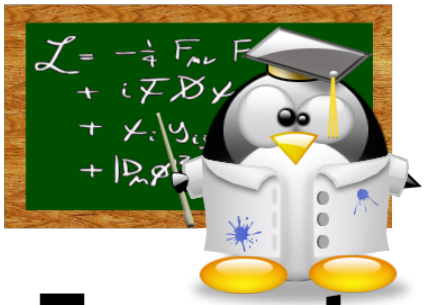


FORMATUX - Support de cours GNU/Linux

Les services sur CentOS et Debian

Version 1.4, 18-08-2017

Table des matières

Préface

GNU/Linux est un **système d'exploitation** libre fonctionnant sur la base d'un **noyau Linux**, également appelé **kernel Linux**.

Linux est une implémentation libre du système **UNIX** et respecte les spécifications **POSIX**.

GNU/Linux est généralement distribué dans un ensemble cohérent de logiciels, assemblés autour du noyau Linux et prêt à être installé. Cet ensemble porte le nom de "**Distribution**".

- La plus ancienne des distributions est la distribution **Slackware**.
- Les plus connues et utilisées sont les distributions **Debian**, **RedHat** et **Arch**, et servent de base pour d'autres distributions comme **Ubuntu**, **CentOS**, **Fedora**, **Mageia** ou **Manjaro**.

Chaque distribution présente des particularités et peut être développée pour répondre à des besoins très précis :

- services d'infrastructure ;
- pare-feu ;
- serveur multimédia ;
- serveur de stockage ;
- etc.

La distribution présentée dans ces pages est la CentOS, qui est le pendant gratuit de la distribution RedHat. La distribution CentOS est particulièrement adaptée pour un usage sur des serveurs d'entreprises.

Crédits

Ce support de cours a été rédigé par les formateurs :

- Patrick Finet ;
- Antoine Le Morvan ;
- Xavier Sauvignon ;
- Nicolas Kovacs.

Licence

respect des droits de l'auteur.

BY : Paternité. Vous devez citer le nom de l'auteur original.

SA : Partage des Conditions Initiales à l'Identique.

- Licence Creative Commons-BY-SA : https://creativecommons.org/licenses/by-sa/3.0/fr/
- Licence Art Libre : http://artlibre.org/

Les documents de Formatux et leurs sources sont librement téléchargeables sur framagit :

- https://framagit.org/alemorvan/formatux.fr-support/

Vous y trouverez la dernière version de ce document.

A partir des sources, vous pouvez générer votre support de formation personnalisé. Nous vous recommandons le logiciel AsciidocFX téléchargeable ici : http://asciidocfx.com/

Gestion des versions

Table 1. Historique des versions du document

Version	Date	Observations
1.0	Avril 2017	Version initiale.
1.1	Juin 2017	Ajout des cours Nginx et Php-fpm.
1.2	Juin 2017	Ajout des cours MySQL et MySQL Master/Master.
1.3	Aout 2017	Ajout du cours Apache LAMP sous CentOS 7 (Nicolas Kovacs).
1.4	Février 2018	Ajout du cours Varnish

Chapitre 1. Network File System

NFS (Network File System) est un système de partage de fichiers via montage réseau.

1.1. Généralités

NFS est basé sur un fonctionnement client-serveur : le serveur met à disposition des ressources pour tout ou partie du réseau (clients).

Le dialogue entre les clients et le serveur se fait grâce aux services RPC (Remote Procedure Call ou procédure d'appel à distance).

Les fichiers distants sont montés dans un répertoire et apparaissent comme un système de fichiers local. Les utilisateurs clients accèdent en toute transparence aux fichiers partagés par le serveur, en parcourant les répertoires comme s'ils étaient locaux.

1.2. Installation

2 services sont nécessaires au fonctionnement de NFS :

- Le service network ;
- Le service rpcbind.

L'état des services peut être visualisé par les commandes :

```
service rpcbind status
service network status
```

Si le paquet nfs-utils n'est pas installé :

```
yum install nfs-utils
```

Le paquet nfs-utils nécessite plusieurs dépendances dont nfs-utils-lib et rpcbind pour s'installer.

Le service NFS peut être démarré :

```
chkconfig nfs on
service nfs start
```

L'installation du service NFS crée deux utilisateurs :

- nfsnobody : utilisé lors des connexions anonymes ;
- rpcuser : pour le fonctionnement du protocole RPC.

1.3. Configuration du serveur

Les droits du répertoire et les droits NFS doivent être cohérents.

Le fichier /etc/exports

Le paramétrage des partages de ressources s'effectue dans le fichier /etc/exports. A une ligne de ce fichier correspond un partage NFS.

Syntaxe du fichier /etc/exports

```
/partage    client1(permissions) client2(permissions)
```

- **/partage** : Chemin **absolu** du répertoire partagé ;
- **clients** : Machines autorisées à accéder aux ressources ;
- **(permissions)** : Permissions sur les ressources.

Les machines autorisées à accéder aux ressources peuvent être déclarées par :

- Adresse IP : 192.168.1.2
- Adresse réseau : 192.168.1.0/255.255.255.0 ou au format CIDR 192.168.1.0/24
- FQDN : client_*.formatux.lan : autorise les FQDN commençant par client_ du domaine formatux.lan ;
- * pour tout le monde.

Plusieurs clients peuvent être spécifiés sur la même ligne en les séparant par un espace.

Permissions sur les ressources

Il existe deux types de permissions :

- ro : lecture seule ;
- rw : modification.

Si aucun droit n'est précisé, alors le droit appliqué sera lecture seule.

Par défaut les UID et GID des utilisateurs du client sont conservés (excepté pour root).

Pour forcer l'utilisation d'un UID ou d'un GID différent de l'utilisateur qui écrit la ressource, il faudra spécifier les options anonuid=UID et anongid=GID ou donner un accès anonyme aux données avec l'option **all squash** (squash dans sens d'écraser).

Il existe un paramètre, no_root_squash, qui permet d'identifier l'utilisateur root du client comme étant celui du serveur. Ce paramètre peut être dangereux pour la sécurité du système.

Par défaut, c'est le paramètre root_squash qui est activé (même si non précisé), identifiant root comme utilisateur anonyme.

Cas concrets

- /partage client(ro,all_squash)

Les utilisateurs du client n'ont accès qu'en lecture seule aux ressources et sont identifiés comme anonyme sur le serveur.

- /partage client(rw)

Les utilisateurs du client peuvent modifier les ressources et gardent leur UID sur le serveur. Seul root est identifié comme anonyme.

- /partage client1(rw) client2(ro)

Les utilisateurs du poste client1 peuvent modifier les ressources tandis que ceux du poste client2 n'ont qu'un accès en lecture seule.

Les UID sont gardés sur le serveur, seul root est identifié comme anonyme.

- /partage client(rw,all_squash,anonuid=1001,anongid=100)

Les utilisateurs du poste client1 peuvent modifier les ressources. Leurs UID sont transformés en 1001 et leur GID en 100 sur le serveur.

La commande exportfs

La commande exportfs (exported file systems) permet de gérer la table des fichiers locaux partagés avec les clients NFS.

Syntaxe de la commande exportfs

```
exportfs [-a] [-r] [-u partage] [-v]
```

Table 2. Options de la commande exportfs

Option	Description
-a	Active les partages NFS
-r	Prend en compte les partages du fichier /etc/exports
-u partage	Désactive un partage donné
-v	Affiche la liste des partages

La commande showmount

La commande showmount permet de surveiller les clients.

Syntaxe de la commande showmount

```
showmount [-a] [-e] [hôte]
```

Table 3. Options de la commande showmount

Option	Description
-e	Affiche les partages du serveur désigné
-a	Affiche tous les partages en cours sur le serveur

Cette commande permet aussi de savoir si le poste client a l'autorisation de monter les ressources partagées.

"showmount" trie et supprime les doublons dans les résultats (sort|uniq), il est donc impossible de déterminer si un client a fait plusieurs montages d'un même répertoire.

1.4. Configuration du client

L'accès aux ressources partagé d'un serveur NFS se fait par point de montage sur le client.

Si besoin, créer le dossier local pour le montage :

```
[root]# mkdir /mnt/nfs
```

Lister les partages NFS disponibles du serveur :

```
[root]# showmount –e 172.16.69.237
/partage *
```

Monter le partage NFS du serveur :

```
[root]# mount –t nfs 172.16.69.237:/partage /mnt/nfs
```

Le montage peut aussi être automatisé au démarrage du système dans le fichier /etc/fstab :

```
[root]# vim /etc/fstab
172.16.69.237:/partage /mnt/nfs nfs defaults 0 0
```

Chapitre 2. Serveur de noms DNS - Bind

Le DNS (Domain Name System) est un système de résolution de nom.

Il s'agit d'une base de données distribuée hiérarchique, dont la racine est symbolisée par un « . ».

L'interrogation de cette base se fait selon le principe de client-serveur.

L'URL www.formatux.lan est appelée FQDN (Fully Qualified Domain Name).

Une table de correspondances permet la traduction d'un FQDN en informations de plusieurs types qui y sont associées, comme par exemple son adresse IP.

2.1. Généralités

Il existe 13 serveurs racines répartis dans le monde, dont une majorité se situe en Amérique du Nord.

La traduction d'un FQDN en adresse IP est le cas que l'on rencontre le plus fréquemment mais DNS permet également de renseigner le système sur les serveurs de messagerie, sur les services disponibles, de faire des alias de noms de service, etc.

Par exemple, il est possible de proposer un FQDN smtp.domain.tld sans qu'il y ait réellement de serveur nommé SMTP.

La résolution est possible en IPv4 comme en IPv6.

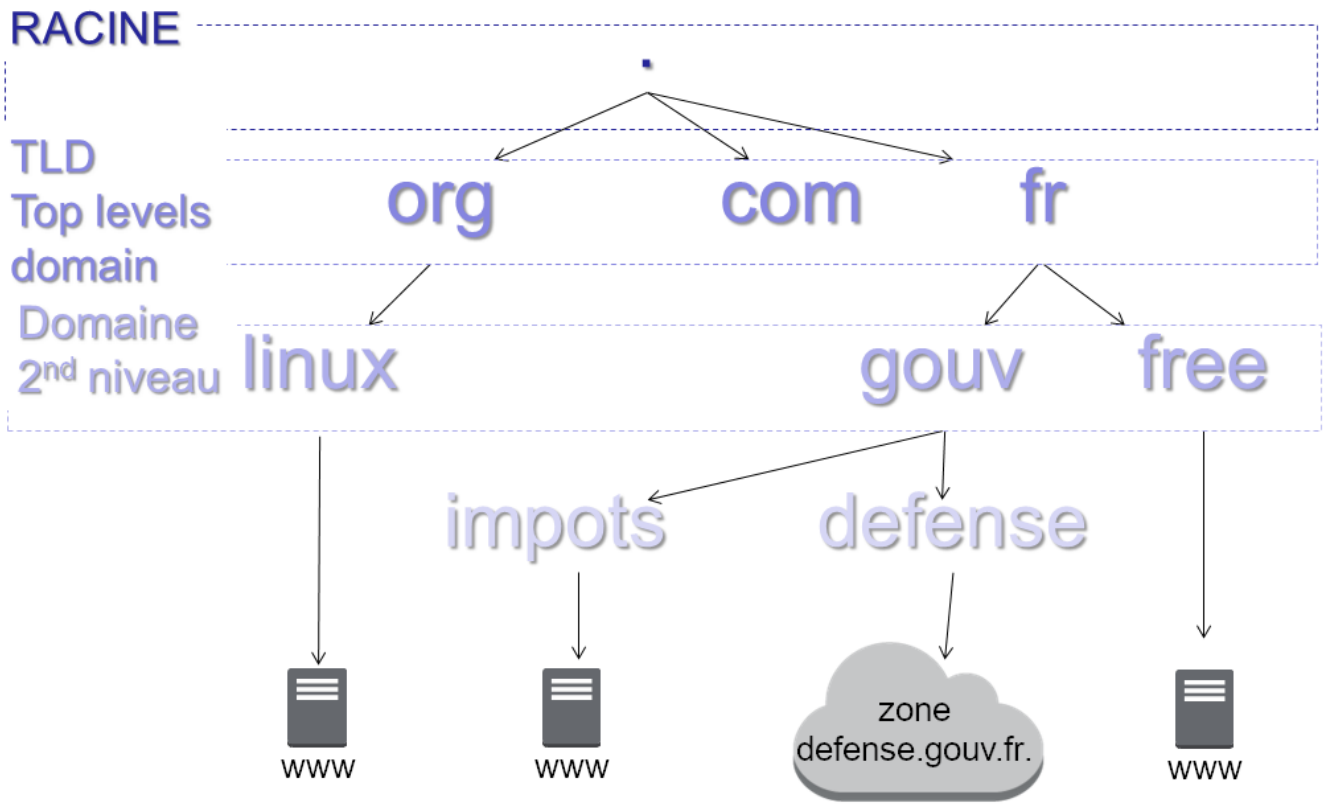


Figure 1. Principe de fonctionnement du DNS

Les Top Levels Domain TLD sont gérés par des organismes bien défini. Les registraires (registars en anglais) de nom de domaine se chargent pour les clients d'enregistrer les noms de domaine auprès de ces organismes.

Le dialogue se fait en général via le protocole UDP (User Datagram Protocol) mais parfois aussi en TCP sur le port 53.

BIND (Berkeley Internet Name Daemon) est un serveur DNS populaire tournant sur système UNIX / Linux.

Bind est disponible sur les serveurs RedHat :

- RHEL 5 : Version 9.3
- RHEL 6 : Version 9.8
- RHEL 7 : Version 9.9

Le protocole peut aussi être utilisé en TCP (connecté) dans certains cas : transferts vers le ou les serveurs secondaires, requêtes dont la réponse est supérieure à la taille d'un paquet UDP, etc.

2.2. Installation du service

Le serveur DNS BIND s'articule autour du service named.

Installation à partir d'un dépôt YUM :

```
[root]# yum install bind
```

L'installation du service BIND ajoute un utilisateur named. Cet utilisateur sera le propriétaire des fichiers de configuration.

```
[root]# grep named /etc/passwd
named:x:25:25:Named:/var/named:/sbin/nologin
```

BIND étant un service réseau, il faut le paramétrer pour un démarrage dans les niveaux 3 et 5, puis le démarrer :

```
[root]# chkconfig --level 35 named on
[root]# service named start
```

Il peut être utile d'installer le paquet bind-utils pour disposer d'outils de test DNS.

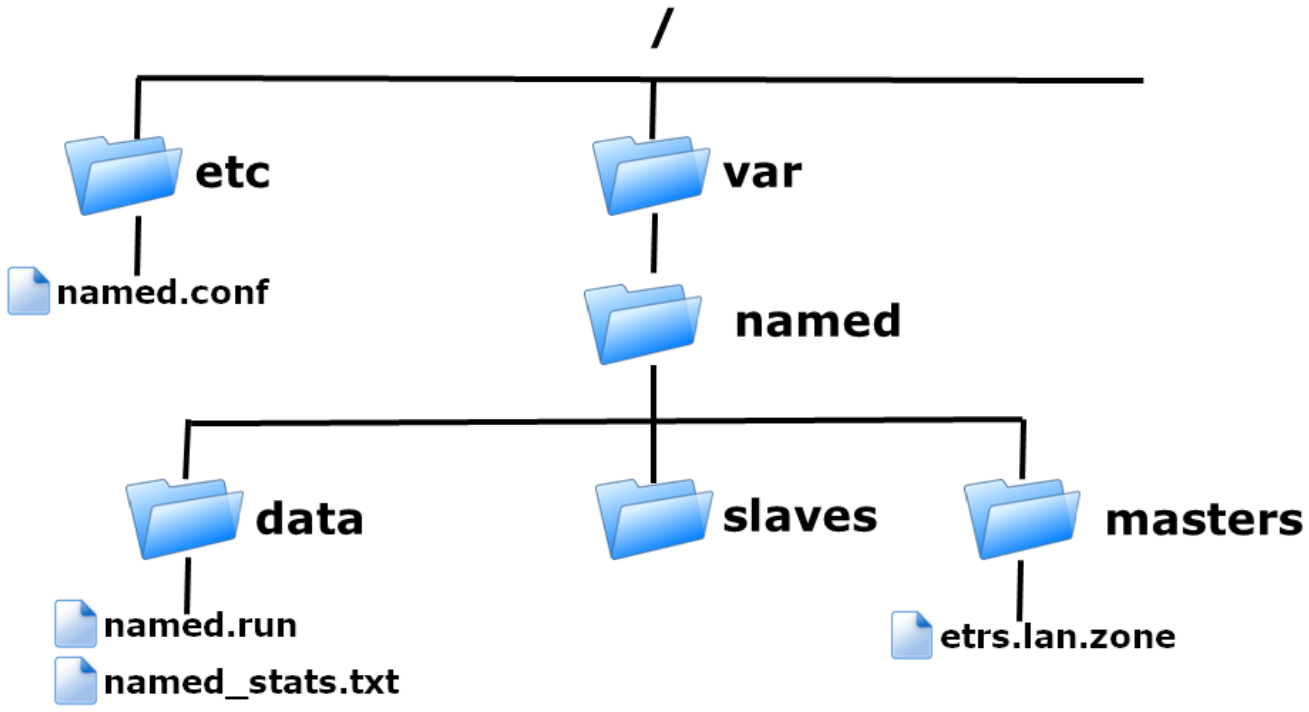


Figure 2. Arborescence du service Bind

Le dossier /var/named/masters n'est pas créé par défaut. Il faudra le créer et ne pas oublier d'y attribuer les droits à l'utilisateur named.

Le fichier de statistiques est généré par la commande rdnc (voir en fin de chapitre).

En environnement de production, les logs des requêtes clientes seront séparés des logs du service lui-même.

Cache DNS

Par défaut, Bind est configuré pour faire office de serveur de proxy-cache DNS (mandataire). Un serveur proxy, ou mandataire, effectue une requête réseau au nom d'un client.

Lorsqu'il résout pour la première fois une requête DNS, la réponse est stockée dans son cache pour la durée de son TTL (Time To Live) restant. Si le même enregistrement est à nouveau demandé par un autre client DNS, le traitement de la requête sera plus rapide, puisque la réponse sera disponible depuis le cache du serveur.

Chaque enregistrement DNS dispose d'un TTL lorsqu'il est fourni par le serveur qui fait autorité pour la zone. Ce TTL est décrémenté jusqu'à la fin de sa validité. Il est ensuite supprimé des caches.

Lorsque l'enregistrement est mis en cache, le TTL qui lui est associé est le TTL restant.

Récursivité

Un serveur est configuré par défaut pour répondre de manière récursive aux requêtes de ses clients.

Il interroge tour à tour les serveurs DNS nécessaires à la résolution d'une requête jusqu'à obtenir la réponse.

Un serveur non-récursif déléguera la résolution du nom DNS à un autre serveur DNS.

Dans quel cadre utiliser un serveur non-récursif ?

Lorsque la latence entre le serveur DNS et le reste du réseau est trop forte, ou quand le débit est limité, il peut être intéressant de configurer un serveur DNS en non-récursif.

Ce sera par exemple le cas pour un site distant ou un élément mobile.

Le serveur DNS local fera autorité sur la zone locale, mais déléguera la résolution des FQDN des autres zones à un serveur distant, qui lui, prendra la requête en charge de manière récursive.

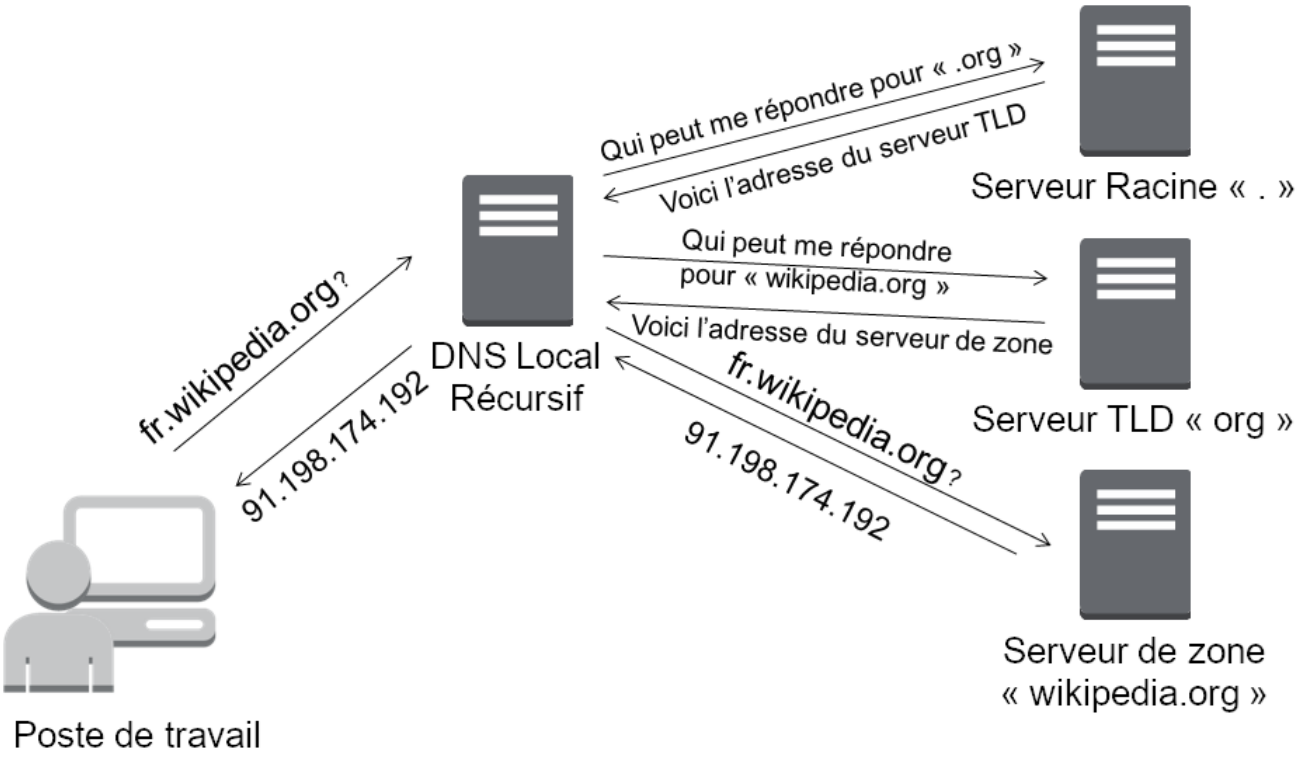


Figure 3. Le mode récursif

Architecture DNS

Un serveur DNS principal dispose d'une copie en écriture de la zone.

Il peut être intéressant de ne le rendre accessible que depuis le domaine local, et ne permettre l'interrogation des DNS depuis l'extérieur que vers les serveurs secondaires. D'un point de vue architectural cela lui évite ainsi les attaques ou les surcharges.

Le serveur est alors appelé serveur furtif.

Un serveur DNS n'est pas nécessairement le serveur maître de toutes les zones pour lesquels il fait autorité.

Un serveur DNS peut très bien être configurer pour être maître d'une zone et esclave d'une autre.

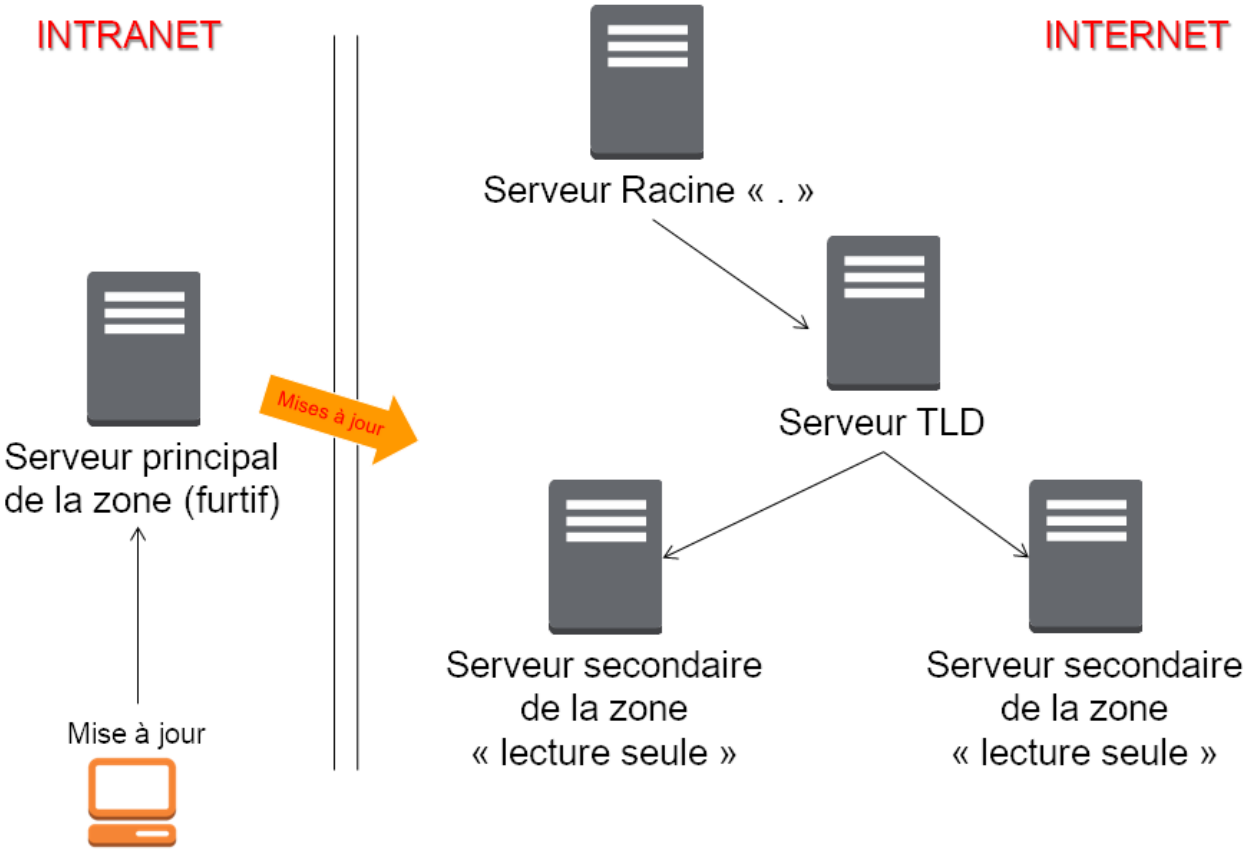


Figure 4. Architecture avec serveur furtif

La mise à jour du serveur se fait depuis le réseau local. Le serveur principal n'est pas accessible depuis l'extérieur.

La mise à jour se propage vers les serveurs secondaires.

L'interrogation par les clients internet ne se fait que sur les serveurs secondaires, ce qui protège le serveur principal des attaques par déni de service.

Pour un réseau complexe, il existe de nombreuses solutions architecturales de l'infrastructure DNS qu'il est important de bien étudier.

2.3. Configuration du serveur

Le fichier /etc/named.conf

Ce fichier contient les paramètres de configuration du service DNS.

```
[root]# less /etc/named.conf
options {
    listen-on port 53 { 192.168.1.200; };
    directory "/var/named";
    allow-query { 192.168.1.0/24; };
};
```

Chaque ligne du fichier /etc/named.conf (même à l'intérieur des accolades) se termine par un point-virgule.

L'oublie de ce ";" est l'erreur la plus fréquente dans la configuration d'un serveur Bind.

Les noms, sous la forme FQDN (Fully Qualified Domain Name) doivent se terminer par ".". En l'absence de ce ".", Bind suffixera automatiquement avec le nom de domaine l'enregistrement.

Eg : www.formatux.lan → www.formatux.lan.formatux.lan.

La rubrique options contient la configuration générale du serveur BIND via différentes directives :

- listen-on : Définit l'interface, l'adresse et le port du service sur le serveur.
- directory : Définit le répertoire de travail de BIND, contenant les fichiers de zone.
- allow-query : Définit les hôtes autorisés à faire des requêtes sur le serveur. Par adresse IP ou réseau.

Permettre qu'un serveur DNS résolve les requêtes de n'importe quel client est une très mauvaise idée. Il faut au moins restreindre les droits aux réseaux locaux.

Il est possible, pour cela, de créer des ACL pour simplifier l'administration du fichier de configuration.

Le fichier de configuration contient également les informations relatives aux fichiers de zone.

```
[root]# less /etc/named.conf
zone "formatux.lan" IN {
    type master;
    file "masters/formatux.lan.direct";
    allow-update { 192.168.1.0/24; };
};

zone "1.168.192.in-addr.arpa" IN {
    type master;
    file "masters/formatux.lan.inverse";
};
```

Les rubriques **zone** contiennent les configurations des zones de résolution de nom, inverses ou directes :

- type : Définit le type de serveur pour cette zone :
  - maître : possède la base de données en écriture
  - esclave : possède la base en lecture seule
  - forwarder : fait office de proxy-cache pour cette zone.
- file : Définit le chemin du fichier de zone.
- allow-update : Définit les hôtes ayant l'autorisation de mettre à jour les enregistrements DNS.

Les fichiers de zone inverse sont nommés en prenant l'adresse réseau de la zone (en inversant les octets) suivi du domaine in-addr.arpa.

Les rôles

Un serveur peut avoir le rôle de maître pour la zone, c'est-à-dire qu'il possède la zone en écriture.

```
[root]# less /etc/named.conf
zone "formatux.lan" IN {
    type master;
    file "masters/formatux.lan.direct";
    allow-update { 192.168.1.0/24; };
};
```

Seuls les clients figurant dans la variable allow-update pourront mettre à jour la base de données du DNS.

Un serveur peut également être un serveur secondaire (slave) pour la zone, c'est-à-dire qu'il possède la zone en lecture.

```
[root]# less /etc/named.conf
zone "formatux.lan" IN {
    type slave;
    file "slaves/formatux.lan.direct";
};
```

Un serveur peut enfin être expéditeur (forwarder) pour la zone, c'est-à-dire qu'il a connaissance de cette zone, et relaie les informations pour celle-ci.

```
[root]# less /etc/named.conf
zone "unautredomaine.fr" IN {
    type forwarder;
    forwarders {221.10.12.1};
};
```

Vous retrouverez cette notion sous Windows en tant que « redirecteur ».

2.4. Fichiers de zone

Les fichiers présents dans le répertoire /var/named doivent appartenir à l'utilisateur système named.

SELinux ne permettra pas l'enregistrement des fichiers de zone en dehors de ce répertoire.

Ces fichiers contiennent des enregistrements (RR : Resource Records) DNS de différents types. Ils permettent la résolution directe de noms (du nom vers l'adresse IP), ou la résolution inverse (de l'adresse IP vers le nom).

En plus de contenir les adresses IP et les noms des machines, les fichiers contiennent les paramètres de durée de vie des enregistrements (Time To Live, TTL).

Lorsqu'un enregistrement DNS est mis en cache, le temps restant sur son TTL est également conservé. À la fin du TTL, l'enregistrement est supprimé des caches.

- Un TTL plus long réduit les échanges DNS.
- Un TTL plus court permet une reconfiguration du réseau plus rapide.

Les types d'enregistrements

Table 4. Les types d'enregistrements

Type	Description
A	Nom attribué à une adresse de type IP V4

Type	Description
AAAA	Nom attribué à une adresse de type IP V6
CNAME	Alias d'un enregistrement A déjà défini **Éviter de faire un alias vers un alias**
MX	Serveur de messagerie destinataire pour la zone concernée
NS	Le ou les serveurs de noms de la zone (type A)
PTR	Enregistrement pointeur pour une zone inverse
SOA	Démarre la configuration (cf: diapos suivantes)
SVR	Service (protocole jabber,...)
TXT	Informations

- Champ MX : Le numéro précise la priorité, la plus faible étant la plus prioritaire. Ceci permet de définir un ou plusieurs serveurs de secours qui stockeront les mails en attendant le retour du serveur principal.
- Champ de type A : Enregistrement standard. Attribue un nom à une adresse IP.

Plusieurs enregistrements identiques de type A vers des adresses différentes permet de faire de l'équilibrage de charge par round-robin (RR).

Exemple :

```
mail    A   192.168.1.10
        A   192.168.1.11
```

- Champ AAAA : On utilise quatre A pour symboliser IPv6 car une adresse IPv6 est codée sur 16 octets, soit 4 fois plus qu'une adresse IPv4.
- CNAME : Permet d'attribuer un ou plusieurs alias à un enregistrement A déjà défini. Plusieurs enregistrements du même alias permettent également de faire de l'équilibrage de charge type RR.

On trouvera des enregistrements typiques, comme autoconfig, qui permet le mécanisme de configuration automatique d'un client de messagerie.

Fichier de zone directe

Ce fichier est nécessaire au fonctionnement du système DNS. C'est par lui que se fait la résolution d'un nom en adresse IP.

```
[root]# less /var/named/formatux.lan.direct
$ORIGIN .
$TTL 3600
formatux.lan.  SOA  inf1-formatux.formatux.lan. contact.formatux.lan. (123; 14400;
3600; 604800; 3600; )

@               IN  NS      inf1-formatux.formatux.lan.
poste1          IN  A       192.168.1.10
inf1-formatux   IN  A       192.168.1.200
formatux.lan.   MX  10      192.168.1.201
inf3-formatux   IN  A       192.168.1.202
www             IN  CNAME   inf1-formatux.formatux.lan.
```

- $ORIGIN : Définit la valeur par défaut du domaine courant pour les renseignements du fichier. Un . signifie la racine.
- $TTL : Durée de vie par défaut des enregistrements de la zone dans le cache, exprimée en secondes. Le TTL peut également être précisé enregistrement par enregistrement.
- SOA : Start Of Authority. La ligne démarre la configuration d'une zone. Définit :
  - le nom du serveur maître principal,
  - l'email de l'administrateur de la zone (un . remplace le @ de l'adresse mail).
  - Entre parenthèses, le numéro de série du fichier (incrémenté à chaque mise à jour) et les délais de mise à jour ou de rafraîchissement, exprimés en secondes.
    - Numéro de zone : Numéro incrémental (voir le paragraphe suivant)
    - Rafraîchissement : Durée en secondes avant une tentative de synchronisation avec le serveur maître
    - Réitération : Intervalle de temps avant réitération si l'essai précédent n'a pas fonctionné
    - Expiration : Durée en secondes avant l'expiration car le serveur maître est injoignable
    - Cache négatif (TTL) : Durée de vie en secondes des enregistrements

Le @ a une signification particulière pour Bind. Il se représente lui même, raison pour laquelle le @ de l'adresse courriel d'administration est remplacée par un .

Le numéro de la zone sert à identifier la dernière modification du DNS maître. Tous les serveurs secondaires utilisent ce numéro pour savoir s'ils doivent se synchroniser.

Il existe deux méthodes d'incrémentation du numéro de zone :

- Incrémentale : 1, puis 2, puis 3 (pourquoi pas ?)
- Basée sur la date : AAAAMMJJXX, qui nous donne par exemple, pour la première modification du jour : 2017210101 (méthode à privilégier)

Le fichier de zone inverse

Bien que non obligatoire, ce fichier est fortement conseillé pour un fonctionnement optimal du système DNS. C'est par lui que se fait la résolution d'une adresse IP en nom.

Des services comme SSH s'appuie sur la résolution inverse.

```
[root]# more /var/named/formatux.lan.inverse
$ORIGIN 1.168.192.in-addr.arpa.
$TTL 259200
@               SOA inf1-formatux.formatux.lan. contact.formatux.lan. ( 123; 14400;
3600; 604800; 3600; )
@               NS      inf1-formatux.formatux.lan.
1   0           PTR     poste1.formatux.lan.
200     PTR     inf1-formatux.formatux.lan.
```

La commande nsupdate

L'usage de la commande nsupdate est exclusive. Il ne faut plus modifier les fichiers de zone manuellement, sous peine de pertes d'informations.

Syntaxe :

```
nsupdate
```

Exemple:

```
[root]# nsupdate
        > server 192.168.1.200
        > zone formatux.lan
        > update add poste2.formatux.lan 3600 A 192.168.1.11
        > update delete poste1
        > send
```

La commande nsupdate est interactive.

À la saisie, elle ouvre un prompt dans lequel il faut saisir les requêtes de mise à jour du fichier de zone.

Ces requêtes peuvent être :

- **server** : Précise le serveur BIND pour lequel les requêtes seront envoyées.
- **zone** : Précise la zone de résolution pour laquelle les requêtes seront envoyées.

- **prereq yxdomain nom** : L'existence de l'enregistrement nom est une condition de mise à jour.
- **update add nom TTL type @IP** : Ajoute l'enregistrement nom, en précisant son type, son adresse IP et son TTL.
- **update delete nom** : Supprime l'enregistrement nom.
- **send** : Valide et envoie les requêtes.

La commande rndc

La commande **rndc** permet de manipuler le serveur DNS à chaud.

Syntaxe de la commande rndc

```
rndc reload
rndc querylog on|off
```

- **reload** : Prend en compte les modifications apportées sans devoir relancer le service
- **querylog** : Active ou non la journalisation

Après modification d'un fichier de zone, il est nécessaire de faire prendre en compte les modifications au service. Les fichiers étant lus au démarrage du service, cela permet de prendre en compte les modifications, mais résulte en la perte des statistiques. Il faut donc privilégier la méthode reload.

Le suivi des logs

La fonction d'enregistrement des fichiers journaux est activée ou désactivée par la commande rndc.

Le fichier de logs est par défaut /var/named/data/named.run

Exemple :

```
[root]# rndc querylog on
[root]# tail –f /var/named/data/named.run
```

Bind propose dans son fichier de configuration des options pour journaliser les informations :

- de transferts,
- de requêtes clients,
- d'erreurs,
- ...

2.5. Configuration du client

NetworkManager est un outil de gestion du réseau. Sur un serveur dont le réseau est défini par cet outil, la configuration cliente de Bind est décrite dans le fichier de l'interface.

```
[root]#less /etc/sysconfig/network-scripts/ifcfg-ethX
DOMAIN="formatux.lan"
DNS1=192.168.1.200
DNS2=192.168.1.201
```

NetworkManager modifiera lui-même le fichier /etc/resolv.conf à chaque relance du service réseau.

Avant de lancer une recherche DNS, le logiciel client vérifiera si la requête porte sur un FQDN ou non. Si le nom n'est pas pleinement qualifié, le client suffixera la requête avec le premier suffixe DNS fourni.

Si la résolution est impossible, le client émettra une nouvelle requête avec le suffixe suivant, ainsi de suite jusqu'à l'obtention d'une réponse.

Par exemple, il est possible de fournir deux suffixes :

```
formatux.fr
formatux.lan
```

Lors d'une requête DNS portant sur, par exemple, portail, une première requête portail.formatux.fr sera faite. En l'absence de réponse positive, une seconde requête sera effectuée portant sur portail.formatux.lan. Pour éviter ce phénomène d'interrogation multiple, il est préférable de fournir une adresse pleinement qualifiée.

Veiller à spécifier au moins deux serveurs DNS pour assurer une redondance en cas de panne du serveur principal.

Sans outil de gestion du réseau, la configuration cliente de Bind est décrite dans le fichier /etc/resolv.conf.

```
[root]# less /etc/resolv.conf
search "formatux.lan"
nameserver 192.168.1.200
nameserver 192.168.1.201
```

NetworkManager, si actif, écrasera les valeurs entrées manuellement dans ce fichier.

L'utilitaire **system-config-network-tui** (**nmtui** sous CentOS 7) permet une configuration graphique

correcte du réseau par une interface ncurse.

La commande dig

La commande **dig** (Domain Information Groper) permet d'interroger des serveurs DNS.

Dig doit être privilégié par rapport à NSLookup qui n'est plus maintenue.

Syntaxe de la commande dig

```
dig [name] [type] [options]
```

Exemple :

```
[root]# dig centos65.formatux.lan A
…
;; QUESTION SECTION:
; centos65.formatux.lan.    IN  A

;; ANSWER SECTION:
centos65.formatux.lan.  86400   IN  A   192.168.253.131

;; AUTHORITY SECTION:
formatux.lan.       86400 IN    NS  centos65.formatux.lan.
…
```

```
[root]# dig –t MX linux.fr
```

```
[root]# dig linux.fr MX +short
```

Mise en cache côté client

Le service NSCD est responsable de la mise en cache des requêtes réseaux type LDAP ou DNS.

Pour profiter de la mise en cache local, il faudra veiller à ce que le service NSCD soit démarré.

```
[root]# service nscd start
[root]# chkconfig nscd on
```

Nscd n'est pas installé par défaut sur les RHEL 6.

Mise à jour dynamique

Les clients peuvent s'enregistrer dynamiquement sur le serveur DNS, ce qui est intéressant dans le cadre d'une attribution de l'adressage IP dynamique avec DHCP.

2.6. Configuration du pare-feu serveur

Les règles iptables a configurer en tcp et udp sont les suivantes :

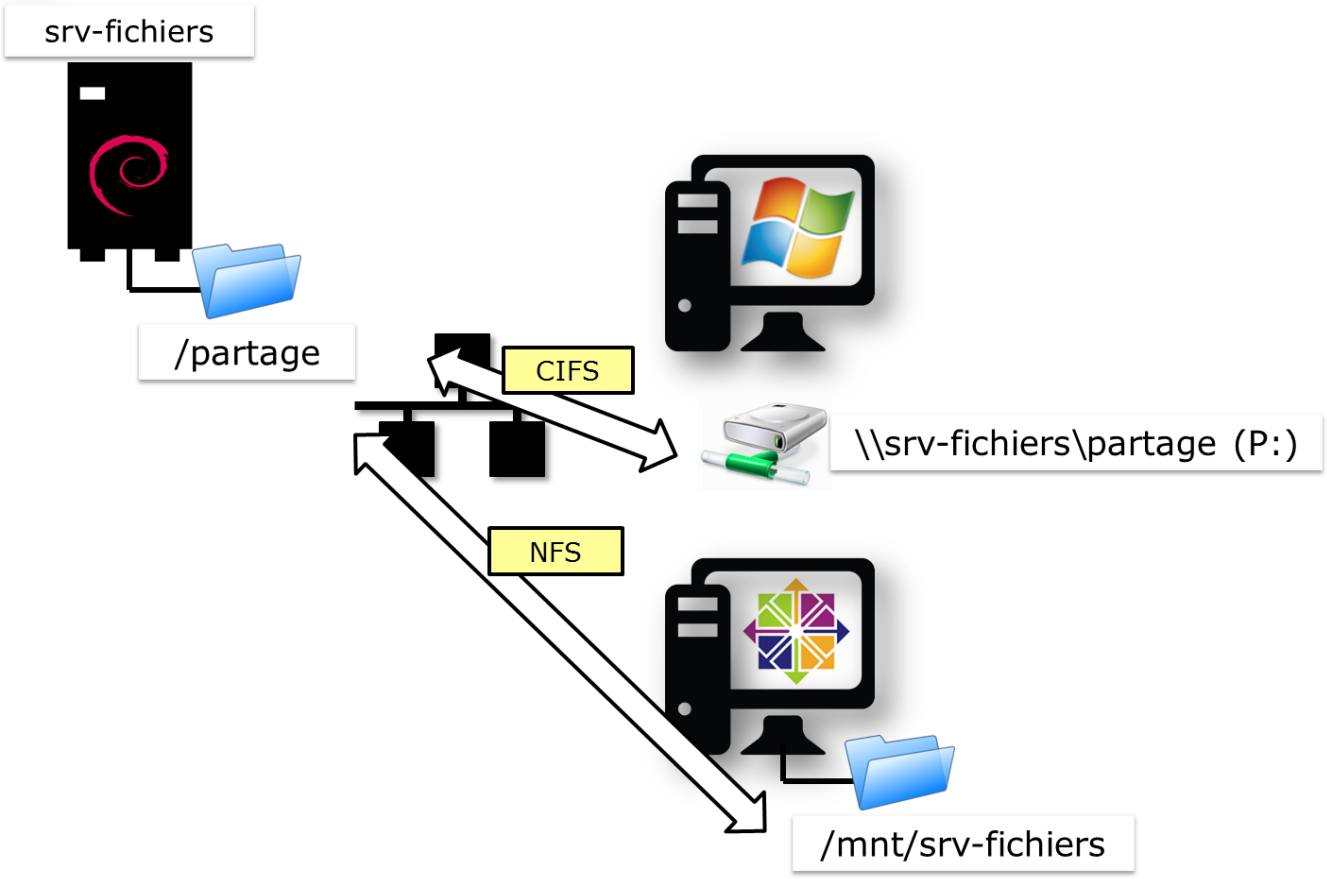
```
[root]# vi /etc/sysconfig/iptables
# Autoriser DNS
iptables –t filter –A INPUT –p tcp –dport 53 –j ACCEPT
iptables –t filter –A INPUT –p udp –dport 53 –j ACCEPT
```

system-config-firewall-tui est l'outil graphique permettant de configurer le pare-feu.

Chapitre 3. Serveur de fichiers Samba

Samba est un serveur de fichiers permettant l'interopérabilité entre divers systèmes, notamment les systèmes Linux et Microsoft. Il permet à des systèmes Linux de créer des partages utilisables par des machines Windows et vice-versa.

Le projet Samba a été initié dès 1992 sous licence GPL (et donc gratuit).



Un même dossier partagé par NFS et par Samba est ainsi accessible depuis toutes les plate-formes clientes.

3.1. Le protocole SMB

Le protocole SMB (Server Message Block) était une extension de Microsoft pour permettre la redirection des entrées/sorties vers NetBIOS (Network Basic Input/Output System).

SMB permettait :

- le transfert de données entre machines Windows : partages de fichiers, impressions et messagerie électronique ;
- le parcours du voisinage réseau (browsing) ;
- la résolution de noms NetBIOS en adresses IP (Windows Internet Name Server) ;
- l'authentification centralisée (notion de domaine).

Le protocole NetBIOS était une interface permettant la mise en place de noms de machines, de groupes de travail, de domaines, etc. Il faisait fonctionner le voisinage réseau jusqu'à Windows 2000, mais son mode de fonctionnement induisait une charge réseau importante.

NetBIOS était le système de noms des réseaux SMB comme l'est aujourd'hui le service DNS.

Les diffusions NetBIOS ne passant pas les routeurs, la notion de voisinage réseau désigne l'ensemble des stations de travail utilisant le protocole NetBIOS sur un même segment de réseau IP. Le **maître explorateur** (master browser) est le poste client ou serveur tenant à jour la liste des ordinateurs utilisés par le service de voisinage réseau.

3.2. Le protocole CIFS

Les améliorations apportées au protocole SMB ont permis de faire aboutir la suite de protocoles clients/serveurs CIFS (Common Internet File System) que le service Samba implémente.

3.3. Installation de Samba

```
[root]# yum install samba smbclient
```

La partie serveur de Samba est basée essentiellement sur 2 démons :

- Le démon **smb** est le serveur SMB : il répond aux requêtes des clients lorsque ceux-ci accèdent aux partages définis et il est le seul à accéder aux systèmes de fichiers Linux.
- Le démon **nmb** est le serveur de noms NetBIOS essentiel au fonctionnement de SMB. Il peut être configuré comme serveur WINS.
- Le fichier de configuration principal est **smb.conf**. C'est dans ce fichier que sont définis les paramètres de fonctionnement des 2 démons, ainsi que la définition des exports de ressources.

La partie cliente de samba (samba-client) contient les outils qui permettent le montage et le parcours des ressources Samba.

Le paquet **samba-client** fournit les binaires :

- **findsmb** : afficher de nombreuses informations sur les stations d'un sous-réseau qui répondent aux requêtes de noms SMB.
- **nmblookup** : interroger le protocole NetBIOS et associe les noms Netbios à des adresses IP.
- **sharesec** : manipuler les droits de partages de fichiers
- **smbcacls** : manipuler les ACL NT sur les partages de fichiers
- **smbclient** : offrir une interface FTP Like pour accéder à des partages de fichiers
- **smbget** : télécharger des fichiers depuis un partage Windows
- **smbspool** : envoyer une impression à un serveur d'impression

- **smbtree** : fournir un explorateur de fichier en mode texte similaire au “Voisinage réseau” des ordinateurs Windows. Il affiche un arbre des domaines connus, des serveurs et des partages accessibles depuis les serveurs.
- ...

Le paquet **samba-common** fournit les binaires :

- **net** : offrir les mêmes fonctionnalités que la commande net du monde Windows.
- **pdbedit** : gérer de la base SAM
- **smbcquotas** : manipuler les quotas NT d’un partage de fichiers
- **smbpasswd** : changer le mot de passe des utilisateurs
- **testparm** : tester la syntaxe d’un fichier de configuration smb.conf
- ...

```
[root]# chkconfig smb on
[root]# chkconfig nmb on
[root]# service smb start
[root]# service nmb start
```

3.4. Sécurité SELinux

Par défaut, la sécurité SELinux est active. Pour vérifier le contexte de sécurité en place sur des fichiers, il faut utiliser la commande :

```
[root]# ls -Zd /export/
```

Le contexte de sécurité *samba_share_t* doit être positionné sur les dossiers partagés :

```
[root]# chcon -R -t samba_share_t /export/
```

Plus d’information sur la sécurité SELinux et Samba sur le site de RedHat. Pensez à stopper le pare-feu ou à le configurer dans le fichier **/etc/sysconfig/iptables** :

```
-A INPUT -m state --state NEW -m udp -p udp --dport 137 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 137 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 138 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 139 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 445 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 445 -j ACCEPT
```

Pour pouvoir utiliser Samba comme contrôleur de domaine et utiliser les commandes useradd et groupadd, il faudra mettre le booléen **samba_domain_controller** à on.

```
[root]# setsebool -P samba_domain_controller on
```

Pour rappel, il est possible d'obtenir tous les booléens SELinux concernant Samba avec la commande suivante :

```
[root]# getsebool -a | grep "samba"
samba_create_home_dirs --> off
samba_domain_controller --> off
samba_enable_home_dirs --> off
samba_export_all_ro --> off
samba_export_all_rw --> off
samba_portmapper --> off
samba_run_unconfined --> off
samba_share_fusefs --> off
samba_share_nfs --> off
sanlock_use_samba --> off
use_samba_home_dirs --> off
virt_use_samba --> off
```

Pour autoriser les partages via Samba des répertoires de connexions des utilisateurs, il faudra positionner le booléen **samba_enable_home_dirs** également à on.

```
[root]# setsebool -P samba_enable_home_dirs on
```

3.5. La configuration de SAMBA

La partie serveur de Samba est basée sur 1 seul fichier de configuration **/etc/samba/smb.conf** qui définit les paramètres de fonctionnement des 2 démons et des exports de ressources.

Chaque paramètre fait l'objet d'une ligne au format :

```
nom = valeur
```

Les commentaires commencent par un ';' ou un '#' et se termine à la fin de la ligne.

Le caractère '\' permet de scinder une ligne logique sur plusieurs lignes physiques.

Ce fichier est constitué de 3 sections spéciales :

- **[global]** : paramètres généraux ;

- **[homes]** : paramètres des répertoires utilisateurs ;
- **[printers]** : paramètres des imprimantes.

Les autres sections, déclarées entre '**[ ]**' sont des déclarations de partage.

Les niveaux de sécurité

Il existe cinq niveaux de sécurité (option security), mais un seul peut être appliqué par serveur :

- **share** : le niveau dit de partage, lié à une ressource. Un mot de passe est associé à chaque partage (Déprécié : ne plus utiliser) ;
- **user** : le niveau de sécurité de l'utilisateur, lié à son authentification. C'est le niveau recommandé et par défaut ;
- **server** : l'authentification est réalisée par un autre serveur (Déprécié : ne plus utiliser) ;
- **domain** : l'authentification est réalisée par un autre serveur, mais le serveur Samba doit être membre du domaine ;
- **ads** : l'authentification est réalisée par un serveur Active Directory.

Les variables internes à Samba

Table 5. Les variables internes à Samba

Variable	Observation
%a	Architecture du client
%I	Adresse IP du client
%M	Nom dns du client
%m	Nom NetBios du client
%u	Identité de l'utilisateur pour le partage concerné
%U	Identité souhaitée par l'utilisateur du partage
%H	Répertoire de connexion de l'utilisateur
%u%g ou %G	Groupe principal de l'utilisateur
%u ou %U%S	Nom du partage
%P	Répertoire racine du partage concerné
%d	PID du processus courant
%h	Nom DNS du serveur Samba
%L	Nom NetBIOS du serveur Samba
%v	Version de samba
%T	Date et heure système
%$var	valeur de la variable d'environnement var

La commande testparm

La commande **testparm** teste la validité du fichier /etc/samba/smb.conf.

L'option -v affiche tous les paramètres applicables.

```
[root]# testparm
Load smb config files from /etc/samba/smb.conf
...
Loaded services file OK.
Server role : ROLE_STANDALONE
...
```

Sans option, cette commande renvoie la configuration du serveur samba sans les commentaires et omet les paramètres positionnés à leur valeur par défaut, ce qui facilite sa lecture.

```
[root]# testparm
```

La section [global]

Table 6. Les directives de la section [global]

Directive	Exemple	Explication
workgroup	workgroup = FORMATUX	Définir le groupe de travail ou le nom de domaine NetBIOS. (À mettre en majuscule).
netbios name	netbios name = inf1-formatux	Nom NetBIOS de la station Maximum 15 caractères Pas de rapport direct avec le nom de la machine Linux
server string	server string = Samba version %v	Description du serveur apparaissant dans l'explorateur Windows
hosts allow	hosts allow = 127. 172.16.1.	Permet de restreindre les clients du serveur aux seuls réseaux mentionnés. Notez la présence d'un point à la fin de l'adresse et l'absence du 0.
log file	log file = /var/log/samba/log.%m	Enregistrer les événements dans un fichier %m représente le nom NetBIOS du client
security	security = user	Modèle de sécurité du serveur

Directive	Exemple	Explication
passdb backend	passdb backend = tdbsam	Stockage des utilisateurs et des mots de passe. Le format tdbsam (Trivial Database) est le format par défaut, limité en performance à 250 utilisateurs. Au delà, il faudra passer au format ldapsam et stocker les utilisateurs et les groupes dans une base LDAP.

La section [homes]

La section [homes] contient la configuration des partages utilisateurs.

C'est une section réservée par Samba, qui lui applique un fonctionnement très particulier. Ce nom de partage ne doit pas être utilisé pour un autre partage ni modifié.

```
[homes]
    comment = Home Directories
    browseable = no
    writable = yes
```

Tous les utilisateurs verront le même partage "homes" mais le contenu sera personnalisé pour chacun.

Attention à bien configurer les booléens SELinux pour autoriser le partage des dossiers personnels.

La section [printers]

La section [printers] contient la configuration du serveur d'impression.

```
[printers]
    comment = All Printers
    browseable = no
    writable = yes
    guest ok = no
    printable = yes
```

Samba peut ainsi faire office de serveur d'impressions, ce qui est une fonctionnalité intéressante (ne nécessite pas l'acquisition de licences clientes).

Partages personnalisés

Avant de paramétrer une nouvelle section du fichier smb.conf qui correspondra à un nouveau

partage, il convient de se poser quelques questions :

- Quel est le chemin du partage ?
- Qui peut modifier le contenu ?
- Le partage doit-il être visible sur le réseau ou au contraire sera-t-il masqué ?
- Y aura-t-il un accès anonyme ?

Un nouveau partage est représenté par une section [nomdupartage] dans le fichier smb.conf. En voici un exemple :

```
[partage]
    comment = Partage
    browseable = yes
    writable = yes
    path = /export/data
    valid users = @users
    read list = georges
    write list = bob, alice
    invalid users = maurice
    create mask = 0664
    directory mask = 0775
    force group = users
```

De nombreuses directives sont disponibles pour configurer les partages :

Directive	**Exemple**	**Explication**
comment	comment = Exemple de partage	Affiche un commentaire dans l'explorateur de fichiers.
browseable	browseable = yes	Affiche le partage dans le voisinage réseau.
writeable	writeable = yes	Le partage est en lecture seule ou en écriture.
path	path = /export/data	Le chemin absolu à partager sur le réseau. Attention au contexte SELinux de ce dossier.
valid users	valid users = @users	Liste les utilisateurs ou les groupes autorisés à accéder au partage.
invalid users	invalid users = alice	Liste les utilisateurs ou les groupes qui ne sont pas autorisés à accéder au partage.
read list	read list = bob	Liste les utilisateurs ou les groupes autorisés à accéder au partage en lecture.

Directive	Exemple	Explication
write list	write list = patrick, alain	Liste les utilisateurs ou les groupes autorisés à accéder au partage en écriture.
create mask	create mask = 0664	Les fichiers créés prendront les droits spécifiés.
directory mask	directory mask = 0775	Les dossiers créés prendront les droits spécifiés.
force group	force group = users	Les nouveaux fichiers et dossiers appartiendront au groupe spécifié. Dans ce cas, il n'y a pas d'@ devant le groupe !

