCGI Process Monitor**

C'est la seule approche qu'NGINX utilise nativement.
Le lien (Apache) HTTPD ⇒ PHP-FPM se fait via le module **mod_proxy_fcgi**

```
<FilesMatch "\.php$">
        SetHandler  "proxy:fcgi://127.0.7.3:9001"
</FilesMatch>
```

Le serveur HTTP **n'exécute pas de code PHP** mais est juste un proxy "Fast-CGI" : **stabilité**.

C'est **plus scalable :** 1 HTTPD devant plusieurs pools PHP-FPM,

PHP-FPM propose des **mécanismes de gestion de charge** similaires à ceux de HTTPD
donc d'entretien de la mémoire, du nombre de processus...

PHP-FPM offre des possibilités intéressantes en terme logs, d'environnement, et de sécurité :
Une **identité** peut être affectée à chaque Pool,… un site = un user
Un **contrôle d'accès** à la socket peut être mis en place.

PHP-FPM permet une configuration "php.ini" (php_value, php_admin_value…) par pool.

Cette méthode autorise aisément des **versions différentes de PHP** sur un même serveur,

Notes personnelles

9.2- Configuration de PHP-FPM

9.2.1- Structure de la configuration

Contenu du package (php-fpm / CentOS 7) :

```
# rpm -ql php-fpm
/etc/logrotate.d/php-fpm
/etc/php-fpm.conf
/etc/php-fpm.d
/etc/php-fpm.d/www.conf
/etc/sysconfig/php-fpm
/run/php-fpm
/usr/lib/systemd/system/php-fpm.service
/usr/lib/tmpfiles.d/php-fpm.conf
/usr/sbin/php-fpm
/usr/share/doc/php-fpm-5.4.16
/usr/share/doc/php-fpm-5.4.16/fpm_LICENSE
/usr/share/doc/php-fpm-5.4.16/php-fpm.conf.default
/usr/share/fpm
/usr/share/fpm/status.html
/usr/share/man/man8/php-fpm.8.gz
/var/log/php-fpm
```

Le point d'entrée est **/etc/php-fpm.conf**,

On y trouve la configuration générale

Et un include pour le répertoire de définition des pools : **/etc/php-fpm.d**.

```
# grep -n ^include /etc/php-fpm.conf
11:include=/etc/php-fpm.d/*.conf
```


Notes personnelles

9.2.2- Fichier php-fpm.conf : configuration générale

La section **[Global]** :

On y définit les paramètres généraux du serveur :
- Gestion des logs et pid-file
- Gestion des limites (nombre de processus, mémoire, open files...)
- Gestion du démarrage / intégration Systemd

Les valeurs par défaut conviennent et les seules directives non commentées sont les suivantes :

```
[global]
pid = /run/php-fpm/php-fpm.pid
error_log = /var/log/php-fpm/error.log
daemonize = no
```

daemonize=no est importante dans le cas d'une intégration à SystemD.

9.2.3- Les paramètres de PHP.INI

Ils ne sont pas fournis par **php-fpm**, mais par le package **php-common**, qui est en dépendance.

Tous les paramètres de **/etc/php.ini** sont donc pris en compte par PHP-FPM,
- On peut les surcharger dans chaque Pool.

Notes personnelles

9.3- Définition de pools PHP-FPM

9.3.1- Un fichier par pool

Chaque Pool devrait faire l'objet d'un fichier qui le défini. Exemple : /etc/php-fpm.d/www.conf

```
[POOL1]
listen = 127.0.0.1:9001
listen.allowed_clients = 127.0.0.1
user = user1
group = apache

#PM : Pool Management, similaire à la gestion de charge d'Apache
pm = dynamic
pm.max_children = 50
pm.start_servers = 5
pm.min_spare_servers = 5
pm.max_spare_servers = 35
slowlog = /var/log/php-fpm/pool1-slow.log

php_admin_value[error_log] = /var/log/php-fpm/pool1-error.log
php_admin_flag[log_errors] = on
php_value[session.save_handler] = files
php_value[session.save_path] = /var/lib/php/session-pool1
```

Ici, on reconnaît des classes de paramètres :

de sécurité : Socket et contrôle d'accès, Identité
de gestion de la charge : **pm**.*
de configuration de l'interpréteur… php*flag / php*value

Notes personnelles

9.3.2- Socket et sécurité

Socket : La socket d'écoute (**listen**) peut être TCP (INET) ou locale (UNIX).
Dans le cas d'une socket Unix, le contrôle d'accès ne peut être que local et est géré par ::

```
;listen.owner = nobody
;listen.group = nobody
;listen.mode = 0666
```

Identité : (**user** / **group**).
L'utilisateur doit pouvoir lire les contenus et pouvoir écrire dans le fichier de log de PHP-FPM.

9.3.3- Gestion de la charge

Il existe plusieurs modes, on en choisit un par la directive "***pm***", obligatoire :

static : on démarre ***pm.max_children***

ondemand (> 5.6) : on démarrera des processus à la demande et il y aura régulation
les processus seront terminés au bout de ***pm.process_idle_timeout*** secondes.

dynamic : on démarre au moins pm.start_servers, et il y a régulation de charge grâce à
pm.max_children, ***pm.start_servers***, ***pm.min_spare_servers***, ***pm.max_spare_servers***

Dans tous les cas, les processus peuvent être terminés au bout de ***pm.max_requests*** requêtes traitées.

Notes personnelles

9.3.4- Le log des "slow queries"

On peut définir un fichier par ***slowlog***.
Il contiendra des logs sur les requêtes trop longues.

Requête trop longue…
Si une requête dépasse le délai indiqué par ***request_slowlog_timeout***, elle sera tracée dans le **slowlog**.

Requête vraiment trop longue…
Si une requête trop longue n'est pas stoppée par la valeur (.ini) **max_execution_time**, alors elle sera tuée au bout de ***request_terminate_timeout*** secondes.

9.3.5- Variables d'environnement

On peut passer des variables d'environnement ou des paramètres de PHP.INI :

```
env[HOSTNAME] = $HOSTNAME
env[PATH] = /usr/local/bin:/usr/bin:/bin
env[TMP] = /tmp
env[TMPDIR] = /tmp
env[TEMP] = /tmp

php_admin_value[sendmail_path] = /usr/sbin/sendmail -t -i -f www@my.domain.com
php_flag[display_errors] = off
php_admin_value[error_log] = /var/log/fpm-php.www.log
php_admin_flag[log_errors] = on
php_admin_value[memory_limit] = 32M
```


Notes personnelles

9.4- Architecture distribuée avec PHP-FPM

Objectif : améliorer les performances grâce à plusieurs pools / plusieurs serveurs

Une approche "classique" :
Déployer plusieurs serveurs HTTP, chacun disposant de son PHP-FPM.
Dans cette approche, on utilisera en complément un mécanisme de reverse proxy

Une approche plus performante :
Déployer un serveur HTTP, utilisant N pools, répartis sur N machines,

Le serveur HTTP peut dispatcher les requêtes vers différents Pool (load-balancing)
Soit directement (mod_proxy_balancer)
Soit sans le savoir (utilisation de keepalived ou haproxy)

Le Load Balancer devient alors le point critique : le dupliquer et utiliser KeepAlived pour gérer une VIP
KeepAlived est aussi très bon dans le rôle de Directeur
il peut se substituer à un reverse proxy dans certains cas.
Cette approche apporte une notion de Failover, en complément du Load Balancing

Implications : dans les deux approches
Mettre en commun les contenus (NFS**,** déploiement automatisé, conteneurs)
Gérer les sessions applicatives (memcached**,** redis)

Notes personnelles

9.5- Introduction à TOMCAT

9.5.1- Présentation

TOMCAT est un moteur Java permettant de servir des applications écrites en Java (Servlet, JSP). Il faisait initialement partie d'un grand projet de la fondation Apache, appelé Jakarta, et est maintenant un projet à part entière, comme "httpd".

Servlet : c'est une Classe java qui s'exécute coté serveur. C'est donc du code Java compilé. On peut comparer la servlet à l'applet, sauf que la servlet s'exécute coté serveur, dans le but de générer du contenu (image, code HTML ou code Javascript) à envoyer au navigateur.

JavaServer Page (JSP) : c'est une page HTML dans laquelle on insère des tags délimitant du code Java, qui s'exécute coté serveur. C'est donc un peu la même approche que PHP.

La différence principale réside dans le fait que les JSP sont compilées lors de la première utilisation, pour devenir des Servlets.

Les logiciels à installer sont :

Le JDK (Sun/Oracle ou OpenJDK) & Le moteur Tomcat.

Éventuellement, un connecteur entre un Apache en frontal et un ou des serveurs Tomcat :

mod_jk (le connecteur "AJP 1.3" originel, à compiler séparément)
mod_proxy_ajp (basé sur mod_proxy, fourni en natif).

