Apache

Un serveur web

Licence Pro. SIL

Année 2007-08

Introduction

Apache est un serveur Web. Fondamentalement il délivre des fichiers par le protocole http à des clients. Les clients sont nombreux :

- des navigateurs graphiques
- des navigateurs texte
- toute sorte de programmes (par exemple des robots).

Fonctions d'un serveur web

Le serveur web se charge par exemple de :

- de la partie de contrôle, d'organisation et de gestion commune à tout type de serveurs (d'applications, de mails, etc ...) :
  - de la liaison qui permet de définir par quels ports les échanges se font.
  - de la journalisation permettant de garder une trace des requêtes
  - de la gestion des performances (nombre de clients acceptés, temps pour garder une connexion ouverte,...)

Fonctions d'un serveur web

- mais il se charge aussi de la gestion spécifique à un serveur web :
  - de la sécurité (authentification, cryptographie etc. . .)
  - du lancement de programmes externes au serveur pour constuire certaines réponses aux requêtes
  - de la négociation qui permet de retourner un contenu adapté à l'émetteur de la requête, selon sa langue, son type etc. . .
  - de la correspondance entre urls et système de fichiers
  - de la réécriture d'urls
  - de la virtualisation qui permet de faire tourner plusieurs serveurs en un seul.

Syntaxe de base

Apache se configure en positionnant des directives en mode texte :

- une directive par ligne, si une directive doit être écrite sur plusieurs lignes : un antislash en dernier caractère de la ligne
- les lignes qui commencent par un # sont des commentaires (pas de commentaire en fin de ligne !) ;
- les lignes blanches et les espaces en début de ligne sont ignorés
- Apache n'est pas sensible à la casse pour le nom des directives, mais il l'est pour les paramètres de ces directives.

Syntaxe de base

Les directives peuvent être placées à l'intérieur de sections :

- elles spécifient un sous-arbre de la zone gérée par Apache et permettent d'écrire des directives spécifiques pour cette zone
- section `Directory` : correspond à un ou des sous-arbres dans le système de fichiers de la machine physique sur laquelle tourne le serveur web
- section `Location` : correspond à un ensemble d'URLs pour lesquelles le serveur reçoit une requête
- section `VirtualHost` : correspond à un serveur virtuel
- ...
- des sections conditionnelles (`IfDefine`, `IfModule`, ...) permettent également de paramétrer finement le comportement du serveur web

Fichiers de configuration

- le fichier principal est `httpd.conf` :
  - souvent éclaté en différents fichiers de configuration
  - la directive `Include` qui permet d'ajouter d'autres fichiers de configuration au fichier principal
- fichiers lus au démarrage du serveur :
  - toute modification doit donc être suivie du redémarrage du serveur pour être prise en compte !
- le fichier `mime.types` contient les types des documents mime :
  - les types MIME sont un moyen de décrire le contenu d'un fichier par un type majeur et un type mineur : text/html, text/css, application/ogg, image/bmp, ...
  - `mime.types` fait une correspondance entre un type MIME et une extension de fichier

Les modules

Apache est modulaire :

- il tourne avec un certain nombre de fonctionnalités de bases
- les extensions sont dans des modules externes qu'il faut charger si on veut les utiliser
- pour charger un nouveau module :
  - si Apache a été compilé avec le support dynamique des modules, les modules peuvent être simplement chargés avec la directive `LoadModule` (Dynamic Shared Objects DSO)
  - sinon, cela nécessite une recompilation complète d'Apache avec les nouveaux modules
- les sections conditionnelles `IfModule` qui permettent de paramétrer le comportement d'Apache en fonction du chargement d'un module

Paramétrage fin

- On définit dans la configuration quelles sont les zones du système de fichiers de la machine qui sont sous la "responsabilité" du serveur
- Basiquement, il s'agit du sous-arbre qui se trouve à la racine

```
DocumentRoot "/usr/local/apache2/htdocs"
```

- les directives placées à la racine du fichier de configuration s'appliquent pour le serveur dans son ensemble. Par exemple :

```
ServerAdmin you@example.com
```

Paramétrage fin

- pour paramétrer le comportement pour une sous-partie du serveur, on peut placer les directives à l'intérieur d'une des sections `Directory`, `DirectoryMatch`, `Location`, `LocationMatch`, `Files` et `FilesMatch`. Par exemple :

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

Paramétrage fin

- le paramétrage peut être délégué à des fichiers `.htaccess` qui sont placés directement dans les répertoires concernés.
- nécessité d'autoriser les `.htaccess` :

```
<Directory /home/*/public_html>
  AllowOverride FileInfo AuthConfig Limit Indexes
</Directory>
<Directory />
  AllowOverride None
</Directory>
```

Description d'une directive

Une directive est généralement présentée dans la documentation en indiquant :

- sa description
- sa syntaxe : la liste de ses paramètres et leur type ;
- sa valeur par défaut (le cas échéant)
- son contexte : la section où on peut la trouver !
- son module : la directive fait elle partie d'un module de base, ou bien du noyau interne d'Apache, ou bien d'une extension,. . .
- et quelques autres éléments : sa compatibilité avec les différentes versions d'Apache, quelles conditions supplémentaires sont requises pour la prise en compte de la directive dans un fichier `.htaccess`, . . .