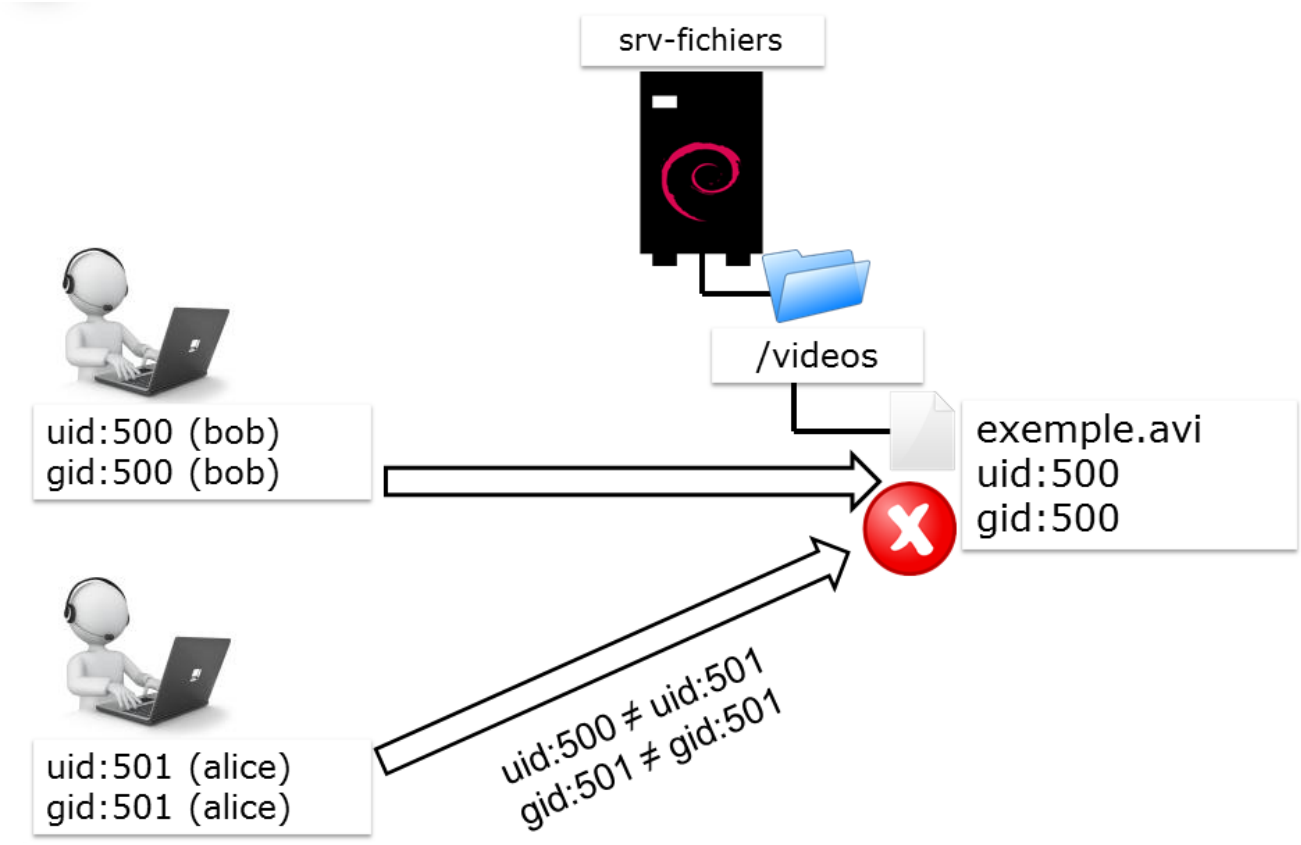
La directive force group

La directive **force group** permet de forcer l'appartenance d'un fichier créé à un groupe spécifique.

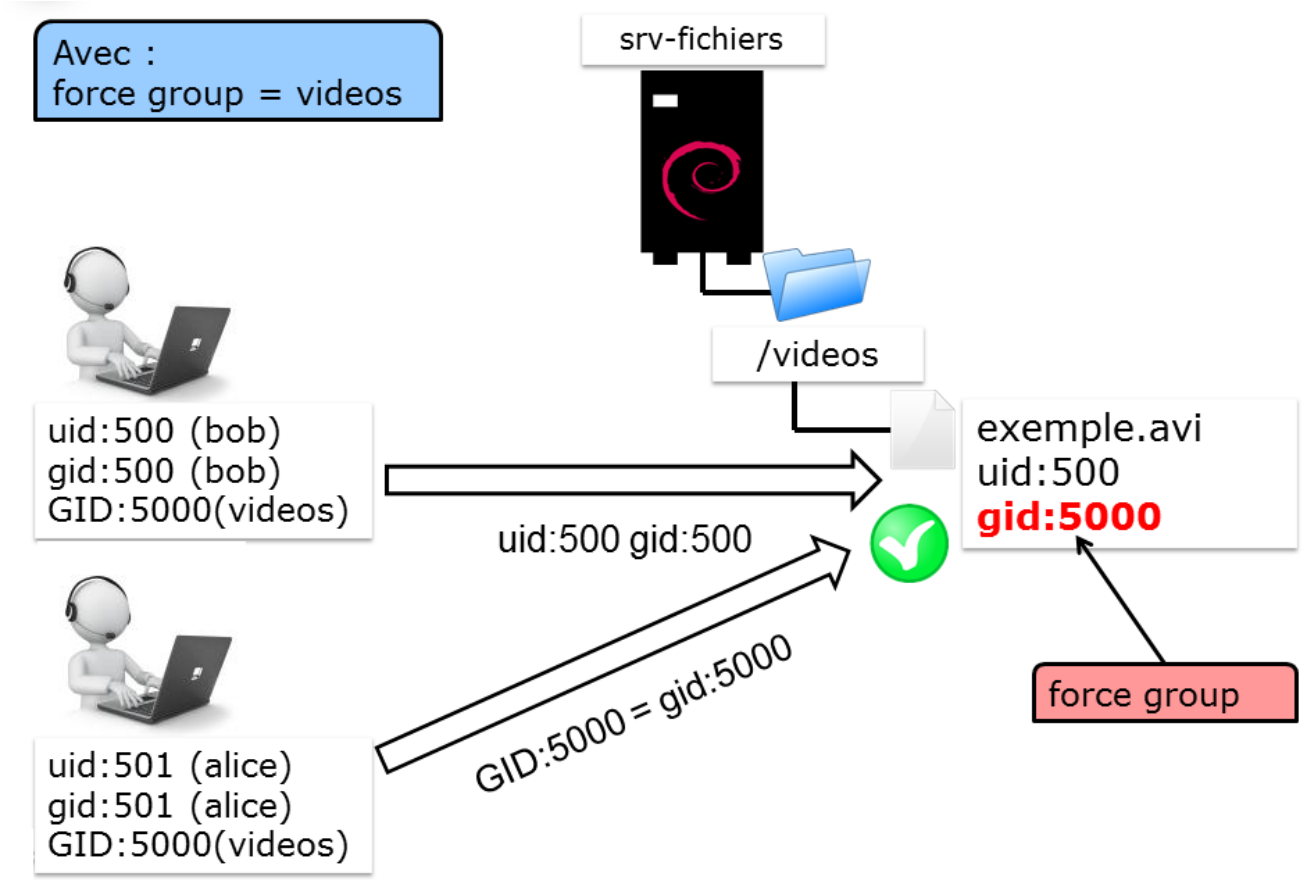
Cette directive est essentielle dans le fonctionnement de Samba, puisqu'elle assure que, quelque soit le groupe principal d'un utilisateur, celui-ci sera autorisé à accéder à un fichier sur le partage s'il fait parti, en tant qu'invité, du groupe spécifié dans la directive.

Les deux exemples ci-dessous mettent en avant ce mécanisme :

Utilisation sans le mécanisme **force group** :



Utilisation avec le mécanisme **force group** :



3.6. Commandes d'administration

La commande pdbedit

La commande pdbedit permet de gérer la base SAM des utilisateurs Samba, que le backend soit au format tdbsam ou ldapsam, contrairement à la commande smbpasswd qui est limitée au format tdbsam.

Syntaxe de la commande pdbedit

```
pdbedit [-a|-r|-x|-L] [-u username] ...
```

Exemple :

```
[root]# pdbedit -L
stagiaire:1000:Stagiaire SYS
```

Table 7. Options principales de la commande pdbedit

Option	Observation
-a	Ajouter un utilisateur

Option	Observation
-r	Modifier un utilisateur
-x	Supprimer un utilisateur
-L	Lister les utilisateurs
-u	Spécifier le nom de l'utilisateur pour les options -a, -r, et -x

Ajouter un utilisateur

Le format de cryptage du mot de passe entre le monde Microsoft et le monde Linux étant différent, Samba doit soit tenir à jour une base de données contenant les mots de passe au bon format ou déléguer cette gestion au serveur LDAP, ce qui explique l'usage de la commande smbpasswd.

```
pdbedit -a -u username [-f description]
```

Exemple :

```
[root]# pdbedit -a -u bob -f "Bob Leponge"
Unix username:     bob
User SID:          S-1-5-21-3024208064-2128810558-4043545969-1000
Full Name:         Bob Leponge
Home Directory:    \\srvfichiers\bob
Domain:            SRVFICHIERS
...
```

Option	Observation
-a	Ajouter un utilisateur. L'utilisateur doit exister dans le fichier /etc/passwd.
-u	Spécifier le nom de l'utilisateur à ajouter.

La commande smbpasswd

La commande smbpasswd permet de gérer les mots de passe des utilisateurs Samba.

Syntaxe de la commande smbpasswd

```
smbpasswd [-d|-e] username
```

Exemple :

```
[root]# smbpasswd bob
New SMB password:
Retype new SMB password:
```

Option	Observation
-e	Réactive un compte.
-d	Désactive un compte.

Il est possible de synchroniser les mots de passe Unix et Samba :

```
[global]
    unix password sync = yes
    obey pam restrictions = yes
```

Directive	Exemple	Explication
unix password sync	unix password sync = yes	Synchronise le mot de passe entre le compte unix et le compte samba. Fonctionne uniquement avec la commande smbpasswd. La directive n'est pas prise en compte par la commande tdbedit.
obey pam restrictions	obey pam restrictions = yes	Applique les restrictions PAM.

La commande smbclient

La commande **smbclient** permet d'accéder à des ressources Windows (ou Samba) depuis le monde Unix.

Syntaxe de la commande smbclient

```
smbclient '//serveur/partage' -U utilisateur
```

Exemple :

```
[root]# smbclient \\\\stat-wind\\partage-wind -U alain
smb: |> help
```

ou :

```
[root]# smbclient '//stat-wind/partage-wind' -U alain
smb: |> help
```

Le programme smbclient est couramment utilisé pour créer un interpréteur de type 'ftp' permettant ainsi d'accéder à des ressources SMB réseau.

Lister les partages :

```
[root]# smbclient -L inf1-formatux
```

Se connecter à un partage data du serveur inf1-formatux avec l'utilisateur bob :

```
[root]#  smbclient -L //inf1-formatux/data -U bob
Enter bob's password:
```

Chapitre 4. Serveur web Apache

🎓 Objectifs

Apprendre aux futurs administrateurs à :

☑ **installer**, **démarrer** et **configurer** le serveur web Apache ;
☑ configurer un ou plusieurs **sites web** ;
☑ utiliser les **principales directives** d'Apache ;
☑ **configurer** et **exploiter** les **journaux d'évènements** ;
☑ **optimiser** le serveur.

Le serveur **HTTP Apache** est le fruit du travail d'un groupe de volontaires : The Apache Group. Ce groupe a voulu réaliser un serveur Web du même niveau que les produits commerciaux mais sous forme de **logiciel libre** (son code source est disponible).

L'équipe d'origine a été rejointe par des centaines d'utilisateurs qui, par leurs idées, leurs tests et leurs lignes de code, ont contribué à faire d'Apache le plus utilisé des serveurs Web du monde.

L'ancêtre d'Apache est le serveur libre développé par le National Center for Supercomputing Applications de l'université de l'Illinois. L'évolution de ce serveur s'est arrêtée lorsque le responsable a quitté le NCSA en 1994. Les utilisateurs ont continué à corriger les bugs et à créer des extensions qu'ils distribuaient sous forme de "patches" d'ou le nom "a patchee server".

La version 1.0 de Apache a été disponible le 1 décembre 1995 (il y a plus de 20 ans !).

L'équipe de développement se coordonne par l'intermédiaire d'une liste de diffusion dans laquelle sont proposées les modifications et discutées les évolutions à apporter au logiciel. Les changements sont soumis à un vote avant d'être intégrés au projet. Tout le monde peut rejoindre l'équipe de développement, il suffit de contribuer activement au projet pour pouvoir être nommé membre de The Apache Group.

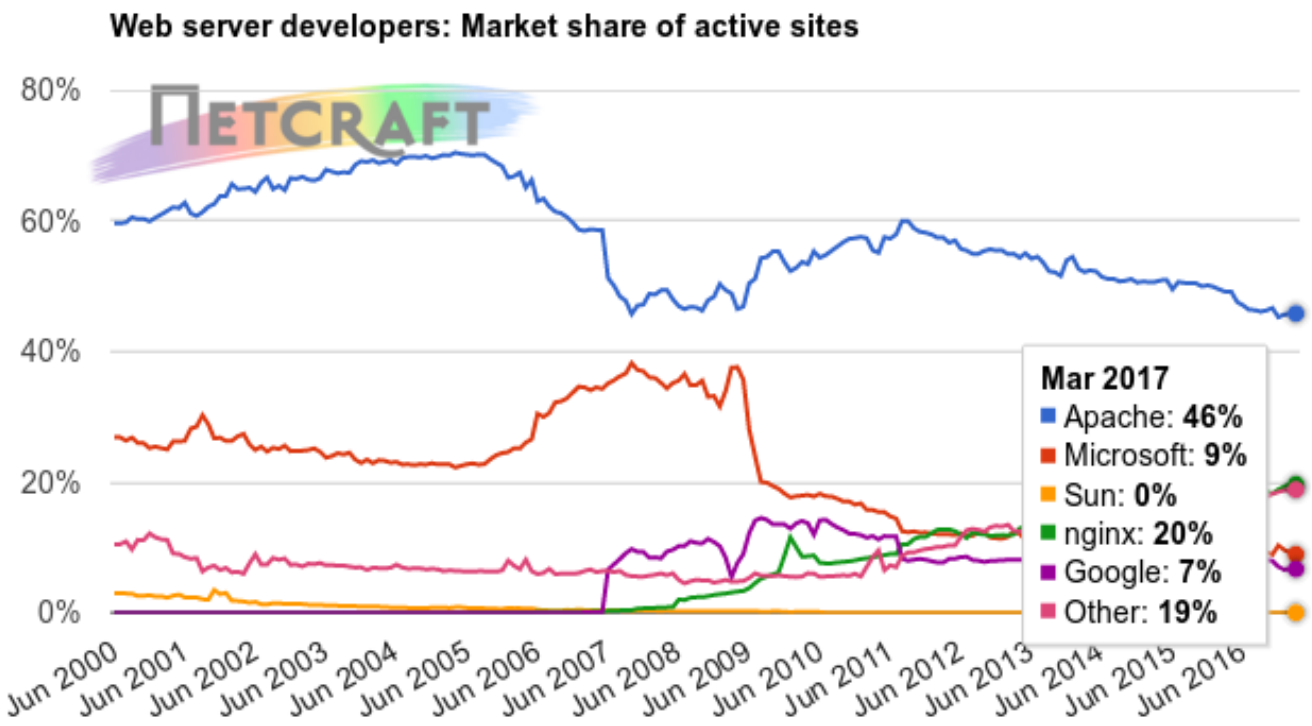


Figure 5. Statistiques NetCraft : Market Share of Active Sites

Le serveur Apache est très présent sur l'Internet, puisqu'il représente encore environ 50% des parts de marché pour l'ensemble des sites actifs.

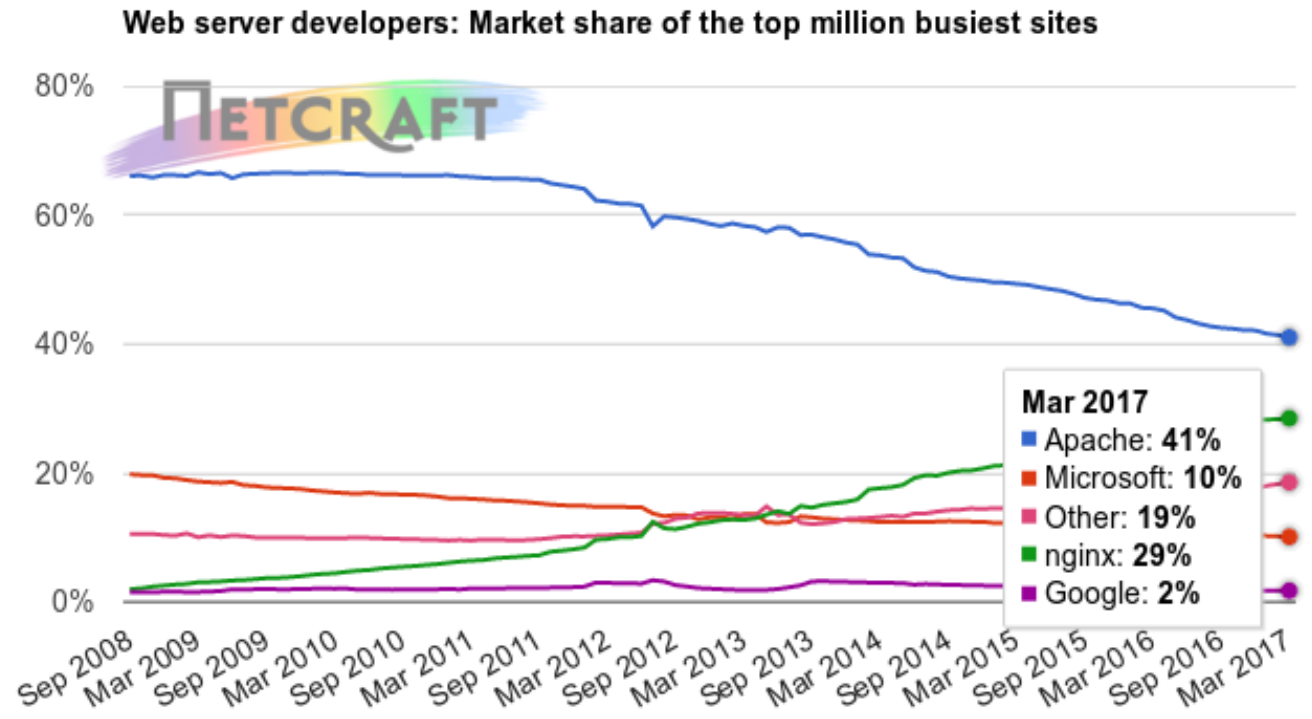


Figure 6. Statistiques NetCraft : Top million busiest sites

Les parts de marché perdues par Apache sont prises par son plus grand challenger : le serveur nginx. Ce dernier, plus rapide pour délivrer les pages web est moins complets fonctionnellement parlant que le géant Apache.

4.1. Le protocole HTTP

Le protocole **HTTP** (HyperText Transfer Protocol) est le protocole le plus utilisé sur Internet depuis 1990.

Ce protocole permet un transfert de fichiers (essentiellement au format HTML, mais aussi au format CSS, JS, AVI…) localisés grâce à une chaîne de caractères appelée URL entre un navigateur (le client) et un serveur Web (appelé d'ailleurs `httpd` sur les machines UNIX).

HTTP est un protocole "requête - réponse" opérant au dessus de **TCP** (Transmission Control Protocol).

1. Le client ouvre une connexion TCP vers le serveur et envoie une requête.
2. Le serveur analyse la requête et répond en fonction de sa configuration.

Le protocole HTTP est en lui même dit "**STATELESS**" : il ne conserve pas d'information sur l'état du client d'une requête à l'autre. Ce sont les langages dynamiques comme le php, le python ou le java qui vont permettre la conservation en mémoire des informations de session d'un client (comme dans le cadre d'un site de e-commerce par exemple).

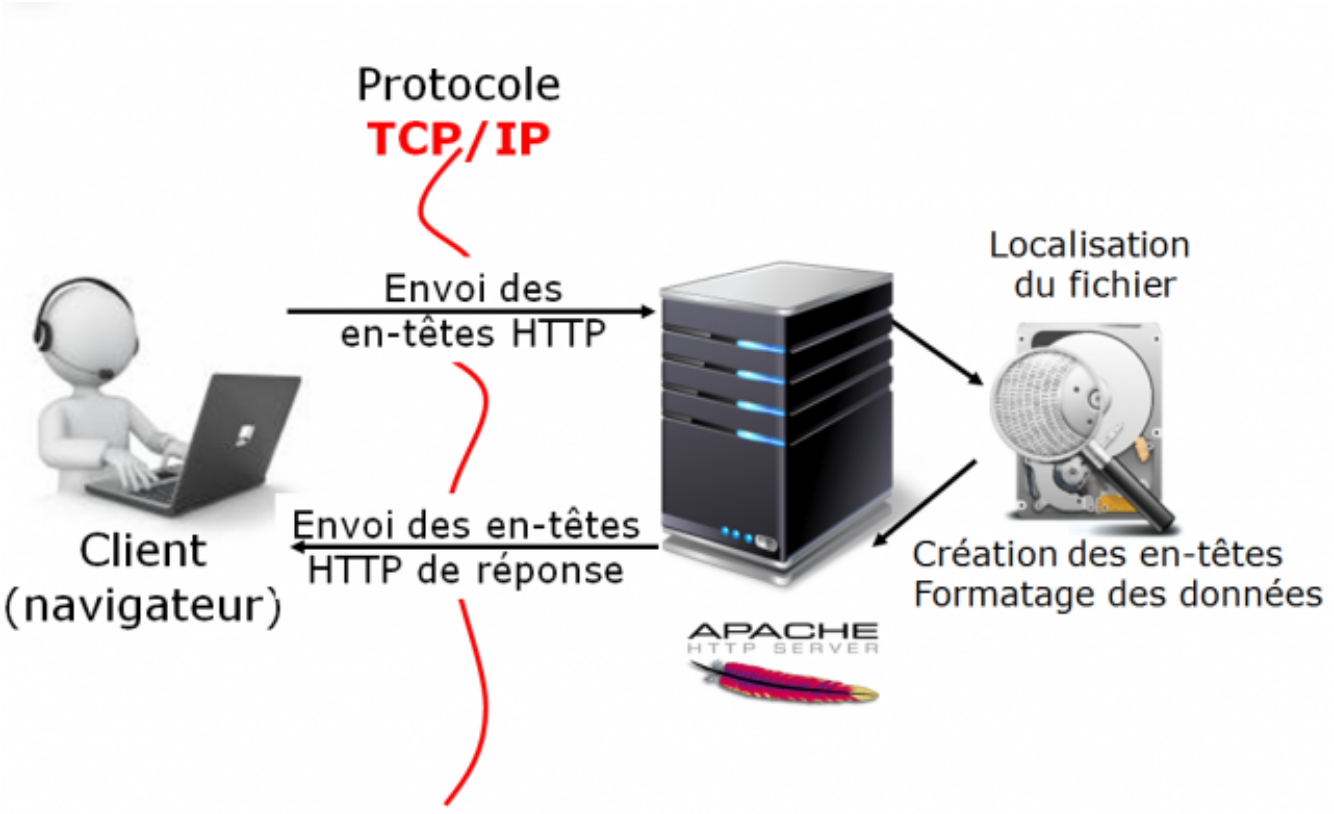


Figure 7. Le protocole HTTP

Le protocole HTTP est en version 1.1. La version 2 est en cours de déploiement.

Une réponse HTTP est un ensemble de lignes envoyées au navigateur par le serveur. Elle comprend :

- **Une ligne de statut** : c'est une ligne précisant la version du protocole utilisé et l'état du traitement de la requête à l'aide d'un code et d'un texte explicatif. La ligne comprend trois éléments devant être séparés par un espace :
  - La version du protocole utilisé ;
  - Le code de statut ;
  - La signification du code .
- **Les champs d'en-tête de la réponse** : il s'agit d'un ensemble de lignes facultatives permettant de donner des informations supplémentaires sur la réponse et/ou le serveur. Chacune de ces lignes est composée d'un nom qualifiant le type d'en-tête, suivi de deux points (`:`) et de la valeur de l'en-tête.
- **Le corps de la réponse** : il contient le document demandé.

Voici un exemple de réponse HTTP :

Exemple de réponse HTTP

```
HTTP/1.1 200 OK
Date : Sat, 15 Jan 2016 14:37:12 GMT Server : Apache/2.17
Content-Type : text/HTML
Content-Length : 1245
Last-Modified : Fri, 14 Jan 2016 08:25:13 GMT
```

Le rôle du serveur web consiste à traduire une URL en ressource locale. Consulter la page http://www.free.fr/, revient à envoyez une requête HTTP à cette machine. Le service DNS joue donc un rôle essentiel.

Les URL

Une URL (**Uniform Ressource Locator** - littéralement "identifiant uniforme de ressources") est une chaîne de caractères ASCII utilisée pour désigner les ressources sur Internet. Elle est informellement appelée adresse web.

Une URL est divisée en trois parties :

Composition d'une URL

```
<protocole>://<hôte>:<port>/<chemin>
```

- **Le nom du protocole** : il s'agit du langage utilisé pour communiquer sur le réseau. Le protocole le plus utilisé est le protocole HTTP (HyperText Transfer Protocol), le protocole permettant

d'échanger des pages Web au format HTML. De nombreux autres protocoles sont toutefois utilisables.

- **Identifiant et mot de passe** : permet de spécifier les paramètres d'accès à un serveur sécurisé. Cette option est déconseillée car le mot de passe est visible dans l'URL (dans le cadre de la sécurité).
- **L'hôte** : Il s'agit du nom de l'ordinateur hébergeant la ressource demandée. Notez qu'il est possible d'utiliser l'adresse IP du serveur, ce qui rend par contre l'URL moins lisible.
- **Le numéro de port** : il s'agit d'un numéro associé à un service permettant au serveur de savoir quel type de ressource est demandé. Le port associé par défaut au protocole est le port numéro 80. Ainsi, lorsque le service Web du serveur est associé au numéro de port 80, le numéro de port est facultatif.
- **Le chemin d'accès à la ressource** : Cette dernière partie permet au serveur de connaître l'emplacement auquel la ressource est située, c'est-à-dire de manière générale l'emplacement (répertoire) et le nom du fichier demandé. Si non renseignée, indique la première page de l'hôte. Sinon indique le chemin de la page à afficher.

Les ports

Une requête HTTP arrivera sur le port 80 (port par défaut pour http) du serveur fonctionnant sur l'hôte. L'administrateur peut toutefois choisir librement le port d'écoute du serveur.

Le protocole HTTP se décline en une version sécurisée : le protocole https (port 443). Ce protocole chiffré s'implémente à partir du module `mod_ssl`.

D'autres ports peuvent être utilisés, comme le port 8080 (serveurs d'applications Java EE) ou le port 10 000 (Serveur webmin).

4.2. Installation du serveur

Apache est **multiplateforme**. Il peut être utilisé sur Linux, Windows, Mac…

L'administrateur devra choisir entre deux méthodes d'installation :

- **Installation par paquets** : l'éditeur de la distribution fourni des versions **stables et soutenues** (mais parfois anciennes) ;
- **Installation depuis les sources** : le logiciel Apache est compilé, l'administrateur peut spécifier les options qui l'intéressent ce qui permet l'optimisation du service. Apache fournissant une architecture modulaire, la re-compilation du logiciel Apache n'est généralement pas nécessaire pour ajouter ou supprimer des fonctionnalités complémentaires (ajout/suppression de modules).

Le choix de la méthode d'installation par paquets est fortement **conseillé**. Des dépôts complémentaires permettent d'installer des versions plus récentes d'Apache sur des versions de distributions anciennes (dépôt `REMI` par exemple) mais, en cas de problème, RedHat n'apportera pas son soutien.

Exemples de modules et de leurs rôles respectifs :

Table 8. Modules et rôles respectifs

Module	Rôle
mod_access	Filtre l'accès des clients par leur nom d'hôte, adresse IP ou autre caractéristique
mod_alias	Permet la création d'alias ou répertoires virtuels
mod_auth	Authentifie les clients
mod_cgi	Exécute les scripts CGI
mod_info	Fournit des informations sur l'état du serveur
mod_mime	Associe les types de fichiers avec l'action correspondante
mod_proxy	Propose un serveur proxy (serveur mandataire)
mod_rewrite	Réécrit les URL
...	

Installation par rpm

Interroger la base de données des rpm :

```
[root]# rpm -qa "http*"
```

Installez à partir de paquets liés à la distribution :

```
[root]# rpm -ivh httpd-xxx.rpm
```

Installer si nécessaire les dépendances demandées.

Installation par yum

Si vous avez à disposition un dépôt yum :

```
[root]# yum install httpd
```

Lancer Apache au démarrage du serveur :

```
[root]# chkconfig httpd on
```

Avant de se lancer dans une installation, il est important de savoir si une version du serveur Apache est installée :

```
rpm -qa "http*"
```

Lors de l'installation, un groupe `apache` et un utilisateur `apache` sont créés.

```
[root]# grep apache /etc/group
apache:x:48

[root]# grep apache /etc/passwd
apache:x:48:48:Apache:/var/www:/sbin/nologin

[root]# grep apache /etc/shadow
apache:!!:14411::::::
```

Le serveur Apache travaille sous l'identité d'un utilisateur `apache` appartenant à un groupe `apache`. Lors de l'installation, ce groupe et cet utilisateur sont créés. L'utilisateur `apache` étant un utilisateur système, aucun mot de passe ne lui est attribué, ce qui rend impossible la connexion au système avec cet utilisateur.

Les fichiers de configuration

Le répertoire `/etc/httpd/conf/` contient le fichier de configuration principal d'Apache `httpd.conf`. Ce fichier est très bien commenté. En général, ces commentaires suffisent pour éclairer l'administrateur sur les options dont il dispose.

Il peut être judicieux de créer un fichier de configuration sans commentaires :

```
[root]# grep -Ev '^#|^$' /etc/httpd/conf/httpd.conf > httpd.conf.lite
```

Le répertoire `/etc/httpd/conf.d/` contient les fichiers de configuration des sites virtuels ou des modules installés (`ssl`, `php`, `welcome.conf`, `phpmyadmin`, etc.)

Manuel d'utilisation

Il existe un package contenant un site faisant office de manuel d'utilisation d'Apache. Il s'agit du package `httpd-manual-xxx.noarch.rpm`. Une fois ce package installé vous pourrez accéder au manuel simplement avec un navigateur web à cette adresse http://127.0.0.1/manual.

Lancement du serveur

- Par le script `init` :

```
[root]# /etc/rc.d/init.d/httpd {start|restart|status
```

- Avec la commande `service` :

```
[root]# service httpd {start|restart|status}
```

- Avec la commande `apachectl` fourni par Apache :

```
[root]# apachectl {start|restart|stop}
```

Il est indispensable de lancer/relancer le serveur :

- après l'installation ;
- après toute modification de la configuration.

Parefeu

Le pare-feu `iptables` interdit l'accès aux ports 80 et 443. Pensez à le configurer (ou à désactiver ce service sur plate-forme de test) !

```
[root]# system-config-firewall-tui
```

Ou :

```
[root]# service iptables stop
[root]# service ip6tables stop
```

La configuration d'un pare-feu fait l'objet d'un autre cours.

4.3. Arborescence

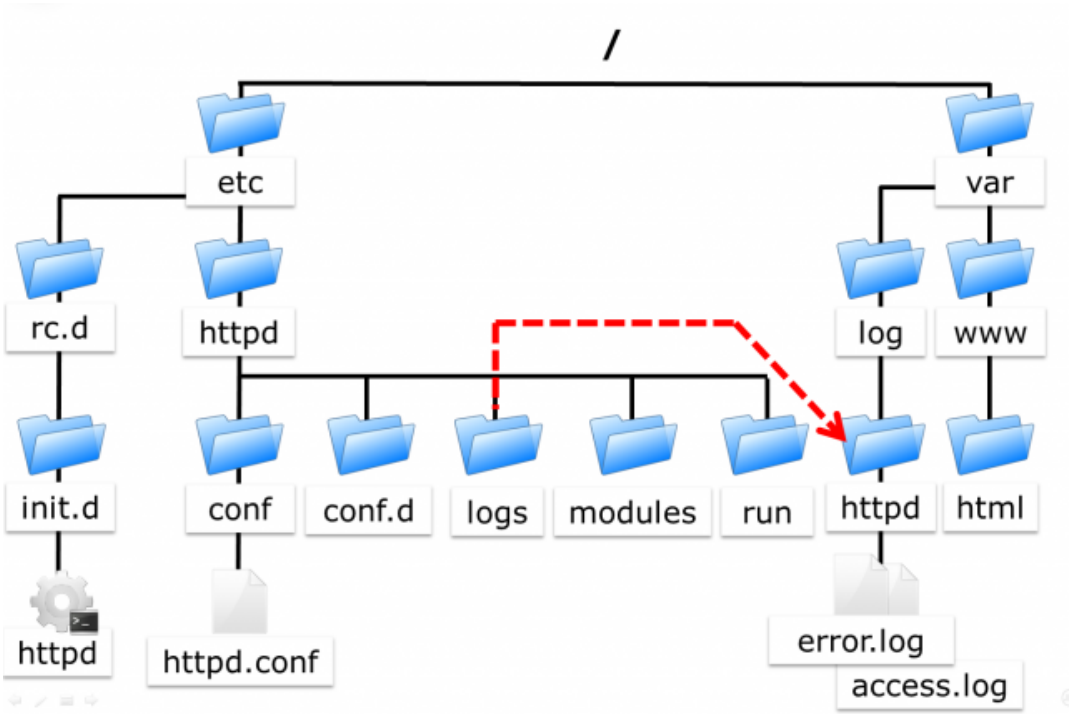


Figure 8. Arborescence d'Apache

L'arborescence peut varier en fonction des distributions.

- `/etc/httpd/` : Ce répertoire est la racine du serveur. Il contient l'ensemble des fichiers du serveur Apache.
- `/etc/httpd/conf/` : Ce répertoire contient l'ensemble des fichiers de configuration du serveur. Il possède des sous-dossiers pour des éléments de configuration précis.
- `/var/www/html/` : Ce répertoire est le répertoire de publication par défaut. Il contient les fichiers nécessaires à l'affichage de la page web par défaut du serveur Apache. Quand l'administrateur veut publier un site, il peut déposer ses fichiers dans ce répertoire.

D'autres répertoires existent sous `/var/www` :

- `cgi-bin` : contient les scripts `CGI` ;
- `icons` : contient des icônes, notamment celles pour identifier le type de fichier ;
- `error` : contient les messages d'erreur d'Apache, ce sont ces fichiers qu'il faudra modifier pour personnaliser les mesages d'erreur.
- `/var/log/httpd/` : Ce répertoire contient les fichiers de logs du serveur Apache.
  - Le fichier `access-log` garde une trace des différents accès au serveur ;
  - Le fichier `error-log` contient la liste des erreurs rencontrées pendant l'exécution du service ;
  - Les fichiers logs sont personnalisables, l'administrateur peut aussi en créer de nouveaux.
- `/etc/httpd/modules` : Répertoire contenant les liens vers le répertoire `/usr/lib/httpd/modules` contenant les modules utilisables par Apache. Un module est une extension logicielle d'Apache, lui permettant par exemple d'interpréter le PHP (ex: `mod-php5.so`).
- `/etc/rc.d/init.d/httpd` : Script de démarrage du serveur `httpd`.

4.4. Configuration du serveur

La configuration globale du serveur se fait dans `/etc/httpd/conf/httpd.conf`.

Ce fichier est découpé en 3 sections qui permettent de configurer :

- en **section 1** l'environnement global ;
- en **section 2** le site par défaut et les paramètres par défaut des sites virtuels ;
- en **section 3** les hôtes virtuels.

L'**hébergement virtuel** permet de mettre en ligne **plusieurs sites virtuels** sur le même serveur. Les sites sont alors différenciés en fonction de leurs noms de domaines, de leurs adresses IP, etc.

La modification d'une valeur en section 1 ou 2 impacte l'ensemble des sites hébergés.

En environnement mutualisé, les modifications seront donc effectuées en section 3.

Pour faciliter les mises à jour futures, il est vivement recommandé de créer un fichier de configuration section 3 pour chaque site virtuel.

Section 1

Les différentes directives rencontrées en section 1 sont :

Table 9. Directives principales de la section 1

Directive	Fonction
`ServerTokens`	Cette directive sera vue dans le cours Apache – sécurité.
`ServerRoot`	Indique le chemin du répertoire contenant l'ensemble des fichiers constituant le serveur Apache.
`PidFile`	Le fichier cible de la directive contient le numéro de `PID` du serveur à son démarrage.
`Timeout`	Le nombre de secondes avant le délai d'expiration d'une requête trop longue (entrante ou sortante).
`KeepAlive`	Connexion persistante (plusieurs requêtes par connexion TCP).
`MaxKeepAliveRequests`	Nombre maximum de connexions persistantes.
`KeepAliveTimeout`	Nombre de secondes à attendre la requête suivante du client avant fermeture de la connexion TCP.
`Listen`	Permettre à Apache d'écouter sur des adresses ou des ports spécifiques.
`LoadModule`	Charger des modules complémentaires (moins de modules = plus de sécurité).
`Include`	Inclure d'autres fichiers de configuration au serveur.

Directive	Fonction
`ExtendedStatus`	Afficher plus d'information sur le serveur dans le module server-status.
`User` et `Group`	Permet de lancer les processus `apache` avec différents utilisateurs. Apache se lance toujours en tant que `root` puis change son propriétaire et son groupe.

Le serveur Apache a été conçu comme un serveur puissant et flexible, pouvant fonctionner sur une grande variété de plate-formes.

Plate-formes différentes et environnements différents signifient souvent fonctionnalités différentes, ou utilisation des méthodes méthodes pour implémenter la même fonctionnalité le plus efficacement possible.

La conception modulaire d'Apache autorise l'administrateur à choisir quelles fonctionnalités seront incluses dans le serveur en choisissant les modules à charger soit à la compilation, soit à l'exécution.

Cette modularité comprend également les fonctions les plus élémentaires du serveur web.

Certains modules, les Modules Multi-Processus (`MPM`) sont responsables de l'association aux ports réseau de la machine, acceptent les requêtes, et se chargent de les répartir entre les différents processus enfants.

Pour la version d'Apache de Windows, le MPM utilisé sera `mpm-winnt`.

Sous Linux, les sites très sollicités utiliseront un MPM threadé comme `worker` ou `event`, tandis que les sites privilégiant la stabilité utiliseront `prefork`.

Voir la page http://httpd.apache.org/docs/2.2/fr/mpm.html

Configuration par défaut des modules `prefork` et `worker` :

Configuration des modules MPM dans /etc/http/conf/httpd.conf

```
<IfModule prefork.c>
StartServers        8
MinSpareServers     5
MaxSpareServers     20
ServerLimit     256
MaxClients      256
MaxRequestPerChild  4000
</IfModule>

<IfModule worker.c>
StartServers        4
MaxClients      300
MinSpareThreads     25
MaxSpareThreads     75
ThreadsPerChild     25
MaxRequestsPerChild 0
</IfModule>
```

Le module `prefork`, activé par défaut, s'appuie sur des processus. Il est donc plus stable mais nécessite plus de mémoire pour fonctionner. Le module `worker`, quand à lui, s'appuie sur des threads, ce qui le rend plus performant, mais des modules comme `php` ne sont pas compatibles.

Choisir un module plutôt qu'un autre est donc une tâche complexe, tout autant que l'optimisation du module MPM retenu (nombre de client, de requêtes, etc.).

Par défaut Apache est configuré pour un service moyennement sollicité (256 clients max).

La configuration minimal d'un serveur Apache ressemble à ceci :

Configuration d'Apache minimal dans /etc/httpd/conf/httpd.conf

```
ServerRoot /etc/httpd
KeepAlive On
Listen 80
Listen 160.210.150.50:1981
LoadModule ...
Include conf.d/*.conf
User apache
Group apache
```

SELinux

Attention par défaut la sécurité via SELinux est active. Elle empêche la lecture d'un site sur un autre répertoire que `/var/www/`.

Le répertoire contenant le site doit posséder le contexte de sécurité `httpd_sys_content_t`.

Le contexte actuel se vérifie par la commande :

```
[root]# ls –Z /rep
```

Rajouter le contexte via la commande :

```
[root]# chcon -vR --type=httpd_sys_content_t /rep
```

Elle empêche également l'ouverture d'un port non standard. Il faut ouvrir manuellement le port désiré à l'aide de la commande `semanage` (non installée par défaut).

```
[root]# semanage port -a -t http_port_t -p tcp 1664
```

Directives User et Group

Définir un compte et un groupe de gestion d'Apache

Historiquement, Apache était lancé par `root`, ce qui posait des problèmes de sécurité. Apache est toujours lancé par `root` mais change ensuite son identité. Généralement `User apache` et `Group apache`.

Attention jamais ROOT !!!

Le serveur Apache (processus `httpd`) est lancé par le compte de super-utilisateur `root`. Chaque requête d'un client déclenche la création d'un processus "fils". Pour limiter les risques, il faut lancer ces processus enfants avec un compte moins privilégié.

Les directives `User` et `Group` servent à déclarer le compte et le groupe utilisés pour la création des processus enfants.

```
User apache
Group apache
```

Ce compte et ce groupe doivent avoir été créés dans le système (par défaut cela est fait à l'installation). Par mesure de précaution supplémentaire, s'assurer que le compte n'est pas interactif (ne peut pas ouvrir de session).

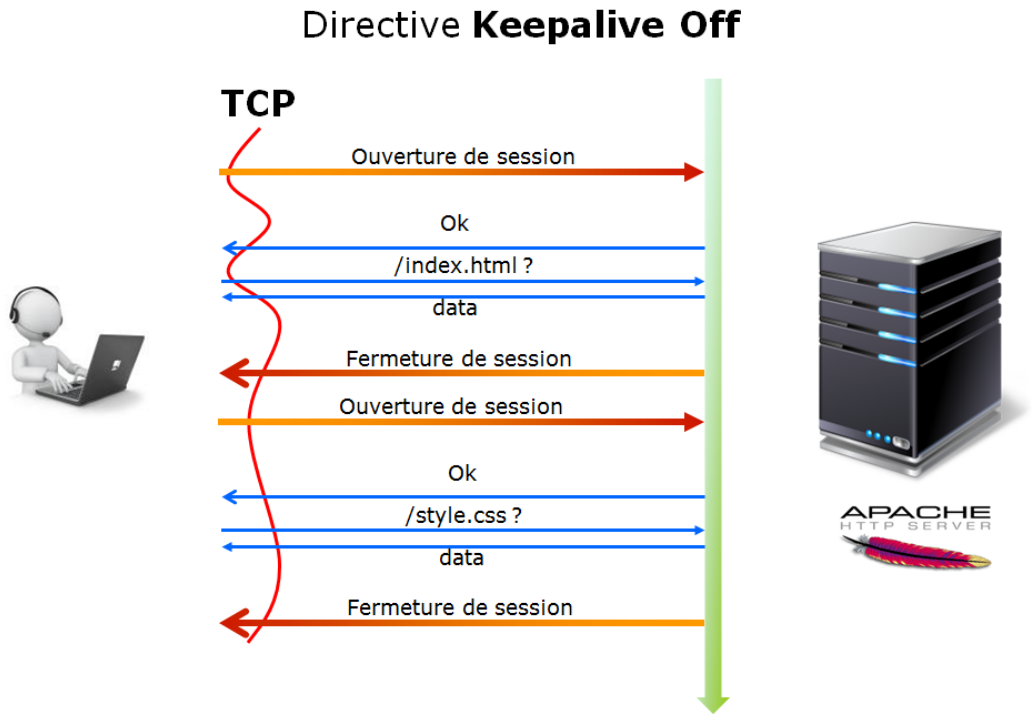


Figure 9. Directive KeepAlive off

Avec la directive `KeepAlive` désactivée, chaque demande de ressource sur le serveur nécessite une ouverture de connexion TCP, ce qui est long à effectuer d'un point de vue réseau et gourmand en ressource système.

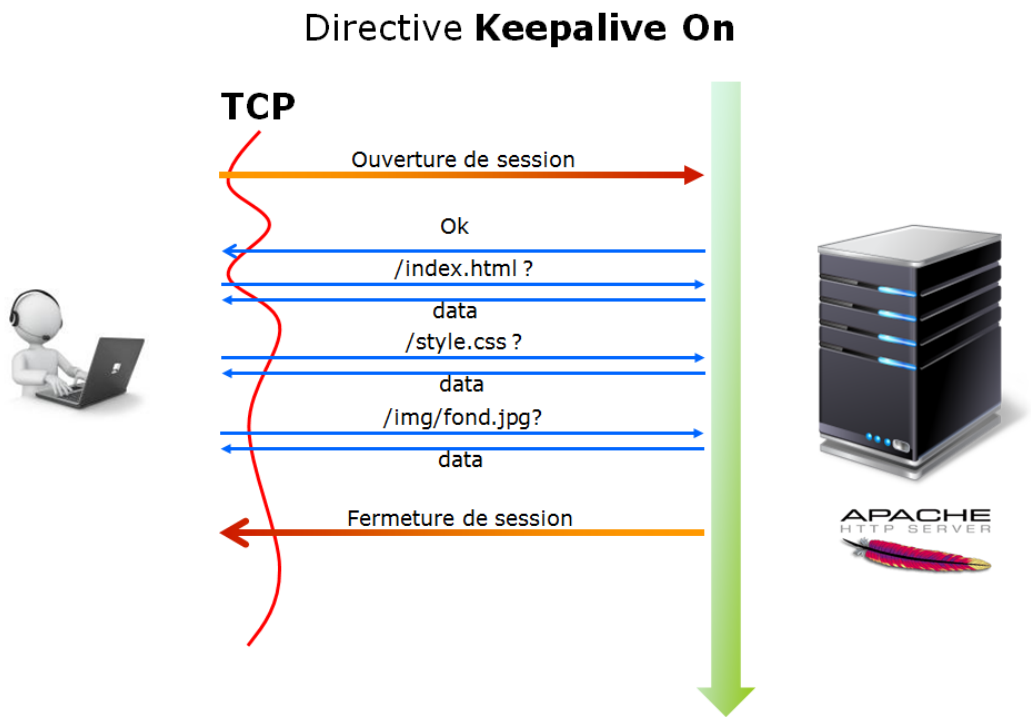


Figure 10. Directive KeepAlive on

Avec la directive `KeepAlive` à `On`, le serveur conserve la connexion ouverte avec le client le temps du `KeepAlive`.

Sachant qu'une page web est constituée de plusieurs fichiers (images, feuilles de styles, javascripts,

etc.), cette stratégie est rapidement gagnante.

Il est toutefois nécessaire de bien paramétrer cette valeur au plus juste :

- Une valeur trop courte pénalise le client,
- Une valeur trop longue pénalise les ressources du serveur.

Des demandes de configuration spécifiques peuvent être faîtes par le client en hébergement mutualisé. Auquel cas, les valeurs de `KeepAlive` seront paramétrées directement dans le `VirtualHost` du client ou au niveau du mandataire (`ProxyKeepAlive` et `ProxyKeepAliveTimeout`).

Figure 11. La page de bon fonctionnement d'Apache

L'affichage de cette page prouve que le serveur est fonctionnel. Mais le serveur ne dispose pas encore de site à publier.

Paramétrons la section 2.

Section 2

La section 2 paramètre les valeurs utilisées par le serveur principal. Le serveur principal répond à toutes les requêtes qui ne sont pas prises en charge par un des VirtualHosts de la sections 3.

Les valeurs sont également utilisées comme valeur par défaut pour les sites virtuels.

Table 10. Directives de la section 2

Directive	Fonction
`ServerAdmin`	Spécifie une adresse de messagerie qui apparaîtra dans certaines pages auto-générées, comme dans les pages d'erreurs.
`ServerName`	Spécifie le nom qui servira d'identification pour le serveur. Peut être déterminé automatiquement, mais il est recommandé de le spécifier explicitement (adresse IP ou nom DNS).
`DocumentRoot`	Spécifie le répertoire contenant les fichiers à servir aux clients. Par défaut `/var/www/html/`.
`ErrorLog`	Spécifie le chemin vers le fichier d'erreurs.
`LogLevel`	`debug`, `info`, `notice`, `warn`, `error`, `crit`, `alert`, `emerg`.
`LogFormat`	Définir un format spécifique de log à utiliser avec la directive `CustomLog`.
`CustomLog`	Spécifie le chemin vers le fichier d'accès.
`ServerSignature`	Vue dans le cours sécurité.
`Alias`	Spécifie un répertoire extérieur à l'arborescence et le rend accessible par un contexte. La présence ou l'absence du dernier slash dans le contexte à son importance.
`ScriptAlias`	Spécifie le dossier contenant les scripts serveurs (idem `alias`) et les rend exécutables.
`Directory`	Spécifie des comportements et des droits d'accès par répertoire.
`AddDefaultCharset`	Spécifie le format d'encodage des pages envoyés (les caractères accentués peuvent être remplacés par des ?...).
`ErrorDocument`	personnaliser les pages d'erreurs.
`server-status`	Rapport sur l'état du serveur.
`server-info`	Rapport sur la configuration du serveur.

La directive ErrorLog

La directive `ErrorLog` permet de définir le journal des erreurs.

Cette directive défini le nom du fichier dans lequel le serveur enregistre toutes les erreurs qu'il rencontre. Si le `file-path` n'est pas absolu, il est supposé être relatif à `ServerRoot`.

Syntaxe :

```
ErrorLog file-path
```

Exemple :

```
ErrorLog logs/error-log
```

La directive DirectoryIndex

La directive `DirectoryIndex` permet de définir la page d'accueil du site.

Cette directive indique le nom du fichier qui sera chargé en premier, qui fera office d'index du site ou de page d'accueil.

Syntaxe :

```
DirectoryIndex page-à-afficher
```

Le chemin complet n'est pas précisé car le fichier est recherché dans le répertoire spécifié par `DocumentRoot`.

Exemple :

```
DocumentRoot    /var/www/html
DirectoryIndex  index.php, index.htm
```

Cette directive indique le nom du fichier index du site web. L'index est la page par défaut qui s'ouvre quand le client tape l'URL du site (sans avoir à taper le nom de cet index). Ce fichier doit se trouver dans le répertoire indiqué par la directive DocumentRoot.

La directive `DirectoryIndex` peut spécifier plusieurs noms de fichiers index séparés par des espaces. Par exemple, une page d'index par défaut au contenu dynamique et en deuxième choix une page statique.

La directive ServerAdmin

La directive `ServerAdmin` permet d'indiquer le mail de l'administrateur.

Syntaxe :

```
ServerAdmin email-adresse
```

Exemple :

```
ServerAdmin webmaster@formatux.fr
```

La balise Directory

La balise `Directory` permet de définir des directives propre à un répertoire.

Cette balise permet d'appliquer des droits à un ou plusieurs répertoires. Le chemin du répertoire

sera saisi en absolu.

Syntaxe :

```
<Directory directory-path>
Définition des droits des utilisateurs
</Directory>
```

Exemple :

```
<Directory /home/SitesWeb/SiteTest>
Allow from all   # nous autorisons tout le monde
</Directory>
```

La section `Directory` sert à définir un bloc de consignes s'appliquant à une partie du système de fichiers du serveur. Les directives contenues dans la section ne s'appliqueront qu'au répertoire spécifié (et ses sous-répertoires).

La syntaxe de ce bloc accepte les caractères génériques mais il faudra préférer alors utiliser le bloc `DirectoryMatch`.

Dans l'exemple suivant, nous allons refuser l'accès au disque dur local du serveur quelque soit le client. Le répertoire `/` représente la racine du disque dur.

```
<Directory />
    Order deny, allow
    Deny from all
</Directory>
```

Dans l'exemple suivant, nous allons autoriser l'accès au répertoire de publication `/var/www/html` pour tous les clients.

```
<Directory /var/www/html>
    Order allow, deny
    Allow from all
</Directory>
```

Prise en compte des modifications

La commande `apachectl` permet de tester la syntaxe du fichier de conf :

```
[root]# service httpd configtest
```

ou :

```
[root]# apachectl -t
```

puis :

```
[root]# /etc/rc.d/init.d/httpd {start|restart|status}
```

ou :

```
[root]# service httpd {start|restart|status}
```

ou :

```
[root]# apachectl {start|restart|stop}
```

Les commandes précédentes ont pour effet de couper les connexions en cours. Apache propose une solution plus élégante, qui lance de nouveaux serveurs et attends la fin du timeout pour détruire les anciens processus :

Ne pas couper les connexions TCP actives :

```
[root]# service httpd graceful
```

4.5. Configuration avancée du serveur

Le mod_status

Le `mod_status` permet d'afficher une page `/server-status` ou `/server-info` récapitulant l'état du serveur :

Configuration des directives server-status et server-info

```
<Location /server-status>
    SetHandler server-status
    Order allow,deny
    Allow from 127.0.0.1
</Location>

<Location /server-info>
    SetHandler server-info
    Order allow,deny
    Allow from 127.0.0.1
    Allow from 172.16.96.105
</Location>
```

La page `/server-status` :

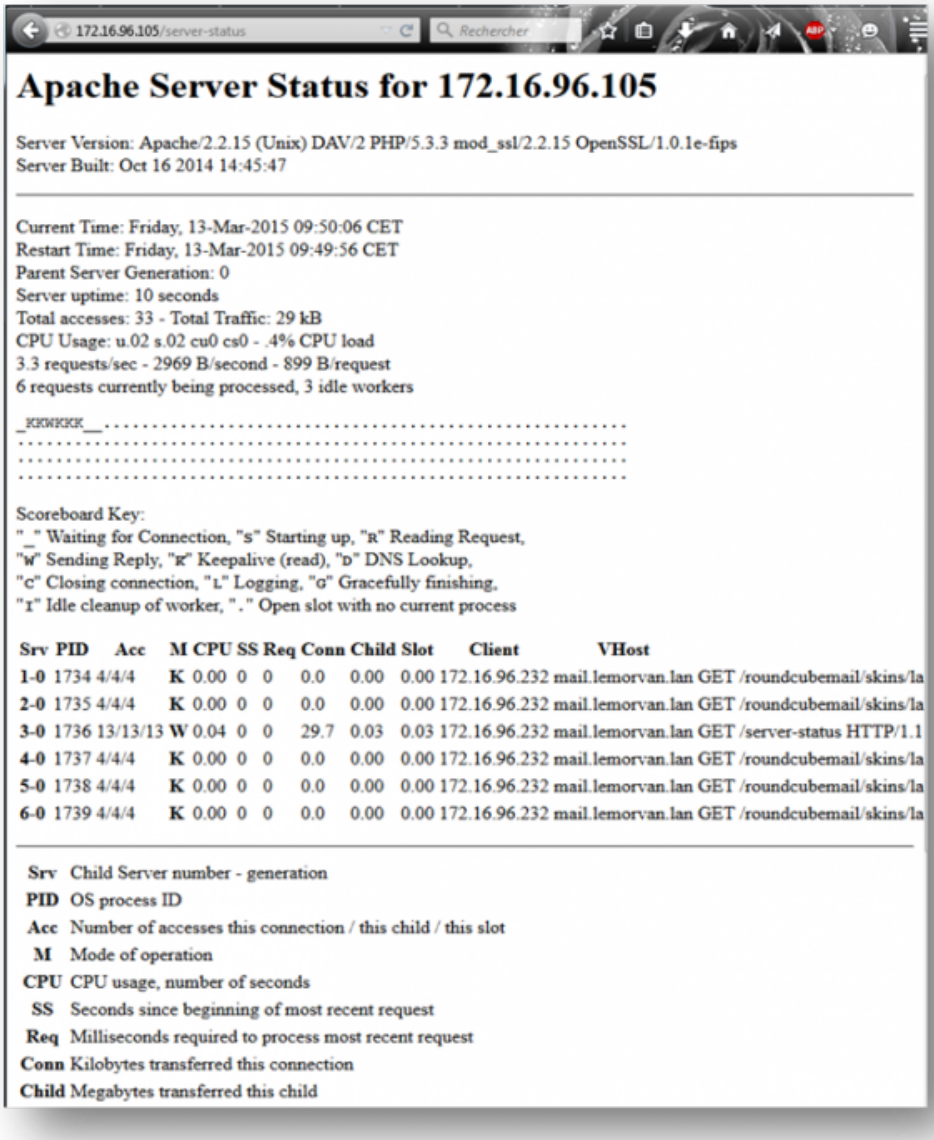


Figure 12. La page server-status

La page `/server-info` :

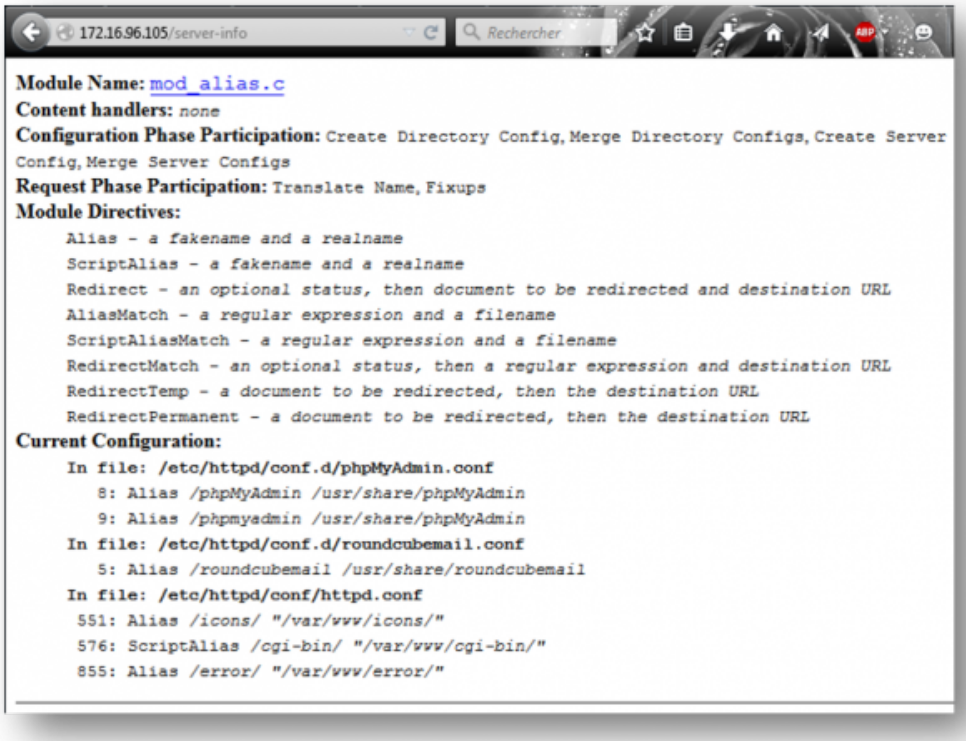


Figure 13. La page server-info

Hébergement mutualisé (section 3)

Dans le cas d'un hébergement mutualisé, le client pense visiter plusieurs serveurs. En réalité, il n'existe qu'un seul serveur et plusieurs sites virtuels.

Pour mettre en place un hébergement mutualisé, il faut mettre en place des hôtes virtuels :

- en déclarant plusieurs ports d'écoute ;
- en déclarant plusieurs adresses IP d'écoute (hébergement virtuel par **IP**) ;
- en déclarant plusieurs noms de serveur (hébergement virtuel par **nom**);

Chaque site virtuel correspond à une arborescence différente.

La section 3 du fichier `httpd.conf` permet de déclarer ces hôtes virtuels.

