Serveur Apache : installation et configuration

I. Le protocole HTTP

II. Les principaux serveurs Web

III. Installation sous Linux avec SSL

IV. La configuration

Fonctionnement d'un serveur Web ?

Un serveur Web s'appuie sur le protocole HTTP

(HyperText Transfer Protocol – port 80 – RFC 1945)

Version 0.9 (1991): uniquement destinée au transfert de données sur Internet (URL)

Version 1.x (1996 …) : permet les connexions persistantes, les requêtes multiples simultanées, l'annonce de la version HTTP supportée …

Actuellement : V1.1 reste la plus employée

Version 2.x (2015 …) : implémentation de SPDY …

Version 3 : passage au protocole QUIC

Fonctionnement d'un serveur Web ?

HTTP 1.0 : traitement des requêtes séquentielles par défaut
➔ Nécessité d'ouvrir pour chaque requête une connexion TCP

HTTP 1.1 : Technique du pipeline basée sur la connexion persistante (keep-alive) par défaut

Pipelining = envoi de plusieurs requêtes HTTP dans une seule connexion TCP sans attendre la réponse du serveur
Le serveur doit impérativement renvoyer les réponses dans l'ordre d'arrivée des requêtes

Le serveur laisse la connexion TCP ouverte après avoir traité une première requête
➔ Le navigateur du client peut alors relancer une nouvelle requête sur la même connexion

Fonctionnement d'un serveur Web ?

C'est au client d'envoyer le signal de fermeture de session

Connection: close
➔ La session doit être fermée après traitement de la requête en cours

<u>Problèmes</u>

Connexion persistante : connexion qui reste ouverte au cas où

Connexion persistante = processus = occupation mémoire

Risque d'écroulement des performances du serveur (attaque DoS)

Fonctionnement d'un serveur Web ?

Problèmes

Connexion persistante : connexion qui reste ouverte au cas où

Connexion persistante = processus = occupation mémoire

Risque d'écroulement des performances du serveur (attaque DoS)

Solutions

Réduire la durée du keep-alive (5s par défaut sur Apache)

Mettre en place un proxy inversé : c'est lui qui gérera les connexions persistantes avec les clients

Le protocole HTTP

HTTP = standard requête / réponse entre serveur et client

HTTP est totalement découplé de TCP

HTTP nécessite juste un protocole de transport (peu importe lequel)

Le client envoie sa requête HTTP sur le port 80 par défaut

Le protocole HTTP

HTTP est principalement utilisé avec TCP (protocole de transport) car:

- **beaucoup de données envoyées pour une page Web**
- **TCP implémente un contrôle de la transmission des données (gestion des retransmissions de paquets perdus)**
- **TCP permet d'obtenir les paquets dans l'ordre (assemblage des paquets pour en faire un datagramme transmis au client)**
- **TCP sait corriger les erreurs de transmission**

NB: TCP = Transmission Control Protocol

Remarques

Classement des protocoles de transport en deux types :

Orientés connexion : contrôle des données pendant la communication établie entre deux machines (le récepteur envoie des accusés de réception à l'émetteur)

Ex: TCP

Non orientés connexion : la machine émettrice envoie les données sans prévenir la machine réceptrice et cette dernière reçoit les données sans émettre d'accusé

Ex: UDP

Les requêtes HTTP

Ensemble de lignes envoyées par le navigateur Web au serveur Web :