Notes personnelles

9.5.2- Installation des logiciels

TOMCAT possède un serveur HTTP et est fourni avec des exemples (webapps) permettant de le tester :

Le package **tomcat-webapps** installe ces "webapps" et par dépendance le serveur Tomcat.

```
# yum -y install tomcat-webapps
```

On peut ensuite démarrer et tester le serveur.

```
# systemctl start tomcat
# systemctl status tomcat
● tomcat.service - Apache Tomcat Web Application Container
   Loaded: loaded (/usr/lib/systemd/system/tomcat.service; disabled; vendor preset: disabled)
   Active: active (running) since mar. 2017-02-07 10:45:33 CET; 1min 52s ago
 Main PID: 1660 (java)
   Memory: 50.5M
   CGroup: /system.slice/tomcat.service
           └─1660 /usr/lib/jvm/jre/bin/java -classpath /usr/share/tomcat/bin/bootstrap.jar:/…
```

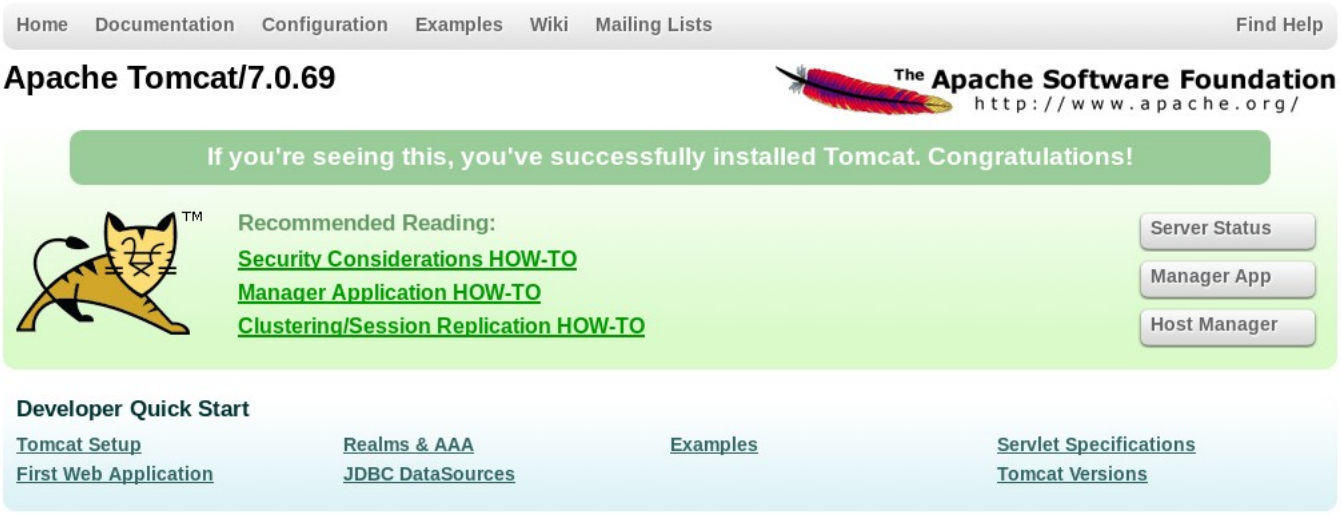
Au démarrage, Tomcat (Java) écoute sur les ports suivants :

8005: port d'administration du serveur Tomcat
8080: port du serveur HTTP de TOMCAT
8009: un des connecteurs (*ajp13*)

Notes personnelles

```
# netstat -pltn
Connexions Internet actives (seulement serrs)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1033/sshd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      1109/master
tcp        0      0 :::8080                 :::*                    LISTEN      1660/java
tcp        0      0 :::22                   :::*                    LISTEN      1033/sshd
tcp        0      0 ::1:25                  :::*                    LISTEN      1109/master
tcp        0      0 ::ffff:127.0.0.1:8005   :::*                    LISTEN      1660/java
tcp        0      0 :::8009                 :::*                    LISTEN      1660/java
```

On peut tenter une connexion avec un navigateur sur le port **8080** : http://votre-serveur:8080/



Notes personnelles

9.5.3- Tester les webapps

On accède aux pages d'exemple du package"webapps" en pointant le navigateur sur :
http://serveur:8080/examples/servlets/ ou http://serveur:8080/examples/jsp/ ,

9.5.4- Configuration de Tomcat

Le fichier de configuration principal est **/etc/tomcat/server.xml**. La partie qui nous intéresse ici est celle déclarant les **Connecteurs : HTTP (port 8080) et AJP (port 8009)**...

```
<!-- A "Connector" represents an endpoint by which requests are received
     and responses are returned. Documentation at :
     Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
     Java AJP  Connector: /docs/config/ajp.html
     APR (HTTP/AJP) Connector: /docs/apr.html
     Define a non-SSL HTTP/1.1 Connector on port 8080
-->
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />

<!-- Define an AJP 1.3 Connector on port 8009 -->
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

Pour placer Apache HTTPD en frontal, deux approches existent :
Utiliser **mod_proxy_http** et donc le connecteur HTTP (port 8080) de Tomcat
ou
Utiliser **mod_proxy_ajp** (ou **mod_jk**) et donc le connecteur AJP (port 8009) de Tomcat.

Notes personnelles

9.6- Connecter Apache et TOMCAT : le connecteur AJP

La configuration à mettre en place vise à cantonner Tomcat au service des servlets/JSP et à laisser Apache faire le reste.

Cette approche permet de bâtir une solution de répartition de charge sur plusieurs serveurs TOMCAT.

9.6.1- Inhibition du connecteur HTTP de Tomcat

Ce n'est pas obligatoire, mais comme Tomcat sera contacté via le connecteur AJP, on peut inhiber le connecteur HTTP. Nous réalisons cela en le mettant en commentaire :

```
<!--
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
-->
```

Procédure :
1/ On arrête TOMCAT
2/ On modifie "server.xml" :
début de commentaire : <!--
fin de commentaire : -->
3/ On redémarre TOMCAT

Notes personnelles

9.7- Mise en œuvre avec mod_proxy_ajp

Il s'agit d'une configuration en reverse-proxy dans laquelle le client ne contacte qu'Apache.
Si l'URL est **/servlets** (ou **/jsp**), alors Apache passe la requête à Tomcat.
Et met en correspondance **/examples/servlets** (ou **/examples/jsp**)
⇒ Implique la règle de Proxy dans les deux sens (ProxyPass et ProxyPassReverse)

9.7.1- Côté Apache

Vérifier que les modules mod_proxy et mod_proxy_ajp sont chargés.

```
# grep -E "mod_proxy(_ajp)*.so"  /etc/httpd/conf.modules.d/00-proxy.conf
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
```

Configurer les règles de mandat inverse :

```
# cat /etc/httpd/conf.d/mod_proxy_ajp.conf
ProxyPass "/servlets"         "ajp://127.0.0.1:8009/examples/servlets"
ProxyPassReverse "/servlets" "ajp://127.0.0.1:8009/examples/servlets"

ProxyPass "/jsp"              "ajp://127.0.0.1:8009/examples/jsp"
ProxyPassReverse "/jsp"       "ajp://127.0.0.1:8009/examples/jsp"
```

Redémarrer HTTPD

```
# systemctl restart httpd
```


Notes personnelles

9.8- Mise en œuvre avec mod_jk

9.8.1- Installation du source de mod_jk

On télécharge l'archive sur http://tomcat.apache.org/download-connectors.cgi
Télécharger le fichier suivant : **tomcat-connectors/jk/tomcat-connectors-1.x.y-src.tar.gz**

Installer le source dans le répertoire /usr/local/src :

```
# tar xzf tomcat-connectors-*.tar.gz  && chown -R root.root tomcat-connectors-*
```

9.8.2- Environnement de compilation

Nous aurons besoin des outils **libtool**, **autoconf** et **automake**.
Ils sont fournis sous forme de paquetages du même nom sur les distributions de Linux.

```
# yum -y install libtool autoconf automake
```

9.8.3- Compilation de mod_jk

Se rendre dans le répertoire "native" et suivre les instructions du fichier "BUILDING.txt"...

```
# cd tomcat-connectors*/native
# ./buildconf.sh
# ./configure --with-apxs=/là/où/est/apache/bin/apxs
# make && make install
```

Produit / met à niveau "./configure"

Notes personnelles

9.8.4- Intégration avec Apache

Le module généré et installé par "make install" est <SERVER-ROOT>/modules/**mod_jk.so**.

Le principe

Vue de Apache, on définit des « **workers** ».
Ce sont des « sous-traitants Tomcat »
Ils désignent des services capables de prendre en charge des requêtes Servlet ou JSP.

On peut définir plusieurs workers, et considérer chacun d'entre-eux :

en fonction du répertoire (ou de l'application)

en fonction du type : JSP ou Servlet

de manière identique, en répartition de charge

Ils sont définis dans un fichier principe nommé "**workers.properties**".
La directive **JkWorkersFile** indique quel est le fichier à Apache.

Apache choisit un Worker en fonction d'une association faite par **JkMount**.
Cette directive associe un contexte web à un worker.

Les traces sont définies par les directives **JkLogFile** et **JkLogLevel**.

Notes personnelles

9.8.5- Mise en œuvre côté Apache

Chargement du module, par exemple en dernier dans notre configuration d'Apache :

```
        LoadModule jk_module          modules/mod_jk.so
```

Puis on décrit un contexte le concernant à la fin de **httpd.conf** :

```
<IfModule mod_jk.c>

        JkWorkersFile /opt/apache22/conf/workers.properties
        JkLogFile     logs/mod_jk.log
        JkLogLevel    info

        JkMount      /examples/servlets/*         worker1
        JkMount      /examples/jsp/*              worker1


        <LocationMatch WEB-INF>
                Deny from all
        </LocationMatch>

</IfModule>
```

On y définit nos workers

C'est le “worker” choisi

Avec l'exemple précédent, si on crée (du côté d'Apache) un répertoire "examples" contenant un fichier "test", celui-ci sera servi par Apache.

Notes personnelles

Le fichier **workers.properties** doit définir les instances Tomcat que nous connaissons