Pour faciliter les mises à jour futures, il est vivement recommandé de créer un fichier de configuration section 3 pour chaque site virtuel.

Choisissez un hébergement virtuel "par IP" ou "par nom". En production, il est déconseillé de mixer les deux solutions.

- Chaque site virtuel peut être configuré dans un fichier indépendant ;
- Les VirtualHosts sont stockés dans `/etc/httpd/conf.d/` ;
- L'extension du fichier est `.conf`.

La balise VirtualHost

La balise `VirtualHost` permet de définir des hôtes virtuels.

Syntaxe :

Syntaxe d'un fichier VirtualHostXXX.conf

```
<VirtualHost adresse-IP[:port]>
    # si la directive "NameVirtualHost" est présente
    # alors "adresse-IP" doit correspondre à celle saisie
    # sous "NameVirtualHost" ainsi que pour le "port".
 ...
 </VirtualHost>
```

Si nous configurons le serveur Apache avec les directives de base vues précédemment, nous ne pourrons publier qu'un seul site. En effet, nous ne pouvons pas publier plusieurs sites avec les paramètres par défaut : même adresse IP, même port TCP et absence de nom d'hôte ou nom d'hôte unique.

L'usage des sites virtuels va nous permettre de publier plusieurs sites web sur un même serveur Apache. Nous allons définir des blocs qui décriront chacun un site web. Ainsi chaque site aura sa propre configuration.

Pour des facilités de compréhension, nous associons souvent un site web à une machine unique. Les sites virtuels ou hôtes virtuels (`VirtualHost`) sont appelés ainsi parce qu'ils dématérialisent le lien entre machine et site web.

Exemple 1 :

```
Listen 192.168.0.10:8080
<VirtualHost 192.168.0.10:8080>
  DocumentRoot /var/www/site1/
  ErrorLog /var/log/httpd/site1-error.log
</VirtualHost>
```

```
Listen 192.168.0.11:9090
<VirtualHost 192.168.0.11:9090>
  DocumentRoot /var/www/site2/
  ErrorLog /var/log/httpd/site2-error.log
</VirtualHost>
```

L'hébergement virtuel basé sur IP est une méthode permettant d'appliquer certaines directives en fonction de l'adresse IP et du port sur lesquels la requête est reçue. En général, il s'agit de servir différents sites web sur des ports ou des interfaces différents.

La directive NameVirtualHost

La directive `NameVirtualHost` permet de définir des hôtes virtuels à base de nom.

Cette directive est obligatoire pour configurer des hôtes virtuels à base de nom. Nous spécifions avec cette directive l'adresse IP sur laquelle le serveur recevra des demandes des hôtes virtuels à base de nom.

Syntaxe :

```
NameVirtualHost adresse-IP[:port]
```

Exemple :

```
NameVirtualHost 160.210.169.6:80
```

Il faut placer la directive avant les blocs descriptifs de sites virtuels. Elle désigne les adresses IP utilisées pour écouter les requêtes des clients vers les sites virtuels. La syntaxe est la suivante :

Pour écouter les requêtes sur toutes les adresses IP du serveur il faut utiliser le caractère *****.

4.6. Exemple de publication de sites

Fichier /etc/httpd/conf.d/80-site1.conf :

```
<VirtualHost 160.210.69.6:80>
  # déclaration de l'arborescence du site
  DocumentRoot "/var/sitesweb/site1"
  # déclaration des index du site
  DirectoryIndex "Index1.htm"
  # déclaration des droits sur le site
  <Directory "/var/sitesweb/site1">
    Allow from all
  </Directory>
</VirtualHost>
```

Fichier /etc/httpd/conf.d/1664-site2.conf :

```
Listen 1664
<VirtualHost 160.210.69.6:1664>
  # déclaration de l'arborescence du site
  DocumentRoot "/var/sitesweb/site2"
  # déclaration des index du site
  DirectoryIndex "Index2.htm"
  # déclaration des droits sur le site
  <Directory "/var/sitesweb/site2">
    Allow from all
  </Directory>
</VirtualHost>
```

4.7. Redirection des logs vers syslog

Il est possible, en passant par la commande `logger` de rediriger les logs d'Apache vers le `syslog` local ou vers un serveur `syslog` distant avec le mot clé `apache`.

Modification des lignes `CustomLog` et `ErrorLogs` du fichier `.conf` correspondant au VirtualHost désiré :

fichier .conf du vhost

```
    ErrorLog "|usr/bin/logger -t apache -p local6.info"
    CustomLog "|usr/bin/logger -t apache -p local6.info" combined
```

Chapitre 5. Serveur web Apache - LAMP - sous CentOS 7

Apache est le principal serveur web du monde de l'Open Source. À l'origine, c'était la continuation du serveur libre développé par le NCSA (National Center for Supercomputing Applications) à l'Université de l'Illinois. Lorsque le projet officiel a été abandonné en 1994, une équipe de développeurs volontaires a continué à fournir du code sous forme de nombreux correctifs, ce qui explique la genèse du nom a patchy server, c'est-à-dire "serveur rafistolé".

D'après les statistiques de Netcraft, un peu moins de la moitié des sites Web du monde tournent sur un serveur Apache. Ces dernières années, les parts de marché perdues par Apache sont reprises par Nginx, son principal concurrent, qui est orienté vers la performance tout en offrant moins de fonctionnalités.

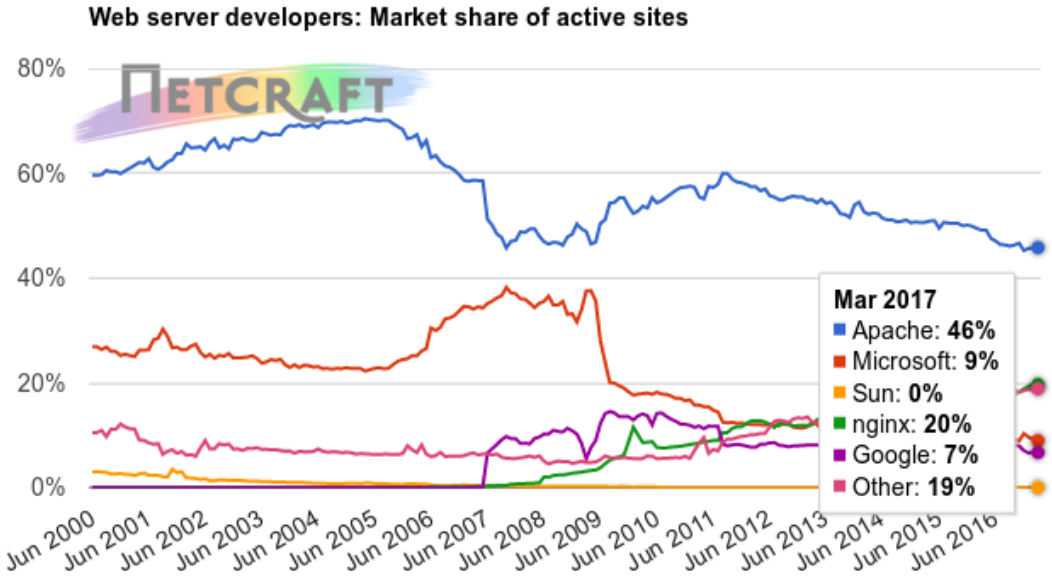


Figure 14. Statistiques NetCraft : sites actifs

Une installation typique d'Apache est généralement constituée d'un assemblage cohérent de paquets.

- le serveur Apache à proprement parler ;
- des bibliothèques diverses et variées ;
- des plug-ins ;
- des langages de programmation ;
- etc.

Ce cours décrit la configuration d'un serveur Web de type LAMP (***L**inux* + ***A**pache* + ***M**ySQL/**M**ariaDB* + ***P**HP*) sur CentOS 7.

5.1. Le protocole HTTP et les URL

Le protocole HTTP (***Hyper**text **T**ransfer **P**rotocol*, c'est-à-dire "protocole de transfert hypertexte") est un protocole de communication client-serveur développé pour le World Wide Web. Il permet un transfert de fichiers (html, css, js, mp4) localisés grâce à une chaîne de caractères appelée URL entre un navigateur (le client) et un serveur web.

HTTP est un protocole "requête-réponse" de la couche application qui utilise le protocole TCP comme couche de transport.

1. Le client ouvre une connexion TCP vers le serveur et envoie une requête.
2. Le serveur analyse la requête et répond en fonction de sa configuration.

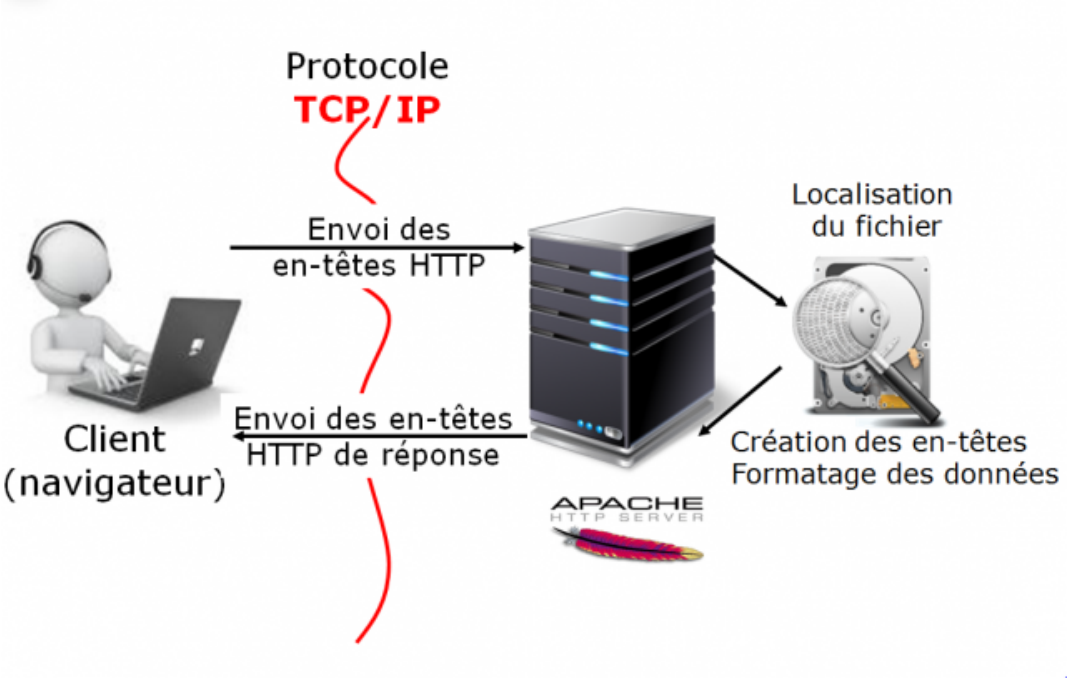


Figure 15. HTTP est un protocole client/serveur

Une réponse HTTP est un ensemble de lignes envoyées au client par le serveur. Elle comprend une ligne de statut, les champs d'en-tête et le corps de la réponse.

Voici un exemple de réponse HTTP :

```
$ curl --head --location https://www.formatux.fr
HTTP/1.1 200 OK
Date: Fri, 18 Aug 2017 18:44:18 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.1e-fips mod_fcgid/2.3.9 PHP/5.4.16
mod_python/3.5.0- Python/2.7.5
X-Powered-By: PHP/7.0.21
Cache-Control: max-age=604800
Expires: Thu, 19 Nov 1981 08:52:00 GMT
domain=www.formatux.fr
Last-Modified: Sun, 23 Jul 2017 18:11:50 GMT
ETag: "faba58243b57482f98e0fa49387257e9"
Content-Type: text/html; charset=UTF-8
```

Le rôle du serveur web consiste à traduire une URL (comme par exemple `http://www.formatux.fr`) en ressource locale. Consulter la page `http://www.formatux.fr` revient à envoyer une requête HTTP à cette machine.

Une URL (***U**niform **R**esource **L**ocator*, autrement dit “identifiant uniforme de ressources”) est une chaîne de caractères ASCII utilisée pour désigner les ressources sur Internet. Elle est informellement appelée “adresse web” et elle est divisée en plusieurs parties, comme ceci.

```
<protocole>://<hôte>:<port>/<chemin>
```

Le protocole

Langage utilisé pour communiquer sur le réseau, comme par exemple `http`, `https`, `ftp`.

L’hôte

Ordinateur qui héberge la ressource demandée. Il est possible d’utiliser l’adresse IP (mais cela rend l’URL moins lisible).

Le numéro de port

Numéro associé à un service permettant de savoir quel type de ressource est demandé. Le port 80 est associé par défaut au protocole HTTP. Si l’on utilise ce port, ce n’est pas la peine de le spécifier explicitement.

Le chemin d’accès

Le chemin d’accès à la ressource permet au serveur de connaître l’emplacement du fichier demandé.

5.2. Ports et pare-feu

Apache utilise le port 80 en TCP pour le protocole HTTP. Il faudra donc songer à ouvrir ce port dans le pare-feu.

Notons que théoriquement, l’administrateur peut choisir librement le port d’écoute du serveur.

5.3. Installation

Le serveur Apache est fourni par le paquet httpd.

```
# yum install httpd
```

L’installation du paquet crée un utilisateur système apache et un groupe système apache correspondant.

```
# grep apache /etc/passwd
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
# grep apache /etc/group
apache:x:48:
# grep apache /etc/shadow
apache:!!:17352::::::
```

5.4. Premier lancement du serveur

Sous Red Hat et CentOS, Apache est préconfiguré pour afficher une page statique par défaut. Il suffit d’activer et de lancer le service.

```
# systemctl enable httpd
# systemctl start httpd
```

Tester le bon fonctionnement du serveur.

```
$ links http://localhost
```

On doit voir quelque chose de ce genre.

```
=================================================================
                           Testing 123..
This page is used to test the proper operation of the Apache HTTP
server after it has been installed. If you can read this page it
means that this site is working properly. This server is powered
by CentOS.
=================================================================
```

Dans le réseau local, ouvrir l’adresse IP du serveur avec un navigateur.

`http://192.168.1.10`

On peut également invoquer le nom d'hôte.

`http://stagiaire.formatux.fr`

Sur un serveur dédié, on essaiera successivement l'adresse IP, le nom de domaine et l'alias associé.

`http://172.16.100.1` `http://stagiaire1.formatux.fr` `http://www.stagiaire1.lan`

Voici à quoi ressemble la page par défaut dans un navigateur graphique.

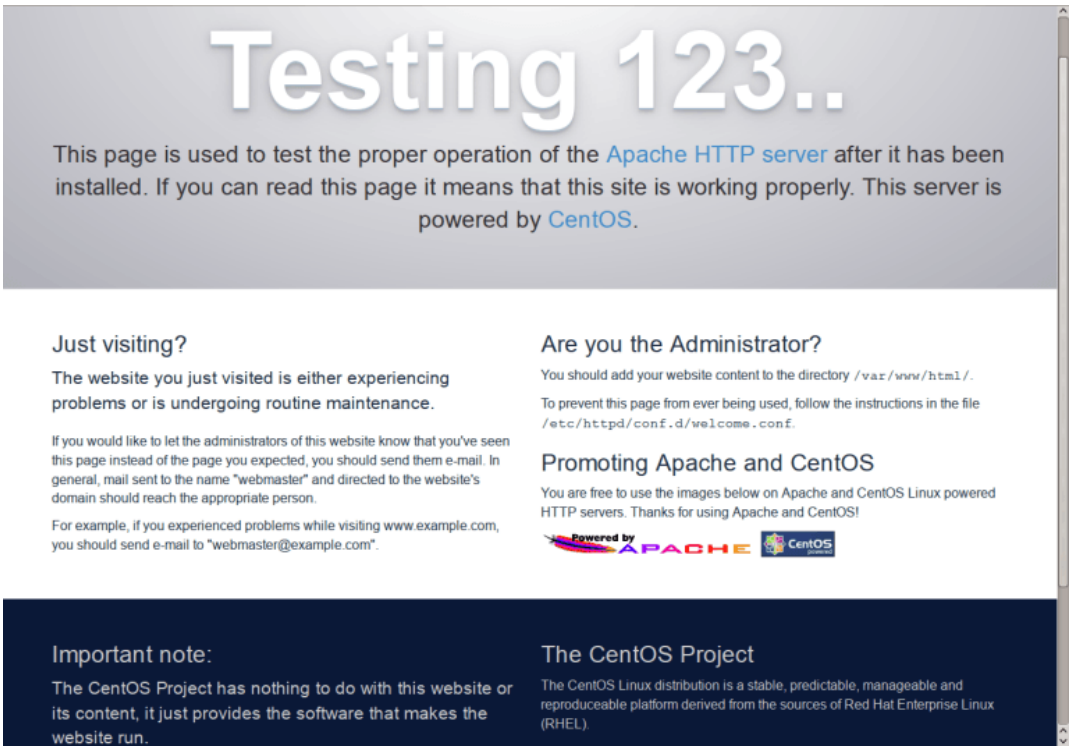


Figure 16. La page de test Apache

5.5. Les fichiers de configuration

Initialement, la configuration du serveur Apache s'effectuait dans un seul fichier `/etc/httpd/conf/httpd.conf`. Avec le temps, ce fichier est devenu de plus en plus volumineux et de moins en moins lisible.

Les distributions modernes ont donc tendance à répartir la configuration d'Apache sur une série de fichiers `*.conf` répartis dans les répertoires `/etc/httpd/conf.d` et `/etc/httpd/conf.modules.d`, rattachés au fichier principal `/etc/httpd/conf/httpd.conf` par la directive `Include`.

Le fichier `/etc/httpd/conf/httpd.conf` est amplement documenté. Pour commencer, nous allons sauvegarder ce fichier par défaut et créer une version dépourvue de commentaires et plus lisible. Ici, la commande `egrep` filtre les lignes qui commencent soit par un commentaire (`^#`), soit par un espace (`^$`), soit par quatre espaces suivis d'un commentaire.

```
# cd /etc/httpd/conf
# cp httpd.conf httpd.conf.orig
# grep -Ev '^#|^$|^    #' httpd.conf.orig > httpd.conf
```

5.6. La configuration par défaut

Jetons un oeil sur le fichier `httpd.conf` par défaut et regardons de plus près les principales directives qui le constituent.

La directive `ServerRoot` permet de définir le répertoire dans lequel le serveur est installé.

```
ServerRoot "/etc/httpd"
```

La directive `Listen` permet à Apache d'écouter sur des adresses ou des ports spécifiques. Notons que cette directive est requise. Si elle est absente du fichier de configuration, Apache refuse de démarrer.

```
Listen 80
```

Comme nous l'avons vu un peu plus haut, la directive `Include` permet l'inclusion d'autres fichiers de configuration dans le fichier de configuration principal du serveur. `IncludeOptional` fonctionne comme `Include`, au détail près que la directive ne produira pas une erreur au cas où le métacaractère `*` ne correspond à aucun fichier. Le chemin est relatif par rapport à l'emplacement spécifié dans la directive `ServerRoot`.

```
Include conf.modules.d/*.conf
...
IncludeOptional conf.d/*.conf
```

Apache ne doit pas être lancé en tant que `root`, mais en tant qu'utilisateur spécial défini par les directives `User` et `Group` dans `/etc/httpd/conf/httpd.conf`. Plus précisément, le processus principal est lancé par `root` qui lance ensuite des processus enfant avec l'utilisateur et le groupe configurés.

```
User apache
Group apache
```

L'adresse mail de l'administrateur, définie par la directive `ServerAdmin`, apparaîtra sur certaines pages générées par le serveur, notamment les pages d'erreur.

```
ServerAdmin root@localhost
```

Les balises `<Directory>` et `</Directory>` permettent de regrouper un ensemble de directives qui ne s'appliquent qu'au répertoire précisé, à ses sous-répertoires et aux fichiers situés dans ces sous-répertoires.

```
<Directory />
  AllowOverride none
  Require all denied
</Directory>
...
<Directory "/var/www">
  AllowOverride None
  Require all granted
</Directory>
<Directory "/var/www/html">
  Options Indexes FollowSymLinks
  AllowOverride None
  Require all granted
</Directory>
```

La directive `DocumentRoot` permet de définir le répertoire à partir duquel Apache va servir des fichiers.

```
DocumentRoot "/var/www/html"
```

Lorsque le serveur trouve un fichier `.htaccess`, il doit savoir quelles directives placées dans ce fichier sont autorisées à modifier la configuration préexistante. Le traitement des fichiers `.htaccess` est contrôlé par la directive `AllowOverride`, qui ne peut être utilisée que dans les sections `<Directory>`. À partir du moment où elle est définie à `none`, les fichiers `.htaccess` sont totalement ignorés.

```
<Directory />
  AllowOverride none
  Require all denied
</Directory>
```

La directive `Require` permet de contrôler l'accès au système de fichiers du serveur. `Require all denied` bloque l'accès pour tous les utilisateurs, `Require all granted` autorise tout le monde.

```
<Directory />
  AllowOverride none
  Require all denied
</Directory>
...
<Directory "/var/www">
  AllowOverride None
  Require all granted
</Directory>
```

Comme son nom l'indique, la directive `Options` permet d'activer ou de désactiver une série d'options (ou de comportements) pour un répertoire donné. Ici par exemple, l'option `Indexes` affiche la liste des fichiers d'un répertoire en cas d'absence de fichier `index.html`. `FollowSymlinks` permet de suivre les liens symboliques.

```
<Directory "/var/www/html">
  Options Indexes FollowSymLinks
  AllowOverride None
  Require all granted
</Directory>
```

Les balises `<IfModule>` et `</IfModule>` contiennent des directives qui ne s'appliquent qu'en fonction de la présence ou de l'absence d'un module spécifique. La directive `DirectoryIndex` spécifie le fichier à envoyer par Apache lorsqu'une URL se termine par `/` et concerne un répertoire entier.

```
<IfModule dir_module>
  DirectoryIndex index.html
</IfModule>
```

Les balises `<Files>` et `</Files>` contiennent des directives qui s'appliquent aux fichiers précisés. Ici par exemple, on interdit l'accès à tous les fichiers dont le nom commence par `.ht`, notamment `.htaccess`.

```
<Files ".ht*">
  Require all denied
</Files>
```

La directive `ErrorLog` définit le chemin vers le journal des erreurs. La verbosité de ce journal est contrôlée par la directive LogLevel.

```
ErrorLog "logs/error_log"
LogLevel warn
```

La directive `CustomLog` permet de contrôler la journalisation des requêtes destinées au serveur. Elle définit le nom et le format du fichier journal.

```
CustomLog "logs/access_log" combined
```

La directive `AddDefaultCharset` paramètre le jeu de caractères par défaut pour les pages de texte. Lorsqu'elle est désactivée (`AddDefaultCharset off`), Apache prend en compte l'encodage spécifié dans la balise `<meta>` des fichiers HTML à envoyer au navigateur.

```
meta http-equiv="Content-Type" content="text/html; charset=utf-8"
```

Ici en revanche, Apache utilise d'emblée le jeu de caractères spécifié en ignorant la balise `<meta>`.

```
AddDefaultCharset UTF-8
```

5.7. Configuration de base

Apache est immédiatement utilisable dans sa configuration par défaut. Avant d'héberger notre premier site, nous allons procéder à quelques ajustements.

Pour commencer, renseigner l'adresse mail de l'administrateur du serveur.

```
ServerAdmin contact@formatux.fr
```

Le nom du serveur peut être déterminé automatiquement, mais il vaut mieux le spécifier explicitement grâce à la directive `ServerName`.

```
ServerName stagiaire.formatux.fr
```

Sur un serveur dédié, on aura ceci.

```
ServerName sd-100246.dedibox.fr
```

Pour la journalisation, on choisira un format un peu moins bavard.

```
CustomLog "logs/access_log" common
```

Enfin, on permettra aux pages hébergées de spécifier leur propre encodage.

```
AddDefaultCharset off
```

Tester la nouvelle configuration :

```
# apachectl configtest
Syntax OK
```

Prendre en compte les modifications :

```
# systemctl reload httpd
```

5.8. Héberger un site statique

Dans la configuration par défaut, Apache est censé servir le contenu de `/var/www/html`. En l'absence de contenu, la page de test s'affiche, en fonction de la configuration prédéfinie dans `/etc/httpd/conf.d/welcome.conf`.

Pour nous épargner la corvée de créer du contenu fictif, nous pouvons récupérer un site web existant. On choisira la documentation de Slackware, qui vient sous forme d'une série de pages HTML statiques.

```
# cd /var/www/html/
# wget -r -np -nH --cut-dirs=1 http://www.slackbook.org/html/
```

Ouvrir le site dans un navigateur (Firefox, Links, Lynx) et apprécier le résultat.

5.9. Apache et les permissions de fichiers

Apache ne doit pas être lancé en tant que root, mais en tant qu'utilisateur spécial défini par les directives `User` et `Group` dans `/etc/httpd/conf/httpd.conf`.

```
User apache
Group apache
```

La règle de sécurité générale veut que les contenus du serveur web ne doivent pas appartenir au processus qui fait tourner le serveur.

Ici, nous attribuons les contenus à l'utilisateur non privilégié `stagiaire1` et au groupe associé `stagiaires`. En passant, nous en profitons pour restreindre les droits d'accès du groupe.

```
# cd /var/www/html
# chown -R stagiaire1:stagiaire ./*
# find ./ -type d -exec chmod 0755 \{} \;
# find ./ -type f -exec chmod 0644 \{} \;
```

5.10. Héberger plusieurs sites sur le même serveur

Le principe des hôtes virtuels (Virtual Hosts) consiste à faire fonctionner un ou plusieurs sites Web sur une même machine. L'utilisateur final ne perçoit pas qu'en fait il s'agit d'un même serveur physique.

Nous allons héberger trois sites :

- http://slackware.formatux.fr hébergera la documentation de Slackware.
- http://freebsd.formatux.fr affichera la documentation de FreeBSD.
- http://formatux.fr pointera vers la page par défaut du serveur.

Pour commencer, on va déplacer le site existant dans un nouveau répertoire slackware/html.

```
# cd /var/www/
# mkdir -pv ./slackware/html
mkdir: création du répertoire « ../slackware »
mkdir: création du répertoire « ../slackware/html »
# mv ./html/* ./slackware/html/
```

Puis, on va créer un autre répertoire freebsd/html, dans lequel on va télécharger un autre site, en l'occurrence la documentation de FreeBSD.

```
# mkdir -pv freebsd/html
mkdir: création du répertoire « freebsd »
mkdir: création du répertoire « freebsd/html »
# cd freebsd/html
# wget -r -p -np -nH --cut-dirs=4 \
  http://www.freebsd.org/doc/fr_FR.ISO8859-1/books/handbook/
```

Enfin, on va mettre en place une page par défaut dans le répertoire default/html. Pour ce faire, on va utiliser la page qui s'affiche lorsqu'il n'y a pas de contenu.

```
# cd /var/www/
# mkdir -pv default/html
mkdir: création du répertoire « default »
mkdir: création du répertoire « default/html »
# cp -R /usr/share/httpd/noindex/* default/html/
```

Au total, nous avons donc :

```
# ls -l
total 12
drwxr-xr-x 3 root root 4096 23 févr. 06:36 default
drwxr-xr-x 3 root root 4096 23 févr. 06:21 freebsd
drwxr-xr-x 3 root root 4096 23 févr. 06:19 slackware
```

Nous allons redéfinir les permissions :

```
# chown -R stagiaire:stagiaire ./*
# find . -type d -exec chmod 0755 \{} \;
# find . -type f -exec chmod 0644 \{} \;
```

Créer un fichier `/etc/httpd/conf.d/00-formatux.fr.conf`. Ce fichier définira le site affiché par défaut, c'est-à-dire lorsqu'on invoque l'adresse IP ou le nom d'hôte de la machine :

```
# Page par défaut
<VirtualHost *:80>
  ServerAdmin info@formatux.fr
  DocumentRoot "/var/www/default/html"
  ServerName www.formatux.fr
  ServerAlias formatux.fr
  ErrorLog logs/formatux.fr-error_log
  CustomLog logs/formatux.fr-access_log common
</VirtualHost>
```

Prendre en compte les modifications :

```
# systemctl reload httpd
```

Vérifier si la page par défaut du serveur s'affiche comme prévu :

```
$ links http://www.formatux.fr
```

À présent, nous pouvons ajouter les deux autres sites. Le site http://slackware.formatux.fr sera configuré comme ceci :

```
# http://slackware.formatux.fr
<VirtualHost *:80>
  ServerAdmin info@formatux.fr
  DocumentRoot "/var/www/slackware/html"
  ServerName slackware.formatux.fr
  ErrorLog logs/slackware.formatux.fr-error_log
  CustomLog logs/slackware.formatux.fr-access_log common
</VirtualHost>
```

La configuration de http://freebsd.formatux.fr suivra la même logique :

```
# http://freebsd.formatux.fr
<VirtualHost *:80>
  ServerAdmin info@formatux.fr
  DocumentRoot "/var/www/freebsd/html"
  ServerName freebsd.formatux.fr
  ErrorLog logs/freebsd.formatux.fr-error_log
  CustomLog logs/freebsd.formatux.fr-access_log common
</VirtualHost>
```

Pour l'instant, les noms d'hôtes `slackware.formatux.fr` et `freebsd.formatux.fr` ne correspondent à rien dans notre réseau local. Nous devons les ajouter à `/etc/hosts` sur le serveur.

```
192.168.2.5 www.formatux.fr formatux.fr  freebsd.formatux.fr slackware.formatux.fr
```

Redémarrer Dnsmasq pour propager l'info DNS.

```
# systemctl restart dnsmasq
```

Prendre en compte la nouvelle configuration d'Apache.

```
# systemctl reload httpd
```

Tester les deux sites en local avec Links ou Firefox sur une machine du réseau local.

`http://slackware.formatux.fr`
`http://freebsd.formatux.fr`

5.11. Héberger des sites dynamiques avec PHP

Installer PHP :

```
# yum install php
```

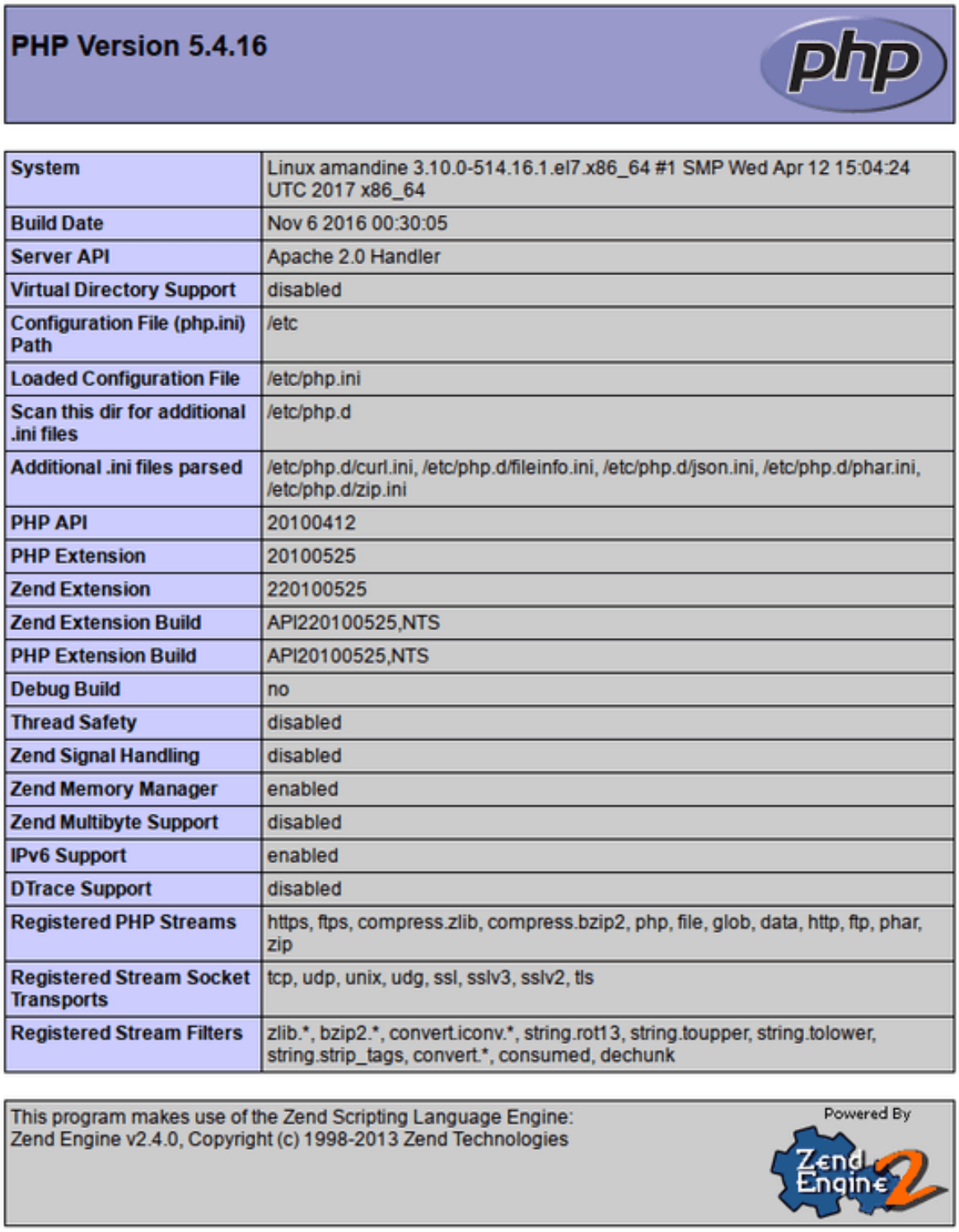
Mettre en place un hôte virtuel http://phpinfo.formatux.fr et éditer la configuration correspondante. L'hôte virtuel contiendra une seule page index.php que l'on éditera comme ceci :

```
<?php
 echo phpinfo();
?>
```

Ajouter une entrée correspondante dans la configuration DNS ou dans le fichier `/etc/hosts` et redémarrer Apache :

```
# systemctl restart httpd
```

Afficher la page `http://phpinfo.formatux.fr` dans un navigateur. On doit obtenir quelque chose qui ressemble grosso modo à ceci :



PHP Version 5.4.16

System	Linux amandine 3.10.0-514.16.1.el7.x86_64 #1 SMP Wed Apr 12 15:04:24 UTC 2017 x86_64
Build Date	Nov 6 2016 00:30:05
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/curl.ini, /etc/php.d/fileinfo.ini, /etc/php.d/json.ini, /etc/php.d/phar.ini, /etc/php.d/zip.ini
PHP API	20100412
PHP Extension	20100525
Zend Extension	220100525
Zend Extension Build	API220100525,NTS
PHP Extension Build	API20100525,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	https, ftps, compress.zlib, compress.bzip2, php, file, glob, data, http, ftp, phar, zip
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, sslv2, tls
Registered Stream Filters	zlib.*, bzip2.*, convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk

This program makes use of the Zend Scripting Language Engine:
Zend Engine v2.4.0, Copyright (c) 1998-2013 Zend Technologies

Powered By Zend Engine 2

Figure 17. La page phpinfo() de PHP

Le fichier `/etc/php.ini` contient la configuration de PHP. On peut commencer par définir le fuseau horaire du serveur, nécessaire pour le bon fonctionnement de certaines applications :

```
[Date]
; Defines the default timezone used by the date functions
; http://php.net/date.timezone
date.timezone = Europe/Paris
```

Redémarrer Apache et vérifier les données correspondantes dans la page qui affiche les infos PHP.

Vous pouvez en apprendre plus sur PHP en suivant notre cours SVR-121-Php-fpm.

5.12. Utiliser MySQL/MariaDB à partir de PHP

Pour utiliser MySQL/MariaDB à partir de PHP, il suffit d'installer le module correspondant et de redémarrer Apache.

```
# yum install php-mysql
# systemctl restart httpd
```

La gestion d'une base de données MariaDB/MySQL est abordée dans le cours SVR-131-MySQL.

5.13. Documentation

- La documentation officielle du projet
- Blog Micro Linux
- Michael Kofler – Apache, Linux 2017

Chapitre 6. Sécurisation du serveur web Apache

6.1. Introduction

La sécurisation d'une site internet nécessite la mise en place de trois mesures :

- La sécurité du service en lui-même.

Par défaut un serveur Apache va envoyer des informations avec ses pages web. Ces informations contiennent des détails sur sa version ou ses modules. Pour un pirate, ces informations vont lui permettre de cibler les attaques en fonction des failles connues. Il est donc impératif de masquer ces informations sur un serveur de production.

- La sécurité des accès aux données.

Pour empêcher l'accès aux données, il est nécessaire de mettre en place une authentification. Cette authentification peut être d'ordre applicative, et s'appuyer par exemple sur une base de données, ou être gérée par le service Apache.

- La sécurité des échanges (protocole https).

La mise en place d'un processus d'authentification sur le serveur pour protéger l'accès aux données n'empêche pas l'attaquant d'intercepter les requêtes sur le réseau, et d'ainsi obtenir les documents désirés voir les informations d'identification (utilisateur et mot de passe). L'authentification va donc de pair avec le chiffrement des échanges.

Moins d'informations = intrusion plus difficile = Masquer l'identité du serveur Apache.

6.2. Masquage de l'identité d'Apache

Directive ServerSignature

Afficher une ligne de bas de page pour les documents générés par le serveur (messages d'erreur)

Syntaxe de la directive ServerSignature

```
ServerSignature  On | Off | EMAil
```

Exemple :

```
ServerSignature  Off
```

La directive "ServerSignature" ajoute un pied de page aux documents générés par le serveur (messages d'erreur, liste des répertoires, ftp, etc.).

L'utilité d'ajouter ces informations apparaît par exemple lorsqu'une réponse à une requête traverse plusieurs proxys. Dans ces conditions, sans ces informations, il devient difficile pour l'utilisateur de déterminer quel élément de la chaîne de proxy a produit un message d'erreur.

Sur un serveur de production, pour des raisons de sécurité, il est préférable de positionner cette option à Off.

ServerSignature On :

ServerSignature Off :

L'information fournie par cette page semble anodine mais pourtant est très précieuse pour un attaquant.

En effet, un éventuel attaquant apprend que le serveur apache, disponible à l'adresse IP 160.210.125.126 est un serveur Apache 2.2.6. La version actuelle d'apache étant la version 2.2.29 (en décembre 2014).

L'attaquant peut donc utiliser le document situé à cette adresse : http://www.apache.org/dist/httpd/

CHANGES_2.2 pour cibler ses attaques.

Directive ServerTokens

Renseigner le champ "server" de l'entête http.

Syntaxe de la directive ServerTokens

```
ServerTokens  Prod[uctOnly] | Min[imal] | OS | Full
```

Exemple :

```
ServerTokens Prod
```

Dans un navigateur web, la page internet que nous consultons n'est que la partie visible par l'utilisateur du contenu d'une requête HTTP. Les en-têtes HTTP fournissent de nombreuses informations, que ce soit du client vers le serveur ou du serveur vers le client.

Ces en-têtes peuvent être visualisée par exemple avec :

- la commande wget, qui permet le téléchargement d'une URL en ligne de commande,
- avec le module "Développement Web" fourni en standard avec le navigateur Firefox.

En-têtes sans ServerTokens (full)

```
[root]# wget –S  http://160.210.125.126

--14:30:07--  http://160.210.125.126/
=> `index.html'
Connexion vers 160.210.125.126:80...connecté.
requête HTTP transmise, en attente de la réponse...
  HTTP/1.1 200 OK
  Date: Mon, 01 Mar 2010 13:30:07 GMT
  Server: Apache/2.2.6 (Fedora) DAV/2
  Last-Modified: Mon, 01 Mar 2010 08:26:10 GMT
  ETag: "16e9e-3c55-594c80"
  Accept-Ranges: bytes
  Content-Length: 15445
  Connection: close
  Content-Type: text/html; charset=UTF-8
  Longueur: 15445 (15K) [text/html]

  100%[====================================>] 15445         --.--K/s

  14:30:07 (86.86 MB/s) - « index.html » sauvegardé [15445/15445]
```

Pour les mêmes raisons que pour la directive ServerSignature vue précédemment, il est impératif de limiter les informations transmises par un serveur de production.

Table 11. La directive ServerTokens

Valeur	Information
Prod[uctOnly]	Server : Apache
Min[imal]	Server : Apache/1.3.0
OS	Server : Apache/1.3.0 (Unix)
Full	Server : Apache/1.3.0 (Unix) PHP/3.0 MyMod/1.2

En-têtes avec ServerTokens Prod

```
[root]# wget –S  http://160.210.125.126

--14:30:07--  http://160.210.125.126/
=> `index.html'
Connexion vers 160.210.125.126:80...connecté.
requête HTTP transmise, en attente de la réponse...
  HTTP/1.1 200 OK
  Date: Mon, 01 Mar 2010 13:30:07 GMT
  Server: Apache
  Last-Modified: Mon, 01 Mar 2010 08:26:10 GMT
  ETag: "16e9e-3c55-594c80"
  Accept-Ranges: bytes
  Content-Length: 15445
  Connection: close
  Content-Type: text/html; charset=UTF-8
  Longueur: 15445 (15K) [text/html]

  100%[====================================>] 15445         --.--K/s

  14:30:07 (86.86 MB/s) - « index.html » sauvegardé [15445/15445]
```

Le choix le plus judicieux est généralement de modifier le fichier /etc/httpd/conf/httpd.conf pour mettre la directive ServerTokens à Prod

6.3. Gestion des authentifications

L'authentification est la procédure qui consiste à vérifier l'identité d'une personne ou d'un ordinateur afin d'autoriser l'accès de cette entité à des ressources (systèmes, réseaux, applications,...).

Il existe de nombreuses méthodes d'authentifications :

- Authentification classique (simple) ;

- Authentification LDAP ;
- Authentification via serveur de base de données ;
- Authentification via PAM ;
- Authentification via SSO.

Apache permet par défaut d'assurer ce processus, avec une authentification classique ou simple, qui s'appuie sur des fichiers de textes contenant les informations de connexion des utilisateurs.

Cette fonctionnalité basique peut être enrichie par des modules, et permettre à apache de s'appuyer sur :

- Une authentification via un serveur LDAP. Ce procédé permet de déléguer l'authentification des utilisateurs (et leur appartenance à des groupes) à un serveur LDAP, dont c'est la fonctionnalité première. Apache utilise alors le module mod_authnz_ldap, qui dépend du module Apache d'accès à LDAP, mod_ldap, qu'il faut aussi installer.
- Une authentification via un serveur de base de données. Ce procédé s'appuiera sur le langage SQL (Structured Query Language) pour la gestion des utilisateurs, de leurs mots de passe et leur appartenance à des groupes.
- Le module PAM (Pluggable Authentification Modules). Ce procédé permet de déléguer l'authentification au système d'exploitation. Pour mettre en place cette procédure, il faut installer deux modules : mod_auth_pam et pam_auth_external.

Ces modules ne sont pas actifs par défaut mais sont présents dans les dépôts.

Dans ce chapitre nous ne verrons que l'authentification classique (dite aussi « simple »).

Authentification classique

Authentification classique : protéger l'accès à un site ou à un dossier d'un site par la mise en place d'une authentification par mot de passe.

L'activation du module d'authentification pour un site peut se faire :

- Soit dans la configuration globale du site, si l'administrateur du site est également administrateur du serveur, ou que l'administrateur lui en a laissé l'accès.

Les directives de configuration se positionne entre les balises <Directory> ou <Location>.

C'est la méthode à privilégier.

- Si la modification du fichier de configuration globale n'est pas possible, ce qui est souvent le cas d'un serveur mutualisé, la protection se fera alors dans un fichier .htaccess directement dans le dossier à protéger.

Dans ce cas, l'administrateur du serveur aura explicitement configuré cette possibilité dans la configuration globale avec la directive AllowOverride à All ou à AuthConfig.

Ce fichier étant évalué à chaque accès, cela peut induire une petite perte au niveau des performances.

L'authentification sécurise l'accès aux données, mais la donnée transite toujours en clair durant la transmission au client.

Directive AuthType

Renseigner le type de contrôle des autorisations

Directives associées : AuthName, AuthUserFile

Syntaxe de la directive AuthType

```
AuthType Basic | Digest
```

Exemple :

```
AuthType Basic
```

AuthType indique à Apache d'utiliser le protocole Basic ou Digest pour authentifier l'utilisateur :

- Authentification Basic : Transmission du mot de passe client en clair

Pour mettre en place cette méthode, il faut utiliser la commande htpasswd qui permet de créer un fichier qui va contenir les logins (ou les groupes) et les mots de passe des utilisateurs (ou les groupes) habilités à accéder au dossier Web sécurisé (la commande étudiée plus loin).

- Authentification Digest : Hachage MD5 128 bits du mot de passe avant transmission

Ce module implémente l'authentification HTTP basée sur les condensés MD5, et fournit une alternative à mod_auth_basic en ne transmettant plus le mot de passe en clair.

Cependant, cela ne suffit pas pour améliorer la sécurité de manière significative par rapport à l'authentification basique. En outre, le stockage du mot de passe sur le serveur est encore moins sûr dans le cas d'une authentification à base de condensés que dans le cas d'une authentification basique.

C'est pourquoi l'utilisation de l'authentification basique associée à un chiffrement de la connexion via mod_ssl constitue une bien meilleure alternative.

Plus d'informations : http://httpd.apache.org/docs/2.2/fr/mod/mod_auth_digest.html.

Pour des raisons de sécurité, seul le mécanisme Basic sera utilisé par la suite.

Configuration du fichier /etc/httpd/conf/httpd.conf :

Dans les balises <Directory> ou <Location>, ajouter les directives d'authentification :

Syntaxe Apache pour protéger l'accès à un dossier

```
<Directory directory-path >
    AuthType Basic | Digest
    AuthName "text"
    AuthUserFile directory-path/.PrivPasswd
    Require user | group | valid-user
</Directory>
```

- AuthName est le message qui est affiché dans la boite de dialogue de saisie du login et du mot de passe.
- AuthUserFile indique le chemin et le nom du fichier contenant le nom des utilisateurs et leur mot de passe. Pour une identification sur un groupe il faut travailler avec la directive AuthGroupFile. Ces deux directives font parties du module mod_auth .
- Require est la règle d'authentification à proprement parler. Elle indique à Apache qu'un utilisateur ayant réussi à s'authentifier à partir du fichier des mots de passe (spécifié dans la directive AuthUserFile) peut accéder au site Web sécurisé.

Exemple :

```
<VirtualHost www.monsite.com >
......
  <Directory /var/www/html/sitetest>
   # Type d'authentification
   AuthType Basic
   # texte affiché dans la boite de dialogue
   AuthName " Acces Securise "
   # fichier contenant les logins et mdp
   AuthUserFile /var/www/private/sitetest/.PrivPasswd
   # Accès par vérification du mot de passe
   require valid-user
  </Directory>
</VirtualHost>
```

Le fichier .PrivPasswd est créé avec la commande htpasswd, que nous verrons plus loin dans ce chapitre.

Le fichier de gestion des logins et des mots de passe .PrivPasswd est un fichier sensible. Il ne faut pas le placer dans le répertoire de publication de votre site, mais plutôt dans un répertoire extérieur à l'arborescence de votre site suivi du nom du site.

Exemple

```
[root]# mkdir -p /var/www/private/SiteTest
```

Le fichier .PrivPasswd" sera ensuite créé dans ce répertoire à l'aide de la commande htpasswd.

Le fichier .htaccess utilise les mêmes directives que précédemment, mais non encadrés par les balises Directory ou Location.

Syntaxe:

```
AuthType Basic | Digest
AuthName "text"
AuthUserFile directory-path/.PrivPasswd
Require user | group | valid-user
```

Avec dans le fichier « /etc/httpd/conf/httpd.conf » <Directory directory-path > AllowOverride AuthConfig </Directory>

Un serveur web répond généralement pour plusieurs sites. Dans ce cas, il est plus simple de configurer ce type de spécificité de configuration directement dans le dossier en question.

- Avantages : l'usage d'un fichier .htaccess a le mérite d'être simple et permet notamment de protéger un dossier Web racine ainsi que les sous-dossiers (sauf si un autre fichier .htaccess y contrevient), il est possible de déléguer la configuration ou la personnalisation du service à un administrateur du site.
- Inconvénients : il n'est pas simple de maintenir un nombre élevé de fichiers .htaccess. L'évaluation du fichier .htaccess par le serveur peut induire sur les performances.

Il ne faut pas oublier d'autoriser la configuration du module d'identification par fichier .htaccess à l'aide de la directive : AllowOverride AuthConfig.

Sans cette directive, apache ne permettra pas au fichier .htaccess d'écraser la configuration définie dans la configuration globale.

```
<VirtualHost www.monsite.com >
    <Directory /home/SitesWeb/SiteTest>
       # déclaration de l'utilisation de .htaccess
      AllowOverride  AuthConfig
    </Directory>
</VirtualHost>
```

Exemple de fichier .htaccess

```
# Type d'authentification
AuthType Basic
# texte affiché dans la boite de dialogue
AuthName " Acces Securise "
# fichier contenant les logins et mdp
AuthUserFile /var/www/private/sitetest/.PrivPasswd
# Accès par vérification du mot de passe
require valid-user
```

Lors du traitement d'une requête, Apache cherche la présence du fichier .htaccess dans tous les répertoires du chemin menant au document depuis la directive DocumentRoot.

Exemple, avec une directive DocumentRoot à /home/sitesweb/ avant de retourner /home/sitesWeb/sitetest/indextest.htm Apache examine les fichiers :

- /home/sitesWeb/.htaccess
- /home/sitesWeb/sitetest/.htaccess

Mieux vaut désactiver cette fonctionnalité sur / (activé par défaut) : <Directory /> AllowOverride None </Directory>

Commande htpasswd

La commande htpasswd permet de gérer les utilisateurs du site.

Syntaxe de la commande htpasswd

```
htpasswd [-options] passwordfile username [password]
```

Exemples :

```
[root]# cd /var/www/private/sitetest
[root]# htpasswd -cb .PrivPasswd user1 mdpuser1
[root]# htpasswd –D .PrivPasswd  user
```

Table 12. La directive ServerTokens

Option	Observation
-c	Créer un nouveau fichier
-b	Indiquer le mot de passe sur la ligne de commande
-m	Chiffrer le mot de passe en md5
-D	Supprimer un utilisateur de la liste d'accès

La commande htpasswd met en place la liste des utilisateurs habilités à accéder au site sécurisé, dans le cas de l'utilisation de la directive « AuthType » avec la valeur « Basic » et vérifie les droits sur le répertoire, afin qu'au moins le groupe « apache » puisse y accéder.

Pour créer le fichier et définir le premier utilisateur :

```
[root]# cd /var/www/private/siteTest
[root]# htpasswd –c .PrivPasswd user1
```

Puis saisir le mot de passe

Pour ajouter les autres utilisateurs dans le fichier :

```
[root]# htpasswd  .PrivPasswd user2
```

Puis saisir le mot de passe

Pour ajouter un utilisateur et définir son mot de passe à la suite de la commande

```
[root]# htpasswd  -b .PrivPasswd user3 mdpuser3
```

Ajouter un utilisateur avec un mot de passe chiffré en md5 :

```
[root]# htpasswd  -bm .PrivPasswd user4 mdpuser4
```

Le résultat donne un fichier /var/www/private/sitetest/.PrivPasswd :

```
user1:$1Pd$EsBY75M
user2:Mku4G$j4p£kl1
user3:Ng7Rd$5F$68f
...
```

6.4. Utilisation du module SSL

Le protocole TLS (Transport Layer Security), successeur du protocole SSL (Secure Socket Layer) est un protocole de sécurisation des échanges sur Internet.

Le protocole SSL a été créé à l'origine par Netscape. Le principe de fonctionnement du TLS repose sur le recours à un tiers, l'Autorité de Certification (CA/Certificate Authority).

C'est un protocole de niveau 4 sur lequel les protocoles des couches OSI supérieures s'appuient. Ils est donc commun pour les protocoles imaps, pops, ldaps, etc.

Le fichier openssl.cnf (/etc/pki/tls/openssl.cnf) peut être configuré de façon à minimiser les données à renseigner à chaque invocation des utilitaires openssl lors de la création des clefs de chiffrement.

Etant donné son caractère hautement critique, l'administrateur veillera à utiliser une version d'openssl à jour de correctifs.

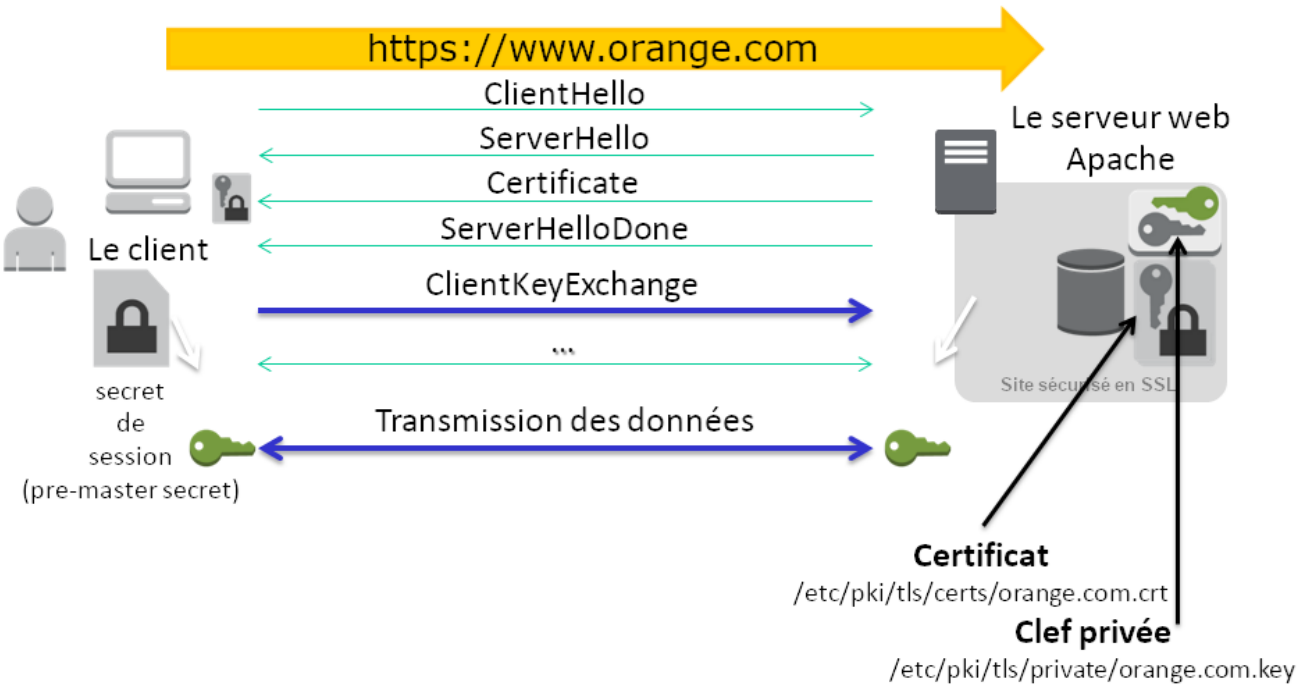
Le protocole SSL/TLS existe en 5 versions :

- SSLv2
- SSLv3
- TLSv1
- TLSv1.1
- TLSv1.2

Prérequis

- Le port 443 (https) est-il ouvert sur les pare-feu ?
- Le logiciel OpenSSL est-il installé et à jour sur le serveur ?
- Le module mod_ssl est-il installé sur le serveur Apache et activé ?

Établissement d'une session TCP https (port 443)



Lors du ClientHello, le client se présente au serveur. Il lui soumet les méthodes de cryptologie, de compression, et les standards SSL qu'il connaît.

Le serveur répond au client avec un message ServerHello. Les 2 parties se sont mises en accord sur le CypherSuite qu'ils allaient utiliser, par exemple avec un CypherSuite TLS_RSA_WITH_AES_128_MD5.

Ils doivent maintenant s'échanger un secret de session : le pre-master secret. Son chiffrement s'effectue avec le certificat publique du serveur transmis pendant le message Certificate.

Le pre-master secret est dérivé en clefs symétriques sur le client et le serveur. Ces clefs symétriques serviront à la transmission des données.

La chaîne (CypherSuite) est composée de 5 éléments distincts :

- Le protocole SSL/TLS
- L'algorithme asymétrique utilisé, principalement RSA et ECDSA
- L'algorithme symétrique utilisé, comme AES, Camelia, SEED, 3DES ou RC4
- Le mécanisme de protection des données pour éviter qu'un assaillant puisse modifier les messages chiffrés (HMAC ou AEAD). Dans le cas du HMAC on choisira une fonction de hachage (MD5, SHA-1, SHA-2)
- La présence ou non de confidentialité persistante ou PFS (Perfect Forward Secrecy)

Mise en place d'un site TLS

La mise en place d'un site TLS respecte les étapes suivantes :

1. Installation du module mod_ssl

2. Configuration du module mod_ssl
3. Création de la clé privée du serveur
4. Création du certificat du serveur depuis sa clé privée
5. Création d'un Certificat Signing Request (CSR) depuis le certificat et transmission à la CA pour signature
6. Installation du certificat
7. L'hôte virtuel peut être configuré en TLS

Le logiciel OpenSSL

Le logiciel OpenSSL, utilitaire cryptographique, implémente les protocoles réseaux :

- Secure Sockets Layer (SSL V2/V3, couche de sockets sécurisés) ;
- Transport Layer Security (TLS v1, sécurité pour la couche de transport).

OpenSSL permet :

- La création de paramètres des clefs RSA, DH et DSA ;
- La création de certificats X.509, CSRs et CRLs ;
- Le calcul de signature de messages ;
- Le chiffrement et déchiffrement ;
- Les tests SSL/TLS client et serveur ;
- La gestion de mail S/MIME signé ou chiffrés.

Création des clés et certificats

Les clés et certificats sont stockés dans le répertoire /etc/pki/tls/.

- Un dossier private accueille les clés privées.
- Un dossier certs accueille les certificats publiques.

Les droits doivent être à 440.

Pour installer le module mod_ssl :

```
[root]# yum install mod_ssl
```

Pour vérifier la présence du module dans le serveur Apache :

```
[root]# httpd –t –D DUMP_MODULES | grep ssl_module
ssl_module (shared)
```

La commande nous indique que le module est disponible mais pas qu'il est activé.