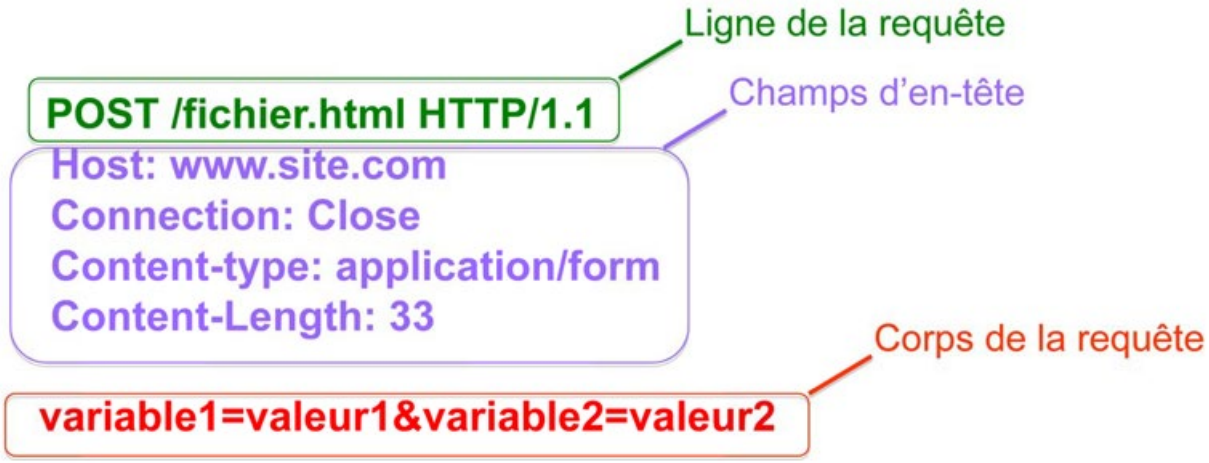
une ligne de requête

+

les champs d'en-tête

+

le corps de la requête



Les requêtes HTTP

HEAD : demande d'information sur une ressource

GET : demande la ressource

POST : ajout d'une ressource (Ex: message sur un forum)

PUT : remplacement ou ajout d'une ressource sur le serveur

DELETE : suppression d'une ressource sur le serveur

TRACE : demande au serveur de retourner ce qu'il vient de recevoir (pour test de la connexion)

OPTIONS : obtention des options de communication d'une ressource

CONNECT : permet d'utiliser un proxy

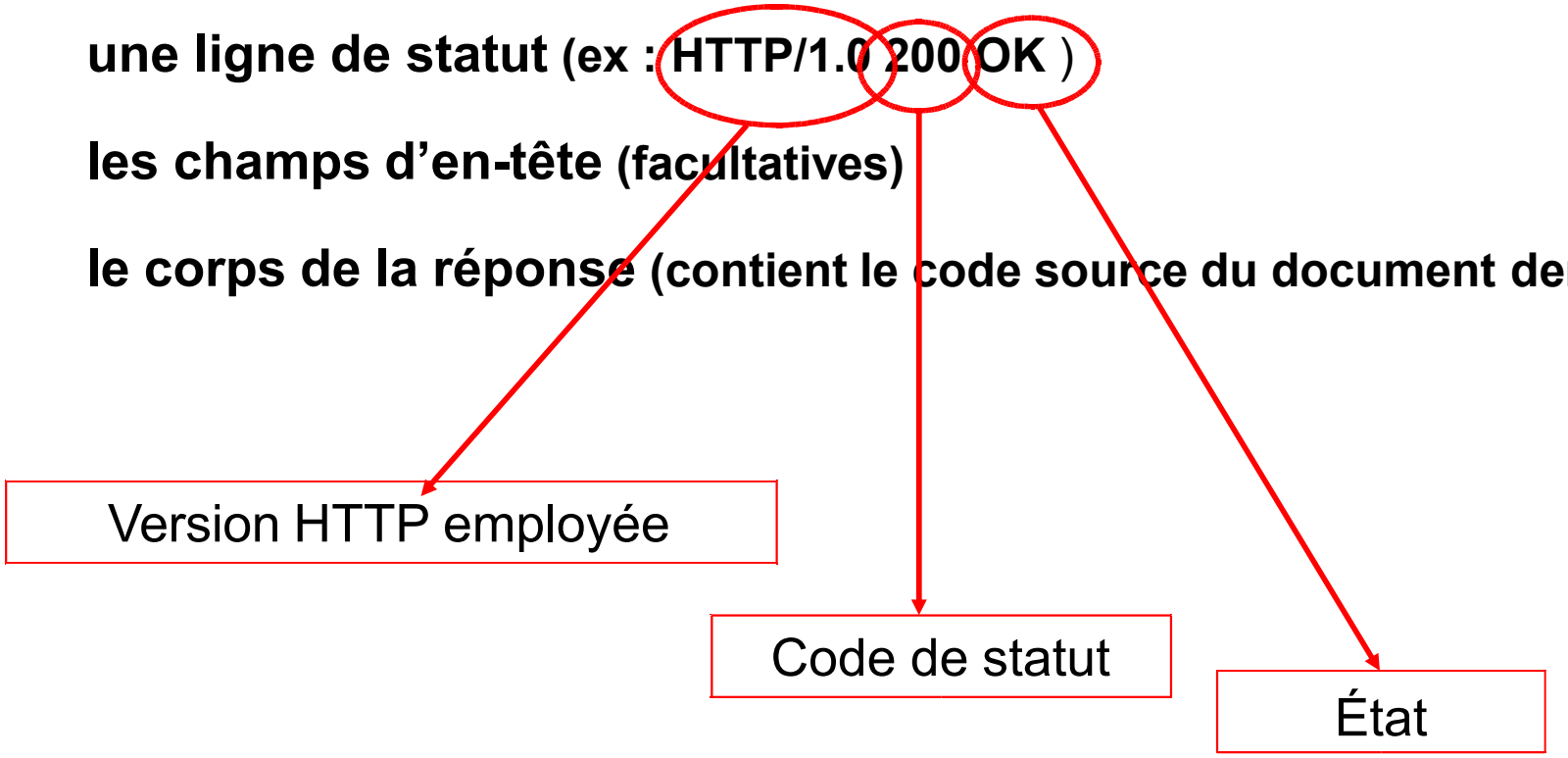
La réponse du serveur après traitement de la requête