```
workers.tomcat_home=/usr/share/tomcat6
#workers.java_home=/opt/java
ps=/

worker.list=worker1,tomcat1,tomcat2

worker.worker1.port=8009
worker.worker1.host=127.0.0.1
worker.worker1.type=ajp13
worker.worker1.lbfactor=1
worker.worker1.connection_pool_size=10
worker.worker1.connection_pool_timeout=600
worker.worker1.socket_keepalive=1

worker.tomcat1.type=ajp13
worker.tomcat1.port=8009
worker.tomcat1.host=192.168.0.10
worker.tomcat1.lbfactor=100

worker.tomcat2.type=ajp13
worker.tomcat2.port=8009
worker.tomcat2.host=192.168.0.20
worker.tomcat2.lbfactor=100
```

Coordonnées, et type de chaque worker connu

Du coté de chaque Tomcat (server.xml), il faut un « Listener » de type AJP/1.3. Dans le fichier de configuration standard, on trouve notamment :

```
<!-- Define an AJP 1.3 Connector on port 8009 -->
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```


Notes personnelles

10- Annexe : Configuration réseau sous RedHat Linux

Notes personnelles

10.1- Noms des interfaces réseau

10.1.1- Nom historique, nameif, mactab

Historiquement, le nom d'une interface réseau était "**eth**" suivi d'un numéro (0, 1, …)

Jusqu'à RHEL 6, les systèmes peuvent renommer les interfaces .
Voir la commande **nameif**[13], et le fichier **/etc/mactab**.

```
# ifconfig -a | grep Link
eth0      Link encap:Ethernet  HWaddr 08:00:27:44:44:00
eth1      Link encap:Ethernet  HWaddr 08:00:27:43:DF:E5
eth2      Link encap:Ethernet  HWaddr 08:00:27:44:F5:83
lo        Link encap:Boucle locale
# nameif
# ifconfig -a | grep Link
NAT       Link encap:Ethernet  HWaddr 08:00:27:44:44:00
PONT      Link encap:Ethernet  HWaddr 08:00:27:43:DF:E5
eth2      Link encap:Ethernet  HWaddr 08:00:27:44:F5:83
lo        Link encap:Boucle locale
```

S'ils ne sont pas cités en argument de **nameif**, les noms d'interface sont déterminés par **/etc/mactab**

```
# cat /etc/mactab
PONT    08:00:27:43:DF:E5
NAT     08:00:27:44:44:00
```

13 Fournie par le package **net-tools**. Ce package n'est plus installé en standard sur RHEL 7.

Notes personnelles

10.1.2- Autre solution : nommage par UDEV

Dans RHEL 5 et 6, les règles de UDEV associaient les adresses MAC à des noms d'interfaces.

```
# grep --color eth0 /etc/udev/rules.d/70-persistent-net.rules
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="52:54:00:00:22:0f", ATTR{type}=="1",
KERNEL=="eth*", NAME="eth0"
```

10.1.3- Nommage par le fichier de configuration

Dans RedHat 7, on utilise uniquement les fichiers "**ifcfg-interface**".

Si le fichier existe, quelque soit son nom, il suffit qu'il contienne
la directive **HWADDR**= aa:bb:cc:dd:ee:ff
et la directive **DEVICE**=NOM…

Le nom NOM sera donné à l'interface ayant l'adresse MAC aa:bb:cc:dd:ee:ff

```
# cat /etc/sysconfig/network-scripts/ifcfg-vboxnet0
HWADDR=08:00:27:c7:f0:01
DEVICE=LAN
ONBOOT=yes
TYPE=Ethernet
BOOTPROTO=dhcp
….
```

Sans ce fichier, ou sans l'association HWADDR/DEVICE, le nom peut être issu de "**biosdevname**" ou lié à **systemd-udev**, qui peut, comme udev en RH6, avoir ses propres règles.

Notes personnelles

10.1.4- La commande ifconfig

Lorsque le pilote est chargé, il est possible de configurer une interface réseau. C'est le rôle de la commande **ifconfig**[14]**,** plus ou moins abandonnée désormais au profit de la commande "ip".

On peut démarrer (up) ou arrêter (down) une interface, même non configurée :

```
# ifconfig eth1[15] up
```

Elle sert aussi à lister les interfaces actuellement configurées (**ifconfig -a**), et si une interface n'apparaît pas dans la liste, c'est qu'elle est arrêtée.

```
# ifconfig eth1
eth1      Lien encap:Ethernet  HWaddr 00:00:B4:65:6B:80
          inet adr:192.168.1.1  Bcast:192.168.1.255  Masque:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Paquets Reçus:0 erreurs:0 jetés:0 débordements:0 trames:0
          Paquets transmis:0 erreurs:44 jetés:0 débordements:0 carrier:44
          collisions:748 lg file transmission:100
          Interruption:10 Adresse de base:0x300
```

14 Package "net-tools"

15 Sous Linux, les interfaces réseau Ethernet s'appelaient historiquement eth0, eth1,...
On peut les renommer
On peut revenir à ce mode de nommage grâce au paramètre noyau net.ifnames (=0)

Notes personnelles

10.1.5- Configurer une interface ethernet

Il suffit de fournir le nom de l'interface et l'adresse IP voulue en argument de la commande **ifconfig**.

```
# ifconfig eth1 192.168.1.1
```

L'interface est alors reconfigurée. Si elle était arrêtée, alors elle est démarrée dans la foulée.

10.1.6- Masque réseau, Adresse Broadcast

Les différentes classes de réseaux possèdent des masques prédéfinis. La commande **ifconfig** en tient compte automatiquement en fonction de la classe de l'adresse.

On peut préciser un masque plus restrictif que le masque par défaut (subnetting) ou plus ouvert (supernetting), et ainsi outrepasser les masques usuels des classes de réseau. Pour cela, il faut préciser les informations « spécifiques » sur la ligne de commande **ifconfig**.

```
# ifconfig eth1 192.168.1.161 netmask 255.255.255.248 broadcast 192.168.1.167
```

L'adresse Broadcast est déduite de l'adresse IP (et du masque de réseau). On peut là aussi préciser une autre valeur. Dans le cas contraire, c'est la dernière adresse de la classe qui est choisie.

Notes personnelles

10.2- Route et table de routage

10.2.1- Notion de route

La portée du réseau local (LAN) est définie par le couple Adresse IP / Masque réseau.

Pour communiquer avec d'autres réseaux, on doit suivre **une route** listée dans **une table de routage**.

10.2.2- Lister la table de routage

La commande "**route**"[16] liste la table de routage (**-n** : pour ne pas faire de résolution de noms)

```
# route -n
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
172.17.1.0      0.0.0.0         255.255.255.0   U     0      0        0 eth2
169.254.0.0     0.0.0.0         255.255.0.0     U     1004   0        0 eth2
0.0.0.0         172.17.1.1      0.0.0.0         UG    0      0        0 eth2
```

Il y a toujours une **route "minimale"**, qui représente celle du réseau LAN :

Sur certains systèmes il y a **une route dite APIPA** (dans un monde parallèle), liée en fait au protocole **ZeroCONF.** Pour ne pas la voir, on déclare la variable NOZEROCONF dans /etc/sysconfig/network.

La **route "par défaut"** ("default", la destination "0.0.0.0") désigne la passerelle (172.17.1.1) qui relayera le trafic venant de ou allant vers un autre réseau que le LAN.

16 Package "net-tools"

Notes personnelles

10.2.3- Éléments de la table de routage

```
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
172.17.1.0      0.0.0.0         255.255.255.0   U     0      0        0 eth2
0.0.0.0         172.17.1.1      0.0.0.0         UG    0      0        0 eth2
```

Destination, Passerelle, Genmask, Iface

Destination et **Genmask** indiquent l'adresse et la portée du réseau (ou de l'hôte) cible.
Passerelle est l'adresse de la passerelle connaissant ce réseau ou la suite de la route à suivre.
Iface indique par quelle interface réseau on peut prendre cette route.

Flags(Indic)

Ce sont des indicateurs donnant une ou des informations sur une route :

U (route is up), **H** (target is a host), **G** (use gateway)
R (reinstate route for dynamic routing), **D** (dynamically installed by daemon or redirect)
M (modified from routing daemon or redirect), **A** (installed by addrconf)
C (cache entry), **!** (reject route)

Metric

Le Metric indique un coût entre le routeur et la destination. Ce paramètre n'est plus utilisé par Linux, sauf pour certains démons de routage.

Notes personnelles

10.2.4- Ajouter une route

La syntaxe est la suivante :

```
route [-v] [-A family] add [-net|-host] target [netmask Nm] [gw Gw] [metric N] [mss M] [window
W] [irtt I] [reject] [mod] [dyn] [reinstate] [[dev] If]
```

Exemple Ajoute une route vers tout le réseau de **172.17.0.0/16** par la passerelle **172.16.18.1**, jointe via **eth1**. La "gateway" doit être joignable directement pour que la route soit utilisable .

```
route add –net 172.17.0.0 netmask 255.255.0.0 gw 172.16.18.1 dev eth1
```

10.2.5- Supprimer une route

La syntaxe est la suivante :

```
route [-v] [-A family] del [-net|-host] target [gw Gw] [netmask Nm] [metric N] [[dev] If]
```

Il faut préciser au moins la destination, et éventuellement d'autres informations pour différencier plusieurs routes vers le même réseau ou la même machine.