Après cette installation, vous devez trouver le module "mod_ssl.so" dans le répertoire /etc/httpd/modules" qui est en fait un lien symbolique sur /usr/lib/httpd/modules.

Pour activer le mod_ssl dans /etc/httpd/conf/httpd.conf, la ligne suivante doit être présente :

```
LoadModule ssl_module modules/mod_ssl.so
```

Le module mod_ssl peut être configuré dans le fichier /etc/httpd/conf.d/ssl.conf

N'oubliez pas de redémarrer le service Apache :

```
[root]# service httpd restart
```

Pour accepter les requêtes TLS, le serveur Apache a besoin de deux fichiers :

- une clé privée : NomDeFichier.key;
- un certificat signé : NomDeFichier.crt

La signature de ce certificat est réalisée par un organisme de certification tiers (tel que Verisign ou Thawte). Cependant vous pouvez signer vous même votre certificat, la seule différence sera un avertissement par le navigateur lors de l'accès à une ressource TLS de votre serveur. La sécurité est la même, que le certificat soit signé par vous même ou pas un organisme.

Si vous décidez d'utiliser une autorité de certification auto-signée, son certificat devra être installée sur l'ensemble de vos postes.

- 1ère étape : Générer la clef privée

```
openssl genrsa \    –out /etc/pki/tls/private/orange.com.key \  2048
Generating RSA private key, 2048 bit long modulus
- - - - - - - - - - - - - - - - +++
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +++
e is 65537 (0x10001)

chmod 400 /etc/pki/tls/private/orange.com.key
```

- 2ème étape : Générer une demande de certificat

```
openssl req    –new     –key /etc/pki/tls/private/orange.com.key     –out
/etc/pki/tls/certs/orange.com.csr
```

- 3ème étape : Envoi de la demande de certificat à l'autorité de certification (CA)
- 4ème étape : L'autorité de certification (CA) renvoie un certificat signé
- 5ème étape : Sécuriser et sauvegarder les certificats.

```
chmod 400 /etc/pki/tls/private/orange.com.key
chmod 400 /etc/pki/tls/certs/orange.com.crt
```

- 6ème étape : Déployer les certificats sur le serveur
- 7ème étape : Configurer le vhost

```
NameVirtualHost 192.168.75.50:443
<VirtualHost 192.168.75.50:443>
  ServerName www.orange.com

  SSLEngine on
  SSLCertificateFile /etc/pki/tls/certs/orange.com.crt
  SSLCertificateKeyFile /etc/pki/tls/private/orange.com.key

  DocumentRoot /home/SitesWeb/SiteOrange
  DirectoryIndex IndexOrange.htm

  <Directory /home/SitesWeb/SiteOrange>
    allow from all
  </Directory>

</VirtualHost>
```

Certificats Auto-Signés

Auto-signer ses certificats revient à créer sa propre autorité de certification.

- 1ère Etape : Générer la paire de clefs de l'autorité de certification

```
openssl genrsa –out /etc/pki/CA/private/cakey.pem
openssl req \    –new \    –x509 \    –key /etc/pki/CA/private/cakey.pem \  -out
/etc/pki/CA/certs/cacert.pem \  -days 365
```

- 2ème Etape : Configurer openssl

/etc/pki/tls/openssl.cnf

```
[ ca ]default_ca   =   CA_default

[ CA_default ]

dir = /etc/pki/CA
certificate = $dir/certs/cacert.pem
private_key = $dir/private/cakey.pem
```

Vérifier la présence du fichier /etc/pki/CA/index.txt.

Si celui-ci s'avère absent, il faut le créer :

```
touch /etc/pki/CA/index.txt
echo '1000' >  /etc/pki/CA/serial
```

Puis faire :

```
echo '1000' > /etc/pki/CA/serial
```

- 3ème Etape : Signer une demande de certificat

```
openssl ca \   -in /etc/pki/tls/certs/orange.com.csr \   -out
/etc/pki/tls/certs/orange.com.crt
```

Pour que ce certificat soit reconnu par les clients, le certificat cacert.pem devrait également être installé sur chaque navigateur.

Le certificat orange.com.crt est à envoyer au client

Chapitre 7. Apache - haute disponibilité

Objectifs de ce cours :

- paramétrer Apache en serveur frontal
- répartir la charge entre plusieurs serveurs applicatifs
- paramétrer Apache pour la tolérance de panne

7.1. Introduction

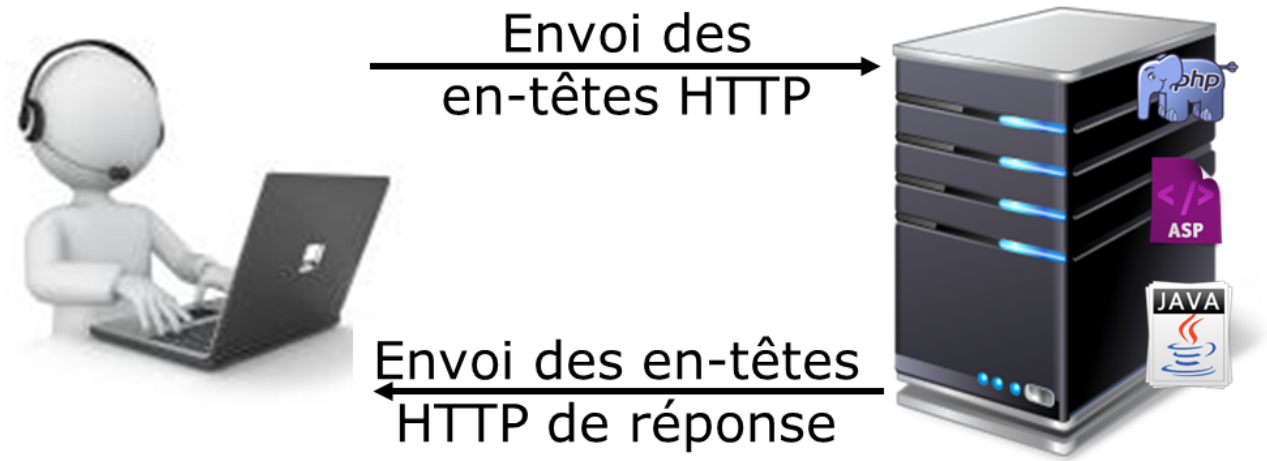
Apache est un serveur **modulaire** web (protocole HTTP).

Il peut :

- Gérer un cache de données ;
- Faire office de serveur mandataire ;
- Compresser les données envoyées aux clients ;
- et plein d'autres choses encore...

7.2. Serveur Applicatif

Un serveur applicatif héberge des applications dynamiques développées dans un langage de programmation web : Php / Asp / Java / Python ...



7.3. Reverse proxy

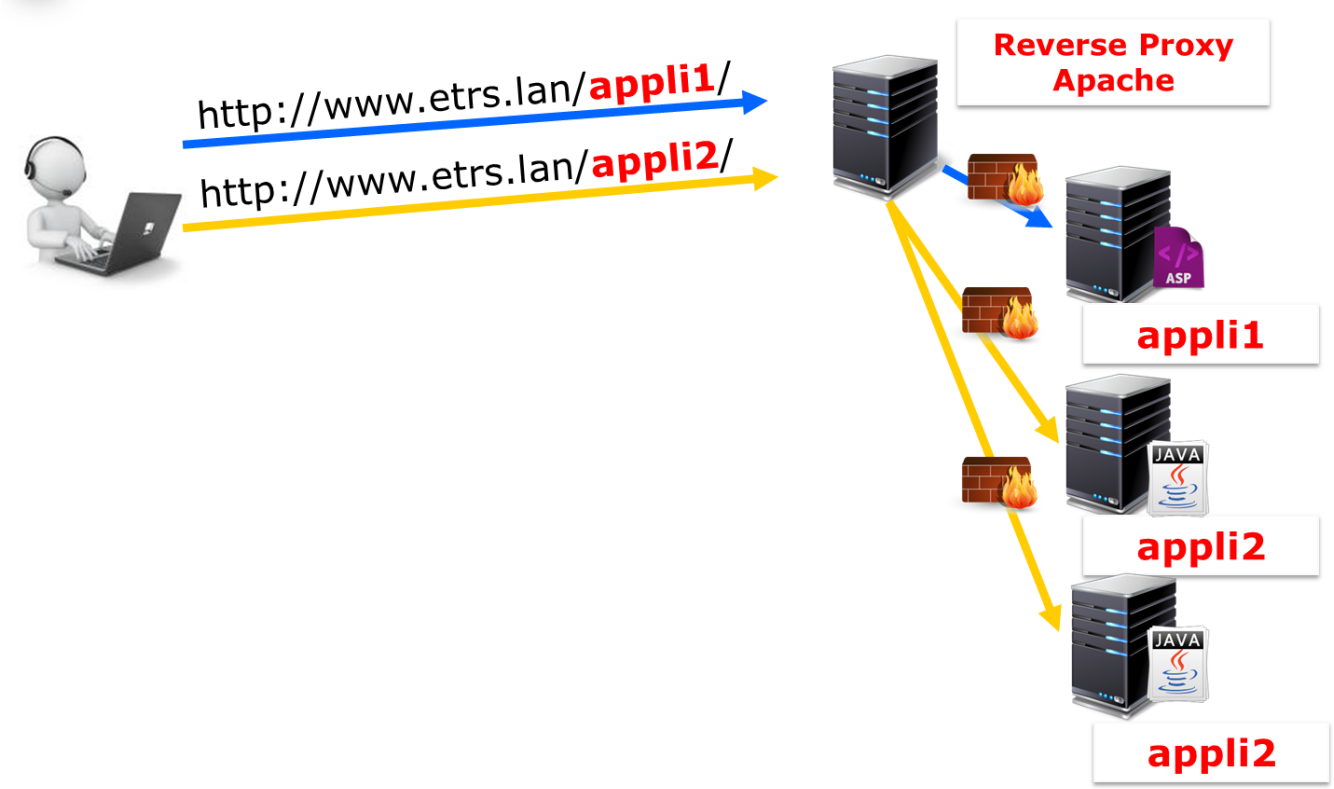
Un serveur reverse proxy (ou serveur mandataire) est mandaté par des clients pour interroger les serveurs applicatifs.

Aux yeux du client, un seul serveur est en ligne pour lui répondre. La complexité du SI lui est

masqué.

Les serveurs applicatifs sont isolés derrière un (ou plusieurs) pare-feu. Leurs ressources sont dédiées aux processus métiers (SQL, Web-services, etc.).

Le reverse proxy se charge de la compression, de la mise en cache, du chiffrement, et de la répartition de charge et/ou tolérance de panne.



Le module mod_proxy

Le module **mod_proxy** est une extension d'apache pour faire office de serveur mandataire / Passerelle HTTP.

- La directive **ProxyPreserveHost On** utilise l'en-tête de requête entrante Host pour la requête du mandataire
- La directive **ProxyPass chemin url** référence des serveurs distants depuis l'espace d'URLs du serveur local
- La directive **ProxyPassReverse chemin url** ajuste l'URL dans les réponses HTTP envoyées par le mandataire en inverse

Les différentes fonctionnalités de mandataire d'Apache sont réparties entre plusieurs modules complémentaires :

- mod_proxy_http,
- mod_proxy_ftp,
- mod_proxy_ajp,

- mod_proxy_balancer
- et mod_proxy_connect.

Apache peut être configuré comme mandataire directe (proxy) ou comme mandataire inverse (reverse proxy ou mode passerelle).

Pour la configuration d'un mandataire direct, reportez vous à la documentation du serveur SQUID.

Un mandataire inverse apparaît au client comme un serveur web standard. Aucune configuration du client n'est nécessaire. Le client adresse ses demandes de contenus ordinaires dans l'espace de nommage du mandataire inverse. Ce dernier décide alors où envoyer ces requêtes et renvoie le contenu au client comme s'il l'hébergeait lui-même.

L'accès des utilisateurs à un serveur situé derrière un pare-feu est une utilisation typique du mandataire inverse.

Configuration exemple d'un VirtualHost avec reverse-proxy

```
<VirtualHost 127.0.0.1:8080>
  ProxyPreserveHost On
  ProxyVia On
  <Proxy *>
    Order deny,allow
    Allow from all
  </Proxy>

  ProxyPass /pagebleue http://127.0.0.1:8080/pagebleue
  ProxyPassReverse /pagebleue http://127.0.0.1:8080/pagebleue
</VirtualHost>
```

- La directive **ProxyPreserveHost on**, lorsqu'elle est activé, va transmettre l'en-tête Host: de la requête entrante vers le serveur mandaté au lieu du nom d'hôte spécifié par la directive ProxyPass.
- La directive **ProxyVia On** permet de contrôler l'utilisation de l'en-tête http **Via:**. Définie à On, chaque requête ou réponse se verra ajouter une ligne d'en-tête Via: pour le serveur courant.

Des solutions spécialisées, comme **HaProxy** ou des équipements physiques comme les **Citrix NetScaler**, existent sur le marché, et sont plus puissantes qu'Apache. Au moment de choisir la solution de mandataire inverse, il faudra prendre en compte les éléments du tableau suivant :

	Points positifs	**Points négatifs**
HaProxy	Plus rapide	Spécifique au reverse proxy (pas de cache, nécessite Varnish ou Memcached)

	Points positifs	Points négatifs
	Développement actif	La syntaxe est différente des autres logiciels utilisés (NginX, Varnish, etc.)
Apache	Grand nombre de fonctionnalités, il peut tout faire	Performances moindres
	Solution unique pour tout le SI	
	Facilité de maintenance	

7.4. Simuler la charge

La commande **ab** (Apache Benchmark) permet d'envoyer des requêtes http à un client

Syntaxe de la commande ab

```
ab –n X –c Y url
```

Exemple :

Exemple d'utilisation de la commande ab

```
ab –n 5000 –c 2 http://127.0.0.1:8080/page1
```

Table 13. Options de la commande ab

Options	Commentaires
-n	Nombre de requêtes à envoyer
-c	Nombre de requêtes à envoyer par paquets

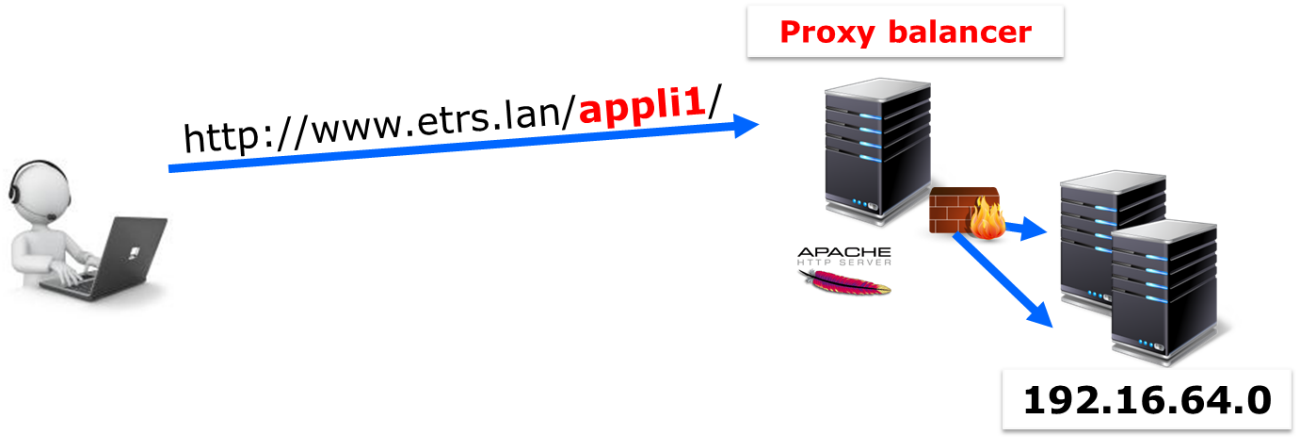
7.5. Répartir la charge

Pourquoi répartir la charge ?

- Améliorer les performances
- Accueillir plus de clients
- Gérer la maintenance
- Faire de la tolérance de panne

Plusieurs petits serveurs seront plus puissants qu'un gros pour un coût moindre.

Pour répartir la charge, Apache utilise un proxy-balancer.



Le module mod_proxy_balancer

Le module **mod_proxy_balancer** est une extension pour la répartition de charge.

- La directive **BalancerMember url [clé=valeur]** ajoute un membre à un groupe de répartition de charge
- La directive **loadfactor=X** est un facteur de charge entre 1 et 100
- La directive **ProxySet url [clé=valeur]** défini les différents paramètres relatifs à la répartition de charge
- La directive **lbmethod=byrequests|bytraffic|bybusyness** spécifie la méthode de répartition de charge (par défaut à byrequests). Les 3 méthodes de répartition de charge sont :
  - byrequests : réparti la charge entre les membres du cluster en fonction du nombre de requêtes traitées
  - bytraffic : réparti la charge entre les membres du cluster en fonction du trafic traitées
  - bybusyness : réparti la charge entre les membres du cluster en fonction de la charge actuelle
- La directive **stickysession=JSESSIONID|PHPSESSIONID** spécifie un nom de session persistant du répartiteur et dépend du serveur d'applications d'arrière plan.

```
<Proxy balancer://mycluster>
  BalancerMember http://192.16.164.1:8080/pages loadfactor=50
  BalancerMember http://192.16.164.2:8080/pages loadfactor=50

  ProxySet lbmethod=bytraffic
</Proxy>

ProxyPass /monsite balancer://mycluster
ProxyPassReverse /monsite balancer://mycluster
```

Le module mod_status

Le module **mod_status** permet de suivre l'état du load-balancer.

- La directive **ProxyStatus On** affiche l'état du répartiteur de charge du mandataire
- La directive **SetHandler balancer-manager** force le traitement par le gestionnaire balancer-manager

```
ProxyStatus On

<Location /balancer-manager>
  SetHandler balancer-manager
  Allow from all
</Location>
```

Vous pouvez maintenant visiter la page /balancer-manager

7.6. Tolérance aux pannes

Apache peut réagir en cas de panne du serveur métier et exclure du cluster le service qui ne répond plus et qui renvoie un code erreur `http`.

```
<Proxy balancer://mycluster>
  BalancerMember http://127.0.0.1:8080/pagebleue loadfactor=50 retry=30
  BalancerMember http://127.0.0.1:8080/pageverte loadfactor=50 retry=30

  ProxySet lbmethod=byrequests failonstatus=500,502,503,504

</Proxy>
```

Chapitre 8. Serveur de messagerie Postfix

8.1. Généralités

Le courrier électronique est apparu sur le réseau **ARPANET** dans les années 1970, ce qui fait du protocole **SMTP** un des plus anciens protocoles de **TCP/IP**.

Avec Internet, les systèmes de messagerie ont pris de l'importance et en 1980, **Sendmail** a été développé. Sendmail est devenu le premier serveur de messagerie important et utilisait déjà le protocole SMTP.

Postfix, dont le développement a été aidé par IBM, est apparu dès **1998**, afin de résoudre les problèmes de sécurité de Sendmail, tout en offrant une **administration beaucoup plus souple et modulaire**.

Le **MTA** (**Mail Transfer Agent**) par défaut de la distribution RedHat (Sendmail) est remplacé par Postfix sur la RedHat 6 (Novembre 2010).

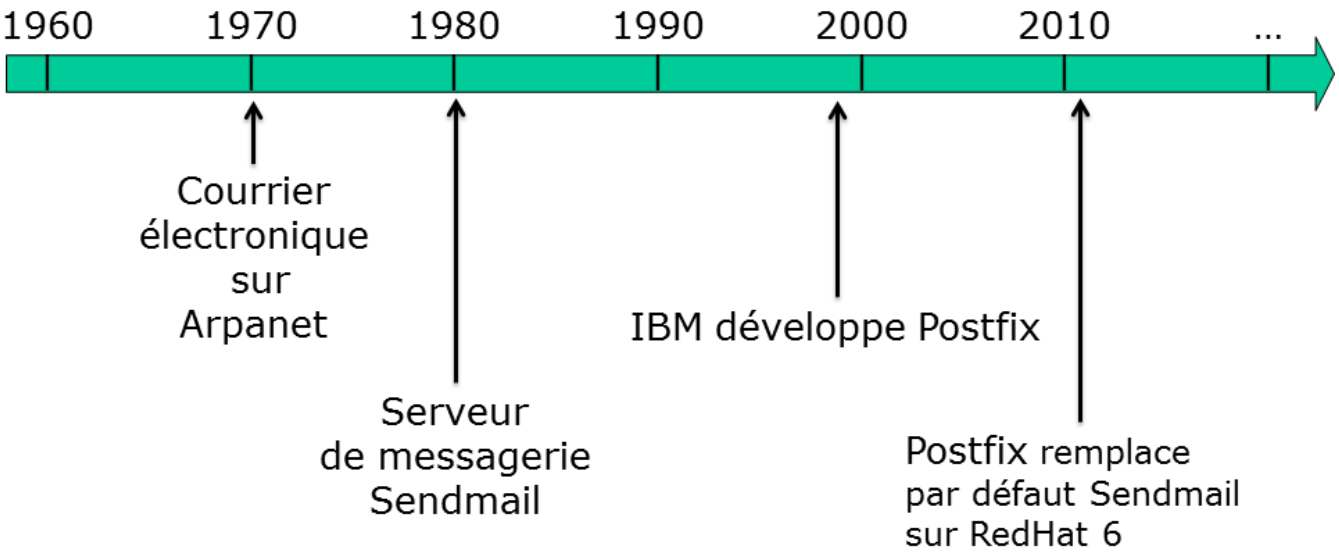


Figure 18. Historique des serveurs de messagerie sous Linux

Les systèmes de messagerie électronique reposent sur plusieurs standards et protocoles, définissant la façon dont sont composés les messages et leur acheminement, du rédacteur au destinataire.

L'**envoi de message** est assuré par le protocole **Simple Mail Transfer Protocol** (SMTP). SMTP est un protocole de communication de type texte.

Le fait que le protocole SMTP soit de type texte est intéressant. En effet, cela permet de tester son bon fonctionnement avec la commande **telnet**.

Les ports utilisés sont :

- **25** (sans authentification) ;
- **587** (avec authentification) ;
- **465** (SSL).

Local Mail Transfer Protocol (**LMTP**) est une variante de **ESMTP**, l'extension de SMTP. LMTP est défini dans la RFC 2033. LMTP a été conçu comme alternative aux échanges SMTP normaux dans les situations où la partie réceptrice ne possède pas de file d'attente des messages reçus. C'est le cas par exemple d'un agent de transfert de courrier agissant en tant qu'agent de distribution du courrier.

La **réception de message** peut se faire à l'aide de deux protocoles :

- **Internet Message Access Protocol** (**IMAP**) ;
- **Post Office Protocol** (**POP**).

IMAP est un protocole permettant de récupérer les courriers électroniques déposés sur des serveurs de messagerie. Les messages sont conservés sur le serveur, ses fonctionnalités avancées en font le protocole de préférence (accès **multi-postes**, accès via **webmail**, etc.).

IMAP utilise le port **143** en clair ou via **STARTTLS** et le port **993** via **IMAPS** (déprécié) en SSL.

POP est également un protocole permettant de récupérer les courriers électroniques de l'utilisateur. En règle générale, POP se connecte sur le serveur, récupère le courrier, efface le courrier sur le serveur et se déconnecte. Ce fonctionnement par défaut ne permet pas l'accès aux mails depuis plusieurs postes, ni l'itinérance.

POP utilise le port **110** en clair ou via **STARTTLS** et le port **995** (déprécié) en SSL.

Le langage Sieve a été conçu pour permettre de filtrer des messages directement sur les serveurs.

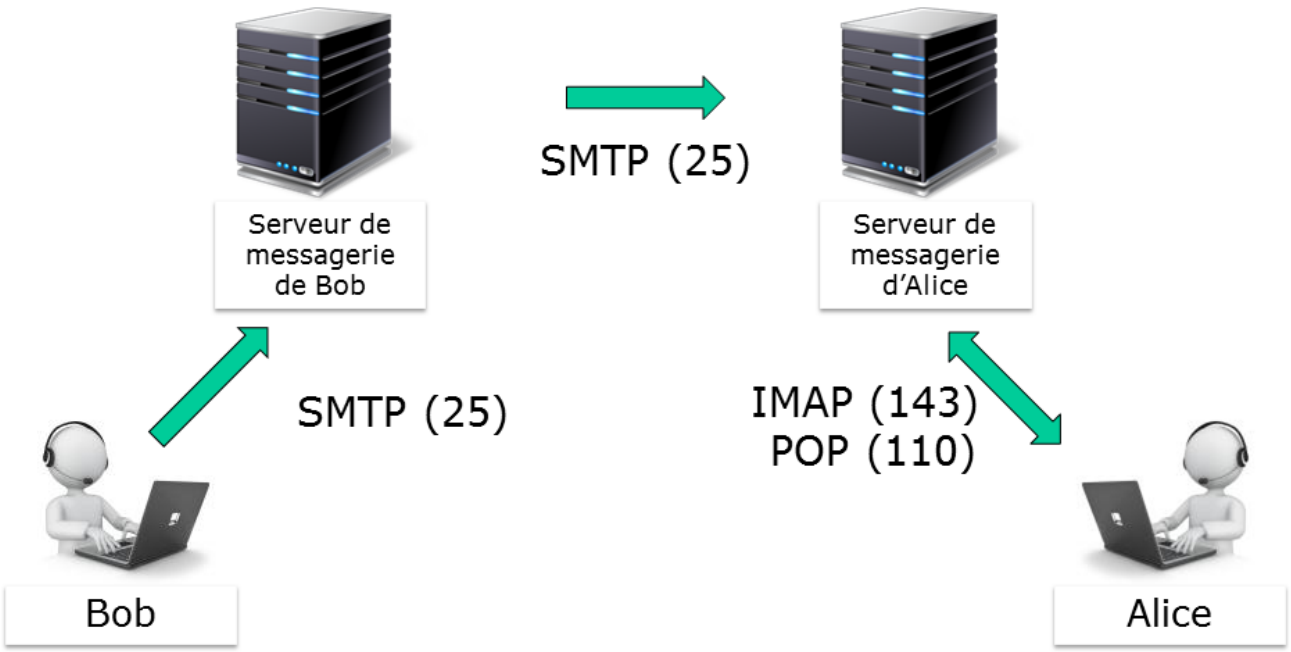


Figure 19. Les protocoles de messagerie mis en oeuvre

Les agents de messagerie

Mail User Agent

Un **Mail User Agent** (**MUA**), ou client de messagerie, est un logiciel servant à **lire** et à **envoyer** des courriers électroniques. Ce sont en général des clients lourds, mais il existe aussi des applications Web : les **webmails**. Les webmails tentent d'offrir les mêmes fonctionnalités qu'un client lourd.

Ces logiciels prennent en charge le protocole **SMTP** pour l'envoi de messages et les protocoles **POP** et **IMAP** pour leur réception.

Parmi les MUA les plus connus, citons : Thunderbird (Mozilla), Evolution (Novell), Outlook et Windows Mail (Microsoft), Kmail, Lotus Notes, Apple Mail, Opéra Mail, etc.

Mail Tranfer Agent

Lorsqu'un message est envoyé depuis un client de messagerie **MUA**, il est transféré au serveur de courrier, le **Mail Transfer Agent** (**MTA**). Un MTA implémente à la fois le **client** (transfert de mail) et le **serveur** (réception) du protocole SMTP.

Les termes Mail Server, Mail Exchanger, Mail Relay et MX Hosts peuvent aussi faire référence à un serveur MTA.

Le système de nom de domaine **DNS** associe à un domaine un ou plusieurs serveur de messagerie (Mail Exchanger - MX) par ordre de priorité (la plus haute priorité étant la plus faible valeur). Un MTA peut donc transférer un mail pour lequel il n'est pas destinataire soit à un serveur relais (si configuré), soit au serveur désigné par l'enregistrement MX du système DNS (le saut suivant - Next Hop).

Les MTA font donc en sorte qu'un message soit délivré d'un système à un autre. Si le MTA ne peut ni accepter, ni relayer un message, il le renvoie à son expéditeur.

Le message arrive au MTA responsable du domaine qui le stocke dans la boîte aux lettres de l'utilisateur.

Parmi les MTA les plus connus, citons : Postfix, Sendmail, Exim, Exchange, Qmail, Lotus Notes.

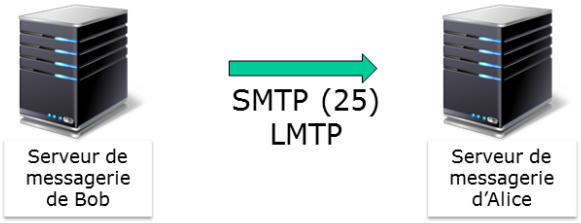


Figure 20. Le transfert de message entre MTA

Mail Delivery Agent

Le **Mail Delivery Agent** (**MDA**), agent de **distribution du courriel**, est le logiciel qui intervient dans la dernière étape du processus de distribution d'un courrier électronique. Il est responsable de la **disposition du message dans la boîte aux lettres** de l'utilisateur. Il peut également être appelé **Local Delivery Agent** (**LDA**).

C'est le MDA qui est chargé de gérer les problèmes comme un disque plein, une corruption de la boîte aux lettres, etc. et de signaler au MTA toute erreur de distribution.

Le MTA communique avec le MDA par l'intermédiaire des canaux d'entrées-sorties standards ou par un protocole spécialisé comme LMTP ou UUCP.

Parmi les MDA les plus connus, citons : Dovecot, Cyrus, Procmail, Local.

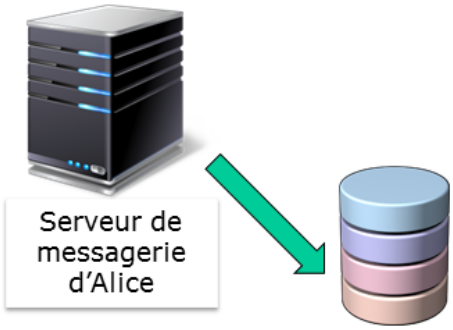


Figure 21. Le rôle de MDA

Mail Access Agent

Le **Mail Access Agent** (**MAA**), permet à l'utilisateur final, après **authentification**, de récupérer le message. Le MUA de l'utilisateur communique avec le MAA par le protocole IMAP ou POP.

Parmi les MAA les plus connus, citons : Dovecot, Cyrus, Courier, Exchange.

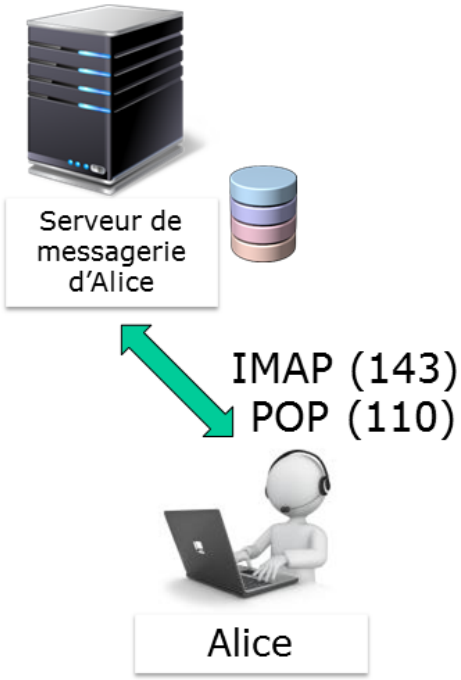


Figure 22. Le rôle de MAA

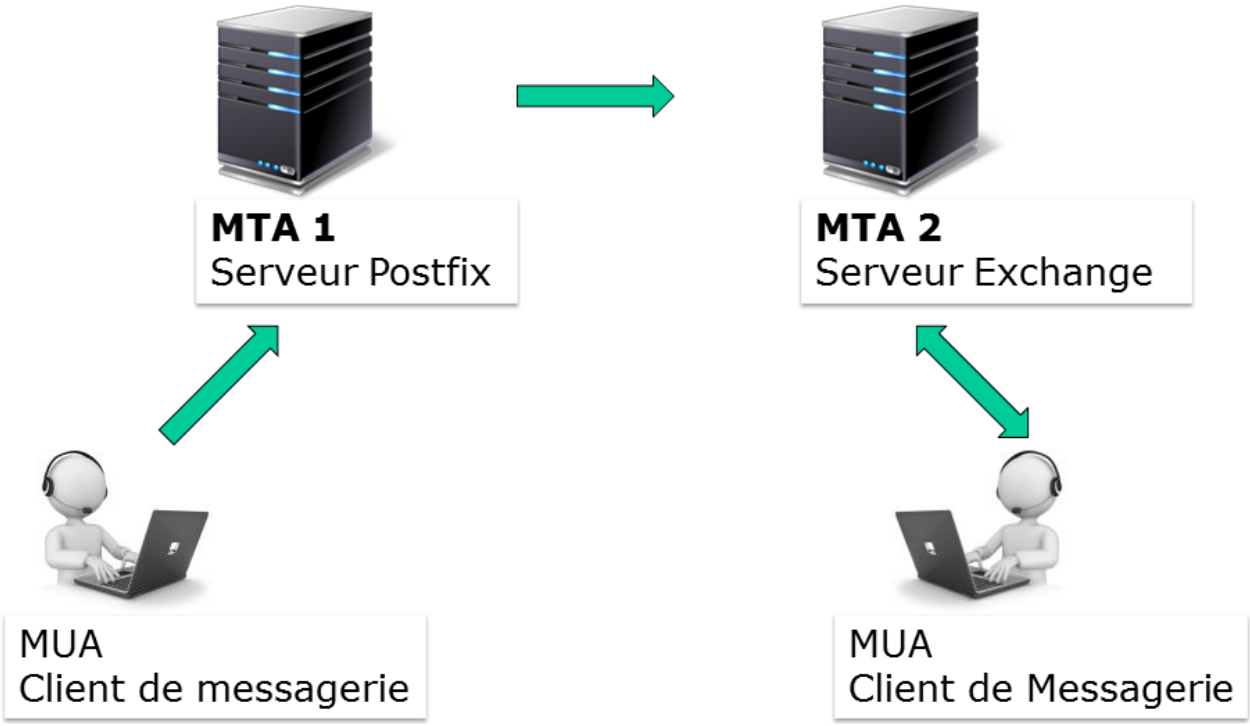


Figure 23. Les agents de transmission de message

Les relais SMTP

Le terme de "**relais de messagerie**" est couramment utilisé, dans le cas d'un MTA qui n'assurerait pas lui même la livraison du message au MTA final, mais qui se contenterait de servir d'intermédiaire entre le MUA du client et le MTA qui prend réellement l'acheminement du message en charge.

Cette fonctionnalité de relais est courante. Elle est par exemple présente dans nos box internet, qui

acheminent l'ensemble des messages émis par un domicile vers un des serveurs MTA centraux. Les messages peuvent ensuite être filtrés (lutte anti-spam ou surveillance ?). Les fournisseurs d'accès à Internet évitent, en bloquant le port 25 en sortie des box, que des serveurs SMTP soient directement sollicités.

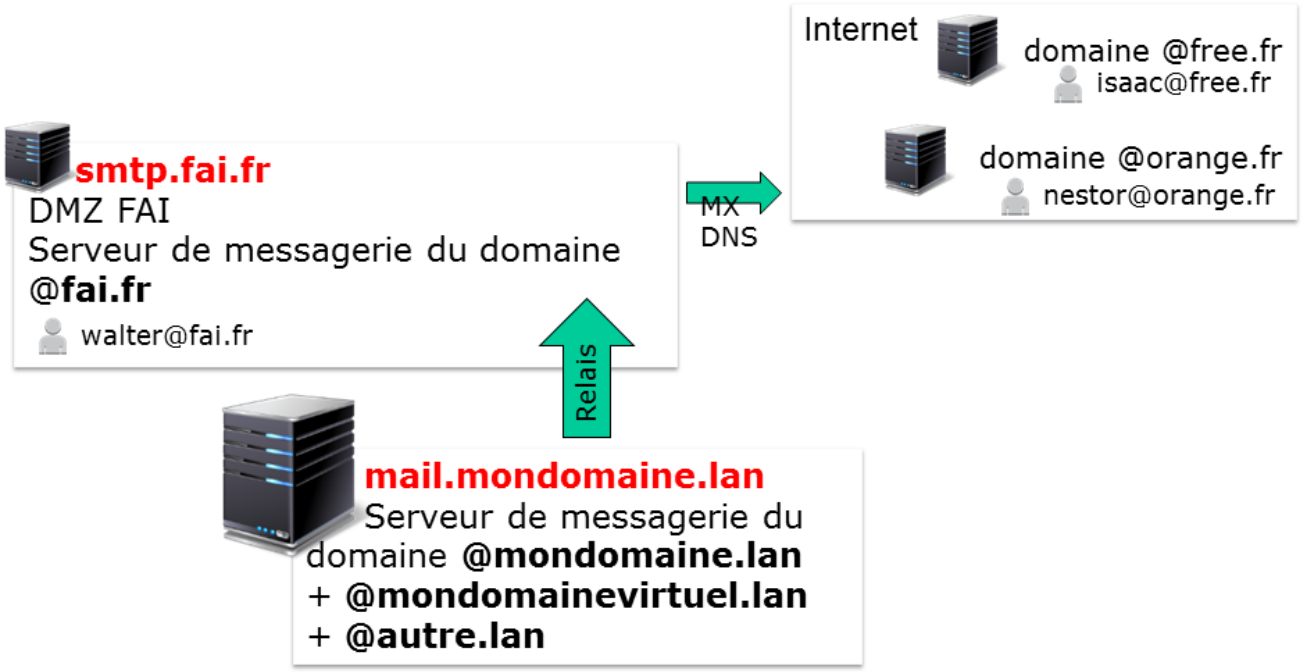


Figure 24. Système de messagerie avec relais

Les formats de stockage

Le format **mbox** est un format ouvert de stockage de courrier. Il repose sur le principe d'attribuer un fichier à chaque dossier (au lieu d'un fichier par message ou d'un répertoire par dossier).

Le format mbox permet un **affichage rapide d'une liste de mail**, puisqu'il ne nécessite l'ouverture que d'un seul fichier. La suppression ou l'ajout de mail est plus long et plus complexe à réaliser d'un point de vue système. L'accès concurrent à la même boîte aux lettres n'est pas possible car un verrou est positionné sur le fichier lors d'une action d'ajout ou de suppression.

Le format mbox est le format par défaut de postfix

Le format **Maildir** est une **structure de répertoires particulière**. Les courriels sont sauvegardés dans un fichier séparé, au sein d'une arborescence spécifique, ce qui lève le problème de verrou du format mbox.

De par son architecture, le format **Maildir est performant et fiable**. Il est plus adapté au protocole IMAP. À noter toutefois que l'affichage d'une liste de mails sera moins rapide qu'avec le format mbox.

Chaque répertoire Maildir contient au moins 3 sous-répertoires : **tmp**, **new**, **cur**. Les mails sont placés dans le répertoire tmp, puis déplacés par le MTA dans le répertoire new, pour enfin être déplacés après accès par un MUA dans le répertoire cur.

Synoptique

Un **MTA** pourra assurer les fonctionnalités suivantes :

- **Serveur de messagerie local** pour les comptes systèmes locaux comme root, bob, alice, etc.
- **Serveur d'un ou de plusieurs domaines de messagerie**, pour des comptes root@mondomain.lan, bob@mondomaine.lan, etc.
- **Serveur relais pour les domaines extérieurs** à son périmètre de gestion.

Un MTA peut également posséder des **routes spéciales**, contenues dans une **table de routage**, pour délivrer des messages à des serveurs de messagerie sans prendre en compte le chemin standard (par le relais ou par le serveur MX désigné lors d'une requête DNS).

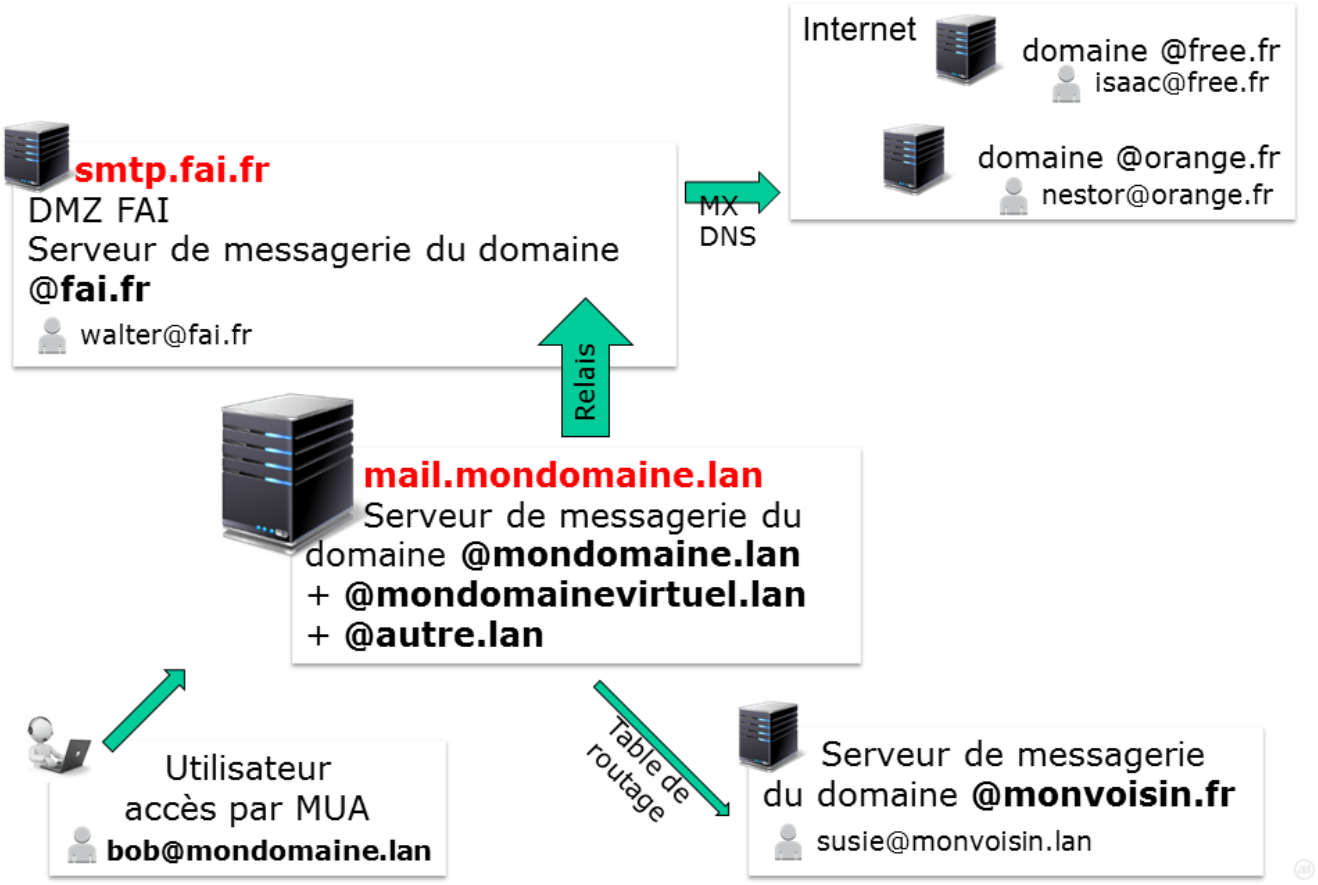


Figure 25. Synoptique global du système de messagerie

8.2. Installation du service

Postfix devrait normalement être installé par défaut sur une RedHat/CentOS 6 ou 7.

```
[root]# yum install postfix
```

Postfix nécessite bien évidemment que le service network soit correctement configuré.

L'installation de postfix ajoute un utilisateur postfix, membre du groupe postfix.

```
[root]# grep postfix /etc/passwd
postfix:x:89:89::/var/spool/postfix:/sbin/nologin

[root]# grep postfix /etc/group
postfix:x:89:x
```

Postfix étant un service réseau, il faut le paramétrer pour un démarrage à minima aux niveaux 3 et 5, puis le démarrer.

L'installation d'un nouveau service sur les distributions RedHat/CentOS n'implique pas leur démarrage automatique.

Après toute installation d'un service, il ne faut pas oublier de le démarrer avec la commande service, et d'automatiser le démarrage au reboot avec la commande chkconfig.

Tout service dépendant du réseau devrait être démarré comme le service network (chkconfig --list network).

```
[root]# chkconfig postfix on
[root]# service postfix start
```

8.3. Arborescence et fichiers

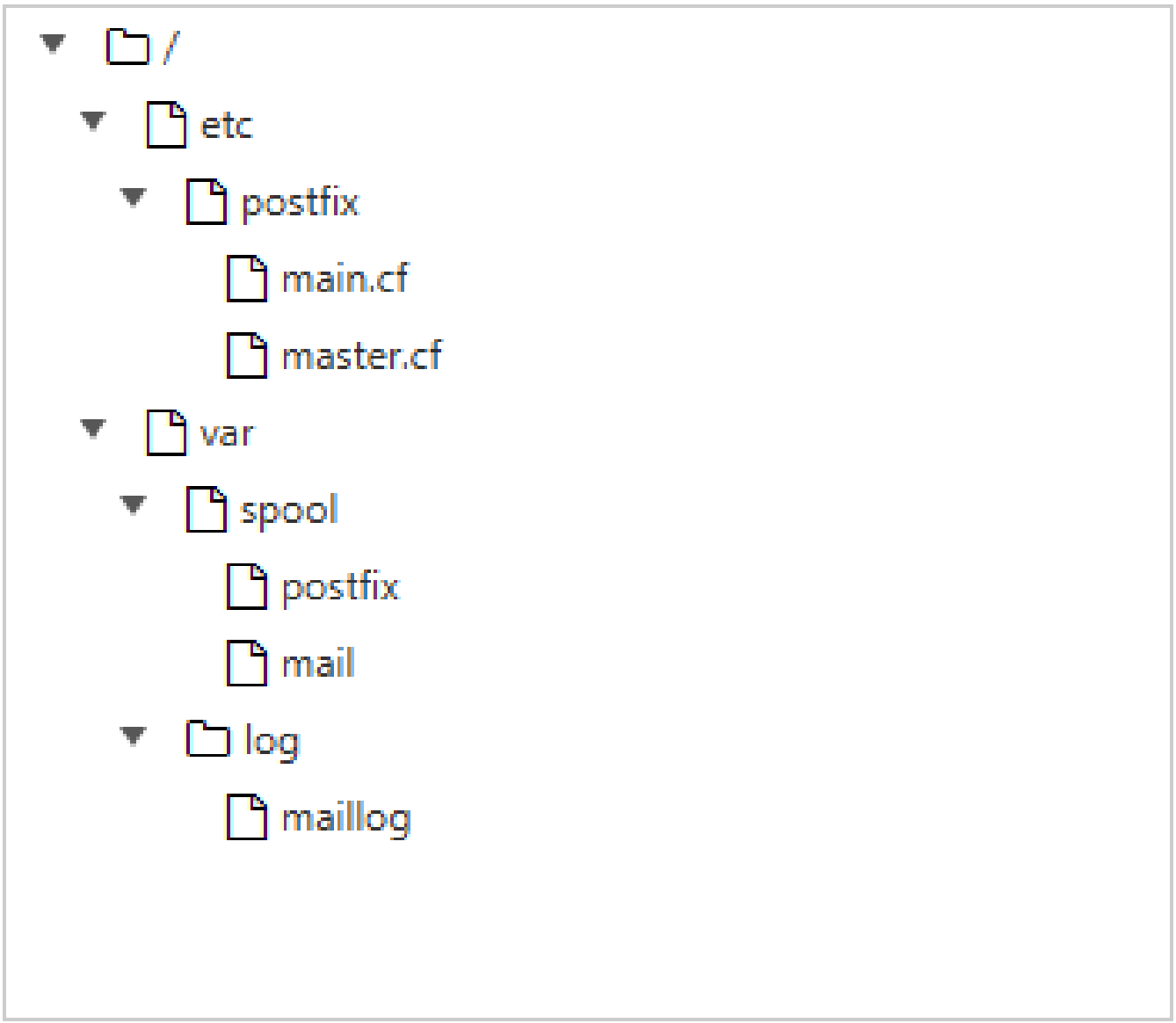


Figure 26. Arborescence et fichiers du serveur postfix

- Le fichier de configuration du serveur Postfix : /etc/postfix/main.cf ;
- Les files d'attente sont groupées dans le répertoire : /var/spool/postfix/ ;
- Les boîtes de messagerie mbox sont stockées dans /var/spool/mail/ ;
- Les logs sont dans le fichier : /var/log/maillog.

Le fichier /etc/postfix/main.cf contient les paramètres de configuration de Postfix. Les paramètres qui n'y sont pas explicitement renseignés sont initialisés avec leur valeur par défaut. Seule la dernière occurrence du paramètre compte lorsque ce paramètre est défini plusieurs fois.

En cas de besoin, un fichier commenté de main.cf existe sous /usr/share/postfix/main.cf.

Les directives sont modifiables avec un éditeur de texte, mais la commande postconf permet l'édition en limitant le risque d'erreur.

Dans le fichier /etc/postfix/main.cf :

- Chaque instruction doit être en début de ligne (pas d'espace avant).
- Les espaces autour du signe "=" sont ignorés, comme ceux situés à la fin de la ligne logique.
- Une ligne démarrant avec un espace continue la ligne logique précédente.

8.4. Mise en oeuvre

La commande **telnet** est particulièrement adaptée pour tester des protocoles en mode texte tel SMTP.

Sa syntaxe est la suivante :

```
[root]# telnet localhost 25
```

Pour communiquer avec le serveur, il faut respecter les étapes attendues par le service.

Déroulé d'une session telnet sur le port 25 :

1. HELO
2. MAIL FROM:
3. RCPT TO:
4. DATA

Pour terminer la rédaction du mail, il conviendra de saisir un . sur une ligne seule.

Déroulé d'une session SMTP

Lancer dans un terminal la commande telnet localhost 25. Voici le déroulé de la session :

```
[root]# telnet localhost 25
Trying ::1...
Connected to localhost.
Escape character is '^]'.
220 mail.mondomaine.lan ESMTP Postfix
HELO mail.mondomaine.lan
250 mail.mondomaine.lan
MAIL FROM: bob
250 2.1.0 Ok
RCPT TO: alice
250 2.1.5 Ok
DATA
354 End data with<CR><LF>.<CR><LF>
From: test
To: monalice
Subject: Test de message

Ceci est un test.
Merci de votre coopération

.
250 2.0.0 Ok: queued as 642A59F6E8
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```

Les champs From: et To: du contenu du mail (après DATA) ne sont pas vérifiés par le serveur SMTP. Ils peuvent différer des valeurs fournies au service SMTP, un peu comme l'adresse du destinataire d'un courrier postal peut différer de l'adresse affichée dans le courrier.

Le message se termine lorsque le serveur reçoit une ligne ne contenant qu'un point.

Suivi du mail

Le traitement du mail par les différents agent peut être suivi dans le fichier **/var/log/maillog**. Dans ce cas, l'utilisation de la commande *tail -f nomdufichier* est particulièrement bien adaptée. L'utilisation de la commande **ccze**, qui permet la coloration syntaxique des fichiers de log, peut également être utilisée via un **grep** : *tail -f /var/log/maillog | grep ccze*.

Voici un extrait du fichier de log, généré par la session telnet précédente :

```
[root]# tail -f /var/log/maillog  | grep ccze
postfix/smtpd[19747]: connect from localhost[::1]
postfix/smtpd[19747]: 642A59F6E8: client=localhost[::1]
postfix/cleanup[19828]: 642A59F6E8: message-id=<...>
postfix/qmgr[19742]: 642A59F6E8: from=<bob@mail.mondomaine.lan>, size=462, nrcpt=1
(queue active)
postfix/local[20100]: 642A59F6E8: to=<alice@mail.mondomaine.lan>, orig_to=<alice>,
   relay=local, ..., status=sent (delivered to mailbox)
postfix/qmgr[19742]: 642A59F6E8: removed
postfix/smtpd[19747]: disconnect from localhost[::1]
```

L'agent **smtpd** a pris en charge la connexion du client par le réseau en passant par **telnet** sur le port **25**. Si la connexion avait été faîte via un outil local, c'est l'agent **pickup** qui aurait alors pris en charge le message, comme nous le verrons à la section suivante.

L'agent **cleanup** a ensuite pris en charge le message. Le destinataire d'origine, bob, n'étant pas **pleinement qualifié** et non conforme à la norme **RFC 822**, **cleanup** l'a fourni au démon **trivial-rewrite** (événement qui est non journalisé) pour permettre la réécriture de l'adresse de messagerie d'origine.

L'agent **cleanup** a ensuite déposé le message dans la file d'attente **incoming**, en attendant que l'agent **qmgr** le déplace dans la file **active** (queue active).

Le message ayant une portée locale (**orig_to=<alice>**), l'agent **trivial-rewrite** est de nouveau appelé pour rendre conforme cette adresse (**to=alice@mail.mondomaine.lan**) par l'agent **qmgr**, qui le délivre à l'agent **local** pour stockage dans la boîte aux lettres d'**Alice**.

L'agent qmgr supprime alors le message de la file active.

La commande mailx

La commande **mailx** est une commande de traitement du courrier (un MUA) dont nous n'étudierons que la partie envoi du courrier.