Également structurée en trois parties :

une ligne de statut (ex : HTTP/1.0 200 OK)

les champs d'en-tête (facultatives)

le corps de la réponse (contient le code source du document demandé)



Exemple de réponse :

HTTP/1.0 200 OK
Date: Fri, 06 Jan 2017 10:00:03 GMT (date de réponse)
Server: Apache/2.4 (modèle de serveur répondant)
Content-Type: text/html (type MIME de la ressource)
Content-Length: 59 (taille en octets de la ressource)
Expires: Sat, 14 Jan 2017 00:59:59 GMT (gestion du cache du nav.)
Last-modified: Fri, 12 Aug 2016 14:21:40 GMT

<TITLE>Exemple</TITLE> <P>La première page.</P> (code de la page)

Le type MIME (Multipurpose Internet Mail Extensions)

Standard (1991) utilisé pour typer :

- les documents attachés à un courrier
- les documents transférés par le HTTP

Transaction entre un serveur web et un navigateur internet

- le serveur envoie le type MIME de ce qu'il va transmettre
- le navigateur sait comment afficher le document

Ex. d'une image GIF : Content-type: image/gif

Serveur Apache : installation et configuration

I. Le protocole HTTP

II. Les principaux serveurs Web

III. La configuration

IV. Installation sous Linux avec SSL

Les principaux serveurs web sur le marché

Apache (wamp, …) ➔ v 2.4.46

nginx (Linux, BSD, Mac OS X, Windows, Solaris) ➔ v1.20

Microsoft IIS (Internet Information Server - Microsoft) ➔ v10.0.17763 (1809)

Lighttpd (Linux, Unix, Windows)