Notes personnelles

10.3- Configurer le réseau

10.3.1- La commande ip

Devenue le standard[17] depuis RedHat 7 (et CentOS 7)… en remplacement de "ifconfig" et ses sœurs.

```
# ip
Usage: ip [ OPTIONS ] OBJECT { COMMAND | help }
       ip [ -force ] -batch filename
where  OBJECT := { link | addr | addrlabel | route | rule | neigh | ntable |
                   tunnel | tuntap | maddr | mroute | mrule | monitor | xfrm |
                   netns | l2tp | tcp_metrics | token }
       OPTIONS := { -V[ersion] | -s[tatistics] | -d[etails] | -r[esolve] |
                    -f[amily] { inet | inet6 | ipx | dnet | bridge | link } |
                    -4 | -6 | -I | -D | -B | -0 |
                    -l[oops] { maximum-addr-flush-attempts } |
                    -o[neline] | -t[imestamp] | -b[atch] [filename] |
                    -rc[vbuf] [size]}
```

Les objets : **addr** et **route** nous intéressent dans un premier temps

Les actions (**COMMAND**) : **add**, **delete**, **show** (ou **rien** ou **list**).

Pour certains objets, certaines actions n'existent pas… d'autres apparaissent en plus.

17 La commande "**ifconfig**" n'est plus installée en installation "de base". C'est le package "**net-tools**" qui la fournit, ainsi que les commandes suivantes : /bin/**netstat**, /sbin/**arp**, /sbin/**ether-wake**, /sbin/**ifconfig**, /sbin/**ipmaddr**, /sbin/**iptunnel**, /sbin/**mii-diag**, /sbin/**mii-tool**, /sbin/**nameif**, /sbin/**plipconfig**, /sbin/**route**, /sbin/**slattach**

Notes personnelles

10.3.2- Lister une interface ou les tables de routage

```
ip addr

ip route
```

ou
ou

```
ip addr show

ip route show
```

Pour tous les objets : l'action **help** décrit la syntaxe précise… **ip addr help** …

```
# ip addr help
Usage: ip addr {add|change|replace} IFADDR dev STRING [ LIFETIME ]
                                                      [ CONFFLAG-LIST ]
       ip addr del IFADDR dev STRING
       ip addr {show|save|flush} [ dev STRING ] [ scope SCOPE-ID ]
                            [ to PREFIX ] [ FLAG-LIST ] [ label PATTERN ] [up]
       ip addr {showdump|restore}
IFADDR := PREFIX | ADDR peer PREFIX
          [ broadcast ADDR ] [ anycast ADDR ]
          [ label STRING ] [ scope SCOPE-ID ]
SCOPE-ID := [ host | link | global | NUMBER ]
FLAG-LIST := [ FLAG-LIST ] FLAG
FLAG  := [ permanent | dynamic | secondary | primary |
           tentative | deprecated | dadfailed | temporary |
           CONFFLAG-LIST ]
CONFFLAG-LIST := [ CONFFLAG-LIST ] CONFFLAG
CONFFLAG  := [ home | nodad ]
LIFETIME := [ valid_lft LFT ] [ preferred_lft LFT ]
LFT := forever | SECONDS
```


Notes personnelles

10.3.3- Ajouter, lister, retirer les adresses IP

```
ip addr {add|change|replace} IFADDR dev STRING [ LIFETIME ]
                                                [ CONFFLAG-LIST ]
```

Exemple :

```
# ip addr add 172.17.1.56/32 dev enp0s3 valid_lft 30 preferred_lft 20
```

Le listage...

```
# ip addr show enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 08:00:27:c7:f0:01 brd ff:ff:ff:ff:ff:ff
    inet 172.17.1.202/24 brd 172.17.1.255 scope global dynamic enp0s3
       valid_lft 467sec preferred_lft 467sec
    inet 172.17.1.56/32 scope global enp0s3
       valid_lft 25sec preferred_lft 10sec
    inet6 fe80::a00:27ff:fec7:f001/64 scope link
       valid_lft forever preferred_lft forever
```

Le retrait d'une adresse

```
# ip addr del 172.17.1.56/32 dev enp0s3
```

Sinon, il peut être automatique au bout du "**valid_lft**" si un "LIFETIME" a été précisé à l'activation.

Notes personnelles

10.3.4- Manipuler les routes

```
Usage: ip route { list | flush } SELECTOR
       ip route save SELECTOR
       ip route restore
       ip route showdump
       ip route get ADDRESS [ from ADDRESS iif STRING ]
                            [ oif STRING ]  [ tos TOS ]
                            [ mark NUMBER ]
       ip route { add | del | change | append | replace } ROUTE
SELECTOR := [ root PREFIX ] [ match PREFIX ] [ exact PREFIX ]
            [ table TABLE_ID ] [ proto RTPROTO ]
            [ type TYPE ] [ scope SCOPE ]
ROUTE := NODE_SPEC [ INFO_SPEC ]
NODE_SPEC := [ TYPE ] PREFIX [ tos TOS ]
             [ table TABLE_ID ] [ proto RTPROTO ]
             [ scope SCOPE ] [ metric METRIC ]
INFO_SPEC := NH OPTIONS FLAGS [ nexthop NH ]...
NH := [ via ADDRESS ] [ dev STRING ] [ weight NUMBER ] NHFLAGS
OPTIONS := FLAGS [ mtu NUMBER ] [ advmss NUMBER ]
           [ rtt TIME ] [ rttvar TIME ] [reordering NUMBER ]
           [ window NUMBER] [ cwnd NUMBER ] [ initcwnd NUMBER ]
           [ ssthresh NUMBER ] [ realms REALM ] [ src ADDRESS ]
           [ rto_min TIME ] [ hoplimit NUMBER ] [ initrwnd NUMBER ]
TYPE := [ unicast | local | broadcast | multicast | throw |
          unreachable | prohibit | blackhole | nat ]
TABLE_ID := [ local | main | default | all | NUMBER ]
SCOPE := [ host | link | global | NUMBER ]
NHFLAGS := [ onlink | pervasive ]
RTPROTO := [ kernel | boot | static | NUMBER ]
TIME := NUMBER[s|ms]
```

Exemple pour ajouter une route : **ip route add 8.8.8.8/32 *via 172.17.1.1* *dev eth2***

Notes personnelles

10.4- Fichiers de configuration des interfaces réseau

10.4.1- Panorama des fichiers sur RedHat / Fedora

Script de démarrage : /etc/init.d/network
C'est le script chargé de démarrer le service, il est exécuté par « rc ».
Ce script en appelle d'autres se trouvant dans **/etc/sysconfig/network-scripts**.

Fichier de configuration général : /etc/sysconfig/network
Configuration générale de la machine (Nom, Routage, Gateway, etc…)

Fichier de configuration de chaque interface : /etc/sysconfig/network-scripts/ifcfg-INTERFACE
Un fichier par interface réseau (Adresse IP, LAN @, Broadcast @, …)

Pour les alias d'interface : /etc/sysconfig/network-scripts/ifcfg-INTERFACE:X
Un fichier par Alias d'@ IP (forme identique à ifcfg-IFACE)

10.4.2- Panorama des fichiers sur Debian / Ubuntu

Script de démarrage : /etc/init.d/networking
C'est le script chargé de démarrer le service réseau

Fichier de configuration : /etc/network/interfaces
Le fichier de configuration des interfaces réseau et des alias d'IP

Notes personnelles

10.5- Analyse et diagnostic réseau : ping

10.5.1- La commande ping

Elle permet de valider qu'une interface réseau est configurée et qu'une adresse IP est joignable.

On peut s'en servir pour surcharger un réseau, mesurer des temps d'accès vers un hôte distant, etc.

Ping utilise le protocole **ICMP**, et en particulier des datagrammes **ECHO_REQUEST**, chargés de demander à l'hôte distant un envoi de datagramme de type **ECHO_RESPONSE**.

L'usage le plus courant est **ping ADRESSE**, l'adresse peut être un nom d'hôte ou une adresse IP.

Lorsque le ping ne passe pas, l'affichage est le suivant :

```
# ping -c 5 www.pentagon.org
PING www.pentagon.org (208.4.62.5): 56 octets data

--- www.pentagon.org ping statistics ---
5 packets transmitted, 0 packets received, 100% packet loss
```

Si l'on ne précise pas l'option **-c X**, **ping** continue indéfiniment, à raison d'un paquet par seconde.

Un hôte peut ne pas répondre au ping : paramètre noyau **net.ipv4.icmp_echo_ignore_all**

Notes personnelles

10.5.2- Quelques options

Interface de sortie : -I interface

Le délai entre deux envois : **-i** X (secondes)

```
# ping -i 5 www.actilis.fr
```

On peut abaisser le délai bien en dessous de la seconde (en tant que root si < 0.2 sec).

```
# ping -i 0.01 -c 10 localhost
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from tupai.actilis.net (127.0.0.1): icmp_seq=1 ttl=64 time=0.077 ms
64 bytes from tupai.actilis.net (127.0.0.1): icmp_seq=2 ttl=64 time=0.034 ms
64 bytes from tupai.actilis.net (127.0.0.1): icmp_seq=3 ttl=64 time=0.011 ms
...
64 bytes from tupai.actilis.net (127.0.0.1): icmp_seq=10 ttl=64 time=0.010 ms

--- localhost ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 80ms
rtt min/avg/max/mdev = 0.010/0.020/0.077/0.020 ms
```

Temps d'attente d'une réponse : **-W** X (secondes)

Ping sur l'adresse de broadcast : **-b (et utiliser l'adresse de broadcast comme cible)**
Ne pas répondre au ping broadcast : paramètre noyau **net.ipv4.icmp_echo_ignore_broadcast**

Notes personnelles

10.5.3- Ping et le flooding

En dessous d'un intervalle (-i) de 0.2 sec, c'est assimilé à du flooding.