```
mailx [-iInv] [-s sujet] [-a en-tete] [-c adresses cc] [-b adresses bcc] adresse[s]
```

Le tableau suivant récapitule les principales options :

Table 14. Options principales de la commande mailx

Options	Information
-v	Affiche les détails de la livraison sur le terminal
-s	Spécifie le sujet en ligne de commande (seul le premier argument après le flag -s est utilisé en tant que sujet ; pensez à mettre des guillemets autour des sujets contenant des espaces).

Options	Information
-c liste	Liste les destinataires en copie carbone. 'liste' doit être une liste de noms séparés par des virgules.
-b	Liste les destinataires en copie cachée (Blind Carbon Copy).

Quelques options supplémentaires :

Table 15. Options supplémentaires de la commande mailx

Options	Information
-i	Ignore les signaux d'interruption du terminal. Cela est particulièrement utile lors de l'utilisation de mail sur des lignes téléphoniques à bruit.
-l	Force mailx à se lancer en mode interactif même lorsque l'entrée n'est pas un terminal. En particulier, le caractère de commande spécial ~, utilisé lors de l'envoi d'un courrier, est seulement disponible interactivement.
-N	Désactive l'affichage initial des en-têtes du message lors de la lecture d'un courrier ou de l'édition d'un dossier de courriers.
-a	Spécifie des champs d'en-tête additionnels dans la ligne de commande comme "X-Loop: foo@bar", etc. Vous devez utiliser des guillemets si la chaîne contient des espaces. Cet argument peut être spécifié plus d'une fois, les en-têtes étant dans ce cas concaténés.
-e	N'envoie pas de courrier vide. Si le corps est vide, le message est sauté.
-f nom	Procède à la lecture du contenu de votre boîte aux lettres (ou le fichier spécifié nom) ; lorsque vous quittez, mail réécrit les messages non supprimés dans ce fichier.
-u utilisateur	Est équivalent à "mail -f /var/mail/utilisateur" sauf qu'il y a verrouillage.

Exemples :

```
[stagiaire]$ less corps
    Bonjour
    Au revoir
[stagiaire]$ mailx -s "coucou" "alice,bob" < corps
```

```
[stagiaire]$ mailx -s "coucou" alice@mail.formatux.lan
    ->Bonjour
    ->Au revoir
    ->. (ou ctrl+d)
```

Suivi du mail

Voici un extrait du fichier de log, généré par la session mailx précédente :

```
[root]# tail -f /var/log/maillog
postfix/pickup[19741]: 5707A9F8A: uid=1000 from=<stagiaire>
postfix/cleanup[22647]: 5707A9F8A: message-id=<...>
postfix/qmgr[19742]: 5707A9F8A: from=<stagiaire@mail.mondomaine.lan>, size=549,
nrcpt=2 (queue active)
postfix/local[22649]: 5707A9F8A: to=<alice@mail.mondomaine.lan>, orig_to=<alice>,
   relay=local, ..., status=sent (delivered to mailbox)
postfix/local[22650]: 5707A9F8A: to=<bob@mail.mondomaine.lan>, orig_to=<bob>,
   relay=local, ..., status=sent (delivered to mailbox)
postfix/qmgr[19742]: 5707A9F8A: removed
```

Les messages générés **localement** (mailx, php, scripts, etc.) sont placés par **sendmail** dans la file d'attente **maildrop**.

Le message ayant été émis par la commande local mailx, c'est l'agent **pickup** qui a pris en charge le message placé dans maildrop.

L'agent **cleanup** a ensuite pris en charge le message, notamment en le fournissant au démon **trivial-rewrite** (non journalisé par défaut) pour permettre la réécriture des adresses de messagerie locale (FROM: root et TO: alice et TO: bob) en adresses conformes à la norme RFC 822 (FROM: root@mail.formatux.lan, TO: alice@mail.formatux.lan et TO: bob@mail.formatux.lan).

Cleanup a ensuite déposé le message dans la file d'attente **incoming**. De la file d'attente **incoming**, le message est passé dans la file active (**queue active**) par l'agent **qmgr**.

Le message ayant une portée local, il est transmis à l'agent **local** pour stockage dans la boîte aux lettres d'Alice puis de nouveau transmis à l'agent **local** pour stockage dans la boîte aux lettres de Bob.

qmgr supprime alors le message de la file **active**.

Utilisation interactive de mailx

```
[alice]$ mailx
Heirloom Mail version 12.4 7/29/08. Type ? for help.
"/var/spool/mail/alice": 2 messages 2 new
>N 1 test@mail.formatux.lan ...    ... "Test de mail"
 N 2 stagiaire                     "coucou"
& _
```

La première ligne identifie la version de mail utilisée.

La deuxième désigne la boite aux lettres.

Le N (new) placé au début de la ligne indique qu'il s'agit d'un nouveau message, tandis que la lettre U (unread) indique qu'elle n'a pas encore été lue lors de la session précédente du programme

mailx.

Pour lire le mail, il faut saisir après l'esperluette (et commercial) le numéro du mail à lire.

Pour quitter, simplement saisir la lettre q.

Lorsque la touche q est saisie, mailx sauvegarde le contenu de la boîte aux lettres dans le fichier mbox du répertoire personnel de l'utilisateur, ainsi que les éventuelles modifications ou suppressions effectuées.

Le fichier /home/alice/mbox doit maintenant contenir les messages qui ont été lus.

Table 16. Gestion des mails avec mailx

Options	Information
num	Affiche le mail n° num
d num	Supprime le mail num
h	Affiche la liste des mails
q	Quitte mailx

La commande swaks

La commande **swaks** (SWiss Army Knife for Smtp) est un outil de test orienté transaction pour SMTP.

Syntaxe de la commande swaks

```
swaks --to user@formatux.lan --server smtp.formatux.lan
```

8.5. Configuration du serveur

La commande **postconf** permet la configuration de Postfix.

Syntaxe de la commande postconf

```
postconf [-d] [-e] [-n] ['directive']
```

Exemple :

```
postconf -e 'myhostname = mail.formatux.fr'
```

Table 17. Options principales de la commande postconf

Options	Information
-d	Affiche les valeurs par défaut des paramètres

Options	Information
-e	Modifie le fichier main.cf avec le paramètre précisé.
-n	Affiche seulement les valeurs qui ne sont pas celles par défaut.

Utilisé sans option ni argument, la commande postconf affiche la configuration courante de Postfix.

La commande postfix check vérifie la configuration du fichier **main.cf** :

```
[root]# postfix check
```

Lorsque la configuration est correcte, la commande ne retourne aucune information.

Les alias

Comment rediriger une adresse vers une autre ? Comment créer une liste de diffusion ? Comment créer une adresse en prenom.nom ? En utilisant les **alias** !

Les alias sont contenus dans le fichier /etc/aliases :

/etc/aliases

```
...
postmaster:    root
...
# Person who shold get root's mail
root:          bob
# Alias locaux
bob.leponge:   bob
# Liste de diffusion
admins:        bob,alice
```

Les modifications sont prises en compte avec la commande **newaliases** :

```
[root]# newaliases
```

Suivi des mails

Suivi d'un mail généré avec mailx à root :

```
postfix/local[25354]: 12C969F84F: to=<bob@mail.formatux.lan>, orig_to=<root>,
relay=local, ..., status=sent (delivered to mailbox)
```

Suivi d'un mail généré avec mailx à admins :

```
postfix/local[25639]: 8DD1A9F84F: to=<bob@mail.formatux.lan>, orig_to=<admins>,
relay=local, ..., status=sent (delivered to mailbox)
postfix/local[25639]: 8DD1A9F84F: to=<alice@mail.formatux.lan>, orig_to=<admins>,
relay=local, ..., status=sent (delivered to mailbox)
```

Suivi d'un mail généré avec mailx à bob.leponge :

```
postfix/local[25920]: 0CD8B9F84F: to=<bob@mail.formatux.lan>, orig_to=<bob.leponge>,
relay=local, ..., status=sent (delivered to mailbox)
```

Configurer un serveur relais

Mon serveur est protégé par une DMZ ! Mon fournisseur d'accès bloque le protocole SMTP ! Comment faire ?

Lorsque le serveur n'est pas directement connecté à Internet, il faut utiliser un serveur relais !

Les messages à destination d'utilisateurs non locaux sont relayés par le serveur MTA de la DMZ ou le MTA du fournisseur d'accès.

Configuration du relais

```
[root]# postconf -e 'relayhost = [svrmail.formatux.lan]'
[root]# service postfix reload
[root]# mailx bob.leponge@free.fr
```

Suivi du mail :

/var/log/maillog

```
postifx/smtp[26595]: 0D7809F84F:
to=<bob.leponge@free.fr>,relay=svrmail.formatux.lan[XXX.XXX.XXX.XXX]:25, ⋯,
status=sent (250 2.0.0 Ok: queued as 2997B686008)
```

Lorsqu'un serveur n'est pas **directement** connecté à l'Internet, les mails qu'il émet devront être envoyés à un serveur intermédiaire : un **MTA relais**.

Il faut alors renseigner la directive **relayhost**.

Pour ne pas utiliser de résolution DNS sur le champ MX du domaine, il convient de mettre le FQDN ou l'adresse IP du serveur relais entre crochets.

Dans l'exemple au dessus, le serveur 'svrmail.formatux.lan' étant le serveur de messagerie pointé par l'enregistrement MX du DNS pour le domaine 'formatux.lan', les deux configurations suivantes

sont identiques, mais la deuxième affranchit le serveur d'une requête DNS :

```
[root]# postconf -e 'relayhost = formatux.lan'
```

est identique à :

```
[root]# postconf -e 'relayhost = [svrmail.formatux.lan]'
```

Le message est cette fois-ci transmis par l'agent **mgr** au démon **smtp**, chargé de transférer le message via le protocole **SMTP**.

Prendre en compte un domaine

Je veux transformer mon serveur de messagerie, je voudrais centraliser les messages pour tout le domaine formatux.lan !

Il faut configurer les directives mydomain et mydestination !

Avant de modifier la configuration du serveur, envoyer un mail à bob@formatux.lan :

```
[root]# mailx bob@formatux.lan
```

Suivi du message :

```
postifx/smtp[27446]: B522C9F84F:
to=<bob@formatux.lan>,relay=svrmail.formatux.lan[172.16.160.7]:25, ⋯, status=sent
(250 2.0.0 Ok: queued as CB201686008)
```

Le serveur ne sait pas pour l'instant qu'il doit délivrer à l'agent local ce message.

Les directives mydomain et mydestination

La directive **mydomain** contient le nom de domain internet du système de messagerie. Par défaut, $mydomain vaut $myhostname oté de son premier composant.

Si $myhostname vaut serveur.formatux.lan alors $mydomain vaut formatux.lan.

La directive **mydestination** liste les domaines livrés par l'agent local.

Pour visualiser la valeur par défaut de la directive mydestination :

```
[root]# postconf 'mydestination'
mydestination = $myhostname, localhost.$mydomain, localhost
```

Le serveur transmettra donc à l'agent local tous les mails correspondant à @serveur.formatux.lan, @localhost.formatux.lan et @localhost.

Pour ajouter le domaine formatux.lan à la liste des domaines gérés localement :

```
[root]# postconf -e 'mydestination = $myhostname, localhost.$mydomain, localhost,
$mydomain'
[root]# postconf -e 'mydomain = formatux.lan'
[root]# service postfix reload
```

Le même test que précédement peut être rejoué :

```
mailx bob@formatux.lan
```

Suivi du mail :

/var/log/maillog

```
postfix/local[28603]: AE6D99F84F: to=<bob@formatux.lan>, relay=local, ⋯, status=sent
delivered to mailbox)
```

Le message est cette fois-ci pris en compte par le serveur est délivré à une boîte aux lettres locale via le démon **local**.

La directive mynetworks

Par défaut, le serveur postfix refuse de prendre en compte des messages provenant du réseau (excepté depuis sa loopback localhost), ce qui aurait pour effet de devenir un serveur OpenRelay à la merci des spammers.

Maintenant que les clients du réseau local disposent d'une boîte mails, ils vont devoir envoyer leur messages au serveur. Ils doivent donc avoir accès à postfix via le réseau.

Il faut configurer postfix pour qu'il accepte les connexions réseaux, tout en prenant soin de le limiter aux connexions du réseau local.

Dans un premier temps, il convient d'autoriser postfix à écouter sur toutes les interfaces réseaux :

```
[root]# postconf -e 'inet_interfaces = all'
```

Dans un second temps, il faut vérifier que le firewall autorise les connexions (au moins sur le port 25).

La directive **mynetworks** précise les réseaux autorisés à envoyer des messages sur le serveur.

```
mynetworks = 192.168.96.0/19
```

La directive **mynetworks_style** est ignorée si **mynetworks** est définie. Sinon elle précise le type d'hôtes autorisés à envoyer des messages au serveur (host, subnet ou class).

```
# autoriser tous les hôtes de mon sous-réseau
mynetworks_style = subnet
```

Attention à ne pas devenir un serveur "open-relay", et ainsi servir de serveur relais pour les spammeurs.

Le format de stockage

Par défaut, postfix stocke les mails au format mbox.

Pour passer du format mbox au format maildir :

```
[root]# postconf –e 'home_mailbox = Maildir/'
[root]# service postfix reload
```

Le changement pourra être vérifié :

```
[root]# mailx bob@formatux.lan
[root]# ls /home/bob/Maildir/new/
```

La table de routage

J'aimerais que les messages vers monvoisin.fr ne passent pas par le serveur relais. Il faudrait définir une route spécifique !

La table de routage va permettre de définir un serveur relais pour un domaine donné. Le fichier par défaut à renseigner est **/etc/postfix/transport**. Plusieurs enregistrements différents peuvent être présent.

L'exemple ci-dessous permet de définir que tous les messages à destination du domaine **monvoisin.fr** seront directement envoyés en smtp au serveur d'adresse IP spécifiée sans tenir compte d'un relais.

Les crochets permettent de s'affranchir de la requête DNS de type MX et d'utiliser directement l'adresse IP spécifiée.

/etc/postfix/transport

```
...
monvoisin.fr        smtp:[172.16.96.100]:25
...
```

Postfix doit également connaitre dans son fichier configuration l'emplacement du fichier */etc/postfix/transport*, dans notre cas :

```
[root]# postconf -e 'transport_maps = hash:/etc/postfix/transport'
```

Après avoir modifié la table des transports, il est nécessaire de lancer la commande **postmap**. Cette commande permet de transformer le fichier en base de données *.db* de type hash (format clef:valeur) interprétable par postfix.

```
[root]# postmap /etc/postfix/transport
[root]# service postfix reload
```

N'oubliez pas de relancer le service postfix !

Certains serveurs SMTP ne tolèrent pas les envois de courriels de masse. Dans ce cas de figure, il va falloir restreindre les envois vers ces domaines et ne procéder qu'à une livraison à la fois vers ces serveurs.

Pour se faire, configurer une entrée dans le fichier `/etc/postfix/transport` pour le domaine concerné avec la directive `slow:`. L'ajout de cette directive aura pour effet de ne procéder qu'à une livraison de mail à la fois vers ce domaine.

```
free.fr      slow:
```

Puis créer cette "file" plus longue dans le fichier `/etc/postfix/master.cf` :

```
slow unix - - n - 5 smtp -o syslog_name=postfix-slow  -o
smtp_destination_concurrency_limit=3   -o slow_destination_rate_delay=1
```

8.6. Protocoles POP/IMAP

Le serveur Dovecot est un serveur POP3/IMAP orienté vers la sécurité.

Installer le serveur dovecot :

```
[root]# yum install dovecot
[root]# chkconfig dovecot on
```

La configuration de dovecot est répartie entre de nombreux fichiers de configuration.

- Activer le protocole IMAP.
- Le serveur dovecot écoutera sur toutes les interfaces réseaux.

/etc/dovecot/dovecot.conf

```
protocols = imap
listen= *
```

- Spécifier à dovecot que le serveur Postfix stocke les mails au format Maildir (maildir:) dans le répertoire Maildir du répertoire de connexion de l'utilisateur (~/).

/etc/dovecot/conf.d/10-mail.conf

```
mail_location = maildir:~/Maildir
```

- L'authentification plaintext permet de gérer l'authentification des clients par login et mot de passe qui transitent en clair sur le réseau. Cette option n'est pas sécurisée si elle n'est pas couplée avec un moyen de chiffrement du flux imap (IMAPs) ce qui explique sa désactivation par défaut.

/etc/dovecot/conf.d/10-auth.conf

```
disable_plaintext_auth = no # signifie authentification plaintext enable
```

Redémarrer le service

```
[root]# service dovecot restart
```

8.7. Architecture de postfix

Postfix est un serveur modulaire basé sur des boîtes aux lettres et régi par le démon principal **master**.

Chaque démon assume une fonction, chaque fonction correspondant à une tâche distincte. Les démons sont gérés par le démon master, qui est le premier a être lancé.

Les messages sont stockés dans des files d'attente, où ils sont récupérés par les démons.

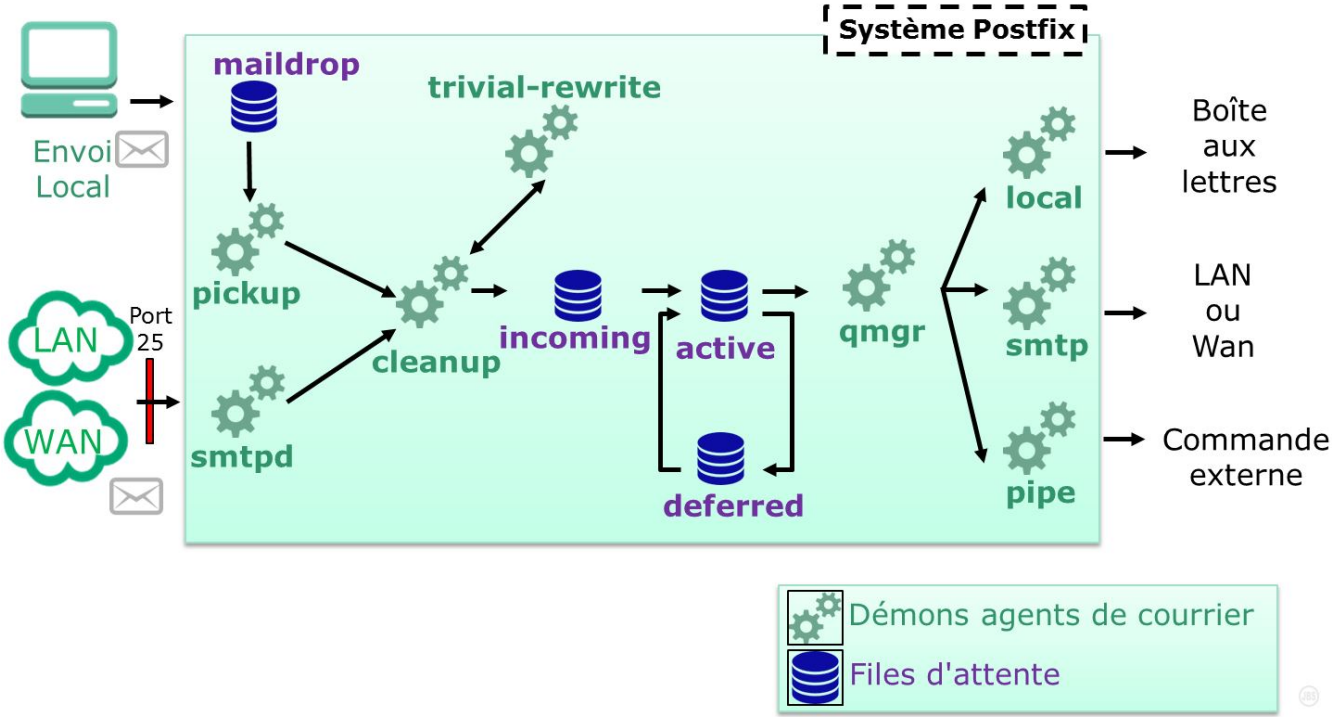


Figure 27. Synoptique global de postfix

Postfix accepte des messages provenant de plusieurs sources :

- Source locale : envoyé par un utilisateur du serveur via un logiciel local (mailx, php, etc.) ;
- En provenance du réseau connecté au serveur ;
- Produit par Postfix lui-même ;
- Un message resoumis pour être transféré à une autre adresse.

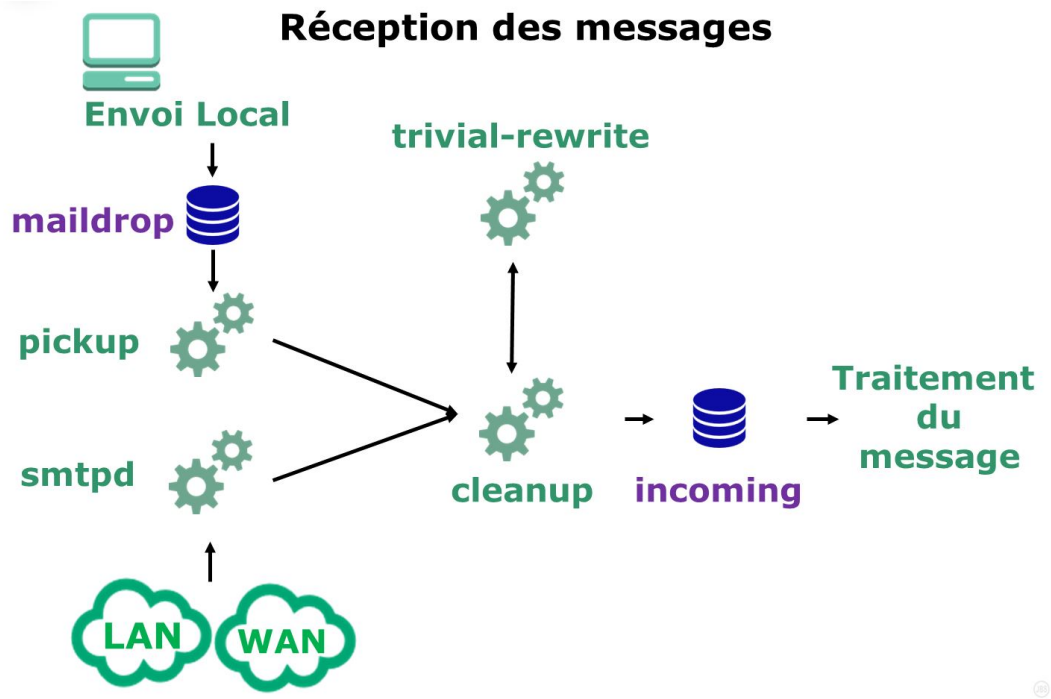


Figure 28. Synoptique de postfix partie réception des messages

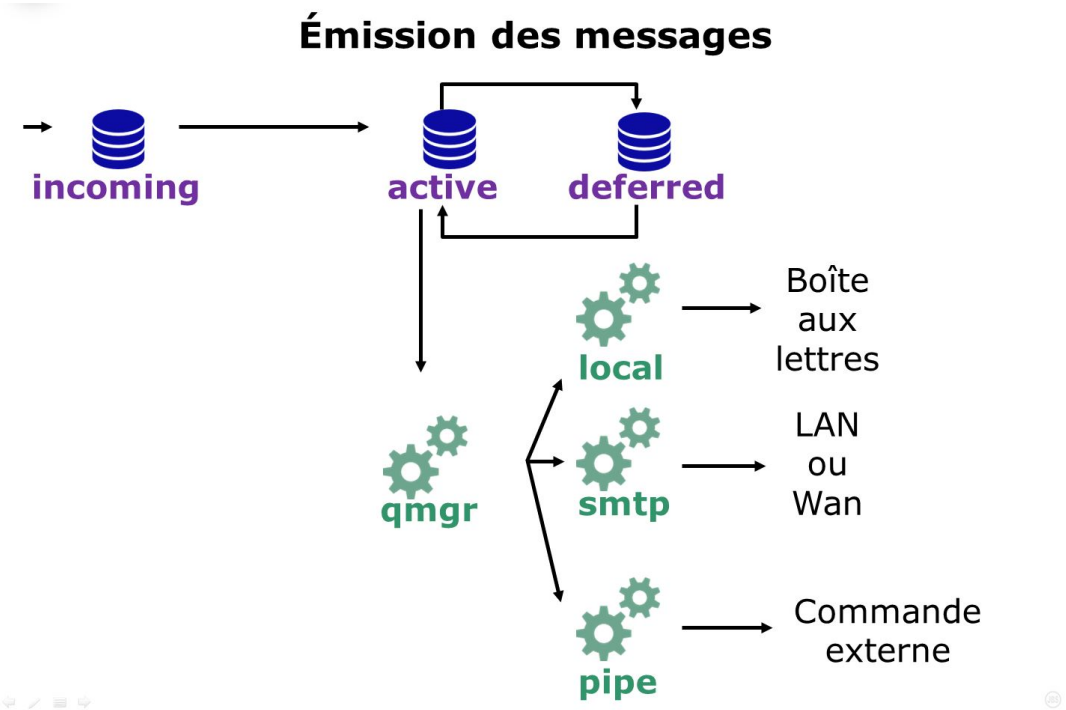


Figure 29. Synoptique de postfix partie émission des messages

Postfix produit lui-même les messages de service pour indiquer la non-réception d'un message ou son report. Ces messages suivent le même cheminement que les messages locaux.

Les messages locaux sont déposés dans la file d'attente maildrop.

Le démon pickup prend les messages dans la file d'attente et les passe à cleanup.

Les messages provenant du réseau sont pris en charge par le démon smtpd. Ce dernier vérifie qu'ils vont pouvoir être traités par le serveur et ensuite il les transfère à cleanup.

Un message se doit de respecter certaines normes de formatage. Les adresses de provenance ou de destination doivent être pleinement qualifiées, il ne doit pas manquer d'en-têtes, etc. Un message ne respectant pas ces règles sera reformaté (remis en forme) par trivial-rewrite. Une fois corrigé, il sera de nouveau pris en charge par cleanup qui le placera dans la file incoming et préviendra le gestionnaire qmgr.

Le gestionnaire de file d'attente qmgr effectue l'essentiel du traitement du courrier. Il gère les files d'attente incoming, active et deferred.

Après traitement par cleanup, les messages sont placés dans la file incoming. Si les ressources systèmes sont disponibles, qmgr déplace alors le message dans la file active et appelle l'un des agents de distribution pour le délivrer.

Les messages qui ne peuvent pas être distribués sont mis dans la file deferred où ils attendent que qmgr tente de nouveau de les distribuer.

Si le message n'est pas à destination d'un utilisateur géré par le serveur, qmgr le transfère au démon smtp qui l'expédie au MTA concerné.

Si le destinataire fait bien partie du domaine géré par le serveur Postfix, le démon qmgr achemine le message vers local.

Le démon local dépose les messages dans l'espace de stockage local des messages. Il contrôle également les alias et vérifie si les messages doivent être délivrés ailleurs.

Le message peut être délivré à un autre processus, comme un gestionnaire de liste de diffusion, ou tout autre processus.

Démons agents de courrier

- **pickup** : collecteur de messages locaux acheminés par *maildrop*. Il fournit à l'agent cleanup les messages déposés dans la file maildrop.
- **smtpd** : collecteur de messages reçus du réseau. L'agent smtpd accepte les connexions réseaux et effectue les transactions SMTP pour fournir les messages à l'agent cleanup.
- **trivial-rewrite** : agent de résolution et de réécriture des adresses (en lien avec le domaine). L'agent trivial-rewrite offre 3 types de services aux requêtes des clients :
  - Réécriture du contexte d'adresse vers une forme standard : ajoute le nom de domaine spécifié par $myorigin ou par $mydomain aux adresses incomplètes des messages postés localement,
  - Résolution de l'adresse en un quadruple (transport, saut suivant, récipient, drapeaux) :
    - Le transport correspond à l'agent de délivrance à utiliser,
    - Le MTA suivant à qui délivrer le mail,
    - L'adresse du destinataire à fournir au prochain MTA,
    - Les drapeaux : la classe d'adresse, si l'adresse nécessite un relais, si l'adresse a un problème ou si la requête a échoué.
  - Résolution de l'adresse de l'envoyeur (pour des besoins de vérifications).
- **cleanup** : agent de formatage des messages selon la norme RFC822. Il traite les messages entrant, les insère dans la file d'attente incoming puis informe qmgr de leur arrivée.
  - L'agent cleanup opère toujours ces transformations :
    - Ajout des en-têtes manquantes : From:, To:, Message-Id: et Date:.
    - Transforme au besoin les adresses de l'enveloppe et des en-têtes au standard utilisateur@fqdn qui est attendu par les autres agents postfix. Cette tâche est déléguée à l'agent trivial-rewrite.
    - Supprime les adresses dupliquées de l'enveloppe,
    - Supprime les en-têtes : Bcc:, Content-Length:, Resent-Bcc, Return-Path:.
  - Optionnellement, les adresses peuvent être transformées en fonction de la table canonical ou virtual et du paramètre masquerade_domains,
- **qmgr** : agent de gestion des files d'attente *active* et *deferred*. L'agent qmgr attend l'arrivée de message entrant et s'assure de leur livraison via un des agents de livraison. La stratégie de

routage des messages est délégué au démon trivial-rewrite.

  - qmgr maintient les files d'attente incoming, active, deferred, corrupt et hold.
  - qmgr surveille les rapports de livraison par message dans les répertoires suivant et demande aux agent concernés d'envoyer les rapports :
    - bounce : rapport des messages refusés. Cette file est maintenue par l'agent bounce.
    - defer : rapport des messages retardés. Cette file est maintenue par l'agent bounce.
    - trace : rapport des messages suivis. Cette file est maintenue par l'agent trace.

- **local** : agent de livraison des messages locaux, MDA. L'agent local met à jour les files d'attente des messages et marque les messages si finis ou informe qmgr si les messages doivent être retraités plus tard. Les messages de livraison sont transmis à l'agent approprié (bounce, defer ou trace).
- **smtp** : agent de livraison des messages vers le réseau. Il implémente les protocoles SMTP et LMTP. Il procède à la livraison des messages à la demande de qmgr. L'agent met à jour les files d'attente des messages et marque les messages si finis ou informe qmgr si les messages doivent être retraités plus tard. Les messages de livraison sont transmis à l'agent approprié (bounce, defer ou trace).
  - L'agent smtp interroge le service DNS pour obtenir une liste d'adresses de serveurs de messagerie MX du domaine du destinataire de message, les trie par ordre de préférence et tente une connexion vers chacun jusqu'à ce qu'il en trouve un qui réponde.
- **pipe** : agent de livraison des messages vers une commande externe.
- **bounce** : agent de suivi des messages (informations sur la délivrance des messages). Ce démon procède à deux types de requêtes :
  - Ajoute un enregistrement de (non-)délivrance à un fichier de suivi de message (un fichier par message).
  - Génère un message de notification de délivrance, avec une copie du fichier de suivi du message correspondant. Lorsque le message est correctement généré, le fichier de suivi est supprimé.

Files d'attente

- **maildrop** : messages locaux postés par *sendmail*.
- **incoming** : messages après formatage en attente de traitement. Tous les messages entrant dans le système Postfix sont écrits par l'agent cleanup dans la file incoming. Les nouveaux fichiers sont créés avec comme propriétaire l'utilisateur "postfix" avec des droits à 0600. Une fois que le fichier est prêt a être traité, l'agent cleanup change les droits à 0700 et notifie qmgr d'une nouvelle arrivée. Les messages qui n'ont pas les droits à 0700 sont tout simplement ignorés car considérés comme en cours d'écriture par cleanup.
- **active** : messages prêts a être acheminés. La file d'attente active n'est pas uniquement un ensemble de fichiers sur le disque. La file d'attente "réelle" active comprend également un ensemble de structures de données dans la mémoire de l'agent gmgr, ce qui explique que la

quantité de message traités dans la file active soit limitée, pour éviter un dépassement de mémoire libre.

- **deferred** : messages n'ayant pas pu être livrés et pour lesquels un envoi ultérieur pourrait réussir.

8.8. Boites aux lettres virtuelles

Est-il vraiment utile de créer un compte système Linux pour chaque adresse de messagerie ?

Il est possible d'héberger la messagerie de différents domaines sans associer les boîtes aux lettres à des comptes système.

Les boîtes seront stockées (par exemple) sous /var/mail/vmail et gérées par l'utilisateur vmail (uid=5000, gid=5000).

Créer l'utilisateur virtual mailbox:

```
[root]# groupadd -g 5000 vmail
[root]# useradd vmail -u 5000 -g 5000 -s /sbin/nologin -d /var/mail/vmail
```

/etc/postfix/main.cf

```
virtual_mailbox_domains = mondomaine.com, autre.com
virtual_mailbox_base = /var/mail/vmail
virtual_mailbox_maps = hash:/etc/postfix/vmailbox
virtual_minimum_id = 100
virtual_uid_maps = static:5000
virtual_gid_maps = static:5000
virtual_alias_maps = hash:/etc/postfix/virtual
```

Ne jamais lister ici un domaine renseigné dans la directive mydestination ou virtual_alias_domain.

/etc/postfix/vmailbox

```
bob@mondomaine.com      mondomaine.com/bob/
alice@mondomaine.com        mondomaine.com/alice/
bob@autre.com           autre.com/bob/
```

Le / à la fin des chemins vers les boîtes aux lettres précise qu'ici le format de stockage est Maildir.

/etc/postfix/vmailbox

```
postmaster@mondomaine.com   postmaster
```

Postfix ne peut pas traiter directement les fichiers vmailbox et virtual dans leur format humain. Il a besoin de générer une base de données au format clé-valeur, plus couramment appelée hash-table.

Générer les tables clés/valeurs (hachées) :

```
[root]# postmap /etc/postfix/vmailbox
[root]# postmap /etc/postfix/virtual
```

Ce rôle est rempli par la commande postmap, qui dans notre exemple, générera deux fichiers : vmailbox.db et virtual.db.

Créer les répertoires de stockage :

```
[root]# su – vmail –s /bin/bash
[vmail]$ mkdir /var/mail/vmail/mondomaine.com/
[vmail]$ mkdir /var/mail/vmail/autre.com/
```

Authentification des utilisateurs :

/etc/dovecot/users

```
bob@mondomaine.com:{PLAIN}password:5000:5000::/var/mail/vmail/mondomaine.com/bob/
alice@mondomaine.com:{PLAIN}password:5000:5000::/var/mail/vmail/mondomaine.com/alice/
```

Authentification par fichier plat :

/etc/dovecot/conf.d/10-auth.conf

```
disable_plaintext_auth = no
!include auth-passwdfile.conf.ext
```

Nouvel emplacement de stockage :

/etc/dovecot/conf.d/10-mail.conf

```
mail_location = maildir:/var/mail/vmail/%d/%n
```

Les macros dovecot :

Dovecot est capable de remplacer dynamiquement des valeurs renseignées dans ses fichiers de configuration.

Dans notre exemple, le %d sera remplacé par le domaine de messagerie et le %n par le nom de la boîte aux lettres. Par exemple, un mail destiné à bob@mondomaine.lan sera stocké dans le sous-dossier mondomaine.lan/bob/.

Les comptes de messagerie ne nécessitent plus de compte système, ce qui facilite l'administration et améliore la sécurité.

Les fichiers plats utilisés dans nos exemples peuvent facilement être remplacés par une table MySQL ou un annuaire LDAP.

Pourquoi ne pas utiliser uniquement des utilisateurs virtuels dans ce cas ?

8.9. Suivi des messages à des fins légales

Il peut être demandé de conserver le sujet, le rédacteur et le destinataire d'un message.

Pour cela, le processus cleanup doit vérifier les entêtes des messages et générer un log lorsqu'il rencontre la valeur attendue. Ces valeurs sont stockées dans le fichier /etc/postfix/header_checks sous forme de regex :

/etc/postfix/header_checks

```
/^subject:/     WARN
/^Subject:/     WARN
/^to:/          WARN
/^To:/          WARN
/^from:/        WARN
/^From:/        WARN
```

Postfix utilise la directive **header_checks** :

```
postconf -e 'header_checks = regexp:/etc/postfix/header_checks'
service postfix restart
```

Un message généré avec la commande swaks :

```
swaks --to alice@formatux.lan --from bob@formatux.lan --header "Subject: test test
test" --server 127.0.0.1
```

Générera les logs suivants :

```
tail -f /var/log/maillog | grep warning
postfix/cleanup[13423]: 125D162F88: warning: header To: alice@formatux.lan [...]
postfix/cleanup[13423]: 125D162F88: warning: header From: bob@formatux.lan [...]
postfix/cleanup[13423]: 125D162F88: warning: header Subject: test test test [...]
```

Chapitre 9. Serveur d'annuaire OpenLDAP

9.1. Généralités

Le protocole **LDAP (LightWeight Directory Access Protocol)** est une suite **standardisée** de protocoles permettant l'accès à un annuaire centralisé. Cet annuaire centralisé stocke des informations diverses comme :

- des noms ;
- des adresses ;
- des numéros de téléphone ;
- des utilisateurs ;
- des groupes ;
- etc.

La version actuelle de ce protocole est la version 3.

Le protocole LDAP s'inspire de la spécification **X.500** mais de manière moins complexe. La norme X.500 désigne l'ensemble des normes informatiques sur les services d'annuaires définies par l'IUT (International Union Telecommunication). Il s'agit donc d'un annuaire version électronique dont l'organisation est hiérarchique, à la manière du DNS, qui correspond généralement à l'organisation de l'entreprise. Le terme Lightweight ne doit pas être compris comme "allégé", ce qui signifierait qu'il serait amputé de certaines fonctionnalités, mais bien dans ce cas de "simplifié", car la spécification X.500 est très lourde.

Le système LDAP est né en 1993 à l'université du Michigan. Le dérivé libre **OpenLDAP** est lui apparu en 1998.

Une base de données LDAP, de part sa nature, est optimisée pour la lecture d'informations :

- authentification ;
- recherche dans l'annuaire.

Les ports utilisés par le protocole LDAP sont les ports **389** (en clair comme en chiffré par **startTLS**) et **636** pour une connexion en TLS (solution dépréciée).

L'implémentation la plus célèbre du protocole LDAP sous Windows est l'Active Directory.

Sous Linux, les choix sont nombreux :

- OpenLDAP ;
- RedHat Directory Studio ;
- Samba4 qui intègre un serveur LDAP ;

- LinID de Linagora ;
- 389 DS ;
- IBM Tivoli ;
- ...

À noter que la suite OpenLDAP au sein de RedHat 6 (et versions supérieures) n'utilise plus OpenSSL. Elle utilise à la place l'implémentation de Mozilla de **NSS** (Network Security Services).

La DIT

Un annuaire LDAP est **un arbre de données**. Cette **structure hiérarchique** est appelée **DIT** (**Directory Information Tree**).

Une entrée de l'arbre est un ensemble d'**attributs**.

Chaque entrée dispose d'un identifiant unique : son **DN** (**Distinguished Name**).

L'OLC

OpenLDAP est le serveur d'annuaire de référence pour les distributions Linux.

Dans les versions récentes d'OpenLDAP (>2.4), sa configuration n'est plus stockée dans un fichier de configuration.

La configuration réside directement dans la base de données elle-même, au sein d'une **DIT** spécifique : c'est la fonctionnalité **OLC** (**On-Line Configuration**), aussi connue sous le nom **cn=config**.

L'approche consistant à stocker 'en ligne' la configuration LDAP peut paraître complexe, mais est justifiée par la criticité du service LDAP. Stocker la configuration dans des fichiers plats imposait un redémarrage du service à chaque modification, ce qui représentait beaucoup de temps d'arrêt pour de grosses bases de données.

Il n'y a plus de fichiers .conf à modifier dans les versions récentes d'OpenLDAP.

Le schéma

Le contenu des entrées d'un annuaire est régi par des schémas. Les schémas définissent les types d'attributs d'une entrée regroupés par classe d'objets.

- schéma : ensemble des classes et des attributs disponibles.
- objectClass : une classe objet rassemble un ensemble d'attributs obligatoires ou facultatifs (par exemple la classe inetOrgPerson).
- attribut : exemple

- mail: john.doe@formatux.lan ;
- preferredLanguage: french.

Couche d'Authentification et de Sécurité Simple SASL

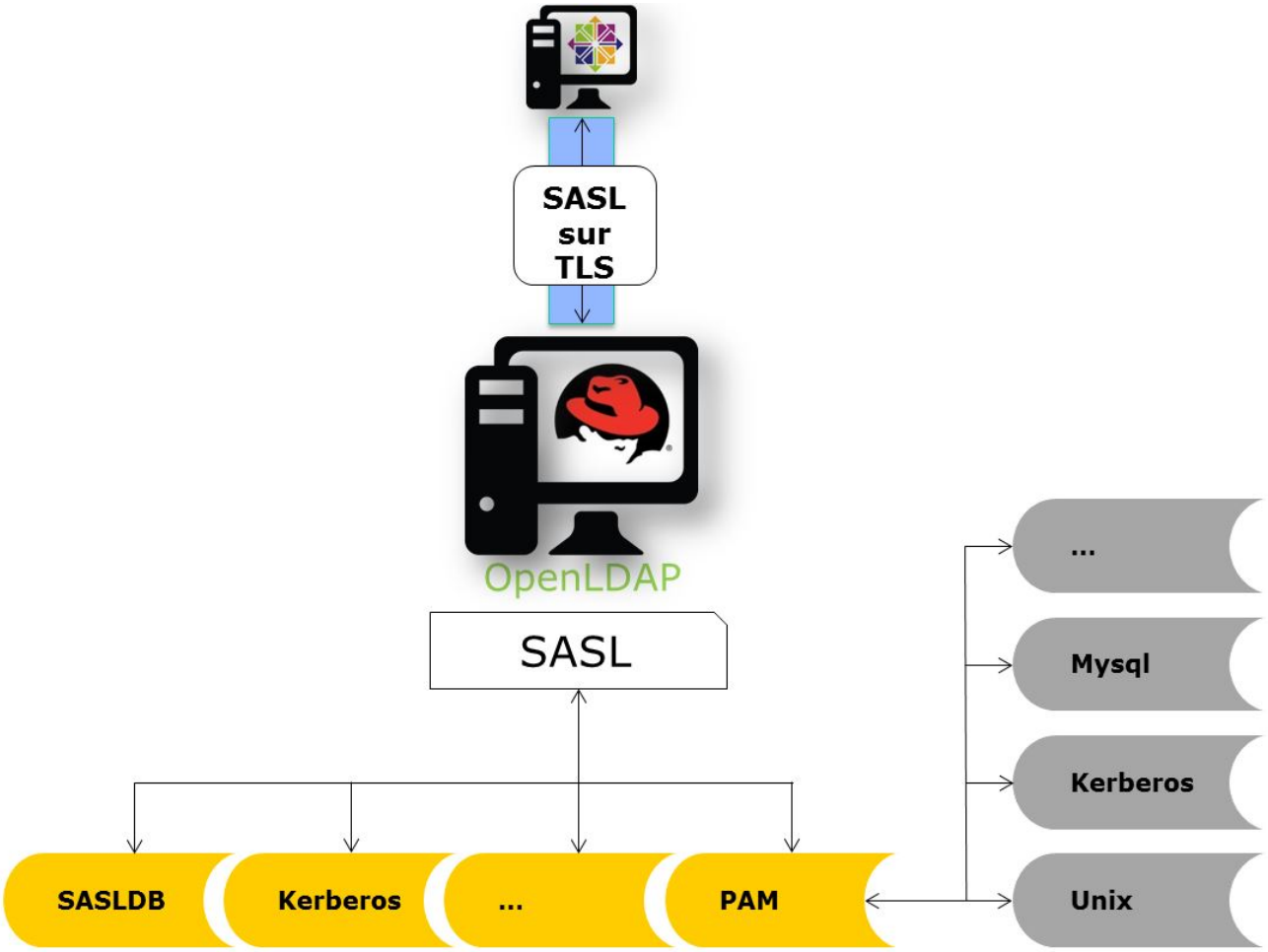
SASL (Couche d'Authentification et de Sécurité Simple) est une méthode pour ajouter le support d'**authentification** aux **protocoles basés sur la connexion** (LDAP, SMTP, IMAP, XMPP, IRC, etc.). Pour utiliser SASL, un protocole inclut une **commande d'identification et d'authentification** d'un utilisateur sur un serveur et la **négociation éventuelle de la protection** des interactions consécutives du protocole.

Si son utilisation est négociée, **une couche de sécurité est insérée entre le protocole et la connexion**.

Séparer ainsi la couche d'authentification de la couche applicative permet en théorie à n'importe quel mécanisme d'authentification pris en charge par SASL d'être employé à partir de n'importe quel protocole d'application capable d'utiliser SASL.

Les mécanismes principaux SASL sont :

- **EXTERNAL** : l'authentification est dérivée du contexte (authentification système);
- **ANONYMOUS** : accès anonyme sans authentification ;
- **PLAIN** : mot de passe en clair ;
- **OTP** : mot de passe unique (One Time Password) ;
- **CRAM-MD5** et **DIGEST-MD5** : basés sur MD5 ;
- **NTLM** : authentification pour réseau local NT ;
- **GSSAPI** : authentification Kerberos via GSSAPI.



Le format LDIF

Le format **LDIF** (LDAP Data Interchange Format) est un format de fichier texte utilisé lors des échanges d'informations en client/serveur ou entre serveurs.

Exemple de fichiers LDIF :

```
dn: cn=John Doe,dc=example,dc=com
  objectClass: top
  objectClass: person
  objectClass: inetOrgPerson
  objectClass: organizationalPerson
  cn: John Doe
  givenName: John
  sn: Doe
  mail: john.doe@example.com
```

Dans cette entrée, nous retrouvons, outre le Distinguished Name (DN) de l'objet :

- 4 objectClass : inetOrgPerson, organizationPerson, person et top ;
- 4 attributs : cn, givenName, sn et mail.

Les objectClass permettent d'inclure des attributs obligatoires ou optionnels dans une entrée. Toujours dans notre exemple, c'est l'ajout de l'objectClass inetOrgPerson qui va permettre d'ajouter un attribut mail.

L'ensemble des objectClass et des attributs sont définis dans des schémas, qu'il conviendra d'ajouter en fonction du rôle du serveur. Par exemple, pour permettre l'authentification Samba depuis le serveur LDAP, il faudra ajouter le schéma samba.schema à la configuration du serveur.

Le format LDIF a des caractéristiques très importantes :

- les séries d'entrées sont séparées par des lignes vides ;
- la dernière ligne doit être vide, sinon la dernière entrée pourrait ne pas être prise en compte.

Les outils clients LDAP

Des outils en ligne de commande permettent l'administration du serveur en utilisant en entrée des fichiers LDIF ou la console.

- **ldapadd** : ajouter des entrées ;
- **ldapdelete** : supprimer des entrées ;
- **ldapmodify** : modifier des entrées ;
- **ldappasswd** : modifier un mot de passe ;
- **ldapsearch** : rechercher dans l'annuaire.

9.2. Installation du serveur

Prérequis à l'installation :

- disposer des droits root ou sudo ;
- disposer d'un dépôt yum configuré ;
- avoir ouvert les ports 389 et 636 sur le pare-feu local et sur les éléments actifs

Installation :

```
[root]# yum install openldap-servers openldap-clients
[root]# cp /usr/share/openldap-servers/DB_CONFIG.example /var/lib/ldap/DB_CONFIG
[root]# chown ldap:ldap /var/lib/ldap/DB_CONFIG
[root]# chkconfig slapd on
[root]# service slapd start
```

LDAP a besoin d'un fichier de configuration (/var/lib/ldap/DB_CONFIG) pour sa base de données. Le fichier fourni en exemple lors de l'installation convient parfaitement.

Le fichier /etc/openldap/ldap.conf contient la configuration pour les clients openldap comme ldapsearch, ldapadd, etc. Toutes les informations inscrites dans ce fichier allégeront d'autant les lignes de commande interactives, puisqu'il ne sera plus nécessaire de préciser les options positionnées ici.

Le fichier /etc/openldap/ldap.conf

```
#
# LDAP Defaults
#

#BASE dc=example,dc=com
#URI ldap://ldap.example.com ldaps://ldap.example.com:666

#SIZELIMIT 12
#TIMELIMIT 15
#DEREF     never

TLS_CACERTDIR /etc/openldap/certs
```

Le répertoire /etc/openldap/slapd.d/ contient les bases de données et le schéma :

Arborescence du service OpenLDAP

```
# /etc/openldap/slapd.d/
|-- cn=config
|   |-- cn=schema # les schémas disponibles
|   |   |-- cn={10}ppolicy.ldif
|   |   |-- cn={1}core.ldif
|   |   |-- cn={2}cosine.ldif
|   |   |-- cn={5}inetorgperson.ldif
|   |   |-- cn={8}nis.ldif
|   |   |-- cn={9}openldap.ldif
|   |-- cn=schema.ldif # le schéma du serveur
|   |-- olcDatabase={0}config.ldif
|   |-- olcDatabase={-1}frontend.ldif
|   |-- olcDatabase={1}monitor.ldif
|   |-- olcDatabase={2}bdb.ldif # la DIT principale au format BDB
|-- cn=config.ldif # configuration globale du serveur
```

Les bases de données du serveur LDAP ne doivent jamais être modifiées manuellement !!!

9.3. Configuration du serveur

Avant de pouvoir utiliser les outils en ligne de commande, il convient de configurer les options par

défaut :

```
BASE dc=formatux,dc=lan
URI ldap://localhost
```

En version TLS sécurisée, l'URI doit impérativement correspondre au FQDN renseigné dans le certificat !

Pour la suite du cours, nous retiendrons que :

- le dn de base est : dc=formatux,dc=lan ;
- l'administrateur LDAP est cn=admin,dc=formatux,dc=lan ;
- les utilisateurs sont stockés dans l'unité d'organisation : ou=users,dc=formatux,dc=lan.

Il est intéressant à ce stade de visualiser la configuration par défaut avec la commande slapcat :

```
[root]# slapcat -b cn=config | less
...
dn: olcDatabase={2}bdb,cn=config # base de données de l'annuaire
...
olcSuffix: dc=my-domain,dc=com # suffix par défaut
olcRootDN: cn=Manager,dc=my-domain,dc=com # administrateur par défaut
...
```

À noter que l'administrateur n'a pas de mot de passe (olcRootPW)

Le suffixe

Le suffixe représente la racine de l'organisation. C'est l'identité même de l'entreprise. Elle correspond habituellement au suffixe DNS.

Nous allons le changer avec la commande ldapmodify :

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:///
dn: olcDatabase={2}bdb,cn=config
changetype: modify
replace: olcSuffix
olcSuffix: dc=formatux,dc=lan
```

Le suffixe étant défini à l'installation, il faut le modifier !

Le RootDN et son mot de passe

L'entrée **RootDN** contient le DN de l'utilisateur autorisé à faire des modifications de l'annuaire.

Son mot de passe est défini par **RootPW**.

Nous allons le configurer avec la commande ldapmodify :

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:///
dn: olcDatabase={2}bdb,cn=config
changetype: modify
replace: olcRootDN
olcRootDN: cn=admin,dc=formatux,dc=lan
```

Le suffixe étant défini à l'installation, il faut le modifier !

Pour définir un mot de passe utilisable par openldap, il faut utiliser la commande slappasswd.

Ajouter le RootPW :

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:///
dn: olcDatabase={2}bdb,cn=config
changetype: modify
add: olcRootPW
olcRootPW: {SSHA}Eke0fnWgD90xZWPT/UivZEBjzBgC/Z+
```

Cette fois-ci, le RootPW n'ayant pas été défini à l'installation, il faudra l'ajouter!

Les trois commandes auraient pu être regroupées en une seule. Dans ce cas, il faut séparer chaque modification de l'objet par une ligne contenant un "-".

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:///
dn: olcDatabase={2}bdb,cn=config
changetype: modify
replace: olcSuffix
olcSuffix: dc=formatux.lan
-
replace: olcRootDN
olcRootDN: cn=admin,dc=formatux,dc=lan
-
add: olcRootPW
olcRootPW: {SSHA}Eke0fnWgD90xZWPT/UivZEBjzBgC/Z+
```

Connexion avec le RootDN

Un RootDN et son mot de passe ayant maintenant été définis dans la DIT dc=formatux,dc=lan, il est possible de les utiliser pour se connecter :

```
[root]# ldapmodify -x -D cn=admin,dc=formatux,dc=lan -W
```

Il n'est pas nécessaire de préciser ici le serveur à contacter (options -H ou -h), la commande ldapmodify utilisera les informations du fichier /etc/openldap/ldap.conf qui a été renseigné précédemment.

La commande slapcat

Exporter le contenu de l'annuaire au format LDIF avec la commande **slapcat**.

Syntaxe de la commande slapcat

```
slapcat -b suffix
```

Exemple :

```
[root]# slapcat -b cn=config | less
...
dn: olcDatabase={2}bdb,cn=config
...
olcSuffix: dc=my-domain,dc=com
olcRootDN: cn=Manager,dc=my-domain,dc=com
...
```

Option	Description
-b	Détermine quelle base de données est exportée.

La commande ldapmodify

La commande **ldapmodify** permet de modifier le contenu de l'annuaire.

Authentification (binding) par SASL

Syntaxe de la commande ldapmodify avec authentification SASL

```
ldapmodify [-y SASLMecanisme] [-H host] [-v] [-f fichier.ldif]
```

Exemple :

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:/// -v -f modldap.ldif
```

Option	Description
-Y	Mécanisme SASL à utiliser pour l'authentification.
-v	Mode verbeux pour diagnostique.
-H	Spécifier un serveur. Le protocole **ldapi** permet une communication sécurisée via une socket UNIX (nécessaire pour utiliser SASL).

Authentification (binding) simple

Syntaxe de la commande ldapmodify avec authentification simple

```
ldapmodify [-x] [-D RootDN] [-W|-w pwd] [-H host] [-f fichier.ldif]
```

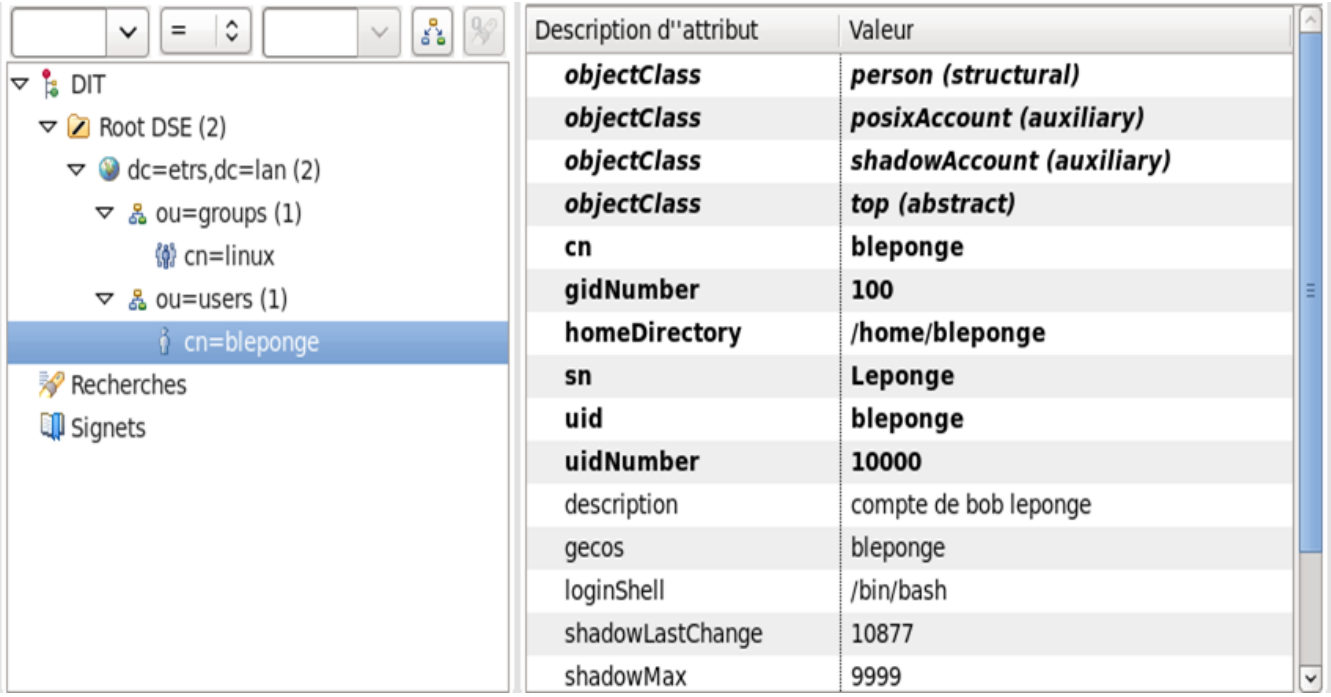
Exemple :

```
[root]# ldapmodify -x -D cn=admin,dc=formatux,dc=lan -W
```

Option	Description
-x	Utiliser l'authentification simple au lieu de SASL
-D	BindDN à utiliser pour la connexion à la base.
-W ou -w	Demander le mot de passe (interactif ou non).
-f	Lire les modifications à effectuer depuis un fichier

Il n'est pas nécessaire de préciser le serveur à contacter (options -H ou -h) si celui-ci est renseigné dans le fichier /etc/openldap/ldap.conf

Exemples



Il faut séparer les cas suivants :

- Ajouter/Supprimer un objet. Supprimer bleponge ou ajouter asaglisse.
- Modifier un objet en lui ajoutant, supprimant ou modifiant un attribut.

Ajouter un objet

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:///
dn:dndelobjetaajouter
changetype: add
...
```

Supprimer un objet

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:///
dn:dndelobjetasupprimer
changetype: delete
```

Modifier un objet

- Ajouter un attribut :

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:///
dn: dndelobjetamodifier
changetype: modify
add: nomdelatrribut
nomdelattribut: nouvellevaleur
```

- Supprimer un attribut

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:///
dn: dndelobjetamodifier
changetype: modify
delete: nomdelattribut
```

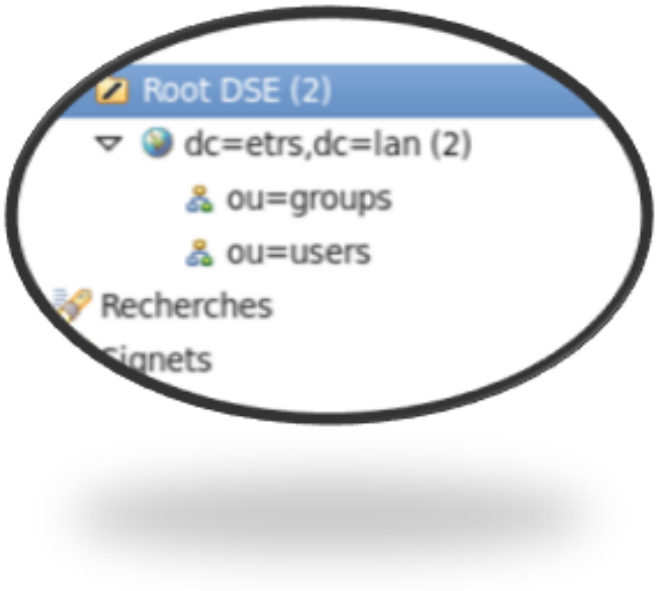
- Modifier un attribut

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:///
dn: dndelobjetamodifier
changetype: modify
replace: nomdelattribut
nomdelattribut: nouvellevaleur
```

La structure de la DIT

Les données de l'arbre de l'annuaire doivent être rangées dans des unités d'organisation (**OU**).

Les OU **users** et **groups** sont généralement utilisées.



Commencer par ajouter une entrée dans l'annuaire correspondant à l'organisation de l'entité :

```
[root]# ldapmodify -x -D cn=admin,dc=formatux,dc=lan -W
Enter LDAP Password:
dn: dc=formatux,dc=lan
changetype: add
objectClass: dcObject
objectClass: organization
dc: formatux
o: formatux
description: Serveur formatux
```

Puis les deux OU concernées :

Le fichier /root/structure.ldif

```
dn: ou=users,dc=formatux,dc=lan
changetype: add
objectClass: top
objectClass: organizationalUnit
ou: users
description: Utilisateurs de Formatux

dn: ou=groups,dc=formatux,dc=lan
changetype: add
objectClass: top
objectClass: organizationalUnit
ou: groups
description: Groupes d'utilisateurs de Formatux
```

```
[root]# ldapmodify -x -D cn=admin,dc=formatux,dc=lan -W -f /root/structure.ldif
```

Activation des logs

Dans certains cas, il sera intéressant d'activer la journalisation dans la base cn=config.

Celle-ci étant très verbeuse, elle sera activée ou désactivée au besoin.

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:///
dn: cn=config
changeType: modify
replace: olcLogLevel
olcLogLevel: stats
```

Il faut paramétrer le service syslog pour qu'il accepte les logs (niveau local4).

Le fichier /etc/rsyslog.conf

```
local4.* /var/log/slapd.log
```

sans oublier de redémarrer le démon syslog :

```
[root]# service rsyslogd restart
```

Le logLevel 4 permet de journaliser les requêtes effectuées sur la base.

Activation du TLS

Avant de pouvoir configurer le **TLS** sous OpenLDAP, il convient de disposer du certificat et de la clef pour le serveur ainsi que le certificat de l'autorité de certification, qui est indispensable au bon fonctionnement d'OpenLDAP.

Pour créer ces certificats, il est possible d'utiliser easy-rsa, qui sera abordé dans la quatrième partie du document.

Une autre méthode est d'utiliser l'outil **certtool** du paquet **gnutls-utils**.

Si l'accès au serveur LDAP se fait via le FQDN ldap.formatux.lan, il faudra impérativement créer le certificat qui répondra à ce nom. Il ne sera plus possible par la suite de se connecter en LDAPS ou en starttls via l'adresse de loopback localhost.

Création des certificats avec certtools

Installer le paquet gnutls-utils :

```
[root]# yum install gnutls-utils
```

Dans le cas d'un certificat auto-signé, il faut dans un premier temps créer une clef privée pour l'autorité de certification :

```
[root]# certtool --generate-privkey --outfile /etc/pki/CA/private/ca-key.key
```

et décliner cette clef privée en certificat public.

```
[root]# certtool --generate-self-signed --load-privkey /etc/pki/CA/private/ca-key.key
--outfile /etc/pki/CA/certs/ca-cert.pem
```

Il faut ensuite générer un certificat privé pour le serveur (ldap.formatux.lan par exemple)

```
[root]# certtool --generate-privkey --outfile /etc/pki/tls/private/ldap.key
```

Puis son certificat public signé par la clef privée de l'autorité de certification créée ci-dessus :

```
[root]# certtool --generate-certificate --load-privkey /etc/pki/tls/private/ldap.key
--outfile /etc/pki/tls/certs/ldap.pem --load-ca-certificate /etc/pki/CA/certs/ca-
cert.pem --load-ca-privkey /etc/pki/CA/private/ca-key.key
```

Prise en compte des certificats

Le fichier /root/tls.ldif

```
dn: cn=config
changetype: modify
replace: olcTLSCertificateFile
olcTLSCertificateFile: /etc/pki/certs/ldap.pem
-
replace: olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/pki/private/ldap.key
-
replace: olcTLSCACertificateFile
olcTLSCACertificateFile: /etc/pki/CA/certs/ca-cert.pem
```

```
[root]# ldapmodify -Y EXTERNAL -H ldapi:/// -f /root/tls.ldif
```

La chaîne de connexion du fichier /etc/openldap/ldap.conf doit également être mise à jour :

Le fichier /etc/openldap/ldap.conf

```
BASE dc=formatux,dc=lan
URI ldaps://ldap.formatux.lan

TLS_CACERTDIR /etc/openldap/certs
TLS_REQCERT try
```

La commande **cacertdir_rehash** permet de créer un lien symbolique vers le certificat de la CA dont le nom correspond au hash de ce certificat. Ceci est nécessaire au fonctionnement d'openLDAP en TLS !

```
[root]# cacertdir_rehash /etc/pki/CA/certs
[root]# ls -l /etc/pki/CA/certs
-rw-r--r--. 1 root root 1281 4 déc. 10:52 ca-cert.pem
lrwxrwxrwx. 1 root root 11   4 déc. 10:54 ce6a8cab.0 -> ca-cert.pem
```

Par défaut, le service slapd n'écoute pas sur le port 636 (ldaps) et il faut privilégier le startTLS sur le port 389. Pour activer le ldaps :

Le fichier /etc/sysconfig/ldap

```
SLAPD_LDAPS=yes
```

sans oublier de relancer le serveur :

```
[root]# service slapd restart
```

Tester la connexion

La commande openssl permet de tester la connexion uniquement sur le port 636 :

```
[root]# openssl s_client -connect ldap.formatux.lan:636 -showcerts
```

Le certificat de la CA

De nombreuses applications auront besoin du certificat de la **CA**.