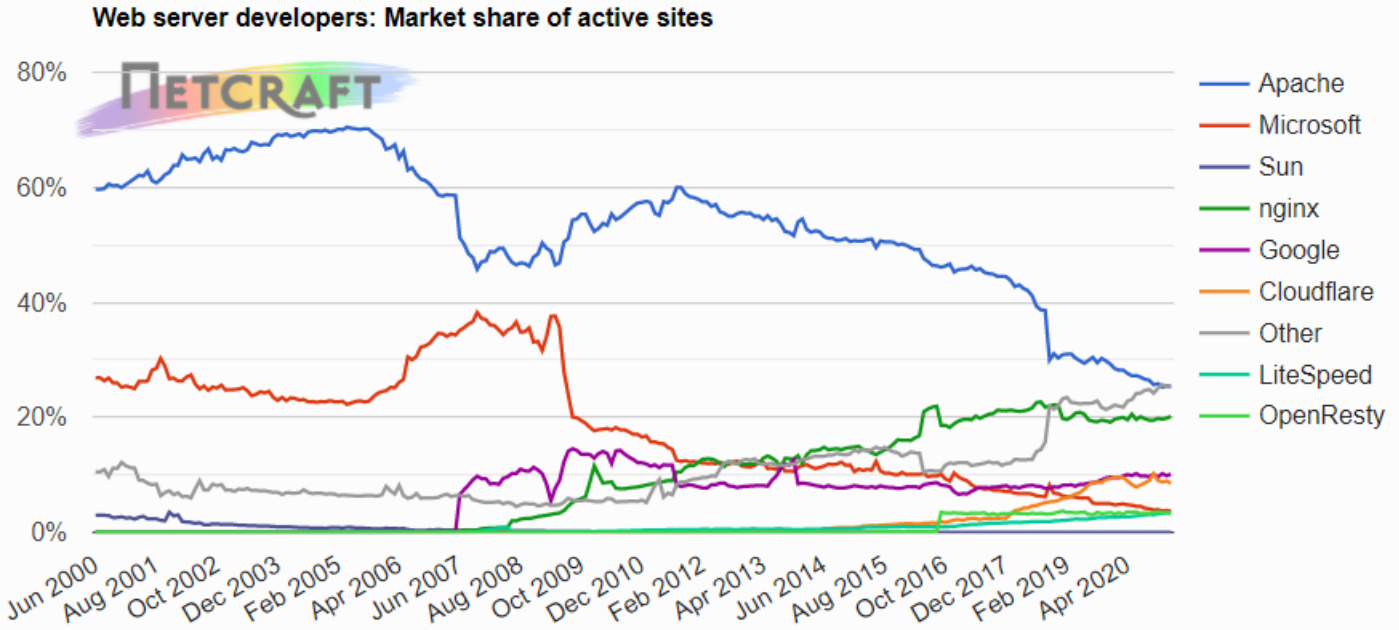
Google Web Server (pour les sites Google)

Sun Java System Web Server (Linux, BSD, OS X, Windows, Solaris)

Monkey Web Server (Linux)

WebStar (Mac OS X)

…



Developer	February 2021	Percent	March 2021	Percent	Change
Apache	50,562,247	25.48%	50,506,614	25.38%	-0.10
nginx	39,371,901	19.84%	40,111,872	20.15%	0.31
Google	19,538,140	9.85%	20,086,396	10.09%	0.25
Cloudflare	17,699,568	8.92%	16,932,600	8.51%	-0.41

Source : http://news.netcraft.com

Serveur Apache : installation et configuration

I. Le protocole HTTP

II. Les principaux serveurs Web

III. Installation sous Linux avec SSL

IV. La configuration

Apache est souvent inclus par défaut dans la distribution

Télécharger le package
- **rpm (fedora, red-hat, …)**
- **deb (debian)**
- **…**

Puis
- **rpm -ivh package1.rpm**
- **apt-get install package1**
- **…**

Pour finir, lancer une mise à jour
- **rpm –Uvh package1.rpm**
- **apt-get upgrade**
- **…**

Installer Apache2 avec SSL sous Debian

✓ **Téléchargement et installation d'Apache2 :**

 # apt-get install apache2 apache2-common apache2-doc

✓**Téléchargement du serveur openSSL:**

 # apt-get install openssl

✓**Activation du mode Apache SSL:**

 # a2enmod ssl

Sécurité

Apache ne doit pas être lancé par root

En cas de détournement d'Apache
➔ gros problème de sécurité si exécution en tant que root

<u>Solutions</u> (debian):