Le Flood Ping, servant à surcharger un réseau : option **-f**.

Il sert à voir combien de paquets sont perdus et donc à tester la qualité du réseau.

2001 , côté client : ADSL 512k, côté serveur : lien 100Mbps

```
# ping -f www.actilis.fr
PING web.pro.proxad.net (212.27.35.100): 56 octets data
.....................................................................................
.................................
--- web.pro.proxad.net ping statistics ---
432 packets transmitted, 269 packets received, 37% packet loss
round-trip min/avg/max = 44.1/380.9/533.3 ms
```

2014, côté client : 4G, côté serveur : lien Gbps.

```
# ping -f www.actilis.fr
PING www.actilis.fr (212.83.186.60) 56(84) bytes of data.
..^C
--- www.actilis.fr ping statistics ---
426 packets transmitted, 424 received, 0% packet loss, time 6256ms
rtt min/avg/max/mdev = 23.453/30.537/85.820/5.031 ms, pipe 6, ipg/ewma 14.721/30.809 ms
```


Notes personnelles

11- Annexes concernant Apache

Notes personnelles

11.1- Compilation du serveur Apache

11.1.1- Pourquoi la compilation

Apache est un **logiciel libre** auquel certaines règles s'appliquent :

Réactivité importante **des développeurs** en cas de « bug »,
Correctifs de **sécurité** à appliquer…
Mises à jour et **évolutions fréquentes**,
Obsolescence rapide des paquetages binaires,

11.1.2- Pré-requis à la compilation

Obtenir les sources… http://httpd.apache.org/download.cgi : propose les dernières versions

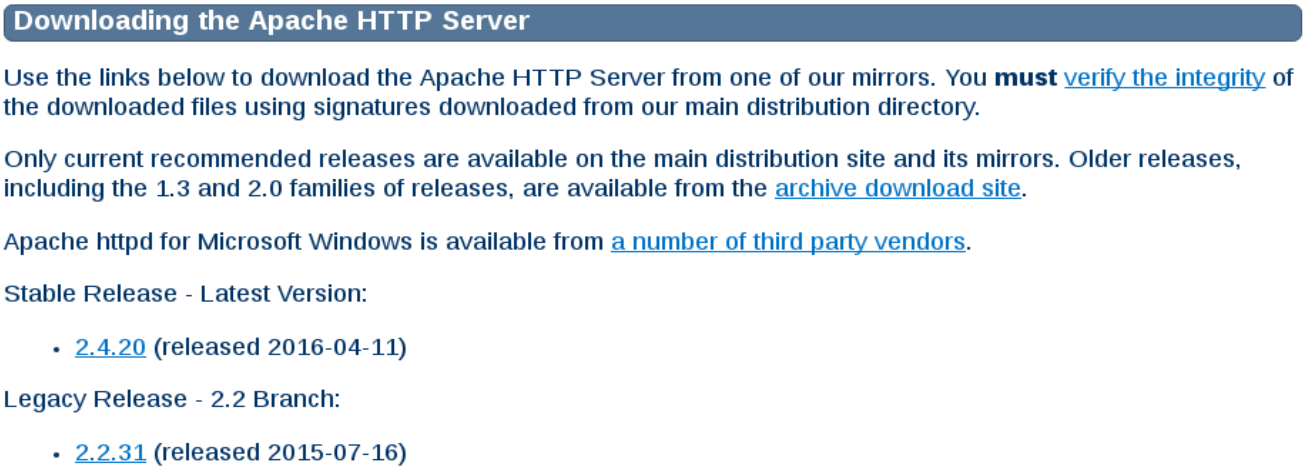
Pour compiler HTTPD-2.4, il faut au minimum les packages suivants :
Les outils suivants sont nécessaires :
bzip2, **gcc**, **make**, **autoconf** et **libtool**
pcre-devel, **lua-devel**, **libnghttp2-devel**, **openssl-devel**,

```
# yum -y install brip2 gcc make autoconf libtool \
  pcre-devel lua-devel libnghttp2-devel openssl-devel
```

Note : httpd-2.4 nécessite aussi **apr** >=1.4 + **apr-util** en phase…
⇒ Ils sont fournis en tant que dépendance du code-source

Notes personnelles

11.1.3- Où trouver le code source ?



Downloading the Apache HTTP Server

Use the links below to download the Apache HTTP Server from one of our mirrors. You **must** verify the integrity of the downloaded files using signatures downloaded from our main distribution directory.

Only current recommended releases are available on the main distribution site and its mirrors. Older releases, including the 1.3 and 2.0 families of releases, are available from the archive download site.

Apache httpd for Microsoft Windows is available from a number of third party vendors.

Stable Release - Latest Version:

- 2.4.20 (released 2016-04-11)

Legacy Release - 2.2 Branch:

- 2.2.31 (released 2015-07-16)

Concernant RHEL 6 / CentOS 6 : Pour Httpd 2.4, il faut disposer de "**apr**" et de "**apr-util**" dans une version plus récente que celle fournie par RHEL **6**.X (ou CentOS **6**.X)...

L'archive "**httpd-x.y.z-deps.tar.bz2**" propose ces composants,
Elle n'est pas censée être systématiquement proposée (voir fichier INSTALL), mais… elle l'est.

Liens directs : Source & Dédendances

http://www-eu.apache.org/dist//httpd/

Notes personnelles

11.1.4- Téléchargement et installation du source

L'endroit souvent choisi pour sauvegarder les archives de sources est le répertoire **/usr/local/src**, mais ce n'est pas très important : /root, c'est possible aussi !

Pour l'exemple, on se positionne dans **/usr/local/src**

```
# cd /usr/local/src
```

On télécharge et extrait les archives :

```
# curl http://www-eu.apache.org/dist//httpd/httpd-2.4.23.tar.bz2 | tar xj
# curl http://www-eu.apache.org/dist//httpd/httpd-2.4.23-deps.tar.bz2 | tar xj
```

Cela doit créer un sous-répertoire du nom de la version.

```
# ls -ltr
total 4
drwxr-xr-x. 11 501 games 4096 30 juin  19:15 httpd-2.4.23
```

On peut commencer par prendre possession, dans une optique de sécurité :

```
# chown -R root.root httpd-*
```


Notes personnelles

11.1.5- Préparation de la compilation : "configure"

Tout se passe dans le répertoire **/usr/local/src/httpd-X.Y.Z.**

Il faut **configurer la compilation** comme pour beaucoup de Logiciels Libres.

Depuis le répertoire des sources, on exécute le script "**configure**".

Il va créer/configurer les Makefiles et différents paramètres de compilation propres à la plateforme.

Il prévoit des options permettant de préciser comment et avec quelles extensions on veut compiler.

L'option help
La première option intéressante est "--help" , voici le début de ce qu'elle affiche :

```
# ./configure --help | less
`configure' configures this package to adapt to many kinds of systems.

Usage: ./configure [OPTION]... [VAR=VALUE]...

To assign environment variables (e.g., CC, CFLAGS...), specify them as
VAR=VALUE.  See below for descriptions of some of the useful variables.

Defaults for the options are specified in brackets.
```


Notes personnelles

Les "Layout"

Il s'agit d'un ensemble d'options, concernant surtout les répertoires d'installation des binaires qui vont être générés.

Sans option, le script configure choisit le premier Layout du fichier : Apache.

C'est exactement comme si on l'avait appelé de la manière suivante :

```
# ./configure --enable-layout=Apache
```

Le préfixe d'installation

On peut modifier le **PREFIX**, c'est dire le point de départ de l'installation du produit :

```
# ./configure --prefix=/usr/local/apache
```

On peut aussi spécifier un par un les différents répertoires de l'installation (config, doc, …)

Autres options

La plupart des autres options servent à activer une fonctionnalité ou un module et se précisent par :

"**--enable-XXX**" pour l'activer, lorsqu'elle est désactivée par défaut,

"**--disable-XXX**" pour la désactiver, lorsqu'elle est activée par défaut,

Notes personnelles

Note

Les différents *Layouts* proposés sont définis dans le fichier **config.layout**.

Dans **Apache 2.X, l'option est "--enable-layout"**

Ce layout « standard » prévoit un **PREFIX=/usr/local/apache**
Tous les répertoires concernant Apache seront déclinés à partir de $**PREFIX**.

Les distributions packagées peuvent utiliser d'autres Layout (GNU, RedHat, opt, etc...)

Concernant les modules (et les MPM)

La syntaxe **--enable-module=NOM** permet de demander la compilation du module **NOM**.

Pour que les modules soient compilés en tant que Dynamic Shared Object (module chargeable à la volée), il faut utiliser la directive suivante :

```
--enable-nom_module=shared
```

Pour demander le plus possible de modules en DSO:

```
--enable-mods-shared=all
```

Le choix concernant les DSO est apparent lors de l'exécution du script "configure" :

```
...
checking whether to enable mod_authn_file... shared (all)
…
```

Idem pour les MPM avec **--enable-mpm-shared**=all

```
…
checking if event MPM supports this platform... yes
checking if mpmt_os2 MPM supports this platform... no
checking if prefork MPM supports this platform... yes
checking if WinNT MPM supports this platform... no
checking if worker MPM supports this platform... yes
checking which MPM to use by default... event
…
```


Notes personnelles

```
config.status: creating docs/conf/extra/httpd-mpm.conf
```


Notes personnelles

11.1.6- "./configure…" par l'exemple

Cette opération doit être exécutée depuis le répertoire "httpd-version"

```
# cd httpd-*
# ./configure  --prefix=/opt/httpd-2.4 --sysconfdir=/etc/httpd-2.4 \
      --enable-session-crypto --with-crypto --enable-lua \
      --enable-mods-shared=all --enable-mpm-shared=all
```

Ici, on choisit le préfixe /opt/httpd-2.4, mais la configuration sera dans /etc/httpd-2.4.
Tous les modules, y compris les MPM, seront compilés en dynamique.