Il est recommandé de le mettre à disposition des utilisateurs sur le serveur web du serveur LDAP :

```
[root]# cp /etc/pki/CA/certs/ca-cert.pem /var/www/html/
```

Le certificat est ainsi accessible via l'adresse http://ldap.formatux.lan/cacert.pem.

Configuration du PAM

PAM peut être configuré pour utiliser le service openldap avec la commande authconfig :

```
[root]# yum install nss-pam-ldapd
```

```
[root]# authconfig --enableldap --enableldapauth --ldapserver=ldap://ldap.formatux.lan
--ldapbasedn="ou=users,dc=formatux,dc=lan" --enableldaptls
--ldaploadcacert=http://ldap.formatux.lan/ca-cert.pem --enablemkhomedir --update
```

Création des utilisateurs

Création du fichier pour l'utilisateur :

```
vim /root/antoine.ldif

dn: cn=alemorvan,ou=users,dc=formatux,dc=lan
objectClass: top
objectClass: person
objectClass: posixAccount
objectClass: shadowAccount
cn: alemorvan
sn: Le Morvan
uid: alemorvan
uidNumber: 10000
gidNumber: 500
homeDirectory: /home/alemorvan
loginShell: /bin/bash
userPassword: {crypt}password
gecos: alemorvan
shadowWarning: 7
```

```
ldapadd -x -D cn=admin,dc=formatux,dc=lan -W -f /root/antoine.ldif
```

Chapitre 10. Installation d'un serveur Shinken

10.1. Généralités

Shinken est un logiciel de supervision créé en 2009 par Jean Gabes. Il est une réécriture de Nagios en Python et en reprend complètement l'esprit. Il va cependant permettre de répondre à des contraintes techniques auxquelles Nagios ne pouvait pas.

Shinken dépend essentiellement de Pyro, librairies Python. Son code est ouvert, libre et communautaire.

Son architecture est décentralisée et se base sur plusieurs processus. Elle prévoit également la haute disponibilité et de hautes performances.

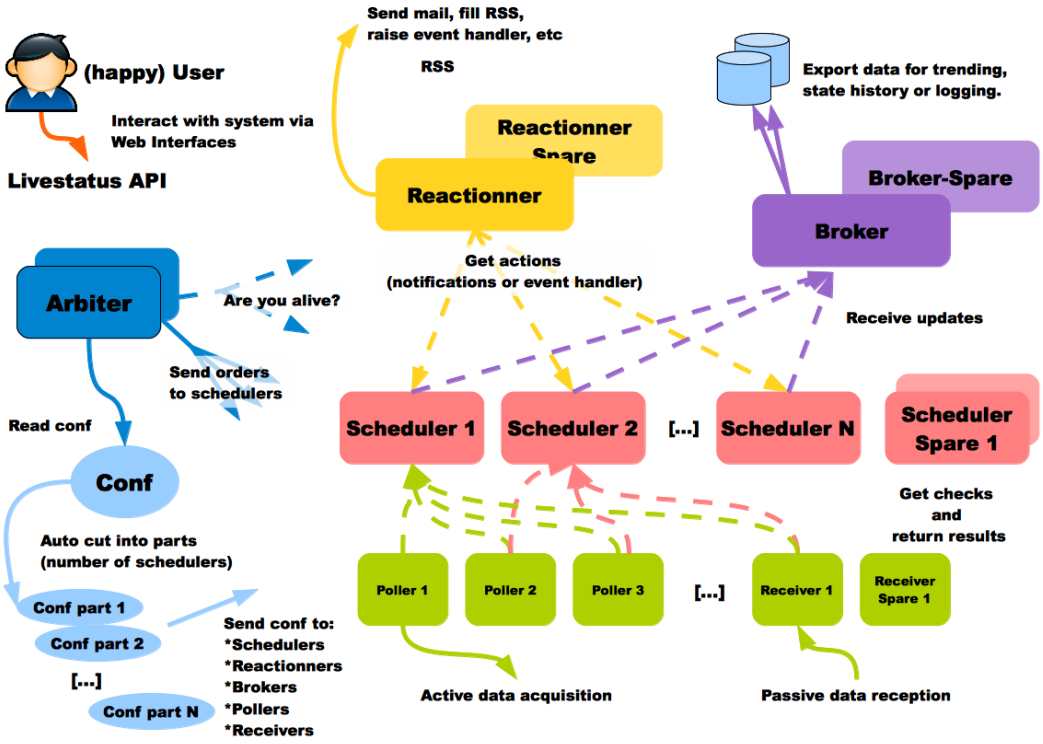


Figure 30. Architecture

10.2. Prérequis

L'installation est effectuée sur un serveur CentOS 7 minimal. Il est nécessaire de posséder un serveur correctement configuré (nom et adressage IP).

Ajouter l'utilisateur **shinken** qui sera utilisé pour le déploiement, lui configurer un mot de passe et l'ajouter au groupe **wheel** pour l'utilisation de sudo.

```
[root@srv-shinken ~]# useradd shinken
[root@srv-shinken ~]# passwd shinken
[root@srv-shinken ~]# usermod -aG wheel shinken
```

Vérifier ou activer si besoin %wheel avec la commande visudo et désactiver requiretty :

```
    Defaults    !requiretty
    %wheel      ALL=(ALL)       ALL
```

Se connecter ensuite avec l'utilisateur shinken pour poursuivre l'installation.

```
[root@localhost ~]# su - shinken
```

Configurer un premier dépôt epel :

```
[shinken@srv-shinken ~]$ sudo yum install epel-release
```

Puis un second dépôt pour mongodb :

```
[shinken@srv-shinken ~]$ sudo vim /etc/yum.repos.d/mongodb.repo
    [mongodb]
    name=MongoDB Repository
    baseurl=http://downloads-distro.mongodb.org/repo/redhat/os/x86_64/
    gpgcheck=0
    enabled=1
```

10.3. Installation

Installation des composants nécessaires à Shinken

Installation des composants Python :

```
[shinken@srv-shinken ~]$ sudo yum install -y python-pip python-pycurl python-
setuptools git python-pymongo python-cherrypy
```

Installation d'une base de données nécessaire à WebUI qui sera utilisé pour l'affichage graphique. Ici, mongodb sera utilisé mais d'autres bases de données peuvent être utilisées.

```
[shinken@srv-shinken ~]$ sudo yum -y install mongodb-org mongodb-org-server
```

Activation et démarrage :

```
[shinken@srv-shinken ~]$ sudo chkconfig mongod on
[shinken@srv-shinken ~]$ sudo systemctl start mongod
```

Revenir en tant que root pour installer pyro.

```
[shinken@srv-shinken ~]$ exit
[root@srv-shinken ~]#
```

Installation de Pyro :

```
[root@srv-shinken ~]# easy_install pyro
```

Installation de shinken

De nouveau avec l'utilisateur shinken, récupérer les sources sur le dépôt github :

```
[root@srv-shinken ~]# su - shinken
[shinken@srv-shinken ~]$ git clone https://github.com/naparuba/shinken.git
```

Se déplacer dans le dossier shinken et lancer l'installation:

```
[shinken@srv-shinken ~]$ cd shinken
[shinken@srv-shinken shinken]$ sudo python setup.py install
```

Installation du package shinken. Ce package installe seulement un outil d'administration et d'installation de modules supplémentaires pour shinken.

```
[shinken@localhost shinken]$ sudo yum install -y shinken
```

Donner les droits nécessaires à l'utilisateur shinken :

```
[shinken@srv-shinken ~]$ sudo chown -R shinken:shinken /var/lib/shinken/
[shinken@srv-shinken ~]$ sudo chown -R shinken:shinken /var/log/shinken/
[shinken@srv-shinken ~]$ sudo chown -R shinken:shinken /var/run/shinken/
```

Activer tous les services shinken au démarrage :

```
[shinken@srv-shinken ~]$ sudo chkconfig shinken on
[shinken@srv-shinken ~]$ sudo chkconfig shinken-arbiter on
[shinken@srv-shinken ~]$ sudo chkconfig shinken-broker on
[shinken@srv-shinken ~]$ sudo chkconfig shinken-poller on
[shinken@srv-shinken ~]$ sudo chkconfig shinken-reactionner on
[shinken@srv-shinken ~]$ sudo chkconfig shinken-receiver on
[shinken@srv-shinken ~]$ sudo chkconfig shinken-scheduler on
```

Initialisation de shinken. Création d'un fichier **.shinken.ini** caché dans la home directory de l'utilisateur shinken.

```
[shinken@srv-shinken ~]$ shinken --init
```

Installation et configuration des modules

- Module permettant d'initialiser l'interface graphique **webui2**

```
[shinken@srv-shinken ~]$ shinken install webui2
Grabbing : webui2
OK webui2
```

Il faut alors modifier le fichier **/etc/shinken/brokers/broker-master.cfg** afin de définir le module **webui2**.

```
[shinken@srv-shinken ~]$ vim /etc/shinken/brokers/broker-master.cfg
    ...
    modules webui2
    ...
```

Pour que **webui2** fonctionne, il faut également installer les dépendances suivantes avec **easy_install** en tant que root.

```
[shinken@srv-shinken ~]$ exit
[root@srv-shinken ~]#
[root@srv-shinken ~]# easy_install bottle
[root@srv-shinken ~]# easy_install pymongo
[root@srv-shinken ~]# easy_install requests
[root@srv-shinken ~]# easy_install arrow
[root@srv-shinken ~]# easy_install passlib
```

Se reconnecter ensuite avec l'utilisateur shinken.

```
[root@srv-shinken ~]# su - shinken
[shinken@srv-shinken ~]$
```

- Module utilisé comme moyen d'authentification.

```
[shinken@srv-shinken ~]$ shinken install auth-cfg-password
Grabbing : auth-cfg-password
OK auth-cfg-password
```

- Module utilisé pour la base de données mongodb.

```
[shinken@srv-shinken ~]$ shinken install mod-mongodb
Grabbing : mod-mongodb
OK mod-mongodb
```

Il faut ensuite modifier le fichier **/etc/shinken/modules/webui2.cfg** afin de définir les deux modules précédemment installés.

```
[shinken@srv-shinken ~]$ vim /etc/shinken/modules/webui2.cfg
    ...
    modules auth-cfg-password,mongodb
    ...
```

Le module **auth-cfg-password** étant utilisé, il faut modifier le mot de passe dans le fichier **/etc/shinken/contacts.admin.cfg**. Ce sont les identifiants définis dans ce fichier (contact_name et password) qui seront utilisés pour s'authentifier.

```
[shinken@srv-shinken ~]$ vim /etc/shinken/contacts/admin.cfg
    ...
    password    **********
    ...
```

10.4. Démarrage de shinken

Afin d'accéder a l'URL de shinken http://srv-shinken:7767, il est nécessaire de configurer le pare-feu en créant une nouvelle règle et de redémarrer le service.

```
[shinken@srv-shinken ~]$ sudo vim /etc/sysconfig/iptables
    ...
    -A INPUT -p tcp -m state --state NEW -m tcp --dport 7767 -j ACCEPT
    ...
[shinken@srv-shinken ~]$ sudo systemctl restart iptables
```

Démarrer ensuite le service shinken afin de prendre en compte la configuration.

```
[shinken@srv-shinken ~]$ sudo systemctl start shinken
```

L'URL donnée ici n'est bien sûr valable que si une résolution de noms permet de définir le nom du serveur (srv-shinken). Il est également possible d'utiliser l'adresse IP (http://@IP:7767).

Le port 7767 est le port de fonctionnement par défaut.

Figure 31. Fenêtre de connexion

10.5. Références

https://shinken.readthedocs.io/en/latest/index.html

https://www.it-connect.fr/installer-shinken-3-0-sur-centos-7-en-10-etapes/

Chapitre 11. Serveur proxy SQUID

11.1. Principes de fonctionnement

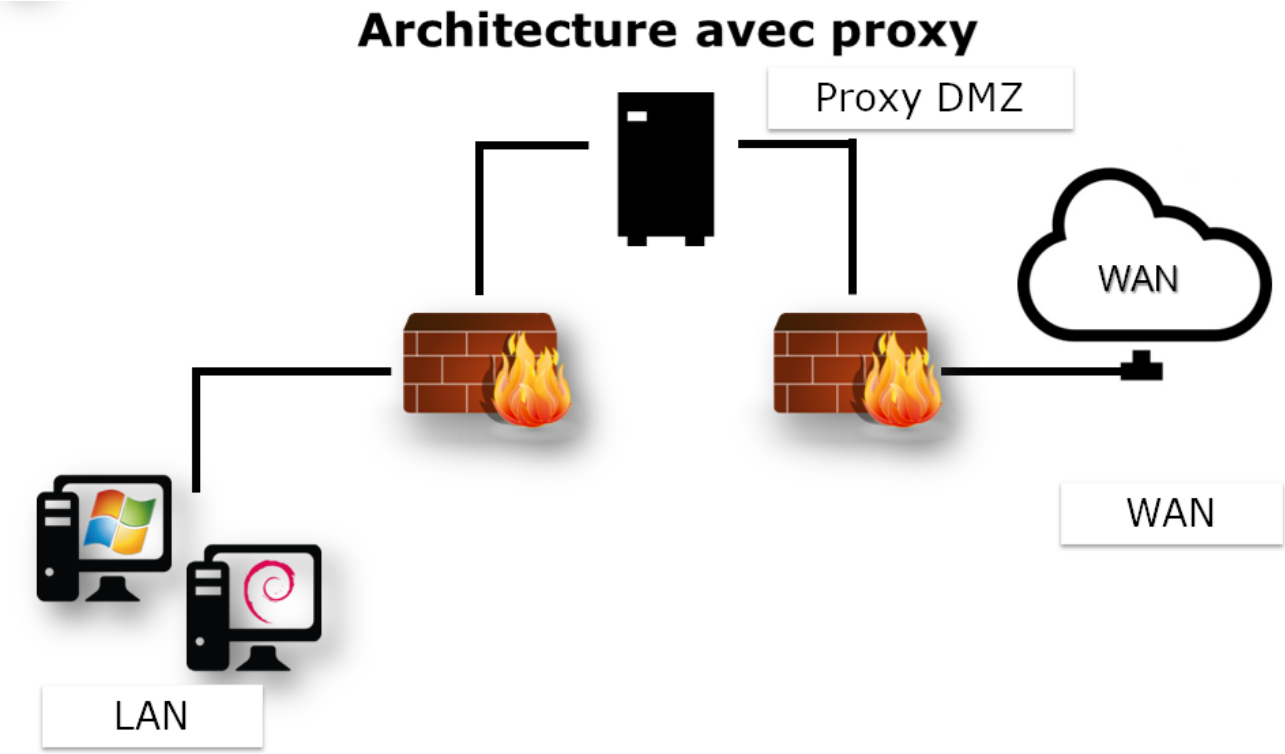
La mise en place d'un serveur proxy nécessite de choisir entre deux types d'architectures :

- Une architecture proxy standard, qui nécessitera une configuration spécifique de chaque client et de leurs navigateurs internet,
- Une architecture dite proxy captif, qui nécessitera l'interception des trames émises par le client et de les réécrire vers le serveur proxy.

Dans un cas comme dans l'autre, il y a rupture au niveau du réseau.

Un client ne peut physiquement plus s'adresser directement à un serveur distant, sans passer par un mandataire, appelé plus couramment serveur proxy.

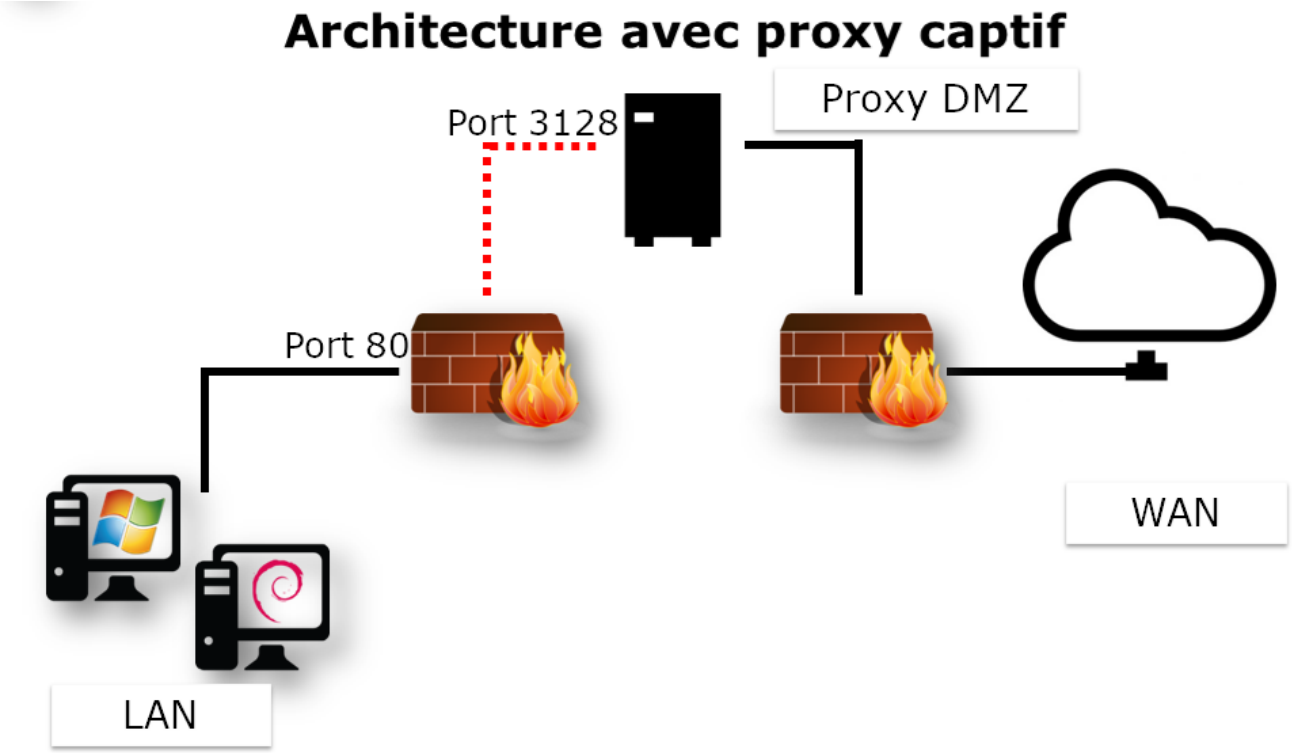
Le poste client est protégé par deux pare-feux et ne communique jamais directement vers le réseau extérieur.



Cette architecture nécessite la configuration du navigateur sur le poste client.

Dans le cas du proxy captif, il n'y a pas de nécessité de configurer l'ensemble des postes clients.

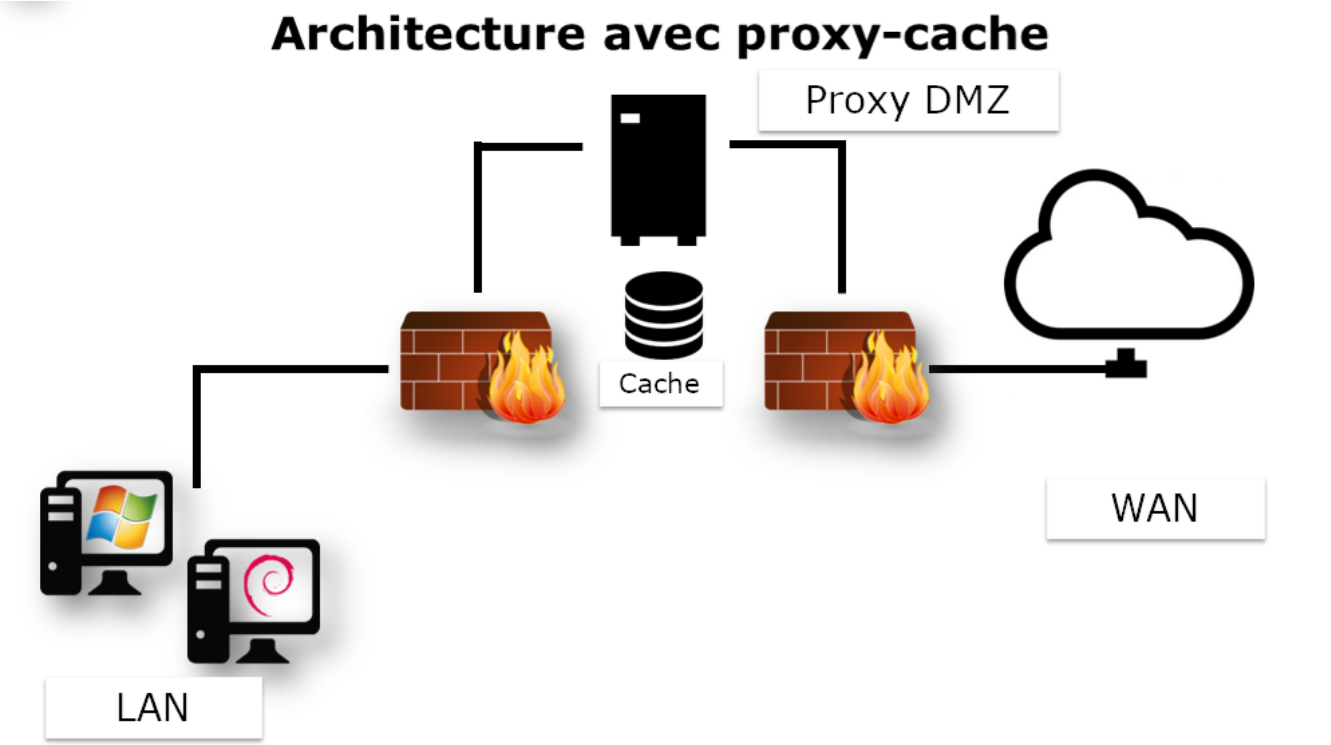
La configuration se passe au niveau de la passerelle, qui reçoit les demandes des clients, et qui va, de manière totalement transparente, réécrire les trames pour les envoyer vers le proxy.



Cette architecture nécessite une configuration spécifique sur le routeur.

Dans le cas de l'architecture proxy standard ou du proxy captif, l'un des premiers intérêts de ce genre de service est bien évidemment de faire office de cache.

Ainsi, un fichier qui aura été téléchargé une première fois depuis le WAN (et donc potentiellement une liaison plus lente que le LAN), sera conservé en mémoire par le proxy-cache pour être resservi au profit des clients suivants. Ainsi, la bande passante de la liaison lente est optimisée.



Comme nous le verrons dans la suite de ce chapitre, ce n'est bien évidemment pas la seule utilité d'un proxy.

Un proxy pourra être déployé pour :

- Interdire l'accès à certaines ressources en fonction de différents paramètres,
- Mettre en place une authentification et un suivi des activités sur internet des clients,
- Mettre en place une hiérarchie de cache distribués,
- Masquer l'architecture du LAN d'un point de vue WAN (combien y a-t-il de clients sur le LAN ?).

Les intérêts sont multiples :

- Anonymat sur Internet ;
- Authentification ;
- Journaliser les activités des clients ;
- Filtrage ;
- Limiter les accès ;
- Optimisation de la bande passante ;
- Sécurité.

Mettre en place l'authentification bloque une grande partie des effets malveillants des virus sur le LAN.

Le service proxy devient un service critique nécessitant une haute disponibilité.

Durant l'exploitation d'un serveur Proxy Squid, l'administrateur est amené à exploiter les logs. Il est donc primordiale de connaître les principaux codes réponses HTTP.

Table 18. Les codes réponses HTTP

Code	Catégories
1XX	Info
2XX	Succès
3XX	Redirection
4XX	Erreur de requête client
5XX	Erreur sur le serveur

Exemples :

- 200 : ok
- 301 : Moved Permanently
- 302 : Moved Temporatly
- 304 : Not modified
- 400 : Bad request
- 401 : Unauthorized
- 404 : Not found

11.2. Le serveur SQUID

Squid prend en charge les protocoles http et ftp.

Les intérêts d'installer une solution basée sur le serveur Squid :

- Les solutions matérielles sont coûteuses ;
- Il est développé depuis 1996 ;
- Il est publié sous licence GNU/GPL.

Dimensionnement

- Prévoir des solutions de haute disponibilité ;
- Privilégier des disques durs rapides pour le cache ;
- Mémoire vive et CPU correctement dimensionnés.

Il faut prévoir 14Mo de RAM par Go de cache sur le disque.

Installation

L'installation du serveur Squid se fait avec le paquet squid.

```
yum install squid
chkconfig squid on
```

Attention à ne pas démarrer le service tant que le cache n'a pas été initialisé !

Arborescence et fichiers du serveur Squid

Le fichier de configuration unique est le fichier **/etc/squid/squid.conf**.

Les logs du service (arrêt et relance) sont enregistré dans le fichier **/var/log/squid.cache.log** tandis que les requêtes des clients **/var/log/squid/access.log**. Les fichiers de cache seront par défaut stockés dans **/var/spool/squid/**.

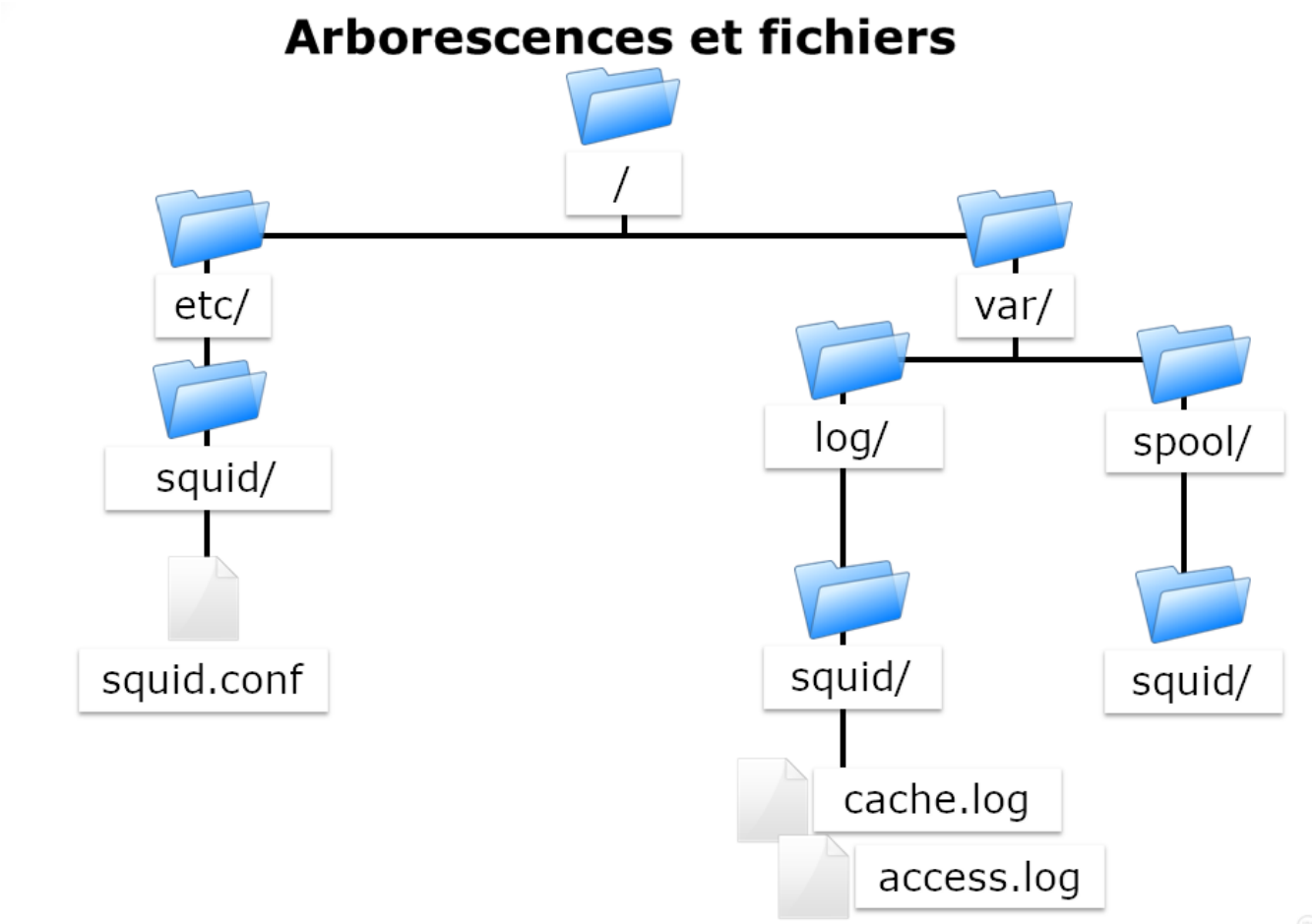


Figure 32. Arborescence et fichiers du serveur Squid

La commande squid

Commande squid permet de contrôler le serveur squid.

Syntaxe de la commande squid

```
squid [-z|-s|-k parse|-k rotate]
```

Option	Description
-z	Initialise les répertoires du cache
-s	Active la journalisation syslog
-k parse	Test le fichier de configuration
-k rotate	Effectue une rotation des logs

Journaliser les requêtes clientes peut rapidement entraîner le stockage des volumes conséquents de données.

Il est opportun de régulièrement créer un nouveau fichier de log et d'archiver l'ancien dans un format compressé.

Cette action peut être effectuée manuellement avec l'option **–k rotate** de la commande squid ou via le service Linux dédié **Logrotate**.

11.3. Configuration basique

La configuration de Squid se fait dans le fichier de configuration **/etc/squid/squid.conf**.

- Numéro de port du proxy (port d'écoute)

Syntaxe de la directive http_port

```
http_port num_port
```

Par défaut, le numéro de port est fixé à 3128 mais il est fréquemment changé à 8080. Il faudra penser à ouvrir le port correspondant du pare-feu !

Par exemple :

```
http_port 8080
```

Au redémarrage du service, le serveur Squid se mettra en écoute sur le port défini par la directive http_port.

- Réservation de la mémoire vive

Syntaxe de la directive cache_mem

```
cache_mem taille KB|taille MB|taille GB
```

Par exemple :

```
cache_mem 1 GB
```

Bonne pratique : 1/3 du total de la mémoire vive allouée

- Protocole de Cache Internet (ICP)

Le protocole ICP (Internet Cache Protocol) permet aux serveurs Squid voisins de s'échanger des requêtes. Il est courant de proposer une hiérarchie de proxy qui se partagent leurs bases d'informations.

La directive icp_port permet de définir le numéro de port sur lequel Squid envoie et reçoit les requêtes ICP des serveurs Squid voisins.

Par exemple :

```
icp_port 3130
```

Positionner à 0 pour le désactiver.

- Utilisateur FTP anonyme

La directive ftp_user permet d'associer un utilisateur FTP aux connexions FTP anonymes. L'utilisateur dit être une adresse de messagerie valide.

```
ftp_user bob@formatux.lan
```

- Mettre en place des Access Control List

Syntaxe des ACL

```
acl nom type argument
http_access allow|deny nomacl
```

Exemple :

```
acl REPAS time 12:00-14:00
http_access deny REPAS
```

Les ACL sont étudiées plus en détail dans la partie "Configuration avancée".

- Taille maximum d'un objet en cache

Syntaxe de la directive maximum_object_size

```
maximum_object_size size
```

Exemple :

```
maximum_object_size 32 MB
```

Si la taille de l'objet est supérieur à la limite maximum_object_size, l'objet n'est pas conservé en cache.

- Nom du serveur proxy

Syntaxe de la directive visible_hostname

```
visible_hostname nom
```

Exemple :

```
visible_hostname proxysquid
```

La valeur fournie peut être différente du nom d'hôte.

- Définir un cache pour squid

```
cache_ufs format chemin taille nbDossierNiv1 nbDossierNiv2
```

Plusieurs caches peuvent être définis sur différents systèmes de fichiers pour optimiser les temps d'accès.

Exemple :

```
cache_dir ufs /var/spool/squid/ 100 16 256
```

Option	Description
ufs	Unix File System
100	Taille en méga
16	16 dossiers de premier niveau
256	256 dossiers de second niveau

Au premier lancement du service, il faut initialiser le dossier de cache :

```
[root]# squid -z
[root]# service squid start
```

11.4. Configurations avancées

Les Access Control List (ACL)

Syntaxe de la directive http_access

```
http_access allow|deny [!]nom_acl
```

Exemple :

```
http_access allow REPAS
http_access deny !REPAS
```

L'ACL !nom_acl est le contraire de l'ACL nom_acl.

Syntaxe de la directive acl

```
acl nom type argument
```

L'ordre des ACL est cumulatif. Plusieurs ACL de même nom représentent une seule ACL.

Exemples :

- Autoriser à l'heure des repas :

```
acl REPAS time 12:00-14:00
http_access allow REPAS
```

- Interdire les vidéos :

```
acl VIDEOS rep_mime_type video/mpeg
acl VIDEOS rep_mime_type video/avi
http_access deny VIDEOS
```

- Gestion des adresses IP :

```
acl XXX src 192.168.0.0/255.255.255.0
acl XXX dst 10.10.10.1
```

- Gestion des FQDN :

```
acl XXX srcdomain .formatux.lan
acl XXX dstdomain .linux.org
```

- Gestion des ports :

```
acl XXX port 80 21
```

- Gestion des protocoles :

```
acl XXX proto HTTP FTP
```

Les algorithmes de cache

Il existe différents algorithmes de cache qui disposent de caractéristiques différentes :

- LRU - **Least Recently Used** : supprime les objets les plus anciens de la mémoire vive.
- LRU-THOLD : copie en fonction de sa taille un objet dans le cache.
- MRU : **Most Recently Used** : les données les moins demandées sont supprimées.
- GDSF : **Greedy Dual Size Frequency** : supprime en fonction de la taille et du temps d'accès d'origine. Les plus petits sont conservés.
- LFUDA : **Least Frequently Used With Dynamic Aging** : idem que GDSF sans notion de taille. Utile pour les caches avec des fichiers de grande taille.

11.5. Authentification des clients

Squid s'appuie sur des programmes externes pour gérer l'authentification. Il peut ainsi s'appuyer sur un simple fichier plat type htpasswd ou sur un service LDAP, SMB, PAM, etc.

L'authentification peut être une nécessité juridique : pensez à faire signer une charte d'usage à vos

utilisateurs !

11.6. Outils

La commande squidclient

La commande squidclient permet de tester une requête vers le serveur squid.

Syntaxe de la commande squidclient

```
squidclient [–s] [-h cible] [-p port] url
```

Exemple :

```
squidclient –s –h localhost –p 8080 http://localhost/
```

Option	Description
-s	Mode silencieux (n'affiche rien dans la console)
-h	Définir un proxy cible
-p	Port d'écoute (par défaut 3128)
-r	Forcer le serveur à recharger l'objet

Analyser les logs

Les enregistrements du journal de Squid peuvent être suivi avec la commande :

```
tail –f /var/log/squid/access.log
```

- Décomposition d'une ligne de log

Option	Description
Date	Horodatage du log
Tps reponse	Temps de réponse pour la requête
@ client	Adresse IP du client
Code status	Code HTTP de la réponse
Taille	Taille du transfert
Méthode	Méthode HTTP (Put / Get / Post / etc.)
URL	URL de la requête
Peer Code	Code de réponse inter-proxy

Option	Description
Type fichier	Type mime de la cible de la requête

La commande sarg

La commande sarg (**Squid Analysis Report Generator**) permet de générer un rapport au format HTML.

Syntaxe de la commande sarg

```
sarg -x
```

- Installer sarg :

```
[root]# yum install httpd
[root]# yum install sarg
```

- Configurer sarg :

```
[root]# vim /etc/sarg/sarg.conf
access_log /var/log/squid/access.log
```

Le rapport est généré sous **/var/www/html/squid_reports/**.

SquidGuard

SquidGuard permet de filtrer les URLs à partir de blacklists (éventuellement disponibles sur Internet).

Sa mise en œuvre dépasse toutefois le cadre de ce support.

Chapitre 12. Serveur de log Syslog

Le système génère des logs qu'il faut surveiller pour la sécurité du système ou pour réagir avant la panne.

Sous Linux, c'est le rôle du protocole Syslog.

12.1. Généralités

Syslog est le protocole de journalisation standard sous Linux. Syslog gère le journal d'événement Linux, que ce soit pour le noyau Linux ou pour les services hébergés sur la station.

- Les logs peuvent être archivés localement (dans ce cas il faut prévoir une rotation des logs).
- Syslog peut également fonctionner en local en mode client/serveur. Syslog utilise le port 514 en UDP ou TCP pour sa communication réseau.

Exemple de fichier /var/log/messages :

```
Nov 23 08:30:00 centos6 dhcp service[warning] 110 message
```

Sous CentOS 6, c'est le logiciel rsyslog qui est utilisé pour gérer les logs du système. A noter que la syntaxe rsyslog est compatible avec les clients syslog standard (syslog, syslog-ng), mais l'inverse n'est pas vrai.

Un journal au format syslog comporte dans l'ordre les informations suivantes :

- la date à laquelle a été émis le log,
- le nom de l'équipement ayant généré le log (hostname),
- une information sur le processus qui a déclenché cette émission,
- le niveau de priorité du log,
- un identifiant du processus ayant généré le log
- le corps de message.

Certaines de ces informations sont optionnelles.

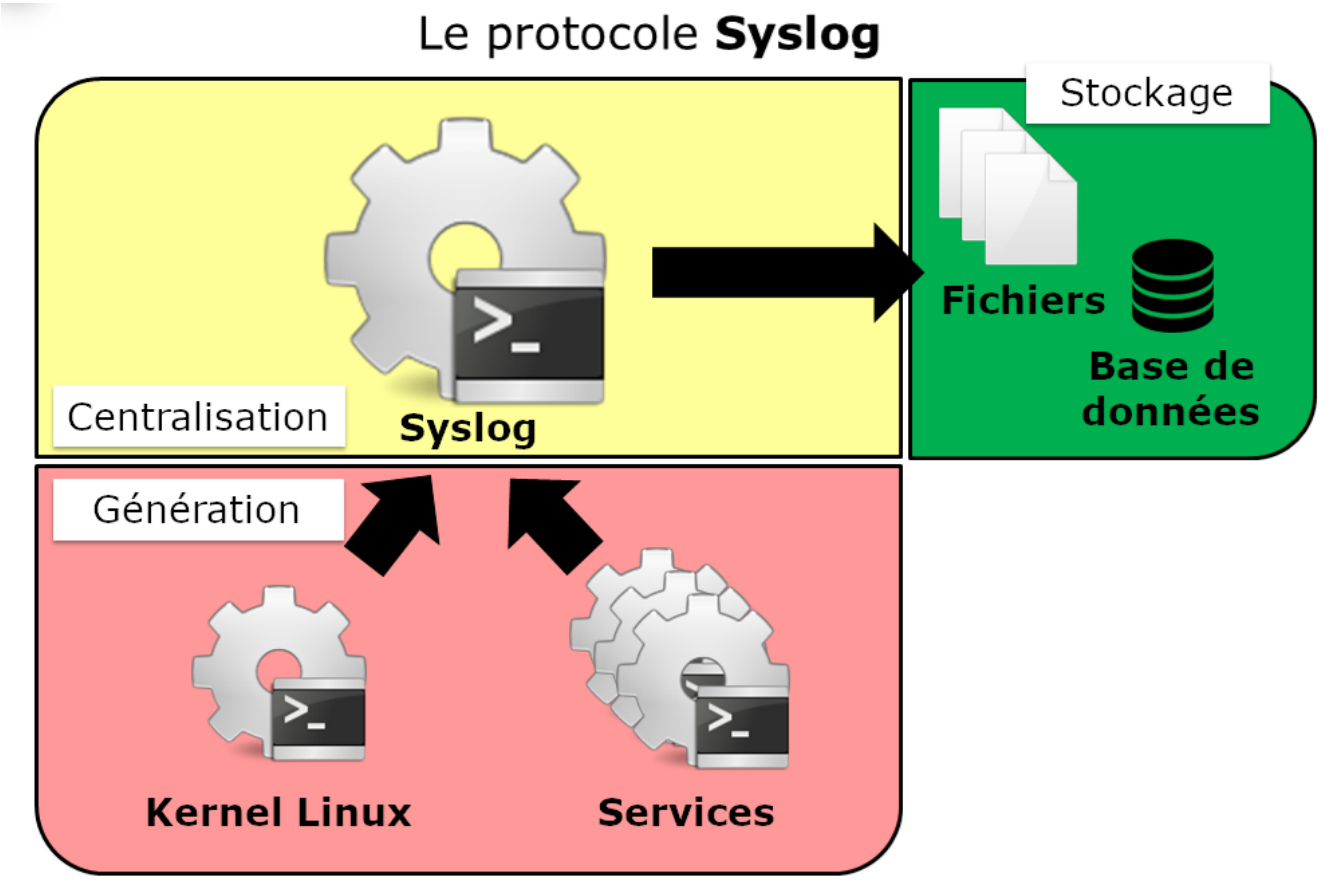


Figure 33. Fonctionnement du protocole Syslog

Le serveur Syslog centralise les messages du kernel Linux ou des services dans des fichiers. Des modules existent pour rediriger les logs vers une base de données.

En mode client/serveur, les clients envoient leurs logs vers un serveur syslog sur le port 514. Ce serveur peut ensuite stocker les logs de ses clients vers un serveur de base de données.

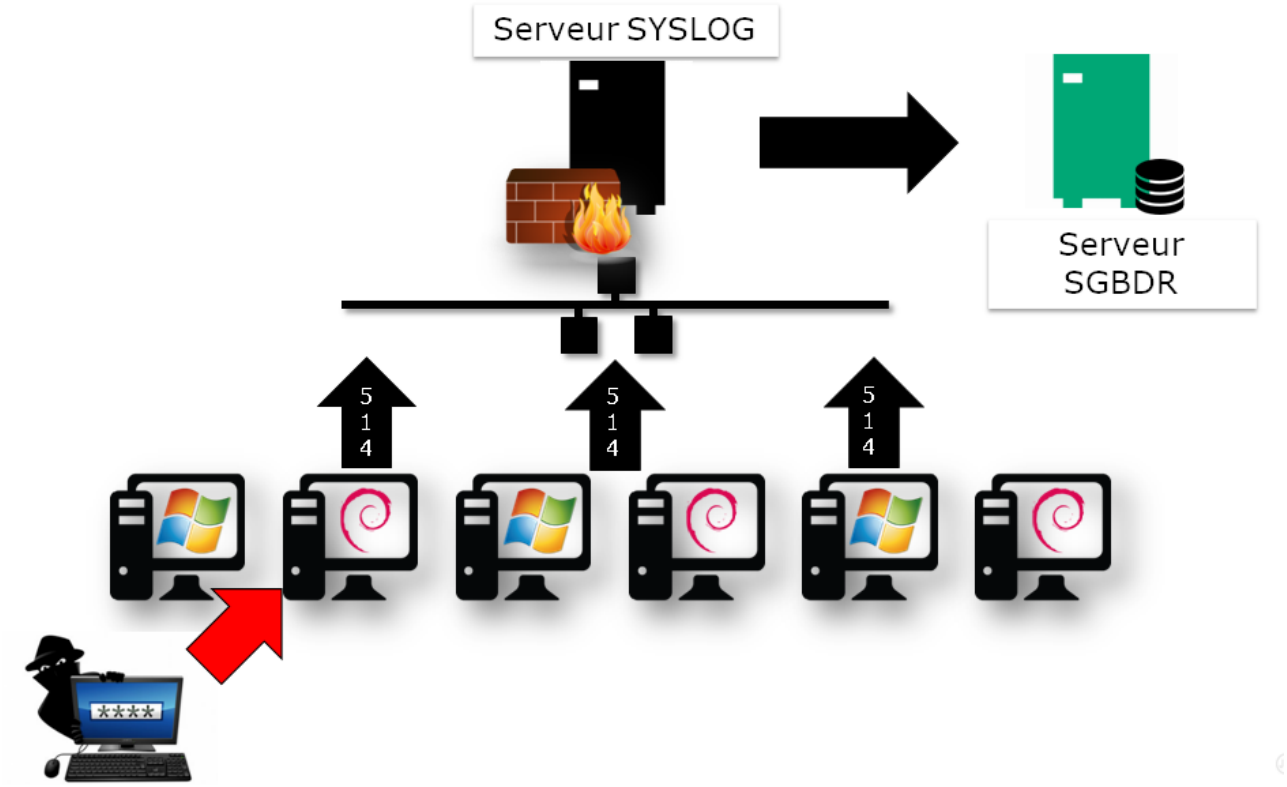


Figure 34. Mise en réseau du protocole Syslog

Ainsi, un attaquant ne peut pas effacer ces traces qui sont déportées sur un serveur distant.

Les catégories de messages

Les messages sont orientés selon leur origine et leur gravité (origine.gravité).

Table 19. Origine des messages Syslog

Code	Mot-clé	Description
0	kern	kernel messages
1	user	user-level messages
2	mail	mail system
3	daemon	system daemons
4	auth	security/authorization messages
5	syslog	messages generated internally by syslogd
6	lpr	line printer subsystem
7	news	network news subsystem
8	uucp	UUCP subsystem
9		clock daemon
10	authpriv	security/authorization messages
11	ftp	FTP daemon

Code	Mot-clé	Description
12	-	NTP subsystem
13	-	log audit
14	-	log alert
15	cron	clock daemon
16	local0	local use 0 (local0)
17	local1	local use 1 (local1)
18	local2	local use 2 (local2)
19	local3	local use 3 (local3)
20	local4	local use 4 (local4)
21	local5	local use 5 (local5)
22	local6	local use 6 (local6)
23	local7	local use 7 (local7)

Table 20. Gravité des messages syslog

Code	Gravité	Mot-clé	Description
0	Emergency	emerg (panic)	Système inutilisable.
1	Alert	alert	Une intervention immédiate est nécessaire.
2	Critical	crit	Erreur critique pour le système.
3	Error	err (error)	Erreur de fonctionnement.
4	Warning	warn (warning)	Avertissement (une erreur peut intervenir si aucune action n'est prise).
5	Notice	notice	Événement normal méritant d'être signalé.
6	Informational	info	Pour information.
7	Debugging	debug	Message de mise au point.

12.2. Client Syslog

La configuration du client et du serveur rsyslog est centralisée dans le fichier **/etc/rsyslog.conf**.

Après toute modification il faut redémarrer le service :

```
service rsyslog restart
```

La commande logger génère une ligne de log.

Syntaxe de la commande logger

```
logger texte
```

Exemple :

```
logger "====> Marqueur"
```

Fichier /var/log/messages après exécution de la commande logger

```
Nov 23 08:30:00 centos6 stagiaire ====> Marqueur
```

Il est possible de rediriger les logs du client vers le serveur :

Modification du fichier /etc/rsyslog.conf pour envoyer les logs vers le réseau

```
*.*     @IPServeur:514
```

```
service rsyslog restart
logger test
```

Le message test est envoyé vers le serveur.

@IPServeur = UDP @IPServeur = TCP

Pour différentier une redirection en TCP d'une redirection en UDP, il faudra doubler l'arobase présent devant l'adresse IP du serveur.

Par exemple :

```
mail.err* @@172.16.96.203
```

Après avoir ajouté cette ligne, le service syslog enverra les logs de la catégorie mail d'un niveau de gravité supérieur à erreur vers le serveur syslog 172.16.96.203 en TCP.

La commande logwatch

La commande logwatch effectue une synthèse journalière des logs et l'envoie par message.

Installation :

```
yum install logwatch
```

LogWatch analyse pour vous quotidiennement les logs pour en extraire les informations du jour, les trie et vous envoie une synthèse quotidienne.

Les logs des services étant généralement très copieux, un outil tel Logwatch (couplé avec la redirection des mails) est nécessaire pour rester informé en un seul coup d'œil.

Voici un exemple de rapport :

```
 ################### Logwatch 7.3.6 (05/19/07) ####################
        Processing Initiated: Fri Oct 23 10:10:04 2015
        Date Range Processed: yesterday
                              ( 2015-Oct-22 )
                              Period is day.
      Detail Level of Output: 0
              Type of Output: unformatted
           Logfiles for Host: srv-instructeurs.formatux.lan
 ##################################################################

 --------------------- Selinux Audit Begin ------------------------


 ---------------------- Selinux Audit End -------------------------


 --------------------- Automount Begin ------------------------


 ---------------------- Automount End -------------------------


 --------------------- Cron Begin ------------------------


 ---------------------- Cron End -------------------------


 --------------------- httpd Begin ------------------------


 Requests with error response codes
    403 Forbidden
       /: 1 Time(s)
    404 Not Found
       /favicon.ico: 2 Time(s)

 ---------------------- httpd End -------------------------
```

```
--------------------- Init Begin -----------------------


---------------------- Init End ------------------------


--------------------- Named Begin -----------------------


Received control channel commands
   reload: 8 Time(s)
   stop: 7 Time(s)

---------------------- Named End ------------------------


--------------------- pam_unix Begin -----------------------

su-l:
   Authentication Failures:
   Sessions Opened:
      pupitre -> root: 1 Time(s)

sudo:
   Authentication Failures:


---------------------- pam_unix End ------------------------


--------------------- Postfix Begin -----------------------

   3.957K  Bytes accepted                              4,052
   3.957K  Bytes delivered                             4,052
========   ================================================

       4   Accepted                                  100.00%
--------   ------------------------------------------------
       4   Total                                     100.00%
========   ================================================

       4   Removed from queue
       2   Sent via SMTP
       2   Forwarded

       6   Postfix start
       6   Postfix stop
       1   Postfix waiting to terminate
```

```
---------------------- Postfix End ------------------------


--------------------- Connections (secure-log) Begin ------------------------

New Users:
   postgres (26)

New Groups:
   postgres (26)


   groupadd: group added to /etc/group: name=postgres, GID=26: 1 Time(s)
   groupadd: group added to /etc/gshadow: name=postgres: 1 Time(s)
   webmin: Successful login as pupitre from 172.16.96.232: 1 Time(s)

---------------------- Connections (secure-log) End -------------------------


---------------------- SSHD Begin ------------------------


---------------------- SSHD End -------------------------


---------------------- Sudo (secure-log) Begin ------------------------



---------------------- Sudo (secure-log) End -------------------------


---------------------- yum Begin ------------------------


Packages Installed:
   postgresql-libs-8.4.20-3.el6_6.x86_64
   postgresql-server-8.4.20-3.el6_6.x86_64
   postgresql-8.4.20-3.el6_6.x86_64
   1:mod_ssl-2.2.15-47.el6.centos.x86_64
   1:net-snmp-libs-5.5-54.el6_7.1.x86_64
   policycoreutils-python-2.0.83-24.el6.x86_64

---------------------- yum End -------------------------


---------------------- Disk Space Begin ------------------------
```

```
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/vg_root-lv_root
                       27G  3.7G   22G  15% /
/dev/sda1             485M   34M  426M   8% /boot
/dev/sdb1             488M  154M  310M  34% /BoiteAOutils


---------------------- Disk Space End -------------------------


######################## Logwatch End #########################
```

12.3. Serveur Syslog

L'architecture de rsyslog est modulaire. Pour activer son mode serveur, il faut charger le module UDP ou TCP et le mettre en écoute sur le port 514 sans oublier de relancer le service.

Activer les serveurs UDP et TCP rsyslog

```
vim /etc/rsyslog.conf
$ModLoad imudp
$UDPServerRun 514

$ModLoad imtcp
$InputTCPServerRun 514
```

```
service rsyslog restart

netstat –tapn | grep 514
udp  0  0  0.0.0.0:514  0.0.0.0:*  LISTEN 3172/rsyslog
```

Stocker les logs dans des fichiers différenciés

Les modifications a apporter au fichier /etc/rsyslog.conf sont les suivantes :

```
## Rules
$template syslog,"/var/log/%fromhost%.log"

mail.* ?syslog
```

Explications :

- **$template** : définir un template qui s'appelera **syslog**
- la variable **%fromhost%** contient le nom du client à l'origine du message

- **mail.*** correspond à tout les messages d'origine mail qui seront redirigés vers le template que nous avons appelé syslog (?syslog)

12.4. Stockage en base de données

Il est particulièrement intéressant de stocker les enregistrements syslog en base de données. Il est ensuite possible de visualiser les logs dans des interfaces web spécialisées (ici LogAnalyzer) :

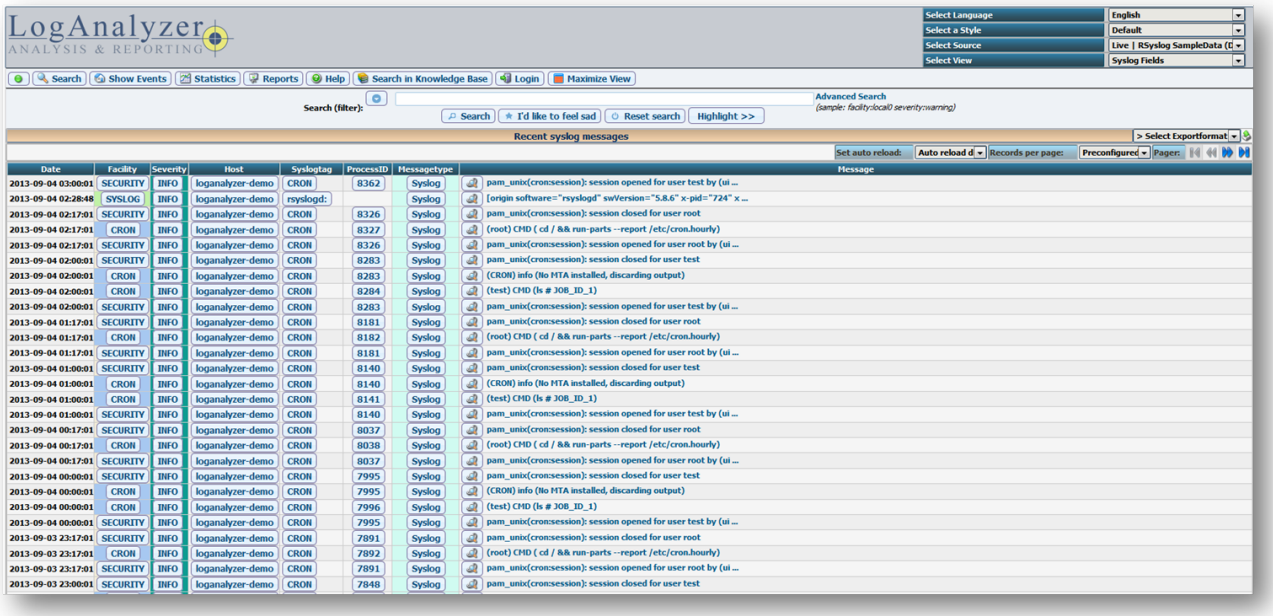


Figure 35. Interface du logiciel LogAnalyzer

Installer le module MySQL :

```
yum install rsyslog-mysql
```

Configurer le module dans /etc/rsyslog.conf :

```
$ModLoad MySQL
*.* > @IPServeur,base,USERMYSQL,PWDMYSQL
```

La création de la base MySQL sort du cadre de ce support.

```
service rsyslog restart
```

Chapitre 13. Serveur web Nginx

Nginx est un serveur web **HTTP libre sous licence BSD**. Son développement commence en Russie en 2002 par Igor Sysoev. Nginx dispose, en plus des fonctionnalités standards d'un serveur web, celles de **proxy inverse** (Reverse Proxy) pour le protocole **HTTP** mais aussi de **proxy** pour les protocoles de messageries **POP et IMAP**.

Le développement du serveur nginx est une réponse au problème **C10K** : supporter 10 000 connexions concurrentes (chose courante sur le web moderne) est un vrai challenge pour des serveurs web.

Un support commercial est possible par Nginx Inc.

13.1. Généralités

L'architecture interne du serveur permet des **performances très élevées** avec une **faible consommation de charge mémoire** en comparaison avec ce que peut faire le serveur web Apache notamment.

Les modules venant compléter les fonctions de base du noyau nginx sont liés à la compilation : ils ne peuvent pas être activés/désactivés à chaud.

Les processus serveurs sont contrôlés par un processus maître, rendant la **modification de la configuration ou la mise à jour du logiciel possible sans arrêt du service**.

Nginx détient une part de marché non négligeable de 28% sur les sites les plus chargés du marché, juste derrière Apache (41%).