Apache2 : www-data:www-data est créé automatiquement

Sécurisation avec SELinux

Serveur Apache : installation et configuration

I. Le protocole HTTP

II. Les principaux serveurs Web

III. Installation sous Linux avec SSL

IV. La configuration

Les fichiers de configuration Apache2 sous Debian

Permettent de séparer la configuration en plusieurs parties (/etc/apache2/):

httpd.conf : fichier utilisé par apacheV1, parfois conservé vide dans ApacheV2 pour assurer la rétrocompatibilité (vide en général)

apache2.conf : fichier principal de configuration à partir duquel les autres sont chargés

ports.conf : contient la directive listen qui spécifie les adresses et les ports d'écoutes

mods-available : contient la liste des modules d'apache installés
mods-enabled : contient la liste des modules utilisés
sites-available : contient la liste des vhosts installés
sites-enabled : contient la liste des vhosts utilisés

Hébergement de plusieurs sites sur un même serveur Web

Le problème

Plusieurs sites et 1 seul port à l'écoute (port 80)

Les solutions

créer des sites virtuels

~~**forcer la prise en compte de différents fichiers de configuration httpd.conf (plus du tout employé)**~~

Les sites virtuels

Ils sont appelés par les clients sous différents noms ou @ IP.

2 possibilités :

-sites virtuels basés sur l'IP : le serveur dispose de plusieurs adresses IP chacune associée à un nom de site (DNS)

-sites virtuels basés sur le nom : le serveur écoute une seule adresse IP, à laquelle sont associés plusieurs noms de sites qui seront utilisés dans les URL clientes

Enregistrement dans un DNS

192.168.9.10	serveurweb	(type A)
192.168.9.10	www.site1.eu	(type CNAME)
192.168.9.10	www.site2.eu	(type CNAME)
192.168.9.10	site1.eu	(type CNAME)
192.168.9.10	site2.eu	(type CNAME)

Enregistrement dans un DNS

Nécessaire pour chaque serveur virtuel :

- **pour être visible par les clients**
- **résolu sur l'adresse IP de la machine qui les héberge**

Possibilité d'utiliser un fichiers hosts pour test local

Les fichiers de configuration Apache2 sous Debian

Fichiers

/etc/apache2/sites-available/monsite.conf ➔ définit le virtualhost

/var/www/html/monsite ➔ héberge le code du site

Activation du site

 # a2ensite monsite
➔ crée un lien symbolique dans /etc/apache2/sites-enabled qui pointe vers /etc/apache2/sites-available/monsite (à défaut, le site n'est pas publié)

Paramétrer Apache2 avec SSL sous Debian

Générer une clef privée (sert au déchiffrement)

Générer un certificat (appairé à la clef privée + clef publique) pour chaque site Web à chiffrer (certificat signé par une autorité de certification - CA -)

Modifier/vérifier les fichiers suivants pour prendre en compte SSL:

/etc/apache2/apache2.conf

qui appelle

include* /etc/apache2/mods-enabled/*.*load

include* /etc/apache2/mods-enabled/*.*conf

Paramétrer Apache2 avec SSL sous Debian

Déclarer le site virtuel correspondant au site disponible en https.

Pour cela, créer le fichier :

/etc/apache2/sites-available/monsite-ssl.conf

```
NameVirtualHost test
<VirtualHost *:443>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html/test/

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

    <Directory /var/www/html/test/>
        Options -Indexes FollowSymLinks MultiViews
        AllowOverride None
                <RequireAll>
                        Require all granted
                </RequireAll>
     </Directory>

        ErrorLog /var/log/apache2/error.log
        # Possible values include: debug, info, notice, warn, error, crit,
        # alert, emerg.
        LogLevel warn
        CustomLog /var/log/apache2/access.log combined
        ServerSignature On
        SSLEngine on
        SSLCertificateFile /etc/apache2/ssl.crt/monsite.crt
        SSLCertificateKeyFile /etc/apache2/ssl.key/monsite.key
</VirtualHost>
```

Gestion de la sécurité : répertoire au cas par cas

Possibilité de gérer les droits pour chaque sous-répertoire :

➡ **fichier .htaccess**

Dans httpd.conf (ou le fichier de conf de l'hôte virtuel):

utilisation sous <directory> de AllowOverride

AllowOverride permet de redéfinir localement les droits d'accès aux fichiers :

- **AllowOverride None : .htaccess ignoré**
- **AllowOverride All : .htaccess appliqué**