11.1.7- Compilation

Il suffit de lancer « **make** » :

```
# make -j $(grep -c ^processor /proc/cpuinfo) | tee make.result
```

11.1.8- Installation

Recopier l'ensemble des programmes, librairies, documentations dans les répertoires du système.

Il suffit de lancer la commande « **make install** » :

```
# make install | tee make.install.result
```


Notes personnelles

12- Annexe : services liés au mail

Notes personnelles

12.1- Acteurs de la messagerie

12.1.1- MUA, MSA, MTA, MDA

MTA : **Mail Transport Agent**

Rôle : prise en charge et acheminement du courrier jusqu'à stockage en BAL
Protocole : SMTP
Logiciels : Sendmail, qmail, Postfix, ...
MSA : Mail Submission Agent : un MTA interne... qui relaye au MTA "exposé"

MDA : **Mail Delivery Agent**

Le stockage d'un mail est réalisé par un programme appelé "**MDA**" (***local***, ***virtual***)
C'est aussi le nom coté serveur des services POP / IMAP et leurs "S-variantes"
Logiciels : Dovecot, Cyrus Imap, Courier Imap (intègrent aussi le service POP)

MUA : **Mail User Agent**

Rôle : stockage local et de classement des messages entrants / sortants sur le client
Protocoles :
SMTP vers le MTA (ou le MSA) pour l'envoi
POP et IMAP vers le MTA pour la réception
Logiciels : Outlook,Thunderbird, Evolution, ...

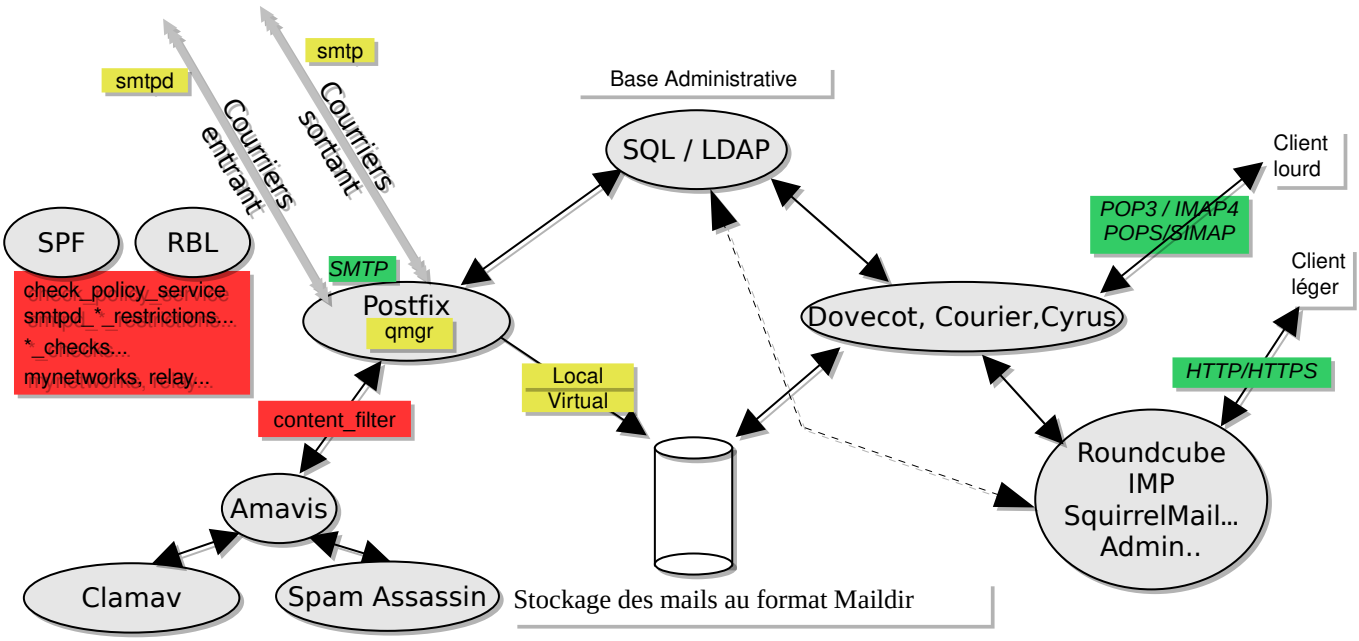
MRA : **Mail Retrieval Agent**

Coté client : logiciels type fetchmail, movemail...
qui récupèrent du courrier via POP(S) ou IMAP(S) pour le livrer localement.

Notes personnelles

12.1.2- Architecture d'une solution de messagerie

Architecture type :



Une solution comme celle-ci est administrable par une application Web.

Son rôle est limité à la manipulation de données (dans la base MySQL / l'annuaire LDAP).

Aucune intervention sur le système en ligne de commande n'est alors nécessaire.

Notes personnelles

12.2- Les protocoles de la messagerie : SMTP

12.2.1- SMTP : Simple Mail Transport Protocol

C'est le protocole utilisé dans les échanges de courrier :
de serveur à serveur (acheminement du courrier)
du client vers le serveur (envoi de mail)

Le protocole **SMTP** est défini par les **RFC 821** et **2821**, et le format des messages par la **RFC 822.**

12.2.2- Le rôle du DNS

Le serveur émetteur réalise une requête **MX** pour identifier le serveur à contacter...

```
~$ host -t MX actilis.net
actilis.net mail is handled by 20 mx02.actilis.net.
actilis.net mail is handled by 10 mx01.actilis.net.
```

Il cherche ensuite l'adresse IP du « MX » principal (celui de priorité la plus basse) :

```
~$ host -t A mx.actilis.net
mx.actilis.net has address 88.190.244.41
```

Il contacte ensuite ce serveur, en se connectant sur son **port SMTP (tcp/25)**.

Si celui-ci ne répond pas ou que le mail est rejeté, le serveur émetteur peut tenter de contacter un autre MX (s'il y a un autre MX).

Notes personnelles

Une session SMTP utilise le protocole SMTP : ELHO, MAIL FROM:, RCPT TO:, DATA, QUIT …

Exemple	Description
EHLO actilis.net	Identification à l'aide de l'adresse IP ou du nom de domaine de l'ordinateur expéditeur Le client se présente... Le serveur aussi, en lui annonçant ses fonctionnalités.
(optionnel) **AUTH** XXXX	Le client s'authentifie (méthode PLAIN, LOGIN...)
MAIL FROM: <fmicaux@actilis.net>	Identification de l'adresse de l'expéditeur
RCPT TO: <formation@actilis.net>	Identification de l'adresse du destinataire
DATA<Enter> Corps du mail <Enter>**.**<Enter>	Quand on termine par "." le corps du mail, le message est soumis. Il entre immédiatement en traitement par le serveur.
QUIT	Sortie du serveur SMTP
HELP	Liste des commandes SMTP supportées par le serveur

L'ordre est important...

Bonjour, j'ai un mail de Alice, il est pour Bob, voici son contenu,
Fin de contenu,
Au-revoir.

ou

J'ai (encore) un mail de Alice, il est pour Chris, voici son contenu,
Fin de contenu,
Au-revoir.

Notes personnelles

12.3- Les protocoles de messagerie : POP3

12.3.1- Post Office Protocol

Le protocole POP3 est défini par la **RFC 1939** (qui remplace la **RFC 1725**).
Il permet au client de télécharger son courrier depuis le serveur POP.
En général, le serveur POP écoute le port **tcp/110**.

C'est ce service que contacte le MUA pour **télécharger les courriers** stockés en boite à lettre.

12.3.2- Le protocole POP3S

Défini par la **RFC 2595**, il s'agit du protocole POP3 par dessus une connexion TLS (port **tcp/995**).

Les commandes de POP3S sont ensuite les mêmes que celles de POP3.

12.3.3- Fonctionnement du protocole POP3

Le client présente des requêtes sont présentées par un mot clé et des arguments pour...
prendre connaissance de la liste des messages disponibles,
rapatrier un message,
supprimer un message

Après chaque requête, le serveur signale un code retour :
Les réponses retournent **+OK** en début de ligne si tout ce passe bien.
Elles retournent **-ERR** en cas de problème.

Notes personnelles

Commande	Description
USER identifiant	Cette commande permet de s'authentifier. Elle doit être suivie du nom de l'utilisateur, c'est-à-dire une chaîne de caractères identifiant l'utilisateur sur le serveur. La commande USER doit précéder la commande PASS.
PASS mot_de_passe	La commande PASS, permet d'indiquer le mot de passe de l'utilisateur dont le nom a été spécifié lors d'une commande USER préalable.
STAT	Information sur les messages contenus sur le serveur
RETR	Suivi du numéro du message à récupérer
DELE	Suivi du numéro du message à supprimer
LIST [msg]	Suivi du numéro du message à afficher
NOOP	Permet de garder les connexion ouverte en cas d'inactivité
TOP <messageID> <n>	Commande affichant n lignes du message, dont le numéro est donné en argument. En cas du réponse positive du serveur, celui-ci renvoie les en-têtes du message, puis une ligne vierge et enfin les n premières lignes du message.
UIDL [msg]	Demande au serveur de renvoyer une ligne contenant des informations sur le message éventuellement donné en argument. Cette ligne contient une chaîne de caractères, appelée listing d'identificateur unique, permettant d'identifier de façon unique le message sur le serveur, indépendamment de la session. L'argument optionnel est un numéro correspondant à un message existant sur le serveur POP, c'est-à-dire un message non effacé).
QUIT	La commande QUIT demande la sortie du serveur POP3. Elle entraîne la suppression de tous les messages marqués comme effacés et renvoie l'état de cette action.