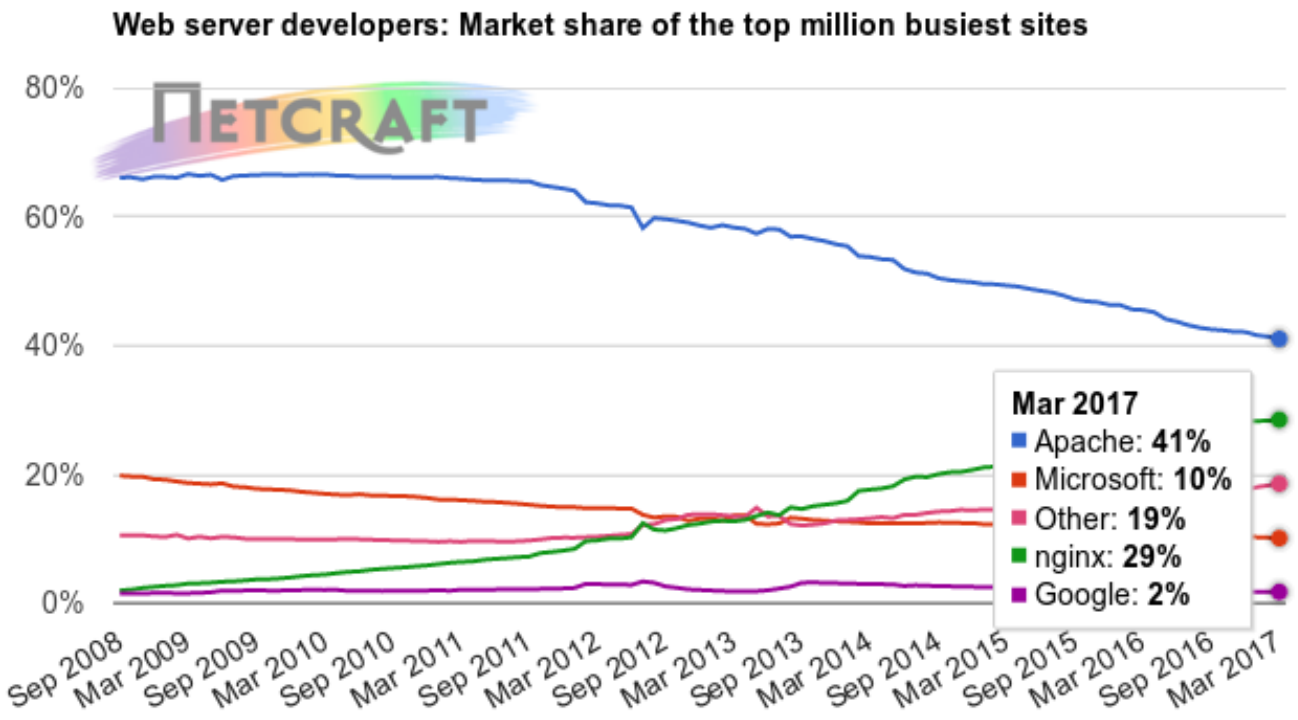


Figure 36. Statistiques NetCraft : top million busiest sites

Fonctionnalités

Nginx offre les fonctionnalités basiques suivantes :

- Hébergement de pages web statiques ;
- Génération de pages d'index automatique ;
- Proxy inverse accéléré avec cache ;
- Répartition de charge ;
- Tolérance de panne ;
- Support avec cache du FastCGI, uWSGI, SCGI et serveur de cache memcached ;
- Filtres divers pour gzip, xslt, ssi, transformation d'images, ...
- Support pour SSL/TLS et SNI ;
- Support du HTTP/2.

Autres fonctionnalités :

- Hébergement par nom ou par adresse IP ;
- Gestion du keepalive des connexions clientes ;
- Gestion des logs : syslog, rotation, buffer ;
- Ré-écriture d'URI ;

- Contrôle d'accès : par IP, mot de passe…
- Streaming FLV et MP4.

13.2. Installation du service

Nginx sous debian

Depuis Debian Wheezy, l'installation de Nginx est proposée par 3 paquets : nginx-light (le moins de modules), nginx-full (installé par le méta-paquet nginx - paquet par défaut), nginx-extras (le plus de modules).

Installation du metapaquet nginx

```
sudo apt-get install nginx
...
Les paquets supplémentaires suivants seront installés :
  libgd3 libvpx1 libxpm4 nginx-common nginx-full
Paquets suggérés :
  libgd-tools fcgiwrap nginx-doc ssl-cert
Les NOUVEAUX paquets suivants seront installés :
  libgd3 libvpx1 libxpm4 nginx nginx-common nginx-full
...
```

Nginx sous RedHat 7

Le dépôt de nginx pour RHEL/CentOS doit être ajouté aux dépôts existants. Créer le fichier /etc/yum.repos.d/nginx.repo:

```
$ sudo vi /etc/yum.repos.d/nginx.repo
```

Et ajouter le contenu suivant :

```
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/centos/7/$basearch/
gpgcheck=0
enabled=1
```

L'installation peut être lancée :

```
$ sudo yum update
$ sudo yum install nginx
```

Configuration de Nginx

La configuration de Nginx se situe sous ***/etc/nginx*** :

- Le fichier **/etc/nginx/nginx.conf** : fichier de configuration globale du serveur. Les paramètres impactent l'ensemble du serveur.
- le répertoire **sites-available** : contient les fichiers de configuration des sites.
- le répertoire **sites-enabled** : contient des liens symboliques vers les fichiers de **sites-available**, ce qui permet d'activer ou de désactiver les sites.
- le répertoire **conf.d** : répertoire contenant les paramètres communs à tous les sites.

La fonctionnalité de fichier .htaccess connues des administrateurs Apache, n'existe pas sous nginx !

Le fichier nginx.conf, épuré de tous ses commentaires, et fourni ci-dessous à titre indicatif :

Fichier nginx.conf par défaut

```
user www-data;
worker_processes 4;
pid /run/nginx.pid;

events {
    worker_connections 768;
}

http {
    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;

    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers on;

    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;

    gzip on;
    gzip_disable "msie6";

    application/javascript text/xml application/xml application/xml+rss
text/javascript;

    include /etc/nginx/conf.d/*.conf;
    include /etc/nginx/sites-enabled/*;
}
```

Table 21. Directives de la configuration par défaut

Directives	**Observations**
user	Définit l'**utilisateur** et le **groupe** propriétaires du processus. Si le groupe n'est pas spécifié, le groupe du même nom que l'utilisateur est utilisé.
worker_processes	Définit le **nombre de processus**. La valeur optimale dépend de nombreux facteurs comme le nombre de coeurs CPU, les spécificités des disques durs, etc. En cas de doute, la documentation de nginx propose comme valeur de départ le nombre équivalent au nombre de coeurs CPU disponibles (la valeur **auto** essaiera de le déterminer).

Directives	Observations
pid	Définit un fichier pour stocker la valeur du **pid**.
worker_connections	Fixe le **nombre maximum de connexions simultanées** qu'un processus worker peut ouvrir (vers le client et vers les serveurs mandatés).
tcp_nopush	tcp_nopush est indissociable de l'option sendfile. Elle permet d'**optimiser la quantité d'information envoyée en une seule fois**. Les paquet ne sont envoyés que lorsque ils ont atteints leur taille maximale.
tcp_nodelay	Activer tcp_nodelay force l'**envoi immédiat des données** contenues dans la socket, quelle que soit la taille du paquet, ce qui est le contraire de ce que fait tcp_nopush.
sendfile	Optimiser l'envoi de **fichiers statiques** (option inutile dans le cadre d'une configuration en proxy-inverse). Si **sendfile** est activée, nginx s'assure que tous les paquets soient bien remplis avant d'être envoyés au client (grâce à tcp_nopush), puis, quand arrive le dernier paquet, nginx désactive tcp_nopush, et force l'envoi des données avec tcp_nodelay.
keepalive_timeout	**temps maximum avant fermeture** d'une connexion inactive.
types_hash_max_size	Nginx entretient des tables de hashage contenant des informations statiques. Permet de définir **la taille maximale de la table de hachage**.
include	Inclure un autre fichier ou d'autres fichiers qui correspondent au modèle fourni dans la configuration.
default_type	Type MIME par défaut d'une requête.
ssl_protocols	Versions du protocole TLS acceptés.
ssl_prefer_server_ciphers	Préférer l'utilisation de la cipher suite du serveur plutôt que celle du client.
access_log	Configurer les **journaux d'accès** (voir paragraphe "gestion des logs").
error_log	Configurer les **journaux d'erreurs** (voir paragraphe "gestion des logs").
gzip	Le module **ngx_http_gzip_module** est un filtre compressant les données transmises au format gzip.
gzip_disable	Désactiver gzip en fonction d'une expression régulière.

La configuration de nginx est articulée de la manière suivante :

```
# directives globales

events {
    # configuration du worker
}

http {
    # configuration du service http

    # Configuration du premier serveur en écoute sur le port 80
    server {
        listen 80 default_server;
        listen [::]:80 default_server;
        root /var/www/html;
        index index.html index.htm index.nginx-debian.html;
        server_name _;
        location / {
            try_files $uri $uri/ =404;
        }
    }
}

mail {
    # configuration du service mail

    # directives globales du service mail

   server {
        # Un premier serveur en écoute sur le protocole pop
        listen     localhost:110;
        protocol   pop3;
        proxy      on;
   }


   server {
        # Un second serveur en écoute sur le protocole imap
       listen     localhost:143;
       protocol   imap;
       proxy      on;
   }
}
```

La configuration du premier serveur en écoute sur le port 80 se situe sous **/etc/nginx/sites-available/default**. Ce fichier est inclue au fichier nginx.conf grâce à la ligne **include /etc/nginx/sites-enabled/*;**

Configuration https

Pour configurer un service https, il faut ajouter un bloc serveur, ou modifier le bloc server existant (un bloc server peut à la fois écouter sur le port 443 et sur le port 80).

Ce bloc peut, par exemple, être ajouté au nouveau fichier *sites-available/default_https* :

```
server {
    listen              443 ssl default_server;
    ssl_protocols       TLSv1.2 TLSv1.1
    ssl_certificate     /chemin/vers/cert.pem;
    ssl_certificate_key /chemin/vers/key.key;
    root                /var/www/html;
    index               index.html index.htm index.nginx-debian.html;
    server_name         _;
    location / {
        try_files       $uri $uri/ =404;
    }
}
```

ou le server par défaut peut être modifié pour prendre en compte le https :

```
server {
    listen              80;
    listen              443 ssl;
    server_name         _;
    ssl_protocols       TLSv1.2 TLSv1.1
    ssl_certificate     /chemin/vers/cert.pem;
    ssl_certificate_key /chemin/vers/key.key;
    ...
}
```

La gestion des logs

La directive **error_log** permet de configurer les journaux d'erreurs.

Syntaxe de la directive error_log

```
error_log fichier [niveau];
```

Le premier paramètre définit un fichier qui va recevoir les logs.

Le second paramètre détermine le niveau des logs : debug, info, notice, warn, error, crit, alert ou emerg (voir le cours syslog).

L'envoi des enregistrements vers syslog peut être effectué en employant le préfixe "syslog:".

```
access_log syslog:server=192.168.1.100:5514,tag=nginx debug;
```

Nginx en proxy inverse

La fonctionnalité de proxy inverse est fourni par le module **ngx_http_upstream_module**. Il permet de définir des groupes de serveurs qui sont ensuite appelés par les directives proxy_pass ou fastcgi_pass, memcached_pass, etc.

Exemple de configuration basique, qui réparti la charge de 2/3 vers le premier serveur et d'1/3 vers le second serveur applicatif :

```
    upstream svrmetiers {
        server metiers1.formatux.fr:8080       weight=2;
        server metiers2.formatux.fr:8080       weight=1;
    }

    server {
        location / {
            proxy_pass http://svrmetiers;
        }
    }
```

Des serveurs peuvent être déclarés en secours :

```
    upstream svrmetiers {
        ...
        server secours1.formatux.fr:8080   backup;
        server secours2.formatux.fr:8080   backup;

    }
```

La directive serveur accepte de nombreux arguments :

- **max_fails=nombredetentative** : fixe le nombre de tentatives de connexion devant être en echec durant le laps de temps défini par la paramètre **fail_timeout** pour que le serveur soit considéré comme indisponible. La valeur par défaut est fixée à 1, la valeur à 0 désactive la fonctionnalité.
- **fail_timeout=time**: fixe la durée durant laquelle un nombre de connexion défini bascule le serveur comme indisponible et fixe la période de temps durant laquelle le serveur sera considéré comme indisponible. La valeur par défaut est de 10 secondes.

13.3. Sources

- https://t37.net/optimisations-nginx-bien-comprendre-sendfile-tcp-nodelay-et-tcp-nopush.html
- http://nginx.org/en/docs/

Chapitre 14. Service de cache HTTP avec Varnish

Varnish est un service de reverse-proxy-cache (mandataire inversé avec cache) HTTP, autrement dit un accélérateur de sites web.

Varnish reçoit les requêtes HTTP des visiteurs :

- s'il dispose de la réponse à la requête en cache, celle-ci est renvoyée directement au client depuis la mémoire du serveur,
- s'il ne dispose pas de la réponse, Varnish s'adresse alors au serveur web. Il lui transmet la requête, récupère la réponse, la stocke dans son cache, et répond au client.

Fournir la réponse depuis le cache en mémoire permet d'améliorer les temps de réponse aux clients. En effet, dans ce cas, il n'y a pas d'accès aux disques physiques.

Par défaut, Varnish écoute sur le port `6081` et utilise le langage VCL (*Varnish Configuration Language*) pour sa configuration. Grâce au langage VCL, il est possible de décider ce qui doit ou ne doit pas être transmit au client, ce qui doit être stocké en cache, depuis quel site et comment la réponse peut être modifiée.

Varnish est extensible par l'utilisation de modules VMOD (Varnish Modules).

14.1. Principe de fonctionnement

Dans un fonctionnement basique d'un service Web, le client communique en TCP sur le port 80 directement avec le service.

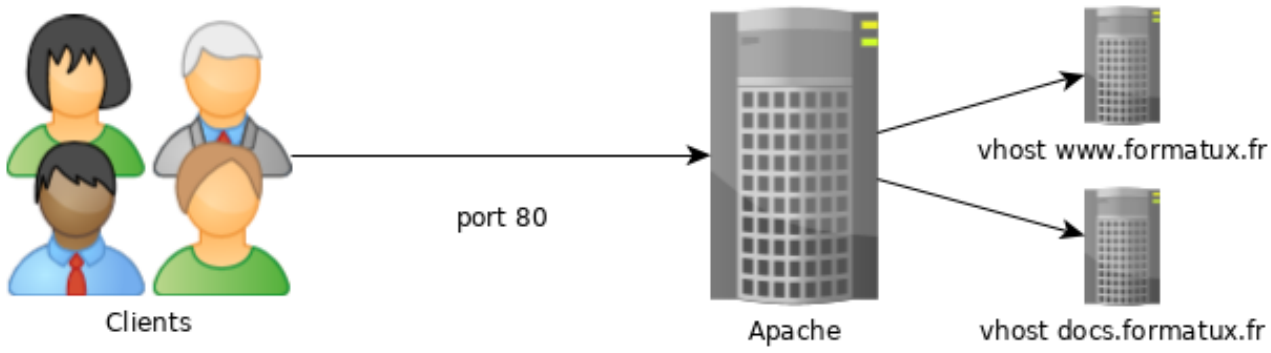


Figure 37. Fonctionnement d'un site web standard

Pour profiter du cache, le client doit communiquer avec le service web sur le port par défaut de Varnish 6081.

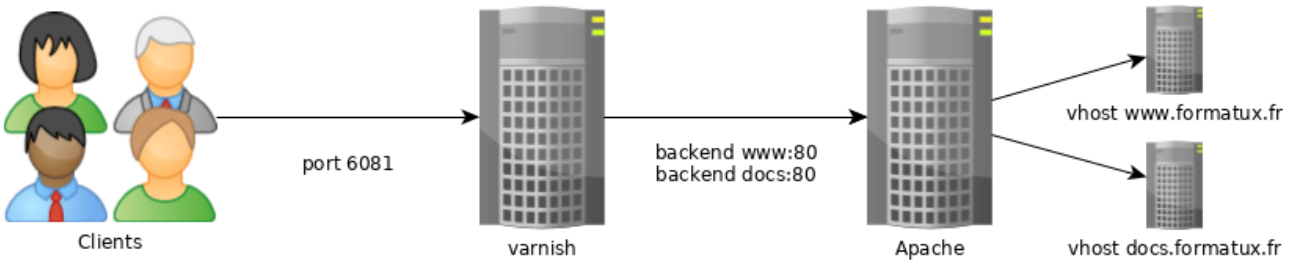


Figure 38. Fonctionnement par défaut de Varnish

Pour rendre le service transparent au client, il faudra changer le port d'écoute par défaut de Varnish et des vhosts du service web.

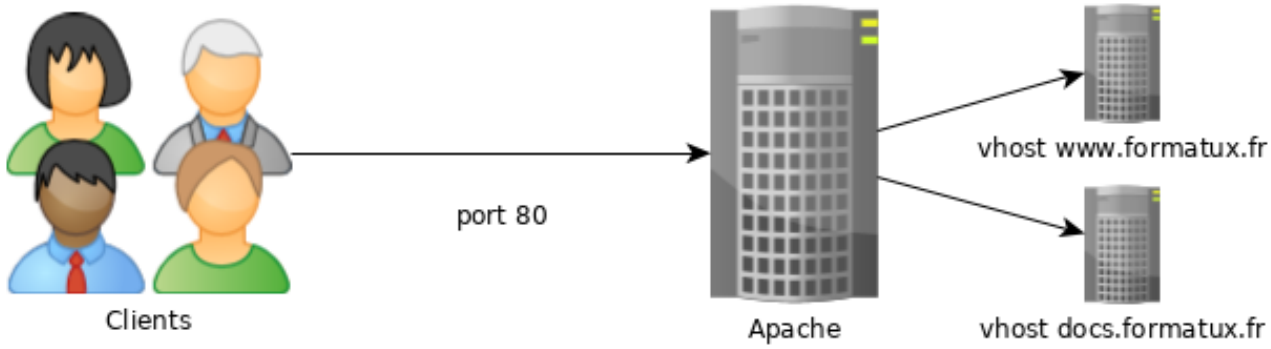


Figure 39. Mise en place transparente pour le client

14.2. Installation de Varnish

- Sous Debian :

```
# apt-get install varnish
```

- Sous RHEL :

```
# yum install varnish
```

14.3. Configuration du démon varnishd

Configuration du démon

La configuration du démon se fait dans le fichier `/etc/varnish/varnish.params` sur RedHat et dans `/etc/default/varnish` sous Debian :

1. Le fichier /etc/varnish/varnish.params sous RedHat

```
# Set this to 1 to make systemd reload try to switch VCL without restart.
RELOAD_VCL=1

# Main configuration file. You probably want to change it.
VARNISH_VCL_CONF=/etc/varnish/default.vcl

# Default address and port to bind to. Blank address means all IPv4
# and IPv6 interfaces, otherwise specify a host name, an IPv4 dotted
# quad, or an IPv6 address in brackets.
# VARNISH_LISTEN_ADDRESS=192.168.1.5
VARNISH_LISTEN_PORT=6081

# Admin interface listen address and port
VARNISH_ADMIN_LISTEN_ADDRESS=127.0.0.1
VARNISH_ADMIN_LISTEN_PORT=6082

# Shared secret file for admin interface
VARNISH_SECRET_FILE=/etc/varnish/secret

# Backend storage specification, see Storage Types in the varnishd(5)
# man page for details.
VARNISH_STORAGE="malloc,256M"

# User and group for the varnishd worker processes
VARNISH_USER=varnish
VARNISH_GROUP=varnish

# Other options, see the man page varnishd(1)
#DAEMON_OPTS="-p thread_pool_min=5 -p thread_pool_max=500 -p thread_pool_timeout=300"
```

Le fichier /etc/default/varnish sous Debian

```
DAEMON_OPTS="-a :6081 \
             -T localhost:6082 \
             -f /etc/varnish/default.vcl \
             -S /etc/varnish/secret \
             -s default,256m"
```

Depuis `systemctl`, changer les valeurs par défaut se fait en créant le fichier `/etc/systemd/system/varnish.service.d/customexec.conf` :

Le ficher `/etc/systemd/system/varnish.service.d/customexec.conf`

```
[Service]
ExecStart=
ExecStart=/usr/sbin/varnishd -a :80 -T localhost:6082 -f /etc/varnish/default.vcl -S
/etc/varnish/secret -s default,256m
```

Cela aura pour effet de surcharger les valeurs par défaut dans la partie `ExecStart` fournies avec Varnish.

Il faut ensuite relancer `systemctl daemon-reload` pour s'assurer que `systemd` prenne en compte les modifications avant de relancer Varnish.

Table 22. Les options du démon varnish

Option	Observation
`-a addr[:port]`	Ecoute les requêtes clientes sur les adresses IP et les ports spécifiés. Par défaut : toutes les adresses et sur le port `80`.
`-T addr[:port]`	Interface de gestion
`-f file`	Fichier de configuration
`-S`	Fichier contenant le secret permettant l'authentification sur le port de gestion
`-s`	Spécifier un backend de stockage du cache. L'option peut être spécifiée plusieurs fois. Les types de stockage possibles sont `malloc` (cache en mémoire puis si besoin dans la swap), ou `file` (création d'un fichier sur le disque puis mapping en mémoire). Les tailles sont exprimées en K/M/G/T (kilobytes, megabytes, ...)
`-C`	Compile la VCL en langage C et l'affiche à l'écran.

Test de la configuration et relance

Il est conseillé de vérifier la syntaxe de la VCL avant de relancer le démon varnishd.

Il s'agit de compiler en langage C le fichier de configuration VCL. Le service peut être redémarré si la compilation fonctionne et qu'aucune alarme n'est affichée.

Vérification de la syntaxe varnishd

```
varnishd -C -f /etc/varnish/default.vcl
```

ou utilisation du script init :

```
# /etc/init.d/varnish configtest
[...]
    .hit_func = VGC_function_vcl_hit,
    .init_func = VGC_function_vcl_init,
    .miss_func = VGC_function_vcl_miss,
    .pass_func = VGC_function_vcl_pass,
    .pipe_func = VGC_function_vcl_pipe,
    .purge_func = VGC_function_vcl_purge,
    .recv_func = VGC_function_vcl_recv,
    .synth_func = VGC_function_vcl_synth,
};
```

Puis purge du cache et rechargement de la configuration : (si `RELOAD_VCL=1`) :

Rechargement de la configuration

```
systemctl reload varnishd
```

ou

Redémarrage complet

```
systemctl restart varnishd
```

Il est possible de vérifier qu'une page provient du cache varnish depuis les en-têtes de la réponse HTTP :

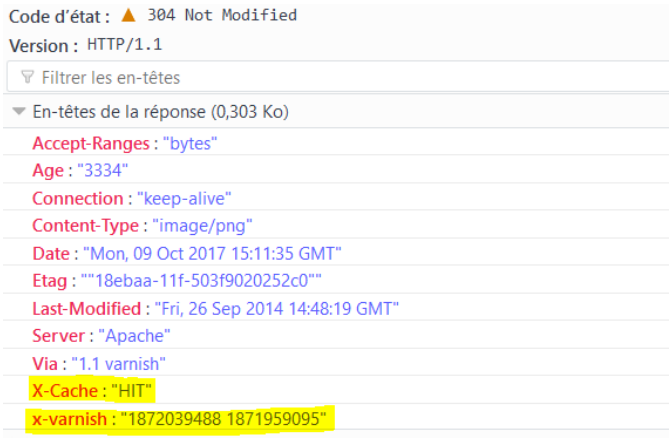


Figure 40. En-tête varnish et cache hit

14.4. La langage VCL

Les sous-routines

Varnish utilise des fichiers VCL, segmentés en sous-routines comportant les actions a exécuter. Ces

sous-routines sont exécutées uniquement dans les cas spécifiques qu'elles définissent. Dans le fichier par défaut `/etc/varnish/default.vcl`, nous trouvons les routines `vcl_recv`, `vcl_backend_response` et `vcl_deliver`.

```
# vim /etc/varnish/default.vcl
cat /etc/varnish/default.vcl
#
# This is an example VCL file for Varnish.
#
# It does not do anything by default, delegating control to the
# builtin VCL. The builtin VCL is called when there is no explicit
# return statement.
#
# See the VCL chapters in the Users Guide at https://www.varnish-cache.org/docs/
# and http://varnish-cache.org/trac/wiki/VCLExamples for more examples.

# Marker to tell the VCL compiler that this VCL has been adapted to the
# new 4.0 format.
vcl 4.0;

# Default backend definition. Set this to point to your content server.
backend default {
    .host = "127.0.0.1";
    .port = "8080";
}

sub vcl_recv {

}

sub vcl_backend_response {

}

sub vcl_deliver {

}
```

Table 23. Les sous-routines VCL

Sous-routine	Action
`vcl_recv`	Cette routine est appelée **avant** l'envoi de la requête vers le backend. Dans cette routine, il est possible de modifier les en-têtes HTTP, cookies, choisir le backend, etc. Voir actions `set req.`.
`vcl_backend_response`	Cette routine est appelée **après** la réception de la réponse du backend (`beresp` = BackEnd RESPonse). Voir actions `set bereq.` et `set beresp.`.

Sous-routine	Action
`vcl_deliver`	Cette routine est utile pour modifier la sortie de Varnish. Si besoin de modifier l'objet final (ajouter ou supprimer un en-tête, ...), il est possible de le faire dans `vcl_deliver`.

Les opérateurs VCL

Table 24. Les opérateurs VCL

Opérateur	Action
`=`	assignation
`==`	comparaison
`~`	comparaison en association avec une expression régulière et des ACL
`!`	négation
`&&`	et logique
`||`	ou logique

Les objets Varnish

`beresp` = BackEnd RESPonse

Table 25. Les différents objets Varnish

Objet	Observation
`req`	l'objet requête. Quand Varnish reçoit la requête, `req` est créé. La plupart du travail dans la sous-routine `vcl_recv` touche à cet objet.
`bereq`	l'objet requête à destination du serveur web. Varnish créé cet objet à partir de `req`.
`beresp`	l'objet réponse du serveur web. Il contient les entêtes de l'objet en provenance de l'application. Il est possible de modifier la réponse du serveur dans la sous-routine `vcl_backend_response`.
`resp`	la réponse HTTP qui va être envoyée au client. Cet objet est modifié dans la sous-routine `vcl_deliver`.
`obj`	l'objet tel que sauvegardé en cache. En lecture seule.

Les actions varnish

Table 26. Les actions les plus fréquentes

Action	Observation
`pass`	Quand `pass` est retourné, la requête et la réponse qui en suivront viendront du serveur d'application. Il n'y aura pas de cache appliqué. `pass` est retourné depuis la sous-routine `vcl_recv`.

Action	Observation
`hash`	Quand `hash` est retourné depuis `vcl_recv`, Varnish servira le contenu depuis le cache même si la requête est configurée pour passer sans cache.
`pipe`	Cette action sert à gérer les flux. Varnish dans ce cas n'inspectera plus chaque requête mais laisse passer tous les bytes au serveur. `pipe` est utilisé par exemple par les websockets ou la gestion des flux vidéos.
`deliver`	Sert l'objet au client. En général depuis la sous-routine `vcl_backend_response`.
`restart`	Recommence le processus de traitement de la requête. Les modifications de l'objet req sont conservées.
`retry`	La requête est de nouveau transférée au serveur d'application. Utilisé depuis `vcl_backend_response` ou `vcl_backend_error` si la réponse de l'application n'est pas satisfaisante.

En résumé, les interactions possibles entre les sous-routines et les actions sont illustrées dans le schéma ci-dessous :

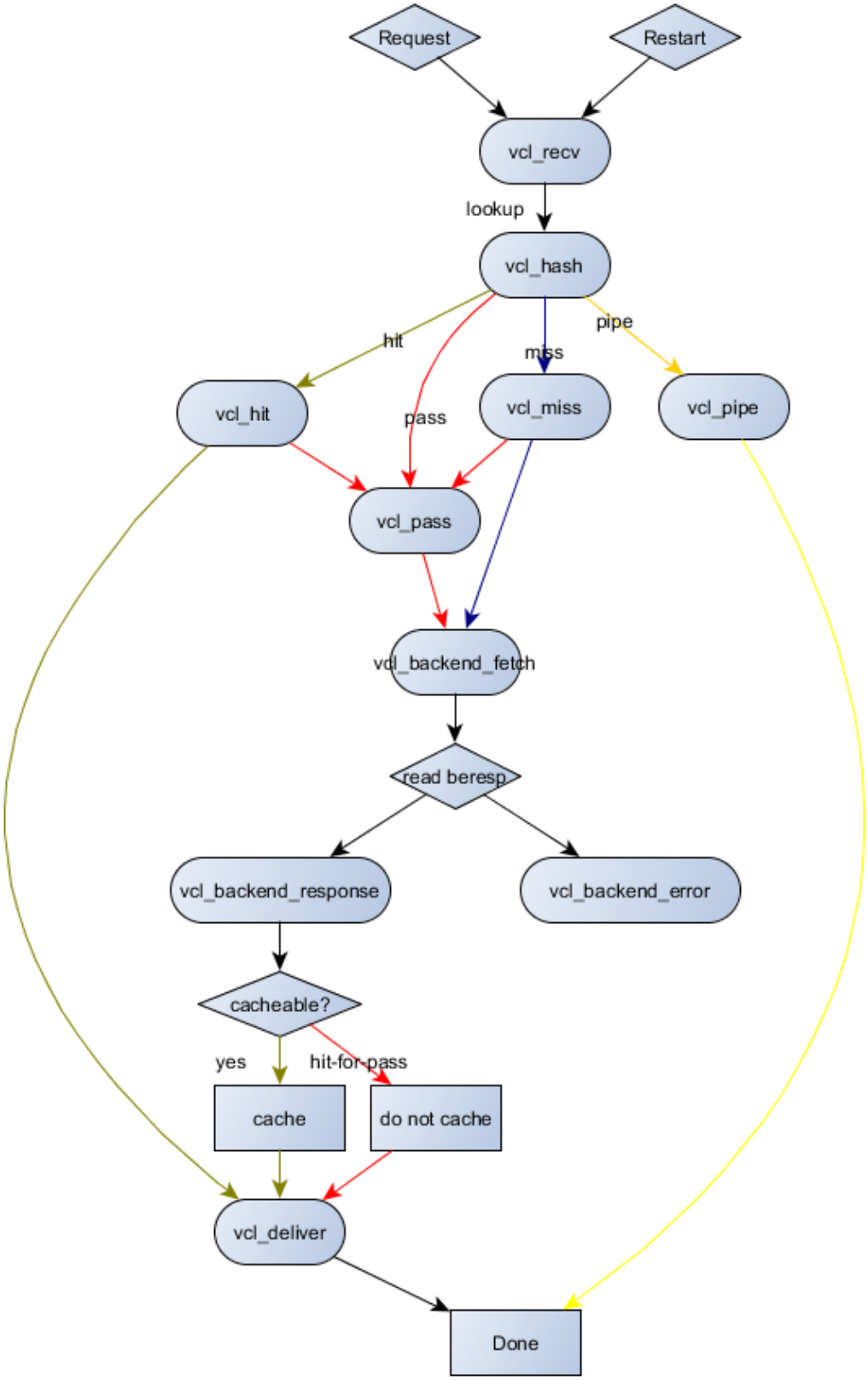


Figure 41. Fonctionnement simplifié de varnish

14.5. Configuration des backends

Varnish utilise le terme backend pour les vhosts qu'il doit mandater.

Plusieurs backend peuvent être défini sur le même serveur Varnish.

La configuration des backends se fait dans le fichier `/etc/varnish/default.vcl`.

Gestion des ACL

```
# ACL de deny
acl deny {
"10.10.0.10"/32;
"192.168.1.0"/24;
}
```

Appliquer l'ACL :

```
# Bloquer les IP de l'ACL deny
if (client.ip ~ forbidden) {
  error 403 "Acces interdit";
}
```

Ne pas cacher certaines pages :

```
# Ne pas mettre en cache les pages login et admin
if (req.url ~ "/(login|admin)") {
  return (pass);
}
```

Paramètres POST et cookies

Varnish ne met jamais en cache les requêtes HTTP de type POST ou celle contenant des cookies (qu'ils proviennent du client ou du backend).

Si le backend utilise des cookies, alors aucun contenu ne sera mis en cache.

Pour corriger ce comportement, il est possible de déréferrencer les cookies de nos requêtes :

```
sub vcl_recv {
    unset req.http.cookie;
}

sub vcl_backend_response {
    unset beresp.http.set-cookie;
}
```

Répartir les requêtes sur différents backend

Dans le cas de l'hébergement de plusieurs sites, comme par exemple un serveur de document (doc.formatux.fr) et un blog (blog.formatux.fr), il est possible de répartir les requêtes vers le bon backend.

Déclaration des backends

```
backend docs {
    .host = "127.0.0.1";
    .port = "8080";
}

backend blog {
    .host = "127.0.0.1";
    .port = "8081";
}
```

L'objet `req.backend` est modifié en fonction de l'hôte appelé dans la requête HTTP dans la sous-routine `vcl_recv` :

Sélection du backend

```
sub vcl_recv {
    if (req.http.host ~ "^doc.formatux.fr$") {
        set req.backend = nginx;
    }

    if (req.http.host ~ "^blog.formatux.fr$") {
        set req.backend = ghost;
    }
}
```

Répartir la charge

Varnish est capable de gérer la répartition de charge via des backends spécifiques appelés directors.

- Le director `round-robin` distribue les requêtes aux backends en round-robin (alternativement). Il est possible d'affecter une pondération à chaque backend.
- Le director `client` distribue les requêtes en fonction d'une affinité de session (sticky session) sur n'importe quel élément de l'en-tête (par exemple avec un cookie de session). Un client est dans ce cas toujours renvoyé vers le même backend.

Déclaration des backends

```
backend docs1 {
    .host = "10.10.11.10";
    .port = "8080";
}

backend docs2 {
    .host = "10.10.11.11";
    .port = "8080";
}
```

Le `director` permet d'associer les 2 backends définis :

Déclaration du director

```
director docs_director round-robin {
    { .backend = docs1; }
    { .backend = docs2; }
}
```

Reste à définir le `director` comme backend aux requêtes :

Association des requêtes au director

```
sub vcl_recv {
    set req.backend = docs_director;
}
```

14.6. Configuration du système

Configuration du port 80 et 443

Modifier, sous debian le fichier `/etc/default/varnish` :

```
DAEMON_OPTS="-a :80
-T localhost:6082
-f /etc/varnish/default.vcl
-S /etc/varnish/secret
...
```

Configuration d'un cache

Configuration d'un cache sur disque d'1G.

Modifier, sous debian le fichier `/etc/default/varnish` :

```
DAEMON_OPTS="-a :80
-T localhost:6082
-f /etc/varnish/default.vcl
-S /etc/varnish/secret
...
-s file,/var/lib/varnish/$INSTANCE/varnish_storage.bin,1G"
```

Adaptation d'Apache

Changement de ports réseau

Si le service Varnish est localisé sur le même serveur que le service Web (Apache ou Nginx), les deux services ne pourront plus écouter en même temps les ports par défaut 80 et 443.

Dans ce cas, il est d'usage de faire écouter le service web sur un port 8080, 8081, 8082 etc. et de laisser le port par défaut à Varnish.

```
#Listen 80
Listen 8080
```

Modification du LogFormat

Le service http étant reverse proxifié, le serveur web n'aura plus accès aux adresses IP du client mais à celui du service Varnish.

Pour prendre en compte le reverse proxy dans les logs Apache, modifier dans le fichier de configuration du serveur le format du journal d'évènement :

```
LogFormat "%{X-Forwarded-For}i %l %u %t "%r" %>s %b "%{Referer}i" "%{User-Agent}i""
varnishcombined
```

et prendre en compte ce nouveau format dans le vhost du site web :

```
CustomLog /var/log/httpd/www-access.log.formatux.fr varnishcombined
```

et rendre Varnish compatible :

```
# Compatibility with Apache format log

if (req.restarts == 0) {
  if (req.http.x-forwarded-for) {
    set req.http.X-Forwarded-For = req.http.X-Forwarded-For + ", " + client.ip;
  } else {
   set req.http.X-Forwarded-For = client.ip;
  }
}
```

14.7. Purge du cache

Le cache peut être purgé :

- en ligne de commande :

```
# varnishadm 'ban req.url ~ .'
```

- en utilisant un secret et un port différent du port par défaut :

```
# varnishadm -S /etc/varnish/secret -T 127.0.0.1:6082 'ban req.url ~ .'
```

- avec l'interface CLI:

```
# varnishadm

varnish> ban req.url ~ ".css$"
200

varnish> ban req.http.host == docs.formatux.fr
200

varnish> ban req.http.host ~ .
200
```

- via une requête HTTP PURGE :

```
$ curl -X PURGE http://www.example.org/foo.txt
```

Varnish doit être configuré pour accepter cette requête :

```
acl local {
    "localhost";
    "10.10.1.50";
}

sub vcl_recv {
    # directive a placer en premier,
    # sinon une autre directive risque de matcher avant
    # et la purge ne sera jamais effectuée
    if (req.method == "PURGE") {
        if (client.ip ~ local) {
            return(purge);
        }
    }
}
```

14.8. La gestion des journaux

Varnish écrit ses journaux en mémoire et en binaire afin de ne pas pénaliser ses performances. Lorsqu'il ne dispose plus d'espace en mémoire, il réécrit les nouveaux enregistrements par dessus les anciens, en repartant du début de son espace mémoire.

Les journaux peuvent être consultés avec les outils varnishstat (statistiques), varnishtop (top pour Varnish), varnishlog (journalisation verbeuse) ou varnishnsca (journaux au format NCSA comme Apache) :

```
# varnishstat
# varnishtop -i ReqURL
# varnishlog
# varnishnsca
```

Filtrer les journaux

L'option `-q` permet d'appliquer des filtres sur les journaux via les commandes précédentes :

```
# varnishlog -q 'TxHeader eq MISS' -q "ReqHeader ~ '^Host: formatux\.fr$'"
# varnishncsa -q "ReqHeader eq 'X-Cache: MISS'"
```

Enregistrer les journaux sur disques

L'enregistrement des journaux sur disque est effectué par les démons `varnishlog` et `varnishnsca` en toute indépendance du démon varnishd. Le démon varnishd continue a renseigner ses journaux en mémoire sans pénaliser ses performances vers les clients, puis les autres démons se charge de

copier les enregistrements sur disque.

14.9. Commandes Varnish

- `varnishlog`: Affichage du log du daemon Varnish.
- `varnishstat`: Affichage des statistiques d'utilisation de Varnish.
- `varnishhist`: Affiche un historique sous forme de graphe des requêtes faites à votre serveur Varnish.
- `varnishadm`: une interface d'administration locale de Varnish

14.10. Sources

- https://wooster.checkmy.ws/2014/03/varnish-performance-cache/
- https://wiki.evolix.org/HowtoVarnish

Chapitre 15. PHP-FPM

PHP-FPM (FastCGI Process Manager) est intégré à PHP depuis sa version 5.3.3. La version FastCGI de php apporte des fonctionnalités complémentaires.

15.1. Généralités

CGI (Common Gateway Interface) et **FastCGI** permettent la communication entre le serveur Web (Apache, Nginx) et un langage de développement (Php, Python, Java) :

- Dans le cas du **CGI**, chaque requête entraîne la création d'un **nouveau processus**, ce qui n'est pas efficace en terme de performance.
- **FastCGI** s'appuie, quant à lui, sur un **certain nombre de processus** pour le traitement de ses requêtes clientes.

PHP-FPM, apporte **en plus des meilleurs performances** :

- La possibilité de **mieux cloisonner les applications** : lancement des processus avec des uid/gid différents, avec des fichiers php.ini personnalisés,
- La gestion des statistiques,
- Gestion des journaux,
- Gestion dynamique des processus et redémarrage sans coupure de service ('graceful').

Apache possédant un module php, l'utilisation de php-fpm est moins intéressante que pour le serveur Nginx.

15.2. Installation

Debian 8

L'installation de php-fpm s'effectue depuis les dépôts apt :

```
$ sudo apt-get install php5-fpm
...
Les paquets supplémentaires suivants seront installés :
  libapparmor1 libonig2 libqdbm14 php5-cli php5-common php5-json php5-readline
Paquets suggérés :
  php-pear php5-user-cache
...
```

Arrêt et relance du service

Via systemd, la commande suivante stoppe le service :

```
$ sudo systemctl stop php5-fpm
```

La commande suivante relance le service :

```
$ sudo systemctl restart php5-fpm
```

Pour simplement recharger la configuration et prendre les modifications effectuées en compte :

```
$ sudo systemctl reload php5-fpm
```

15.3. Configuration

Les fichiers de configuration de php-fpm se situent sous ***/etc/php5/fpm***.

php-fpm utilise la syntaxe de php.ini pour ses fichiers de configuration (php-fpm.conf et fichier de configuration des pools).

Le fichier **/etc/php5/fpm/php-fpm.conf**, dans sa version minimale, contient :

```
[global]
pid = /run/php5-fpm.pid
error_log = /var/log/php5-fpm.log


include=/etc/php5/fpm/pool.d/*.conf
```

Le fichier ***/etc/php5/fpm/pool.d/www.conf*** contient, quant à lui, les quelques directives suivantes :

```
[www]
user = www-data
group = www-data
listen = /var/run/php5-fpm.sock
listen.owner = www-data
listen.group = www-data

pm = dynamic
pm.max_children = 5
pm.start_servers = 2
pm.min_spare_servers = 1
pm.max_spare_servers = 3

chdir = /
```

Table 27. Directives de la configuration par défaut

Directives	**Observations**
[pool]	Nom du pool de processus. Le fichier de configuration peut être composé de plusieurs pools de processus (le nom du pool entre crochet commence une nouvelle section)
listen	Définit l'interface d'écoute ou le socket unix utilisé. Exemple : **listen = 127.0.0.1:9000** Ou via une socket Unix : **listen = /var/run/php5-fpm.sock**. L'utilisation d'une socket lorsque le serveur web et le serveur php sont sur la même machine permet de s'affranchir de la couche TCP/IP.
Pour une interface : listen.owner, listen.group, listen.mode	Spécifier le propriétaire, le groupe propriétaire et les droits de la socket Unix. Attention : les deux serveurs (web et php) doivent disposer des droits d'accès sur la socket.
Pour une socket : listen.allowed_clients	restreindre l'accès au serveur php à certaines adresses IP. Exemple : **listen.allowed_clients = 127.0.0.1**

Configuration statique ou dynamique

Les processus de php-fpm peuvent être gérés de manière statique ou dynamique :

- En mode **static** : le nombre de processus fils est fixé par la valeur de pm.max_children ;

Configuration de php-fpm en mode static

```
pm = static
pm.max_children = 10
```

Cette configuration lancera 10 processus.

- En mode **dynamic** : php-fpm lancera au maximum le nombre de processus spécifié par la valeur de pm.max_children, en commençant par lancer un nombre de processus correspondant à pm.start_servers, et en gardant au minimum la valeur de pm.min_spare_servers de processus inactifs et au maximum pm.max_spare_servers processus inactifs.

Exemple :

```
pm                  = dynamic
pm.max_children     =  5
pm.start_servers    =  2
pm.min_spare_servers =  1
pm.max_spare_servers =  3
```

Php-fpm créera un nouveau processus en remplacement d'un processus qui aura traité un nombre de requêtes équivalent à pm.max_requests.

Par défaut, la valeur de pm.max_requests est à 0, ce qui signifie que les processus ne sont jamais recyclés. Utiliser l'option pm.max_requests peut être intéressant pour des applications présentant des fuites mémoires.

Configuration avancée

Status du processus

Php-fpm propose, à l'instar de Apache et de son module mod_status, une page indiquant l'état du processus.

Pour activer la page, il faudra fournir à nginx son chemin d'accès via la directive pm.status_path :

```
pm.status_path = /status
```

Journaliser les requêtes longues

La directive slowlog indique le fichier recevant la journalisation des requêtes trop longues (dont le temps dépasse la valeur de la directive request_slowlog_timeout).

Le fichier généré se situe par défaut ***/var/log/php5-fpm.log.slow***.

```
request_slowlog_timeout = 30
slowlog = /var/log/php5-fpm.log.slow
```

Une valeur à 0 de request_slowlog_timeout désactive la journalisation.

Configuration avec nginx

Le paramètrage par défaut de nginx intègre déjà la configuration nécessaire pour faire fonctionner php avec php-fpm.

Le fichier de configuration fastcgi.conf (ou fastcgi_params) se situe sous /**etc/nginx/** :

```
fastcgi_param  SCRIPT_FILENAME    $document_root$fastcgi_script_name;
fastcgi_param  QUERY_STRING       $query_string;
fastcgi_param  REQUEST_METHOD     $request_method;
fastcgi_param  CONTENT_TYPE       $content_type;
fastcgi_param  CONTENT_LENGTH     $content_length;

fastcgi_param  SCRIPT_NAME        $fastcgi_script_name;
fastcgi_param  REQUEST_URI        $request_uri;
fastcgi_param  DOCUMENT_URI       $document_uri;
fastcgi_param  DOCUMENT_ROOT      $document_root;
fastcgi_param  SERVER_PROTOCOL    $server_protocol;
fastcgi_param  HTTPS              $https if_not_empty;

fastcgi_param  GATEWAY_INTERFACE  CGI/1.1;
fastcgi_param  SERVER_SOFTWARE    nginx/$nginx_version;

fastcgi_param  REMOTE_ADDR        $remote_addr;
fastcgi_param  REMOTE_PORT        $remote_port;
fastcgi_param  SERVER_ADDR        $server_addr;
fastcgi_param  SERVER_PORT        $server_port;
fastcgi_param  SERVER_NAME        $server_name;

# PHP only, required if PHP was built with --enable-force-cgi-redirect
fastcgi_param  REDIRECT_STATUS    200;
```

Pour que nginx traite les fichiers .php, les directives suivantes doivent être ajoutées au fichier de configuration du site :

- Si php-fpm écoute sur le port 9000 :

```
location ~ \.php$ {
  include /etc/nginx/fastcgi_params;
  fastcgi_pass  127.0.0.1:9000;
}
```

- Si php-fpm écoute sur une socket unix :

```
location ~ \.php$ {
  include /etc/nginx/fastcgi_params;
  fastcgi_pass  unix:/var/run/php5-fpm.sock;
}
```

Chapitre 16. Serveur de base de données MySQL - MariaDB

MySQL et **MariaDB** sont des systèmes de gestion de base de données relationnelles (**SGBD-R**) open source.

La littérature anglophone utilise le terme RDBMS (Relational DataBase Managed System).

MySQL a été développé par **Michael "Monty" Widenius** (informaticien Finlandais), qui fonde MySQL AB en 1995. MySQL AB est rachetée par la société **SUN** en 2008, elle-même rachetée par **Oracle** en en 2009 qui est toujours propriétaire du logiciel MySQL et qui le distribue sous double licence GPL et propriétaire.

En 2009, Michael Widenius quitte SUN, fonde la société Monty Program AB et lance le développement de son fork communautaire de MySQL : **MariaDB** sous licence GPL. La gouvernance du projet est confiée à la fondation MariaDB, ce qui assure au projet de rester libre.

Rapidement, la majorité des distributions Linux proposent par défaut les paquets MariaDB à la place de ceux de MySQL et des grands comptes, comme Wikipedia ou Google, adoptent eux aussi le fork communautaire.

MySQL et MariaDB font parti des SGBR-R les plus utilisés au monde (professionnellement et par le grand public), notamment pour des applications web (**LAMP** : Linux+Apache+Mysql-MariaDB+Php).

Les concurrents principaux de Mysql-MariaDB sont :

- Oracle DB,
- Microsoft SQL Server.

Les services applicatifs sont multi-threads et multi-utilisateurs, ils fonctionnent sur la majorité des systèmes d'exploitations (Linux, Unix, BSD, Mac OSx, Windows) et sont accessibles depuis de nombreux langages de programmation (Php, Java, Python, C, C++, Perl, ...).

Plusieurs moteurs sont supportés, ce qui permet, au sein d'une même base, d'attribuer un moteur différent aux différentes tables en fonction du besoin :

- **MyISAM** : le plus simple mais ne supporte pas les transactions ni les clefs étrangères. Il s'agit d'un moteur séquentiel indexé.
- **InnoDB** : gestion de l'intégrité des tables (clé étrangères et transactions) mais occupe plus d'espace sur le disque. C'est le moteur **par défaut** depuis la version 5.6 de MySQL. Il s'agit d'un moteur transactionnel.
- **Memory** : les tables sont stockées en mémoire.
- **Archive** : compression des données à l'insertion ce qui fait gagner de l'espace disque mais

ralenti les requêtes de recherche (données froides).

- ...

Il s'agira d'adopter un moteur selon le besoin : Archive pour le stockage de log, Memory pour des données temporaires, etc.

MySQL utilise la port **3306/tcp** pour sa communication sur le réseau.

16.1. Installation

Sous CentOS

Le serveur MySQL est installé par le paquet **mysql-community-server**, mais le paquet distribué dans les dépôts de base de la distribution est le paquet mariadb-server :

```
$ yum search mysql-community-server mariadb-server

mariadb-server.x86_64 : The MariaDB server and related files

  Name and summary matches mostly, use "search all" for everything.
Warning: No matches found for: mysql-community-server
```

L'installation de mariadb-server peut donc se faire directement :

```
$ sudo yum install mariadb-server
$ sudo systemctl enable mariadb
Created symlink from /etc/systemd/system/multi-user.target.wants/mariadb.service to
/usr/lib/systemd/system/mariadb.service.
$ sudo systemctl start mariadb
```

L'installation ajoute au système un utilisateur :

```
$ cat /etc/passwd
...
mysql:x:27:27:MariaDB Server:/var/lib/mysql:/sbin/nologin
...
```

Pour installer le paquet mysql-server, il faudra passer par l'ajout du dépôt MySQL. Ce dépôt fourni les dernières versions des paquets logiciels pour les applications :

- Serveur MySQL,
- MySQL Cluster
- MySQL Workbench

- MySQL Fabric
- MySQL Router
- MySQL Utilities
- MySQL Connector / ODBC
- MySQL Connector / Python
- MySQL Shell

Les dépôts MySQL se situent sous : http://repo.mysql.com/.

- Pour RHEL 7

Installer le rpm contenant la définition du dépôt :

```
$ sudo yum install http://repo.mysql.com/mysql-community-release-el7.rpm
```

L'installation de ce RPM a pour effet de créer le fichier */etc/yum.repos.d/mysql-community.repo* :

Le fichier de dépôts /etc/yum.repos.d/mysql-community.repo

```
[mysql-connectors-community]
name=MySQL Connectors Community
baseurl=http://repo.mysql.com/yum/mysql-connectors-community/el/7/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql

[mysql-tools-community]
name=MySQL Tools Community
baseurl=http://repo.mysql.com/yum/mysql-tools-community/el/7/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql

# Enable to use MySQL 5.5
[mysql55-community]
name=MySQL 5.5 Community Server
baseurl=http://repo.mysql.com/yum/mysql-5.5-community/el/7/$basearch/
enabled=0
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql

# Enable to use MySQL 5.6
[mysql56-community]
name=MySQL 5.6 Community Server
baseurl=http://repo.mysql.com/yum/mysql-5.6-community/el/7/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql

[mysql57-community]
name=MySQL 5.7 Community Server
baseurl=http://repo.mysql.com/yum/mysql-5.7-community/el/7/$basearch/
enabled=0
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql
```

Comme vous pouvez le constater, le dépôt activé par défaut et le dépôt contenant la version 5.6 du paquet. Pour installer la version 5.7 du serveur, il faudra activer le dépôt correspondant et désactiver les dépôts des autres versions.

```
$ sudo yum install mysql-community-server
$ sudo systemctl enable mysqld
$ sudo systemctl start mysqld
```

L'utilisateur sera cette fois-ci quelque peu différent (uniquement au niveau du commentaire et du shell) :

```
$ cat /etc/passwd
...
mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/false
...
```

Si le serveur Mariadb est déjà installé, celui-ci sera remplacé par le serveur MySQL !

Sous Debian

Sous Debian, les 2 paquets mariadb-server et mysql-server sont disponibles dans les dépôts de base. Il n'est donc pas nécessaire d'ajouter le dépôt officiel.

```
$ sudo apt-cache search mysql-server
mysql-server - MySQL database server (metapackage depending on the latest version)
```

```
$ sudo apt-cache search mariadb-server
mariadb-server - MariaDB database server (metapackage depending on the latest version)
mariadb-server-10.0 - MariaDB database server binaries
```

Installer au choix un des deux paquets (selon vos préférences) :

```
$ sudo apt-get install mariadb-server
```

Un utilisateur est également créé :

```
cat /etc/passwd
...
mysql:x:112:116:MySQL Server,,,:/nonexistent:/bin/false
...
```

A l'installation du paquet mysql-server, l'installateur demande la saisie d'un mot de passe pour l'utilisateur root@localhost.

16.2. Gestion du service

Le nom de l'unité cible systemd est différente selon la distribution et le paquet :

- sous RedHat :
  - mysqld pour mysql
  - mariadb
- sous debian :
  - mysql
  - mariadb

16.3. Sécurisation

- Depuis la version 5.7 :

Au démarrage du serveur, une bi-clef ssl est générée dans le dossier de données, le plugin validate_password est installé puis activé. Un compte utilisateur '**root**' est créé et son mot de passe est stocké dans le fichier journal de log.

Extrait du fichier /var/log/mysqld.log

```
[Note] A temporary password is generated for root@localhost: akTjtLSPq6/5
```

Le plugin validate_password nécessite que le mot de passe soit long d'au moins 8 caractères et qu'il soit composé d'au moins une majuscule, une minuscule, un entier et un caractère spécial.

Ce mot de passe peut immédiatement être changé avec les commandes suivantes :

```
$ mysql -uroot -p
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'M1nNouve@uMDP';
```

Ne pas lancer la commande mysql_secure_installation pour un serveur en version 5.7 (la procédure est effectuée automatiquement au moment de l'installation).

- Avec la version 5.6 et précédente :

Le programme mysql_secure_installation effectue des opérations importantes comme attribuer un mot de passe à l'utilisateur root, supprimer les utilisateurs anonymes, etc.

Pour des raisons évidentes de sécurité, la commande mysql_secure_installation devrait toujours être lancée après l'installation :

```
$ mysql_secure_installation
```

16.4. Configuration

Le fichier ***/etc/my.cnf*** (redhat) ou ***/etc/mysql/my.cnf*** (debian) contient la configuration à la fois du client et du serveur.

Le fichier /etc/my.cf par défaut

```
cat /etc/my.cnf

[mysqld]
#
# Remove leading # and set to the amount of RAM for the most important data
# cache in MySQL. Start at 70% of total RAM for dedicated server, else 10%.
# innodb_buffer_pool_size = 128M
#
# Remove leading # to turn on a very important data integrity option: logging
# changes to the binary log between backups.
# log_bin
#
# Remove leading # to set options mainly useful for reporting servers.
# The server defaults are faster for transactions and fast SELECTs.
# Adjust sizes as needed, experiment to find the optimal values.
# join_buffer_size = 128M
# sort_buffer_size = 2M
# read_rnd_buffer_size = 2M
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock

# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0

log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

Le fichier /etc/mysql/my.cnf

```
# The MariaDB configuration file
#
# The MariaDB/MySQL tools read configuration files in the following order:
# 1. "/etc/mysql/mariadb.cnf" (this file) to set global defaults,
# 2. "/etc/mysql/conf.d/*.cnf" to set global options.
# 3. "/etc/mysql/mariadb.conf.d/*.cnf" to set MariaDB-only options.
# 4. "~/.my.cnf" to set user-specific options.
#
# If the same option is defined multiple times, the last one will apply.
#
# One can use all long options that the program supports.
# Run program with --help to get a list of available options and with
# --print-defaults to see which it would actually understand and use.