Le fichier .htaccess

Permet de :

- protéger l'accès aux fichiers par mot de passe
- protéger l'accès aux répertoires par mot de passe
- ajouter un mime-type
- définir des pages d'erreurs personnalisées
- portabilité

Le fichier .htaccess

- il est placé dans le répertoire où il doit agir

- un fichier .htaccess peut être placé dans un répertoire fils

- le fichier .htaccess du répertoire père reste actif tant que les fonctionnalités n'ont pas été réécrites au niveau du répertoire fils

Page par défaut

Vérifier la(les) valeur(s) de *DirectoryIndex* dans httpd.conf

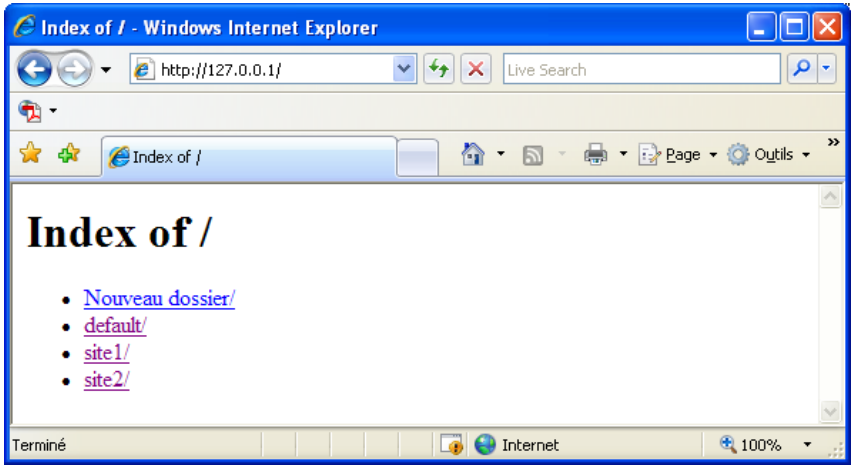
DirectoryIndex index.php index.php3 index.html index.htm index.html.var

Fichier index.html absent :