Notes personnelles

12.4- Les protocoles de messagerie : IMAP4

12.4.1- Internet Message Access Protocol

Ce protocole est défini par la **RFC 2060**, puis après par la **RFC 3501** (IMAP version 4rev1)

Le client et le serveur sont connectés "durablement"...

Le client peut manipuler son courrier en restant connecté. IMAP permet de **manipuler une arborescence de sous-dossiers** stockée **sur le serveur**.

C'est un protocole très utilisé par les applications de type Webmail.

IMAP4 et TLS/SSL : IMAP 4 utilisé par dessus une connexion TLS: IMAPS, (port **tcp/993**)

12.4.2- Fonctionnement du protocole IMAP4

Il s'agit d'échanges entre le client et le serveur.

La syntaxe des commandes du protocole IMAP est la suivante:

```
[ tag ] [ commande ] [ argument ] [ argument ] ( retour chariot )
```

Le tag est un code alphanumérique que l'on peut choisir: en général, on utilise un code croissant pour identifier temporellement les commandes.

Notes personnelles

Commande	Description
LOGIN	prend comme argument le nom et le mot de passe du client.
SELECT	prend comme argument le nom de la boite que l'on veut manipuler.
CREATE	le nom de la nouvelle boite est passée en argument.
DELETE	efface la boite aux lettres qui est passée en argument.
RENAME	prend comme argument le nom de l'ancienne boite et le nouveau.
LIST	prend en argument la référence (ex : etc/mail/....) et le nom de la boite.
STATUS	donne les informations sur une boite.
SEARCH	permet de rechercher des messages selon des critères spécifiques.
UID commande args	Manipulation d'un courrier : arguments de type FETCH, STORE
EXPUNGE	Commet les modifications réalisées par UID sur le dossier.
CLOSE	pas d'argument, ferme la boite mais attend de nouveau un login.

Le serveur peut retourner plusieurs types de réponses :

tag * OK : la commande s'est bien déroulée.
tag * NO : échec de la commande.
BAD : erreur de protocole.
PREAUTH : à l'accueil, indique qu'il n'est pas nécessaire de se logger.
BYE : le serveur va fermer sa session.

Notes personnelles

12.5- Le Serveur SMTP Postfix

Il faut à peine 5 minutes pour installer Postfix et réaliser une configuration simple !

Outre les **performances**, Postfix est conçu pour répondre à 3 autres objectifs principaux :

Compatible avec Sendmail, donc avec :
- Les MUA existants (pine ,mutt, mail, ...)
- Gestionnaires de liste (majordomo, sympa, ...)
- Formats de boîte aux lettres (mh, mbox, qmail-dir, ...)
- Agents d'acheminement local (procmail, deliver, cyrus, ...)
- Configurations (UUCP, réécriture, mailertable, ...)
- Utilisateurs (alias, .forward, ...)

Sécurité : une architecture très unixienne : une fonction = un programme
- Décomposition en agents = programmes plus petits et plus lisibles
- Plus difficile à casser ou circonvenir, chroot plus facile
- Les programmes ne se font pas confiance : isolation de chaque fonction
- Communication par sockets unix ou pipes (FIFO)

Facile à administrer :
- Des commandes d'administration simples
- Un fichier de configuration simple et des directives dont le nom est parlant
- Clarté = Sécurité, car ce qui est facile à comprendre est facile à sécuriser

Notes personnelles

12.6- Postfix et les tables

12.6.1- Rôle

Postfix utilise des tables de correspondance pour stocker les informations de :
contrôle d'accès, filtrage de contenu
ré-écriture d'adresse, transport du courrier, livraison du courrier...

12.6.2- Types de tables

La liste (non exhaustive) des types de table gérés est donnée par la commande **postconf -m**.

Les types possibles dépendent de la façon dont Postfix a été compilé

```
# postconf -m |sort
btree
cidr
environ
hash
ldap
mysql
nis
pcre
proxy
regexp
sdbm
static
unix
```


Notes personnelles

12.7- Commandes d'administration de Postfix

postfix : super-intendant du serveur (démarrage, arrêt, …)

postsuper : contrôle de la file d'attente et de ses répertoires de stockage (delete, hold, unhold, requeue..)

postqueue : contrôle de la file d'attente (flush, et visualisation du contenu de la file)

postmap : requêtes dans les tables et construction des tables plates

postconf : visualisation et édition des paramètres de configuration

postalias : "équivalente" de "newaliases" (= *postalias /etc/aliases*), sais requêter dans une table base.

postlock : exécution d'une commande (unix, sans interprétation) sur après lock exclusif sur un fichier

postcat : visualisation d'un fichier d'une file (*postcat File* ou *postcat -q QID*)

postdrop : création d'un fichier dans le spool (injection dans "maildrop" d'un mail posté par "sendmail")

postkick : envoi d'un signal à un processus de Postfix (*postkick public qmgr IF*)

postlog : émission d'un message de log (*cf logger*)

Notes personnelles

12.8- Postfix et les logs

C'est le service **(r)syslogd** qui est utilisé.

Paramétrer **rsyslogd.conf** en conséquence

```
mail.=info                      -/var/log/mail/info.log
mail.=warn                      -/var/log/mail/warn.log
mail.err                        -/var/log/mail/err.log
```

Deux directives de configuration de Postfix permettent de paramétrer l'émission des logs

syslog_facility = ***mail*** : Postfix émet les messages en tant que sous-système ***mail***

syslog_name = postfix : les message sont taggués avec le mot « postfix » :

```
Date serveur postfix/démon[PID]: IDENTIFIANT: message de trace
```

Exemple :

```
Mar  3 14:09:41 ns31057 postfix/smtpd[13354]: EA9DB3DA4B: client=localhost[127.0.0.1]
Mar  3 14:09:42 ns31057 postfix/cleanup[13485]: EA9DB3DA4B: message-
id=<20060403120912.32E449B261@sgem09.ec-nantes.fr>
Mar  3 14:09:42 ns31057 postfix/qmgr[26829]: EA9DB3DA4B: from=<root@sgem09.ec-nantes.fr>, size=2575,
nrcpt=1 (queue active)
Mar  3 14:09:42 ns31057 postfix/virtual[28314]: EA9DB3DA4B: to=<fmicaux@actilis.net>, relay=virtual,
delay=1, status=sent (delivered to maildir)
Mar  3 14:09:42 ns31057 postfix/qmgr[26829]: EA9DB3DA4B: removed
```


Notes personnelles

12.9- Configuration de base de Postfix

12.9.1- Lister et modifier la configuration

Le fichier **/etc/postfix/master.cf** permet d'instancier les agents de Postfix.

On y déclare les agents et leurs paramètres

Chaque agent peut y être référencé plusieurs fois, avec des paramètres différents

La configuration standard de master.cf est suffisante pour démarrer un service simple.

Y compris du point de vue tuning et réglage de charge

Les directives de configuration communes sont déclarées dans le fichier **/etc/postfix/main.cf**.

Attention :
La commande "**postconf**" permet de lister et modifier les directives (elle ne lit et modifie que "main.cf").

Dans "**master.cf**", on peut surcharger des directives dont la valeur par défaut est définie dans "**main.cf**".

Notes personnelles

12.9.2- Paramètres de base

Nom d'hôte local : myhostname
Il doit pointer vers le serveur et est par défaut obtenu par la fonction gethostname(). Il et est utilisé comme valeur par défaut pour d'autres paramètres.

```
myhostname = mail.actilis.net
```

Nom de domaine local : mydomain :
Par défaut, **mydomain = $myhostname** moins le premier composant.

```
mydomain = actilis.net
```

D'où semblent provenir les mails postés localement : myorigin
Par défaut : $myhostname

Si on veut qu'apparaisse uniquement notre "nom de domaine" :

```
myorigin = $mydomain
```

Interfaces réseau : inet_interfaces
Interfaces réseau sur lesquelles nous écoutons et acceptons la réception et le relais de mails.
Par défaut : inet_interfaces = all

```
inet_interfaces = localhost, 192.168.X.Y
```


Notes personnelles

Quels noms de domaines sont considérés "locaux" : mydestination

C'est une liste de noms acceptés "localement" derrière le signe "@"

```
mydestination = $myhostname, $mydomain, localhost.$mydomain, localhost
```

Qui sont les clients autorisés pour l'envoi SMTP : mynetworks

C'est une liste de réseaux pour lesquels nous acceptons de relayer le courrier qui ne nous est pas destiné.

```
mynetworks = 127.0.0.0/8 [::1]/128
```

Stockage des boites à lettres : home_mailbox

Spécifie l'emplacement et le format des boites à lettres :
Terminée par un "/", c'est le format "Maildir" qui est utilisé, sinon c'est le format "mbox".

```
home_mailbox = Mail/
```


Notes personnelles

Une table de contrôle d'en-tête : header_checks

L'option **header_checks** donne le format et l'emplacement de tables permettant le filtrage de messages sur la base du contenu des champs d'en-têtes.