#
# This group is read both both by the client and the server
# use it for options that affect everything
#
[client-server]

# Import all .cnf files from configuration directory
!includedir /etc/mysql/conf.d/
!includedir /etc/mysql/mariadb.conf.d/
```

Alors que le serveur mysqld est en cours de fonctionnement, la commande *ps* nous renseigne sur les paramètres en cours :

```
ps -ef | grep mysql
root      3324     1  0 08:27 ?        00:00:00 /bin/bash /usr/bin/mysqld_safe
mysql     3468  3324  0 08:27 ?        00:00:01 /usr/sbin/mysqld --basedir=/usr
--datadir=/var/lib/mysql --plugin-dir=/usr/lib/mysql/plugin --user=mysql --skip-log
-error --pid-file=/var/run/mysqld/mysqld.pid --socket=/var/run/mysqld/mysqld.sock
--port=3306
root      3469  3324  0 08:27 ?        00:00:00 logger -t mysqld -p daemon error
```

Le processus **mysqld_safe** est le responsable du démarrage du serveur. La majorité des options de mysqld_safe sont les mêmes que celles de mysqld.

La démon **mysqld** est démarré avec les options suivantes :

- **--basedir=/usr** : chemin vers l'installation de MySQL.
- **--datadir=/var/lib/mysql** : chemin vers le dossier contenant les bases de données.
- **--plugin-dir=/usr/lib/mysql/plugin** : chemin vers le dossier contenant les plugins.
- **--user=mysql** : compte utilisateur sous lequel le serveur est lancé.

- **--skip-log-error** : n'enregistre pas dans le journal d'erreur mais utilise syslog à la place.
- **--pid-file=/var/run/mysqld/mysqld.pid** : fichier contenant le PID du service.
- **--socket=/var/run/mysqld/mysqld.sock** : chemin vers la socket Unix utilisée par les clients locaux.
- **--port=3306** : port TCP utilisé. Le port 3306 est le port par défaut.

La commande **mysqld --help --verbose** donne la liste complète des options disponibles et leurs valeurs actuelles.

Autres options utiles :

- **--bind-address** : par défaut, le serveur écoute sur le réseau et sur l'adresse 0.0.0.0 (toutes les adresses). IL est possible de restreindre l'écoute sur une ou plusieurs interface, comme Debian le fait en n'écoutant que sur l'adresse de loopback localhost.

16.5. Utilisation

La commande mysql

La commande **mysql** permet de se connecter à un serveur MySQL.

Syntaxe de la commande mysql

```
[root]# mysql [–u utilisateur] [-p[password]] [-D database] [-h host]
```

Exemples de la commande shutdown

```
[root]# mysql -u root -pmdparoot
```

Table 28. Options de la commande shutdown

Options	Observations
-u	L'utilisateur MySQL a utilisé lors de la connexion au serveur.
-p	Le mot de passe à utiliser lors de la connexion au serveur. Il ne doit pas y avoir d'espace entre l'option -p et le mot de passe. Sans mot de passe suivant l'option -p, MySQL en demandera un.
-D	La base de données à utiliser.
-h	Se connecter à un serveur distant.

Gestion des utilisateurs

Création/suppression d'un utilisateur et gestion de ses privilèges

- Connexion au serveur MySQL :

```
$ mysql -u root -p
```

- Créer un nouvel utilisateur :

```
CREATE USER 'utilisateur'@'localhost' IDENTIFIED BY 'password';
```

Dans la requête ci-dessus, l'utilisateur ne pourra se connecter au serveur MySQL que localement (localhost). Si l'utilisateur doit pouvoir se connecter à distance depuis l'adresse IP 172.16.1.100, il faudra utiliser la requête suivante :

```
CREATE USER 'utilisateur'@'172.16.1.100' IDENTIFIED BY 'password';
```

- Donner des droits à l'utilisateur :

Si l'utilisateur doit accéder à la totalité des bases en lecture :

```
GRANT SELECT ON *.* TO 'utilisateur'@'localhost';
```

Syntaxe de la requête GRANT

```
GRANT <permission type> ON <database>.<table> TO '<username>'@'<host>';
```

- Suppression des droits :

Pour supprimer les droits d'un utilisateur les mots clefs **GRANT** et **ON** seront remplacés par **REVOKE** et **TO**.

Syntaxe de la commande REVOKE

```
REVOKE <permission type> ON <database>.<table> FROM '<username>'@'<host>';
```

- Suppression d'un utilisateur :

Un utilisateur est supprimé avec le mot clé **DROP** :

```
DROP USER 'utilisateur'@'localhost';
```

- Appliquer les modifications :

Pour que les modifications prennent effet, il faut recharger tous les privilèges :

```
FLUSH PRIVILEGES;
```

- Quitter l'environnement mysql :

Pour quitter l'environnement mysql, il faudra saisir la commande :

```
exit;
```

Les types de permissions

Il existe différents types de permissions à offrir aux utilisateurs :

- **SELECT** : lecture sur les données
- **USAGE** : autorisation de se connecter au serveur (donné par défaut à la création d'un nouvel utilisateur)
- **INSERT** : ajouter de nouveaux tuples dans une table.
- **UPDATE** : modifier des tuples existantes
- **DELETE** : supprimer des tuples
- **CREATE** : créer des nouvelles tables ou des bases de données
- **DROP** : supprimer des tables ou des bases de données existantes
- **ALL PRIVILEGES** : tous les droits
- **GRANT OPTION** : donner ou supprimer des droits aux autres utilisateurs

Définition de **tuple** : Collection ordonnée des valeurs d'un nombre indéfini d'attributs relatifs à un même objet.

16.6. La gestion des journaux

MySQL renseigne différents journaux :

- Le journal des erreurs

Il contient les messages générés au démarrage et à l'arrêt du service ainsi que les événements importants (warning et error).

- Le journal binaire

Ce journal (au format binaire) conserve toutes les actions qui modifient la structure des bases ou les données. En cas de restauration d'une base de données, il faudra restaurer la sauvegarde ET rejouer le journal binaire pour retrouver l'état de la base de données avant le crash.

- Le journal des requêtes

Toutes les requêtes émises par les clients sont consignées dans ce journal.

- Le journal des requêtes lentes

Les requêtes lentes, c'est à dire qui mettent plus qu'un certain temps (ce temps est paramétrable) à s'exécuter sont consignées à part dans ce journal. Une analyse de ce fichier permettra éventuellement de prendre des mesures afin de réduire leur temps d'éxécution (mettre en place des index par exemple ou modifier l'application cliente).

Hormis le journal binaire, ces journaux sont au format texte, ce qui permet de les exploiter directement !

Les paramètres du démon **mysqld** concernant les journaux sont :

Table 29. Paramètres de gestion des journaux d'enregistrements

Paramètres	Observations
--log-error=pathtofile	le journal des erreurs
--log-bin=path	le journal binaire
--log	le journal des requêtes
--slow-queries=pathtofile	le journal des requêtes lentes
--long_query_time=secondes	durée à partir de laquelle une requête est considérée comme longue et donc consignée dans le journal des requêtes lentes.

16.7. La chasse aux requêtes longues

Afin d'activer le journal d'enregistrement des requêtes longues, éditez le fichier de configuration my.cnf pour ajouter les lignes suivantes :

```
slow_query_log      = 1
slow_query_log_file = /var/log/mysql/mysql-slow.log
long_query_time     = 2
```

La valeur minimale pour la variable long_query_time est de 0 et la valeur par défaut est fixée à 10 secondes.

Relancez le service pour que les modifications soient prises en compte.

Une fois que le fichier de log se rempli, il est possible de l'analyser avec la commande **mysqldumpslow**.

Syntaxe de la commande mysqldumpslow

```
mysqldumpslow [options] [log_file ...]
```

Table 30. Options de la commande mysqldumpslow

Option	Observation
-t n	N'affiche que les n premières requêtes
-s sort_type	Trie en fonction du nombre de requête (c), .
-r	Inverse l'affichage des résultats

Les types de tri peuvent être :

- c : en fonction du nombre de requête
- t ou at : en fonction du temps d'exécution ou de la moyenne du temps d'exécution (a pour average)
- l ou al : en fonction du temps de verrou ou de sa moyenne
- r ou ar : en fonction du nombre de lignes renvoyées ou de sa moyenne

16.8. La sauvegarde

Comme pour tout SGBD-R, la sauvegarde d'une base de données doit se faire alors que les données ne sont pas en cours de modification. Cela est possible :

- alors que le service est à l'arrêt : il s'agit d'une **sauvegarde offline** ;
- alors que le service fonctionne mais un verrou est posé (pour momentanément suspendre toutes modifications) : il s'agit d'une **sauvegarde online**
- en utilisant un instantané du système de fichiers type LVM, permettant de sauvegarder les données avec un système de fichiers à froid.

Le format de la sauvegarde peut être un fichier au format ASCII (texte), représentant l'état de la base et de ses données sous forme d'ordres SQL, ou un fichier binaire, correspondant aux fichiers de stockage de MySQL.

Tandis que la sauvegarde d'un fichier binaire peut se faire à l'aide des utilitaires courants comme tar ou cpio, la sauvegarde au format ASCII nécessite l'utilisation d'un utilitaire comme **mysqldump**.

N'oubliez pas qu'après la restauration d'une sauvegarde complète, la restauration des fichiers binaires permettent de compléter la reconstitution des données.

Exemple de sauvegarde d'une base de données mabase :

```
$ mysqldump -u root -p --opt mabase > /tmp/sauvegarde-mabase.sql
```

Lors de la sauvegarde, l'utilitaire mysqldump pose un verrou sur la base.

Le fichier obtenu permet de restaurer les données de la base (la base doit toujours exister ou être recréée au préalable !) :

```
$ mysql -u root -p mabase < /tmp/sauvegarde-mabase.sql
```

16.9. Outils de gestions

- PhpMyAdmin
- MySQL Workbench
- MySQL administrator

Chapitre 17. Serveurs MySQL/MariaDB - Multi-Maîtres

MySQL/MariaDB fonctionne par défaut en Standalone, chaque serveur étant autonome.

L'accès aux données étant crucial pour les entreprises, les administrateurs chercheront des solutions de **répartitions de charge** (répartition des requêtes en écriture sur le serveur maître et celles en lecture sur les serveurs esclaves).

MySQL propose un système de réplication à sens unique, basée sur :

- Un serveur **maître** contenant la base de données en **écriture**,
- Un ou plusieurs serveurs **esclaves**, qui vont se synchroniser sur le serveur maître.

Ce mode de réplication est dit "Asynchrone". La données consultée sur un noeud peut être différente de la valeur stockée sur le noeud maître le temps de la réplication.

Un serveur esclave peut devenir maître pour d'autres serveurs, cascadant ainsi la réplication d'une donnée.

Une telle infrastructure permet de répartir la charge de travail (requêtes en écriture sur le maître, les autres requêtes vers les esclaves, sauvegarde sur un esclave dédié) mais :

- l'application doit avoir été **développée spécifiquement** pour utiliser l'infrastructure,
- en cas de panne sur un des serveurs (panne du maître ou d'un esclave d'esclave), une **action de l'administrateur** sera nécessaire,
- en cas d'évolution de l'infrastructure, l'application devra être modifiée.

Dans ce type de réplication, l'état de la base de données est dans un premier temps dupliqué sur le serveur esclave. Ensuite le serveur esclave entre en contact avec le serveur maître et récupère les données binaires depuis une position de référence et se met à jour.

1. Le journal binaire doit être activé sur tous les serveurs. Un identifiant unique doit être attribué à chaque serveur.
2. Un compte de réplication avec le privilège de 'REPLICATION SLAVE' doit être créé sur le serveur maître permettant à l'esclave de s'y connecter.
3. La base de données du maître est sauvegardée puis exportée vers l'esclave (manuellement).
4. Sur le maître, le fichier binaire en cours et la position dans le journal doivent être récupérés (avec la commande SQL 'show master status;')
5. Sur l'esclave, le maître peut être configuré et le processus de réplication démarré.

Pour des besoins de **hautes disponibilités** (**HA** : High Availability), l'administrateur pourra :

- configurer ses serveurs en Multi-Maîtres : chaque esclave étant également le maître des autres

serveurs. Cette technique est limitée à 64 maîtres maximum.

- utiliser des technologies de clustering avec MySQL Cluster qui propose un système de réplication Synchrone et une répartition de la charge.

Dans ce chapitre, nous allons vous présenter la mise en oeuvre d'une **réplication Multi-Maître à deux noeuds**. La plate-forme de test est constituée de deux noeuds CentOS 7 :

- mirroir1 : 192.168.100.173
- mirroir2 : 192.168.100.211

17.1. Configuration des noeuds

Lors de la mise en place de cluster, il est recommandé de s'assurer de la bonne résolution des noms de machines de chaque membre du cluster dès le démarrage de la machine et même en absence de service DNS.

Configuration du fichier /etc/hosts sur chaque noeud :

Fichier /etc/hosts

```
...
192.168.100.173         mirroir1.local.lan      mirroir1
192.168.100.211         mirroir2.local.lan      mirroir2
```

Installation de mariadb-server :

```
yum install mariadb-server
```

Le service peut être démarré sur chacun des noeuds :

```
systemctl enable mariadb
systemctl start mariadb
```

17.2. Création de la base à répliquer

La base de données à répliquer est crée sur les deux noeuds :

```
mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 5.5.52-MariaDB MariaDB Server

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB > show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
4 rows in set (0.00 sec)

MariaDB > create database mabase;
Query OK, 1 row affected (0.01 sec)
```

17.3. Création des utilisateurs MySQL pour la réplication

Sur chaque noeud doit être créé un utilisateur qui disposera des droits de réplication sur le serveur distant.

- Sur le noeud 1 :

```
MariaDB > create user 'mirroir'@'mirroir2.local.lan' identified by 'm!rro!r';
Query OK, 0 rows affected (0.01 sec)

MariaDB > grant replication slave on *.* to 'mirroir'@'mirroir2.local.lan';
Query OK, 0 rows affected (0.00 sec)
```

- Sur le noeud 2 :

```
MariaDB > create user 'mirroir'@'mirroir1.local.lan' identified by 'm!rro!r';
Query OK, 0 rows affected (0.01 sec)

MariaDB > grant replication slave on *.* to 'mirroir'@'mirroir1.local.lan';
Query OK, 0 rows affected (0.00 sec)
```

17.4. Configuration de MySQL

Le fichier my.cnf peut être modifié comme suit :

- Sur le noeud 1 :

Fichier /etc/my.cnf sur le noeud 1

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
port=3306
innodb_file_per_table=ON
pid-file=/var/run/mariadb/mariadb.pid

server-id=10
log_bin=/var/log/mariadb/mariadb-bin.log
binlog_do_db=mabase
```

- **server-id** : l'identifiant du serveur pour la réplication. Il doit être différent sur chaque noeud.
- **log_bin** : Le fichier de log utilisé pour suivre l'activité de la réplication
- **binlog_do_db** : La base de données concernée par le processus de réplication
- Sur le noeud 2 :

Fichier /etc/my.cnf sur le noeud 2

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
port=3306
innodb_file_per_table=ON
pid-file=/var/run/mariadb/mariadb.pid

server-id=11
log_bin=/var/log/mariadb/mariadb-bin.log
binlog_do_db=mabase
```

Relancer les services sur les deux noeuds pour prendre en compte les modifications :

```
systemctl restart mariadb
```

- Vérification

Sur les différents noeuds, vérifier l'état de la réplication :

```
MariaDB > show master status;
+--------------------+----------+--------------+------------------+
| File               | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+--------------------+----------+--------------+------------------+
| mariadb-bin.000001 |      245 | mabase       |                  |
+--------------------+----------+--------------+------------------+
1 row in set (0.00 sec)

MariaDB > show slave status;
Empty set (0.00 sec)
```

Basculer les serveurs en multi-maîtres :

- Sur le noeud 1 :
  - Arrêter le processus de synchronisation
  - Ajouter le noeud 2 comme maître
  - Redémarrer le processus de synchronisation

Les valeurs de MASTER_LOG_POS et MASTER_LOG_FILE sont à récupérer sur le noeud 2.

```
mariadb > stop slave;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mariadb > CHANGE MASTER TO MASTER_HOST = 'mirroir2.local.lan', MASTER_PORT = 3306,
MASTER_USER = 'mirroir', MASTER_PASSWORD = 'm!rro!r', MASTER_LOG_FILE = 'mariadb-
bin.000001', MASTER_LOG_POS = 245;
Query OK, 0 rows affected, 2 warnings (0.25 sec)

mariadb > start slave;
Query OK, 0 rows affected (0.03 sec)

MariaDB [(none)]> show slave status;
+---------------------------------+-------------+-------------+-------------
+---------------+--------------------+---------------------+--------------------------
+---------------+----------------------+------------------+-------------------
+-----------------+---------------------+--------------------+------------------------
+-------------------------+-----------------------------+------------+------------
+-------------+--------------------+----------------+----------------
+----------------+--------------+--------------------+-------------------
```

```
+--------------------+----------------+-------------------+---------------
+----------------------+------------------------------+--------------
+--------------+----------------+----------------+----------------------------
+------------------+
| Slave_IO_State                    | Master_Host | Master_User | Master_Port |
Connect_Retry | Master_Log_File    | Read_Master_Log_Pos | Relay_Log_File           |
Relay_Log_Pos | Relay_Master_Log_File | Slave_IO_Running | Slave_SQL_Running |
Replicate_Do_DB | Replicate_Ignore_DB | Replicate_Do_Table | Replicate_Ignore_Table |
Replicate_Wild_Do_Table | Replicate_Wild_Ignore_Table | Last_Errno | Last_Error |
Skip_Counter | Exec_Master_Log_Pos | Relay_Log_Space | Until_Condition |
Until_Log_File | Until_Log_Pos | Master_SSL_Allowed | Master_SSL_CA_File |
Master_SSL_CA_Path | Master_SSL_Cert | Master_SSL_Cipher | Master_SSL_Key |
Seconds_Behind_Master | Master_SSL_Verify_Server_Cert | Last_IO_Errno | Last_IO_Error
| Last_SQL_Errno | Last_SQL_Error | Replicate_Ignore_Server_Ids | Master_Server_Id |
+---------------------------------+-------------+-------------+-------------
+--------------+--------------------+--------------------+-------------------------
+--------------+----------------------+------------------+------------------
+----------------+----------------------+--------------------+-----------------------
+-------------------------+-----------------------------+------------+------------
+--------------+--------------------+-----------------+-----------------
+----------------+--------------+--------------------+--------------------
+--------------------+----------------+-------------------+---------------
+----------------------+------------------------------+--------------
+--------------+----------------+----------------+----------------------------
+------------------+
| Waiting for master to send event | mirroir2    | mirroir     |        3306 |
60 | mariadb-bin.000001 |                509 | mariadb-relay-bin.000002 |
531 | mariadb-bin.000001    | Yes              | Yes               |                |
|                    |                       |                          |
|          0 |           |           0 |                509 |             827 |
None            |               |            0 | No                |
|                    |               |                  |               |
0 | No                            |            0 |              |             0 |
|                             |              11 |
+---------------------------------+-------------+-------------+-------------
+--------------+--------------------+--------------------+-------------------------
+--------------+----------------------+------------------+------------------
+----------------+----------------------+--------------------+-----------------------
+-------------------------+-----------------------------+------------+------------
+--------------+--------------------+-----------------+-----------------
+----------------+--------------+--------------------+--------------------
+--------------------+----------------+-------------------+---------------
+----------------------+------------------------------+--------------
+--------------+----------------+----------------+----------------------------
+------------------+
1 row in set (0.00 sec)
```

La requête **CHANGE MASTER [nom_connexion] TO** créé ou modifie une connexion à un serveur

maître. Elle change les paramètres que le serveur esclave utilise pour se connecter et communiquer avec le serveur maître durant les réplications. Sans spécifier de nom de connexion, la connexion par défaut est modifiée.

Les valeurs **MASTER_LOG_FILE** et **MASTER_LOG_POS** sont les coordonnées à partir desquelles l'esclave doit commencer à lire depuis le maître lors de la prochaine réplication.

Sur le noeud 2 :

```
mariadb > stop slave;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mariadb > CHANGE MASTER TO MASTER_HOST = 'mirroir1.local.lan', MASTER_PORT = 3306,
MASTER_USER = 'mirroir', MASTER_PASSWORD = 'm!rro!r', MASTER_LOG_FILE = 'mariadb-
bin.000001', MASTER_LOG_POS = 245;
Query OK, 0 rows affected, 2 warnings (0.07 sec)

mariadb > start slave;
Query OK, 0 rows affected (0.04 sec)
```

17.5. Tests de bon fonctionnement

Pour tester le bon fonctionnement du cluster, une table va être créée sur le noeud 1. Après vérification sur le noeud 2 que la table a bien été répliquée, des données y seront rajoutées. La présence des données sur le noeud 1 permettra de valider la réplication multi-maître.

- Sur le noeud 1 :

```
[root@mirroir1 ~]# mysql -u root -p mabase
Enter password:

mariadb > create table table1( id int(11) primary key auto_increment, nom
varchar(30));
Query OK, 0 rows affected (0.22 sec)

mariadb > show tables in mabase;
+------------------+
| Tables_in_mabase |
+------------------+
| table1 |
+------------------+
1 row in set (0.01 sec)
```

- Vérifier le noeud 2 la présence de la table et ajouter des données :

```
[root@mirroir2 ~]# mysql -u root -p mabase
Enter password:

mariadb > show tables in mabase;
+--------------------+
| Tables_in_test_rep |
+--------------------+
| table1 |
+--------------------+
1 row in set (0.00 sec)

mariadb > insert into table1 ( nom ) values ('antoine'), ('xavier'), ('patrick') ;
Query OK, 3 rows affected (0.05 sec)
Records: 3 Duplicates: 0 Warnings: 0

mariadb > select * from table1;
+----+----------+
| id | fullname |
+----+----------+
| 1 | antoine |
| 2 | xavier |
| 3 | patrick |
+----+----------+
3 rows in set (0.00 sec)

mariadb > commit;
Query OK, 0 rows affected (0.00 sec)
```

- Retour sur le noeud 1 :

```
[root@DB1 ~]# mysql -u root -p mabase
Enter password:

mariadb > select * from table1;
+----+----------+
| id | fullname |
+----+----------+
| 1 | antoine |
| 2 | xavier |
| 3 | patrick |
+----+----------+
3 rows in set (0.01 sec)
```

Chapitre 18. La mise en cluster sous Linux

> La **haute disponibilité** est un terme souvent utilisé en informatique, à propos d'architecture de système ou d'un service pour désigner le fait que cette architecture ou ce service a un taux de disponibilité convenable.
>
> — wikipedia

Cette disponibilité est une mesure de performance exprimée en pourcentage obtenue par le ration **Durée de fonctionnement** / **Durée total de fonctionnement souhaité**.

Table 31. Taux de disponibilité

Taux	Temps d'arrêt annuel
90%	876 heures
95%	438 heures
99%	87 heures et 36 minutes
99,9%	8 heures 45 minutes 36 secondes
99,99%	52 minutes, 33 secondes
99,999%	5 minutes, 15 secondes
99,9999%	31,68 secondes

La "Haute Disponibilité" (en anglais "**High Availability**" - **HA**) concerne donc toutes les mesures prises visant à garantir la plus haute disponibilité d'un service, c'est à dire son bon fonctionnement 24H/24H.

18.1. Généralités

Un **cluster** est une ***"grappe d'ordinateurs"***, un groupe de deux machines ou plus.

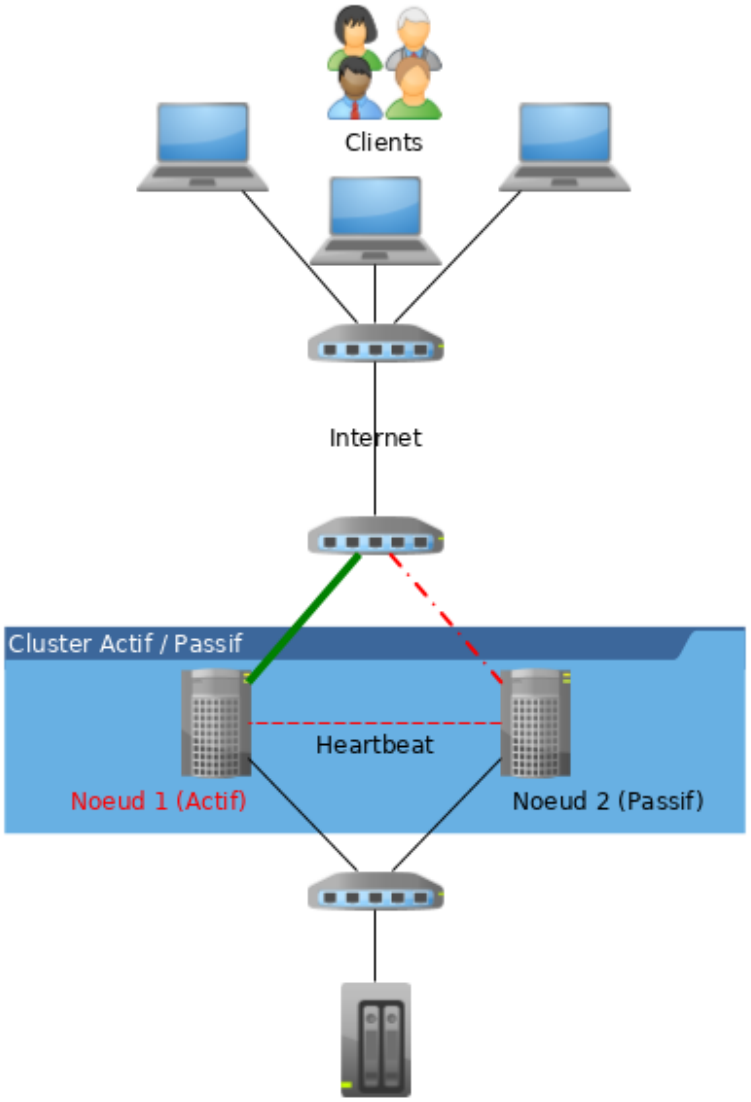


Figure 42. Schéma d'un cluster actif/passif

Un cluster permet :

- le **calcul distribué** utilisant la puissance de calcul de l'ensemble des noeuds,
- la **haute disponibilité**, la continuité de service et le basculement automatique des services en cas de panne d'un noeud.

Les types de services

- Services Actif / Passif

L'installation d'un cluster avec deux noeuds actif/passif utilisant Pacemaker et DRBD est une solution à faible coût pour de nombreux cas de figure nécessitant un système de haute disponibilité.

- Services N+1

Grâce à plusieurs noeuds, Pacemaker peut réduire le coût matériel en permettant à plusieurs

clusters actif/passif de se combiner et de partager un noeud de secours.

- Services N TO N

Avec un stockage partagé, chaque noeud peut potentiellement être utilisé pour la tolérance de panne. Pacemaker peut également faire fonctionner plusieurs copies des services pour répartir la charge de travail.

- Services Sites Distants

Pacemaker inclus des améliorations pour simplifier la création de clusters sur plusieurs sites.

Les VIP

La **VIP** est une adresse IP virtuelle. Cette adresse est attribuée à un cluster Actif/Passif. Elle est assignée au noeud du cluster qui est actif. En cas de rupture de service, la VIP est désactivée sur le noeud en échec et activée sur le noeud prenant le relais : c'est la **tolérance de panne** (ou **failover**).

Les clients s'adressent toujours au cluster en utilisant la VIP, ce qui rend transparent à leurs yeux les bascules de serveur actif.

Le split-brain

Le ***split-brain*** est le risque principal que peut rencontrer un cluster. Cet état arrive lorsque plusieurs nœuds d'un cluster pense son voisin inactif. Le noeud tente alors de démarrer le service redondant et plusieurs noeuds fournissent alors le même service, ce qui peut entraîner des effets de bords génant (VIP en double sur le réseau, accès concurrent aux données, etc.).

Les solutions techniques possibles pour éviter ce problème sont :

- de séparer le trafic réseau public du trafic réseau du cluster,
- d'utiliser un agrégat de lien (bonding) réseau.

18.2. La gestion des ressources : Pacemaker

Pacemaker est la partie logicielle du cluster qui gère ses ressources (les VIP, les services, les données). Il est chargé de faire démarrer, arrêter et superviser les ressources du cluster. Il est le garant de la haute disponibilité des noeuds.

Pacemaker utilise la couche de message fournit par **corosync** (par défaut) ou par **Heartbeat**.

Pacemaker est composé de **5 composants clés** :

- La base d'information du cluster (Cluster Information Base - **CIB**)
- Le démon de gestion des ressources du cluster (Cluster Resource Management daemon - **CRMd**)
- Le démon de gestion des ressources locales (Local Resource Management daemon - **LRMd**)

- Le moteur de stratégie (Policy Engine - **PEngine** ou **PE**)
- Le démon de protection (Fencing daemon - **STONITHd**)

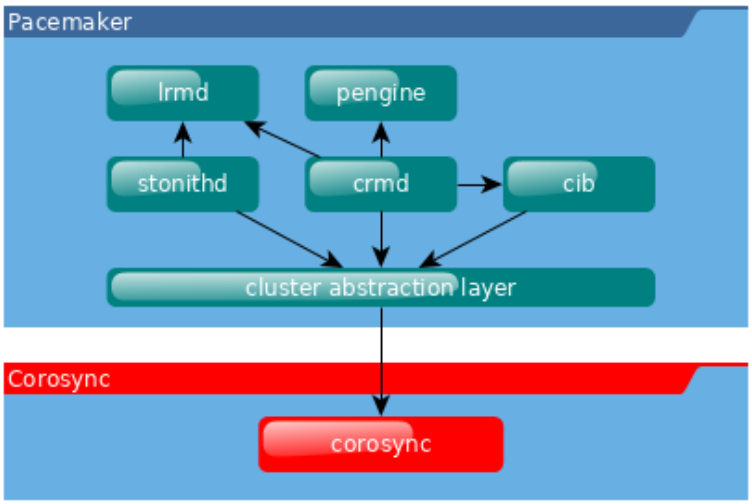


Figure 43. Les composants logiciels du cluster

La **CIB** représente la configuration du cluster et l'état en cours de toutes les ressources du cluster. Le contenu de la CIB est automatiquement synchronisé sur le cluster complet et est utilisé par le **PEngine** pour calculer l'état idéal du cluster et comment il doit être obtenu.

La liste des instructions est alors fournie au controlleur désigné (**Designated Controller** - **DC)**. Pacemaker centralise toutes les décisions du cluster en élisant en tant que maître une des instances de **CRMd**.

Le **DC** execute dans l'ordre requis les instructions du **PEngine** en les transmettant soit au **LRMd** local ou aux **CRMd** des autres noeuds via **Corosync** ou **Heartbeat**.

Dans certains cas, il est nécessaire de stopper des noeuds pour protéger les données partagées ou permettre leur récupération. Pour cela, Pacemaker est livré avec **STONITHd**.

Stonith

Stonith est un composant de Pacemaker. Il est l'acronyme de **Shoot-The-Other-Node-In-The-Head**, une pratique recommandée pour que le noeud dysfonctionnant soit le plus rapidement isolé (éteint ou au moins déconnecté des ressources partagées), ce qui évite la corruption des données.

Un nœud qui ne répond pas ne signifie pas qu'il n'accède plus aux données. La seule solution pour s'assurer qu'un nœud n'accède plus aux données avant de donner la main à un autre nœud et d'utiliser STONITH, qui va soit éteindre, soit redémarrer le serveur en échec.

STONITH a également un rôle à jouer dans le cas où un service en cluster n'arrive pas à s'arrêter. Dans ce cas, Pacemaker utilise STONITH pour forcer le nœud entier à s'arrêter.

La gestion du quorum

Le **quorum** représente le nombre de **noeuds minimum** en fonctionnement qui permettent de valider une décision, comme décider quel noeud de secours doit prendre le relais lorsqu'un des noeuds est en erreur. Pacemaker, par défaut, exige que plus de la moitié des nœuds soient en ligne.

Lorsque des problèmes de communication séparent un cluster en plusieurs groupes de noeuds, le quorum est alors utilisé pour prévenir le démarrage des ressources sur plus de noeuds que prévu. Un cluster a le quorum lorsque plus de la moitié de tous les noeuds étant connus en ligne se retrouvent dans son groupe (noeuds_actifs_groupe > noeuds_total_actifs / 2)

La décision par défaut lorsque le quorum n'est pas atteint est de désactiver toutes les ressources.

```
Exemple de cas :
```

- Sur un cluster à **deux noeuds**, le quorum ne pouvant **jamais être atteint** en cas de panne d'un noeud, il doit donc être ignoré sous peine que le cluster complet soit stoppé.
- Si un cluster à 5 noeuds est coupé en 2 groupes de 3 et 2 noeuds, le groupe de 3 noeuds disposera du quorum et continuera à gérer les ressources.
- Si un cluster à 6 noeuds est coupé en 2 groupes de 3 noeuds alors aucun groupe ne disposera du quorum. Dans ce cas, le comportement par défaut de pacemaker est de stopper toutes les ressources pour éviter la corruption de données.

18.3. Communication du clusters

Pacemaker s'appuie au choix sur **Corosync** ou **Heartbeat** (du projet linux-ha) pour assurer la communication entre les nœuds et la gestion du cluster.

Corosync

Corosync Cluster Engine est une couche de messagerie entre les membres du cluster et intègre des fonctionnalités additionnelles pour l'implémentation de la haute disponibilité au sein des applications. Le projet Corosync est dérivé du projet OpenAIS.

La communication entre les noeuds se fait en mode Client/Serveur via le protocole UDP.

Il permet de gérer des cluster composés de plus de 16 noeuds dans les modes Actif/Passif ou Actif/Actif.

Heartbeat

La technologie Heartbeat est plus limitée que Corosync. Il n'est pas possible de créer un cluster de plus de 2 noeuds et les règles de gestion sont moins abouties que son concurrent.

Le choix de pacemaker/corosync aujourd'hui semble plus opportun, c'est le choix par défaut des distributions RedHat, Debian et Ubuntu.

18.4. La gestion des données

Le raid en réseau drdb

DRDB est un driver de périphérique de type **bloc** qui permet la mise en oeuvre de **RAID 1** (mirroir) **via le réseau**.

La mise en oeuvre de DRDB peut être intéressante lorsque des technologies NAS ou SAN ne sont pas disponibles mais que les données doivent tout de même être synchronisées.

GFS2

18.5. Ateliers Dirigés Cluster

Ces ateliers dirigés s'appuient sur une infrastructure à deux noeuds :

- Noeud 1 :
  - Nom de machine : node1.formatux.fr
  - Disque système 10G ou plus
  - Disque 2 : 1G (pour DRDB)
  - Distribution CentOS7 à jour
  - Adresse IP publique : 192.168.1.100
  - Adresse IP privée : 192.168.100.100
- Noeud 2 :
  - Nom de machine : node2.formatux.fr
  - Disque système 10G ou plus
  - Disque 2 : 1G (pour DRDB)
  - Distribution CentOS7 à jour
  - Adresse IP publique : 192.168.1.101
  - Adresse IP privée : 192.168.100.101

Préparation des hôtes

Sur chaque hôte sera configuré :

- Le fichier /etc/hosts pour assurer la résolution de nom dès le démarrage de la machine et en toute indépendance du serveur DNS

- Le service Corosync
- Le service Pacemaker

A l'issue de cette configuration, les mises en oeuvre de clusters avec VIP, avec services puis avec DRDB pourront être envisagés.

Configuration du fichier /etc/hosts

Le fichier */etc/hosts* doit permettre la résolution des adresses IP publiques et des adresses IP privées.

Le fichier /etc/hosts ressemblera à l'exemple ci-dessous sur l'ensemble des noeuds du cluster :

Fichier /etc/hosts

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.1.100           node1.formatux.fr       node1
192.168.1.101           node2.formatux.fr       node2
192.168.100.100         node1-priv.formatux.fr  node1-priv
192.168.100.101         node2-priv.formatux.fr  node2-priv
```

Depuis le noeud 1, le noeud 2 peut être joint soit par son nom publique ou son nom privé :

```
$ ping -c 4 node2
$ ping -c 4 node2-priv
```

Informations sur les paquets Pacemaker et corosync

Quelques informations sur le paquet pacemaker :

```
$ yum info pacemaker
Paquets disponibles
Nom                 : pacemaker
Résumé              : Scalable High-Availability cluster resource manager
URL                 : http://www.clusterlabs.org
Licence             : GPLv2+ and LGPLv2+
Description         : Pacemaker is an advanced, scalable High-Availability
                    : cluster resource manager for Corosync, CMAN and/or
                    : Linux-HA.
                    :
                    : It supports more than 16 node clusters with significant
                    : capabilities for managing resources and dependencies.
                    :
                    : It will run scripts at initialization, when machines go up
                    : or down, when related resources fail and can be configured
                    : to periodically check resource health.
```

Grâce à la commande repoquery (voir le chapitre commandes avancées), il est possible de connaître les dépendances du paquet pacemaker :

```
$ sudo yum install yum-utils
$ repoquery --requires pacemaker
corosync
pacemaker-cli = 1.1.15-11.el7_3.5
resource-agents
...
```

L'installation de pacemaker installera donc automatiquement corosync et une interface CLI pour pacemaker.

Quelques informations sur le paquet corosync :

```
yum info corosync
Paquets disponibles
Nom                 : corosync
Résumé              : The Corosync Cluster Engine and Application Programming
                    : Interfaces
URL                 : http://corosync.github.io/corosync/
Licence             : BSD
Description         : This package contains the Corosync Cluster Engine
                    : Executive, several default APIs and libraries, default
                    : configuration files, and an init script.
```

Installation des logiciels corosync et pacemaker

```
# yum install pacemaker
```

- Ouverture du firewall :

```
# firewall-cmd --permanent --add-service=high-availability
success
# firewall-cmd --reload
success
```

Les services peuvent maintenant être activés pour le prochain démarrage :

```
# systemctl enable corosync
# systemctl enable pacemaker
```

Gestion du cluster

Le paquet **pcs** fournit des outils la gestion du cluster. La commande **pcs** est une interface en ligne de commande pour gérer la **stack** de haute disponibilité de Pacemaker.

La configuration du cluster pourrait éventuellement être faite à la main, mais le paquet pcs facilite grandement la gestion (création, configuration et dépannage) d'un cluster !

Il existe des alternatives à pcs.

Installation de pcs

Installer le paquet sur l'ensemble des noeuds et activer le démon :

```
# yum install pcs
# systemctl start pcsd
# systemctl enable pcsd
```

L'installation du paquet a créé un utilisateur **hacluster** avec un mot de passe vide. Pour effectuer les tâches de synchronisation des fichiers de configuration de corosync ou redémarrer les noeuds distants, un mot de passe doit lui être attribué.

Sur tous les noeuds, attribuer un mot de passe identique à l'utilisateur hacluster :

```
# echo "mdphacluster" | passwd --stdin hacluster
```

Administration du cluster

Depuis n'importe quel noeud, il est possible de s'authentifier comme utilisateur hacluster sur l'ensemble des noeuds, puis d'utiliser les commandes pcs sur ceux-ci :

```
pcs cluster auth node1-priv node2-priv
Username: hacluster
Password:
node2-priv: Authorized
node1-priv: Authorized
```

Depuis le noeud sur lequel pcs est authentifié, lancer la configuration du cluster :

```
pcs cluster setup --name moncluster node1-priv node2-priv
```

Le cluster peut maintenant être démarré :

```
pcs cluster start --all
node1-priv: Starting Cluster...
node2-priv: Starting Cluster...
```

La commande **pcs cluster setup** prend en charge le problème du quorum des clusters à deux noeuds. Un tel cluster fonctionnera donc correctement en cas de panne d'un des deux noeuds. Si vous configurez manuellement corosync ou utilisez un autre shell de gestion du cluster, vous devrez configurer corosync correctement par vous-même.

Vérifications

La commande **pcs cluster setup** a eu pour effet de générer un fichier **/etc/corosync/corosync.conf** :

```
totem {
    version: 2
    secauth: off
    cluster_name: moncluster
    transport: udpu
}

nodelist {
    node {
        ring0_addr: node1-priv
        nodeid: 1
    }

    node {
        ring0_addr: node2-priv
        nodeid: 2
    }
}

quorum {
    provider: corosync_votequorum
    two_node: 1
}

logging {
    to_logfile: yes
    logfile: /var/log/cluster/corosync.log
    to_syslog: yes
}
```

Des exemples de fichiers de configuration plus complets se trouvent sous /etc/corosync.

La commande **pcs status** renseigne sur l'état global du cluster :

```
pcs status
Cluster name: moncluster
WARNING: no stonith devices and stonith-enabled is not false
Stack: corosync
Current DC: node1-priv (version 1.1.15-11.el7_3.5-e174ec8) - partition with quorum
Last updated: Wed Jul  5 18:22:47 2017      Last change: Wed Jul  5 17:56:27 2017 by
hacluster via crmd on node2-priv

2 nodes and 0 resources configured

Online: [ node1-priv node2-priv ]

No resources


Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

Comme vous pouvez le constater dans le retour de la commande, le processus **stonith** est activé mais non configuré :

```
WARNING: no stonith devices and stonith-enabled is not false
```

Dans un premier temps, nous allons désactiver stonith en attendant d'apprendre à le configurer :

```
pcs property set stonith-enabled=false
```

Attention à ne pas laisser stonith désactivé sur un environnement de production !!!

La commande **pcs status corosync** nous renseigne sur l'état des noeuds corosync :

```
pcs status corosync

Membership information
----------------------
    Nodeid      Votes Name
         1          1 node1-priv (local)
         2          1 node2-priv
```

Les outils standards peuvent également être utilisés :

- La commande **crm_mon** renvoie une configuration correcte du cluster :

```
# crm_mon -1
Stack: corosync
Current DC: node2-priv (version 1.1.15-11.el7_3.5-e174ec8) - partition with quorum
Last updated: Wed Jul  5 15:57:18 2017      Last change: Wed Jul  5 16:08:39 2017 by
hacluster via crmd on node2-priv

2 nodes and 0 resources configured

Online: [ node1-priv node2-priv ]

No active resources
```

- La commande **corosync-cfgtool** verifie si la configuration est correcte et si la communication avec le cluster se fait bien:

```
$ corosync-cfgtool -s

Printing ring status.
Local node ID 1
RING ID 0
        id      = 192.168.122.100
        status  = ring 0 active with no faults
```

- La commande **corosync-cmapctl** est un outil pour accéder à la base d'objets. Elle permet, par exemple, de vérifier le status des noeuds membres du cluster :

```
corosync-cmapctl  | grep members
runtime.totem.pg.mrp.srp.members.1.config_version (u64) = 0
runtime.totem.pg.mrp.srp.members.1.ip (str) = r(0) ip(192.168.100.100)
runtime.totem.pg.mrp.srp.members.1.join_count (u32) = 1
runtime.totem.pg.mrp.srp.members.1.status (str) = joined
runtime.totem.pg.mrp.srp.members.2.config_version (u64) = 0
runtime.totem.pg.mrp.srp.members.2.ip (str) = r(0) ip(192.168.100.101)
runtime.totem.pg.mrp.srp.members.2.join_count (u32) = 2
runtime.totem.pg.mrp.srp.members.2.status (str) = joined
```

TD Configuration d'une VIP

La première ressource que nous allons créer sur notre cluster est une VIP.

Cette VIP, correspondant à l'adresse IP utilisée par les clients pour accéder aux futurs services du cluster, sera attribuée à un des noeuds, puis, en cas de défaillance, le cluster basculera cette ressource d'un noeud à l'autre pour assurer la continuité du service.

```
pcs resource create monclusterVIP ocf:heartbeat:IPaddr1 ip=192.168.1.99
cidr_netmask=24 op monitor interval=30s
```

L'argument **ocf:heartbeat:IPaddr2** est composé de 3 champs qui fournissent à pacemaker :

1. le standard qui est suivi par le script de ressource (ici ocf),
2. l'espace de nom du script,
3. le nom du script de la ressource

Les ressources standards disponibles sont fournies par la commande **pcs resource standards** :

```
pcs resource standards
ocf
lsb
service
systemd
stonith
```

Le résultat est l'ajout d'une adresse IP virtuelle sur un des noeuds :

```
pcs status
Cluster name: moncluster
Stack: corosync
Current DC: node1-priv (version 1.1.15-11.el7_3.5-e174ec8) - partition with quorum
Last updated: Wed Jul  5 18:30:54 2017      Last change: Wed Jul  5 18:29:50 2017 by
root via cibadmin on node1-priv

2 nodes and 1 resource configured

Online: [ node1-priv node2-priv ]

Full list of resources:

 monclusterVIP  (ocf::heartbeat:IPaddr2):   Started node1-priv

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

Dans le cas présent, la VIP est active sur le noeud 1, ce qui est vérifiable avec la commande **ip** :

```
node1 # ip add sh enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen
1000
    link/ether 08:00:27:83:83:11 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.100/24 brd 192.168.1.255 scope global enp0s3
       valid_lft forever preferred_lft forever
    inet 192.168.1.99/24 brd 192.168.1.255 scope global secondary enp0s3
       valid_lft forever preferred_lft forever
```

Tests de bascule

- Depuis un poste du réseau, lancer la commande ping sur la VIP :

```
ping 192.168.1.99
```

- Redémarrer le noeud 1 :

```
node1 # reboot
```

- Sur le noeud 2 :

Constater que le noeud 1 est offline et que la bascule de la ressource s'est correctement effectuée :

```
# pcs status
Cluster name: moncluster
Stack: corosync
Current DC: node2-priv (version 1.1.15-11.el7_3.5-e174ec8) - partition with quorum
Last updated: Wed Jul  5 18:34:19 2017      Last change: Wed Jul  5 18:29:51 2017 by
root via cibadmin on node1-priv

2 nodes and 1 resource configured

Online: [ node2-priv ]
OFFLINE: [ node1-priv ]

Full list of resources:

 monclusterVIP  (ocf::heartbeat:IPaddr2):   Started node2-priv

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

- Vérifier avec la commande ip sur le noeud 2 :

```
node2 # ip add show enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen
1000
    link/ether 08:00:27:55:3d:ca brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.101/24 brd 192.168.1.255 scope global enp0s3
       valid_lft forever preferred_lft forever
    inet 192.168.1.99/24 brd 192.168.1.255 scope global secondary enp0s3
       valid_lft forever preferred_lft forever
```

- Depuis le poste du réseau, constater qu'aucun ping n'a été perdu durant la bascule.
- Noter qu'une fois le noeud 1 redémarré, le cluster retrouve son état normal, la ressource n'est toutefois pas rebasculée vers le noeud 1 : elle reste sur le noeud 2.

```
node2 # pcs status
Cluster name: moncluster
Stack: corosync
Current DC: node2-priv (version 1.1.15-11.el7_3.5-e174ec8) - partition with quorum
Last updated: Wed Jul  5 18:37:23 2017      Last change: Wed Jul  5 18:29:51 2017 by
root via cibadmin on node1-priv

2 nodes and 1 resource configured

Online: [ node1-priv node2-priv ]

Full list of resources:

 monclusterVIP  (ocf::heartbeat:IPaddr2):   Started node2-priv

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

TD Configuration d'un service Apache

L'objectif de cet atelier est d'installer le service Apache sur les deux noeuds de notre cluster. Ce service ne sera démarré que sur le noeud actif est basculera de noeud en même temps que la VIP en cas de panne du noeud actif.

Installation et configuration d'Apache

L'installation doit être faîte sur les 2 noeuds :

```
# yum install -y httpd
# firewall-cmd --permanent --add-service=http
# firewall-cmd --reload
```

Une page HTML contenant le nom du serveur sera affichée par défaut :

```
echo "<html><body>Noeud $(hostname -f)</body></html>" > "/var/www/html/index.html"
```

L'agent de ressource de Pacemaker utilise la page /server-status (voir chapitre apache) pour déterminer son état de santé. Il doit être activé en créant le fichier ***/etc/httpd/conf.d/status.conf*** :

Activation du server-status Apache

```
<Location /server-status>
  SetHandler server-status
  Require local
</Location>
```

Le service ne doit pas être démarré ni activé. La main est totalement laissé à Pacemaker, qui au démarrage du serveur prendra en charge le démarrage du service sur l'un des noeuds !

Configuration du service dans le cluster

Pour créer une ressource que nous appelerons "SiteSeb", nous allons faire appel au script apache de la ressource OCF et dans l'espace de nom de heartbeat.

```
# pcs resource create SiteWeb ocf:heartbeat:apache
configfile=/etc/httpd/conf/httpd.conf statusurl="http://localhost/server-status" op
monitor interval=1min
```

Le cluster va vérifier l'état de santé d'Apache toutes les minutes (**op monitor interval=1min**).

Enfin, pour que le service Apache soit démarré sur le même noeud que l'adresse VIP, il faut ajouter une contrainte au cluster :

```
# pcs constraint colocation add SiteWeb with monclusterVIP INFINITY
```

Le service Apache peut également être configuré pour démarrer après la VIP, ce qui peut être intéressant si les VHosts Apache sont configurés pour écouter l'adresse de la VIP (**Listen 192.168.1.99**) :

```
pcs constraint order monclusterVIP then SiteWeb
```

La page Apache affichée est gérée par le noeud 1 :

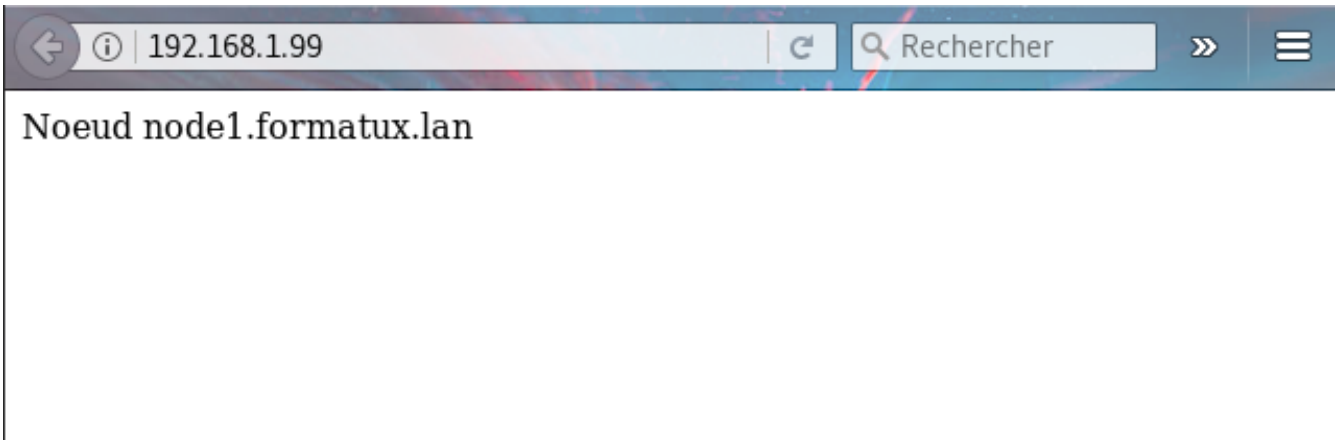


Figure 44. Le service Apache pris en charge par le noeud 1

Après le redémarrage du noeud 1, le service est pris en charge par le noeud 2 :

```
pcs status
Cluster name: moncluster
Stack: corosync
Current DC: node2-priv (version 1.1.15-11.el7_3.5-e174ec8) - partition with quorum
Last updated: Thu Jul  6 15:31:45 2017      Last change: Thu Jul  6 15:13:56 2017 by
root via cibadmin on node1-priv

2 nodes and 2 resources configured

Online: [ node2-priv ]
OFFLINE: [ node1-priv ]

Full list of resources:

 moncluster (ocf::heartbeat:IPaddr2):   Started node2-priv
 SiteWeb    (ocf::heartbeat:apache):    Started node2-priv

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

Et la page affichée provient du noeud 2 :

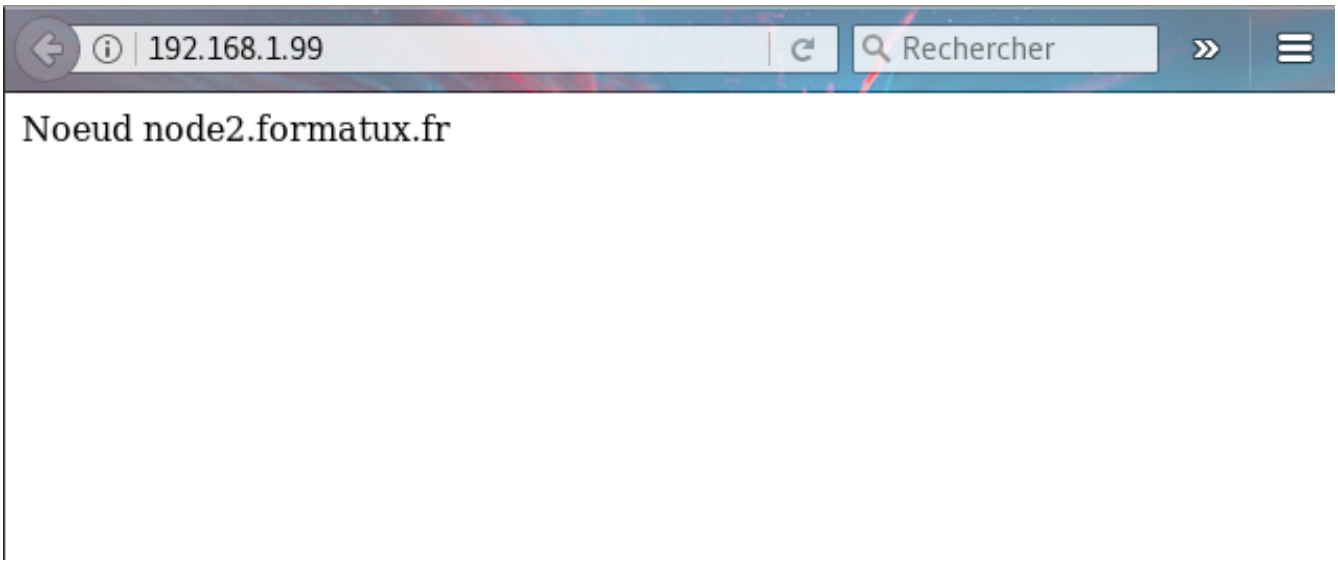


Figure 45. Le service Apache après bascule sur le noeud 2

18.6. Répliquer les données avec DRDB

DRDB permet la synchronisation, en l'absence de stockage partagé type NAS ou SAN, de données entre deux serveurs.

DRDB peut être vu comme un mirroir Réseau de type RAID 1.

Installation

Le module DRBD est inclu dans le noyau Linux, mais il est nécessaire d'installer des outils pour le contrôler.

Les outils sont disponibles depuis le dépôt **el** :

```
# yum install http://www.elrepo.org/elrepo-release-7.0-2.el7.elrepo.noarch.rpm
# yum install kmod-drbd84 drbd84-utils
```

Rendre SELinux permissif uniquement pour DRBD :

```
# yum install policycoreutils-python
# semanage permissive -a drbd_t
```

et configurer le pare-feu :

```
firewall-cmd --permanent --add-port="7789/tcp"
firewall-cmd --reload
```

Configuration des disques

A l'aide de **cfdisk**, créer une partition sur le disque (ou éventuellement la partition) dédiée sur les 2 noeuds :

```
# cfdisk /dev/sdb
```

Créer le fichier de configuration /etc/drbd.d/siteweb.res sur les 2 noeuds :

```
# vim /etc/drbd.d/siteweb.res
resource siteweb {
 protocol C;
 meta-disk internal;
 device /dev/drbd1;
 syncer {
  verify-alg sha1;
 }
 net {
  allow-two-primaries;
 }
 on node1 {
  disk   /dev/sdb1;
  address  192.168.100.100:7789;
 }
 on node2 {
  disk   /dev/sdb1;
  address  192.168.100.101:7789;
 }
}
```

Initialiser et activer le disque sur les 2 noeuds :

```
# drbdadm create-md siteweb
initializing activity log
NOT initializing bitmap
Writing meta data...
New drbd meta data block successfully created.
success

# modprobe drbd
# drbdadm up siteweb
```

Comme les disques ne sont pas encore initialisés, il faut dire à drbd lequel des deux noeuds est la référence (le primary). Depuis le noeud 1 :

```
drbdadm primary --force website
```

La synchronisation des disques peut être suivie avec la commande :

```
cat /proc/drbd
version: 8.4.9-1 (api:1/proto:86-101)
GIT-hash: 9976da086367a2476503ef7f6b13d4567327a280 build by akemi@Build64R7, 2016-12-
04 01:08:48

 1: cs:SyncSource ro:Primary/Secondary ds:UpToDate/Inconsistent C r-----
    ns:1120 nr:0 dw:0 dr:2032 al:16 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:1047356
    [>....................] sync'ed:  0.4% (1047356/1048476)K
    finish: 0:14:32 speed: 1,120 (1,120) K/sec
```

Lorsque la synchronisation est finie :

```
cat /proc/drbd
version: 8.4.9-1 (api:1/proto:86-101)
GIT-hash: 9976da086367a2476503ef7f6b13d4567327a280 build by akemi@Build64R7, 2016-12-
04 01:08:48

 1: cs:Connected ro:Primary/Secondary ds:UpToDate/UpToDate C r-----
    ns:0 nr:1065560 dw:1065560 dr:0 al:8 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:0
```

Le formatage des disques en ext4 ne permettrait pas de monter le système de fichiers en même temps sur les deux noeuds.

Pour monter le même système de fichiers sur deux noeuds en même temps, il faut utiliser des systèmes de fichiers spécialisés, type ocfs ou gfs2, ainsi qu'un gestionnaire de verroux distribués **dlm** (**Distributed Lock Manager**) qu'il faut installer sur les 2 noeuds :

```
yum install gfs2-utils dlm
```

La partition peut être formatée en gfs2 :

```
mkfs.gfs2 -p lock_dlm -j 2 -t moncluster:siteweb /dev/drbd1
It appears to contain an existing filesystem (ext4)
This will destroy any data on /dev/drbd1
Are you sure you want to proceed? [y/n]y
Discarding device contents (may take a while on large devices): Done
Adding journals: Done
Building resource groups: Done
Creating quota file: Done
Writing superblock and syncing: Done
Device:                    /dev/drbd1
Block size:                4096
Device size:               1,00 GB (262119 blocks)
Filesystem size:           1,00 GB (262117 blocks)
Journals:                  2
Resource groups:           5
Locking protocol:          "lock_dlm"
Lock table:                "moncluster:siteweb"
UUID:                      23405f79-cc0c-3dfa-8554-a0cb6dce40ad
```

Table 32. Options de la commande mkfs.gfs2

Options	Commentaires
-j n	Le nombre de journaux à créer. Un journal est nécessaire par noeuds qui monteront simultanément le système de fichiers. Il est préférable de bien réfléchir à choisir le bon nombre de noeuds maximum dès la création plutôt que d'ajouter des journaux par la suite.
-p lock_type	Spécifier le protocole de verroux à utiliser lorsqu'aucun protocole n'est spécifié au moment du montage : soit lock_dlm, soit lock_nolock.
-t clustername:lockspace	La table de verroux utilisée pour identifier le système de fichiers dans le cluster. Le clustername doit correspondre au nom donné à votre cluster durant la configuration. Les seuls membres du cluster sont autorisés à utiliser le système de fichiers. Le lockspace est un nom unique pour le système de fichiers gfs2.

18.7. Sources

- http://clusterlabs.org/doc/en-US/Pacemaker/1.1/html/Clusters_from_Scratch/
- https://binbash.fr/2011/09/19/des-clusters-avec-pacemaker/
- https://binbash.fr/2011/10/27/cluster-pacemaker-apache-actif/passif/
- https://www.sebastien-han.fr/blog/2011/07/04/introduction-au-cluster-sous-linux/
- https://www.yanx.eu/tag/pacemaker/

Glossaire

BASH
: Bourne Again SHell

BIOS
: Basic Input Output System

CIDR
: Classless Inter-Domain Routing

Daemon
: Disk And Execution MONitor

DHCP
: Dynamic Host Control Protocol

DNS
: Domain Name Service

FIFO
: First In First Out

FQDN
: Fully Qualified Domain Name

GNU
: Gnu is Not Unix

HA
: High Availability (Haute Disponibilité)

HTTP
: HyperText Transfer Protocol

ICP
: Internet Cache Protocol

IFS
: Internal Field Separator

LAN
: Local Area Network

LDIF

LDAP Data Interchange Format

NTP

Network Time Protocol

nsswitch

Name Service Switch

OTP

One Time Password

POSIX

Portable Operating System Interface

POST

Power On Self Test

RDBMS

Relational DataBase Managed System

SGBD-R

Systèmes de Gestion de Base de Données Relationnelles

SHELL

En français "coquille". Á traduire par "interface système".

SMB

Server Message Block

SMTP

Simple Mail Tranfer Protocol

SSH

Secure SHell

SSL

Secure Socket Layer

TLS

Transport Layer Security, un protocole de cryptographie pour sécuriser les communications IP.

TTL

Time To Live

TTY

teletypewriter, qui se traduit téléscripteur. C'est la console physique.

UEFI

Unified Extensible Firmware Interface

Index