Affichage de la liste des fichiers du répertoire

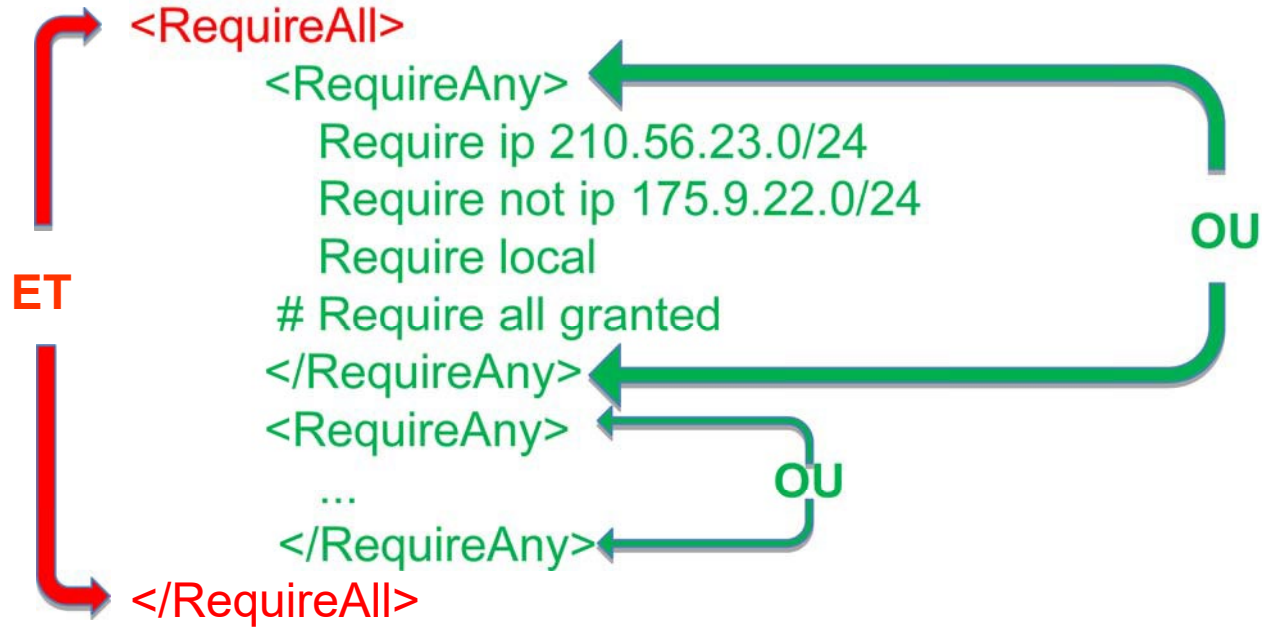
Esthétique : Pas satisfaisant

Sécurité : Faille de sécurité potentielle



Options -Indexes

Les fichiers de configuration Apache2 sous Debian



<u>Les directives de conteneur d'autorisation</u>:

<RequireAll> : groupe de règles qui doivent être toutes vérifiées
<RequireAny> : il suffit d'un règle vérifiée pour que le groupe s'applique
<RequireNone> : groupe de règles dont aucune ne doit être vérifiée

not permet d'inverser un critère

Les fichiers de configuration Apache2 sous Debian

Conteneurs d'autorisation

Les directives de conteneur d'autorisation `<RequireAll>`, `<RequireAny>` et `<RequireNone>` peuvent être combinées entre elles et avec la directive `Require` pour confectionner une logique d'autorisation complexe.

L'exemple ci-dessous illustre la logique d'autorisation suivante. Pour pouvoir accéder à la ressource, l'utilisateur doit être l'utilisateur `superadmin`, ou appartenir aux deux groupes LDAP `admins` et `Administrateurs` et soit appartenir au groupe `ventes` ou avoir `ventes` comme valeur de l'attribut LDAP `dept`. De plus, pour pouvoir accéder à la ressource, l'utilisateur ne doit appartenir ni au groupe `temps`, ni au groupe LDAP `Employés temporaires`.

```
<Directory "/www/mydocs">
    <RequireAll>
        <RequireAny>
            Require user superadmin
            <RequireAll>
            Require group admins
            Require ldap-group cn=Administrators,o=Airius
                <RequireAny>
                Require group sales
                Require ldap-attribute dept="sales"
                </RequireAny>
            </RequireAll>
        </RequireAny>
        <RequireNone>
            Require group temps
            Require ldap-group cn=Temporary Employees,o=Airius
        </RequireNone>
    </RequireAll>
</Directory>
```

Source :
https://httpd.apache.org/docs/2.4/fr/mod/mod_authz_core.html#logic

Les fichiers de configuration Apache2 sous Debian

(/etc/apache2/sites-available/monsite)

Sous apache 2.2

```
<VirtualHost *:80>

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/monsite
        ServerName www.monsite.univ-orleans.fr
        ServerAlias monsite.univ-orleans.fr
<Directory />
      Order allow,deny
      deny from all
      AllowOverride none
      Options -Indexes
  </Directory>
  <Directory /var/www/monsite>
        Options -Indexes
        AllowOverride All
        Order deny,allow
        allow from 210.56.23.0/2
        allow from 175.9.22.0/24
        allow from 127.0.0.1
        deny from all
     # allow from all
   </Directory>

ErrorLog /var/log/apache2/error.log
LogLevel warn
CustomLog /var/log/apache2/access.log combined

</VirtualHost>
```

Sous apache 2.4

```
<VirtualHost *:80>

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/monsite
        ServerName www.monsite.univ-orleans.fr
        ServerAlias monsite.univ-orleans.fr
    <Directory "/">
      Require all denied
      AllowOverride none
      Options -Indexes
    </Directory>
   <Directory "/var/www/monsite">
    AllowOverride All
    Options -Indexes
     <RequireAll>
        <RequireAny>
           Require ip 210.56.23.0/24
           Require ip 175.9.22.0/24
           Require local
         # Require all granted
        </RequireAny>
     </RequireAll>
    </Directory>

# LogLevel info ssl:warn
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

</VirtualHost>
```