La syntaxe utilisée est celle des expressions régulières standard (**regexp**) ou compatible PERL (pcre).

```
header_checks = regexp:/etc/postfix/blacklist
```

Voici un extrait de table :

```
/^from: .+@.hotmail.com/     REJECT
```


Notes personnelles

12.10- Démarrage et maintenance de Postfix

Le script "postfix" est dans **/etc/init.d**, le service peut alors être démarré par la commande suivante :

```
# service postfix start
```

12.10.1- Automatisation du démarrage

Pour permettre au système de démarrer automatiquement Postfix au prochain reboot, utiliser la commande **chkconfig.**

Le service est paramétré pour démarrer dans les runlevels 2,3,4 et 5

```
# chkconfig --level 2345 postfix on
```

et s'arrêter dans les runlevels 0, 1 et 6

```
# chkconfig --level 016 postfix off
```

Vérification :

```
# LANG=C chkconfig --list postfix
postfix         0:off   1:off   2:on    3:on    4:on    5:on    6:off
```


Notes personnelles

12.10.2- Maintenance de Postfix & processus

12.10.2.1- Processus visibles en permanence

On doit le plus souvent trouver (au moins) ces processus lancés :

```
# ps -fauxw | grep postfix
root       382  0.0  1.0  3740 1288 ?        S    12:19   0:00 /usr/lib/postfix/master
postfix    384  0.0  1.0  3764 1284 ?        S    12:19   0:00  \_ pickup -l –t fifo
postfix    385  0.0  1.1  3900 1416 ?        S    12:19   0:00  \_ qmgr -l -t fifo -u
```

Seul le premier processus (master) est lancé sous l'identité de "root".
Les processus fils du serveur Postfix sont lancés sous l'identité de l'utilisateur "postfix".

12.10.2.2- Listage de la file d'attentes

La commande **postqueue** permet de lister et manipiler la file d'attente :

postqueue -p : listage
postqueue -f : flush : tentative de vidage (par forçage du traitement d'acheminement)
postqueue -i NUMERO : tentative de livraison du courrier spécifié.
postqueue -s DOMAINE : tentative de livraison de tout courrier en attente vers DOMAINE.

La commande postsuper permet de nettoyer la file d'attente :

postsuper -d ALL : vidage de toute la file d'attente
postsuper -d NUMERO : suppression du courrier spécifié.

Notes personnelles

12.11- Dovecot : un serveur POP / IMAP

Très documenté et fourni avec des exemples de configuration, Dovecot est simple à prendre en main.

Le package fourni par CentOS est "dovecot" :

```
# yum -y install dovecot
```

Le fichier de configuration est **/etc/dovecot/dovecot.conf**.

Celui-ci utilise l'inclusion des fichiers du répertoire **/etc/dovecot/conf.d**.

12.11.1- Emplacement des boîtes à lettres

Le {premier|seul} fichier à éditer est **/etc/dovecot/conf.d/10-mail.conf**, pour y déclarer :

mail_location : Format et emplacement des boites à lettres, (à aligner avec home_mailbox côté Postfix)

```
mail_location = maildir:~/Mail
```

Au démarrage, le serveur dovecot offre les services POP, IMAP, POPS et IMAPS.

```
# service dovecot restart
```


Notes personnelles

12.11.2- Méthode d'authentification et réseau de confiance

Par défaut, l'authentification "plaintext" est désactivée,

sauf si le client est "localhost" ou membre de "réseaux de confiance".

Deux solutions pour permettre à un client de se connecter en authentification "en clair" :

Dans /etc/dovecot/conf.d/10-auth.conf :

```
disable_plaintext_authentication = yes ==> no
```

ou

Dans /etc/dovecot/dovecot.conf :

```
login_trusted_networks = 192.168.0.0/24 10.0.0.0/8 ...
```

Puis redémarrage.

Notes personnelles

13- Annexe : Serveur FTP

Notes personnelles

13.1- Le Serveur VSFTPD

Red Hat Entreprise Linux est fournie avec deux serveurs FTP :

tux (intégré dans le noyau), qui propose un service HTTP et FTP (anonyme uniquement)

vsftpd : le serveur FTP choisi par défaut.

Le serveur **vsftpd** est rapide et sécurisé, et accepte le mode anonymous ou les connexions authentifiées. Il s'appuie sur les mécanismes standards d'authentification (PAM).

Le paquetage vsftpd est prêt à fonctionner dès l'installation et le démarrage du service :

par défaut, l'accès anonyme est activé.

```
$ ftp 10.44.56.160
Connected to 10.44.56.160.
220 (vsFTPd 2.2.2)
Name (10.44.56.160:fmicaux): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls -l
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x    2 0        0            4096 Mar 04  2011 pub
226 Directory send OK.
ftp> pwd
257 "/"
```


Notes personnelles

13.1.1- Le fichier de configuration

C'est **/etc/vsftpd/vsftpd.conf**

On y indique lignes de la forme suivante

```
# commentaire
directive=valeur
```

Les lignes par défaut non commentées sont les suivantes :

```
# grep -v ^# /etc/vsftpd/vsftpd.conf
anonymous_enable=YES
local_enable=YES

write_enable=YES
local_umask=022

dirmessage_enable=YES
xferlog_enable=YES
connect_from_port_20=YES
xferlog_std_format=YES

listen=YES

pam_service_name=vsftpd
userlist_enable=YES
tcp_wrappers=YES
```


Notes personnelles

13.1.2- Activer / Inhiber l'accès anonyme

C'est la directive **anonymous_enable** qui contrôle cette fonctionnalité. Par défaut, elle vaut "YES".

Le répertoire dans lequel on se trouve est **/var/ftp** (le HOME du compte "ftp"). On y est "chrooté".

13.1.3- Activer / Inhiber l'accès authentifié

Il s'agit d'accepter les connexions pour les utilisateurs déclarés sur le système.

C'est la directive **local_enable** qui contrôle cette fonctionnalité.

Par défaut, elle vaut "YES", mais un problème de changement de répertoire peut se produire après l'authentification, lié à SELinux, qui interdit au service d'accéder à un autre répertoire que **/var/ftp**.

```
$ ftp 10.44.56.160
Connected to 10.44.56.160.
220 (vsFTPd 2.2.2)
Name (10.44.56.160:fmicaux): fmicaux
331 Please specify the password.
Password:
500 OOPS: cannot change directory:/home/fmicaux
Login failed.
```

On peut passer SELinux en mode "Permissif" et vérifier que cela fonctionne.

```
# setenforce Permissive
```


Notes personnelles

13.1.4- Permissions d'écriture

Les directives **write_enable=YES** autorise l'écriture (donc l'upload) sur le serveur FTP.

La directive **local_umask=022** définit les permissions portées par les fichiers et répertoires créés sur le serveur.

Pourtant, en accès FTP, on ne peut pas écrire, ni dans le répertoire d'accueil, ni dans le répertoire "pub". Cela est du aux permissions Unix portées par le répertoire /var/ftp (et /var/ftp/pub)

Pour activer un répertoire d'upload en mode anonyme, il faut activer les options suivantes :
pour les transferts de fichiers : anon_upload_enable=YES
pour les créations de répertoires : anon_mkdir_write_enable=YES

Cela ne suffit pas, car le répertoire d'upload doit être "writable" par l'utilisateur Unix "ftp".

SELinux empêche tout de même d'écrire vers le serveur, car on sort des policies définies par RedHat.

Un passage en mode "Permissif" montre bien que cela fonctionne.

Notes personnelles

Index lexical

A & AAAA....28
Ab....107
AccessFileName....87
Alias....74
AllowOverride....88
APIPA....174
Autobench....109
bench2graph....109
CCTLD....22
contexte applicatif....11
Directory....76
DirectoryIndex....74
DirectoryMatch....76
DocumentRoot....74
en-têtes HTTP ....16 sv
ErrorDocument....74
Files....76
FQDN....22
GTLD....22
GTLD-servers....26
header_checks....210
home_mailbox....209
Httperf....109
ICMP....182
identifiant de session....11
ifconfig....172
IMAP....201
IMAPS....201
inet_interfaces....208
Keepalive....99
KeepaliveTimeout....99
LOC....33
Location....74, 76
LogLevel....56
logresolve....90
mactab....170
MaxKeepaliveRequests....99
MaxSpareThreads....104
MDA....195
méthode HTTP....16
MinSpareThreads....104
modules....66
MRA....195
MSA 195
MTA....195
MUA....195
MX....31
mydestination....209
mydomain....208
myhostname....208
mynetworks....209
myorigin....208
nameif....170
NameVirtualHost....127
NS....30
ping....182
POP3....199
POP3S....199
ProxyTimeout....98
PTR....34
Redirect....82
RedirectMatch....83
RedirectPermanent....83
RedirectTemp....83
RequireAll....116
RequireAny....116
RequireNone....116
Resource Records....28
root-servers....24
ServerAlias....127
ServerLimit....104
ServerName....127
session....11
SMTP....197
SOA....29

Notes personnelles

SRV.....32
ThreadLimit.....104
ThreadsPerChild.....104
Timeout.....98
TLD.....22
VirtualDocumentRoot.....131
VirtualDocumentRootIP.....131
VirtualHost.....125, 127
VirtualScriptAlias.....131
VirtualScriptAliasIP.....131
vsftpd 216
ZeroCONF.....174
.htaccess.....76, 87
/etc/hosts.....21
/etc/nsswitch.conf.....20
/etc/resolv.conf.....21

Notes personnelles