Liaison

- Apache se lie à un port et à une adresse sur la machine et se met en attente de requêtes entrantes
- Apache accepte d'écouter sur un ou plusieurs ports spécifiques, ou sur une seule ou plusieurs adresses, ou encore une combinaison des deux
- les serveurs virtuels permettent de faire répondre Apache de manière différente en fonction de l'adresse IP, du nom ou du port

Liaison (2)

- Le serveur utilise la directive `Listen` pour n'accepter que des requêtes provenant de ports spécifiques ou d'une combinaison adresse IP + port passés en argument.
- Par exemple, pour que le serveur accepte les connexions à la fois sur les ports 80 et 8000, spécifiez :

```
Listen 80
Listen 8000
```

- Pour qu'Apache accepte les connexions sur deux combinaisons adresses + ports, spécifiez :

```
Listen 192.170.2.1:80
Listen 192.170.2.5:8000
```

La négociation de contenus

- pour une même requête http, plusieurs documents peuvent être retournés. Par exemple, pour une requête `http ://www.mon-asso.fr`, le serveur web va retourner un document qui est à la racine du site web. Mais quel document ? L'index en français ou en anglais ? un .html ou un .txt ?
- Le choix est effectué sur le serveur en fonction d'éléments dans l'entête de la requête du client. Voici un exemple d'entête de requête :

```
Accept-Language : fr ; q=1.0, en ; q=0.5
Accept : text/html ; q=1.0, text/* ; q=0.8,
         image/gif ; q=0.6, image/jpeg ;
         q=0.6, image/* ; q=0.5, */* ; q=0.1
```

La négociation de contenus

- On appelle ressource une entité conceptuelle qui correspond à une URI,
- et variantes les différentes représentations possibles pour cette ressource.
- S'il existe plusieurs variantes, alors la ressource est négociable :
  - soit il possède un fichier associé à la ressource qui contient une représentation pour chacune des variantes. On appelle ce fichier une table de types.

La négociation de contenus

Voilà ce qu'on pourrait trouver dans un fichier `foo.var`

```
URI : foo

URI : foo.en.html
Content-type : text/html
Content-language : en

URI : foo.fr.de.html
Content-type : text/html ;charset=iso-8859-2
Content-language : fr, de
```

La négociation de contenus

- soit c'est le serveur lui-même qui construit sa table de types à partir des extensions des fichiers disponibles pour la ressource et en fonction de `mod_mime` :
  - mais il faut que l'`Option Multiviews` ait été posée sur la partie du site où se trouve la ressource

La journalisation

Le répertoire `<serverRoot>/logs` contient différents journaux d'activités mis à jour par le serveur Apache :

- `error_log` contient tous les problèmes rencontrés : les problèmes systèmes et les problèmes concernant les documents publiés
  - stocké dans le fichier défini par

    ```
    ErrorLog logs/error_log
    ```

  - contient les informations de niveau `emerg`, `alert`, `crit`, `error`, `warn`, `notice`, `info`, ou `debug`

    ```
    LogLevel warn
    ```

La journalisation

- `acces_log` contient des informations concernant toutes les requêtes traitées par le serveur
  - stocké dans le fichier défini par la directive `CustomLog`

    `CustomLog désignationDuFichierLog formatUtilisé`
  - où le format donne les informations conservées :

    `LogFormat ''Adresse IP du client -> %h'' monFormat`

Serveurs virtuels

- accéder à plusieurs serveurs web virtuels qui fonctionnent en fait sur une même machine physique. Par exemple, proposer un accès à la machine `www.mon-asso.fr` et un autre à la machine `www.mon-autre-asso.org`.
- On peut avoir des serveurs virtuels basés sur IP, i.e. chaque machine virtuelle possède une adresse IP différente,
- ou bien des serveurs virtuels basés sur le nom, i.e. une même IP correspond à plusieurs noms de machines
- le fait que ces serveurs virtuels tournent en définitive sur la même machine physique est complètement transparent pour l'utilisateur final.

Avec des serveurs virtuels

- `Listen` sert simplement à indiquer au serveur principal sur quels adresses et ports écouter
- si aucune section `<VirtualHost>` n'est utilisée, le serveur répondra de la même manière pour toutes les requêtes qu'il recevra
- les sections `<VirtualHost>` peuvent être utilisées pour qu'Apache réagisse différemment selon que la requête est destinée à telle adresse ou à tel port
- une section `VirtualHost` peut comprendre simplement un ensemble de directives (négociation de contenu, fichiers de journalisation,...), mais il peut aussi contenir toutes les sections `Directory`, `Location`, `Files`, ... pour paramétrer le comportement du serveur virtuel.

Plusieurs serveurs

- Une autre possibilité est de faire tourner plusieurs démons httpd (un pour chaque serveur virtuel).
- Cette solution est la plus sûre,
- mais c'est bien sûr aussi une solution plus coûteuse en terme de performances.

La correspondance des URLs et du système de fichiers

La directive `DocumentRoot` détermine la racine du site web. C'est ici qu'on trouve le fichier qui est retourné lorsque la requête s'adresse à la racine du site `http://www.mon-asso.org`
Exemple :

```
DocumentRoot /usr/web
```

L'accès à `http://www.mon-asso.org/index.html` correspond à `/usr/web/index.html`

La correspondance des URLs et du système de fichiers

- **Serveurs virtuels** : on peut définir un document racine spécifique pour chaque serveur virtuel.
- **Directive** `Alias` : un alias permet de définir comment on réécrit un chemin dans une URL en un autre chemin dans le système de fichiers.
  Exemple :
  ```
  Alias /docs /var/web
  ```
  Ainsi, l'URL
  `http://www.example.com/docs/dir/file.html`
  correspondra au fichier `/var/web/dir/file.html`
- **Directive** `ScriptAlias` idem sauf que les document positionnés à l'endroit spécifié seront vus comme des scripts CGI
- les directives `AliasMatch` et `ScriptAliasMatch` fonctionnent avec des expression régulières en paramètre.

La correspondance des URLs et du système de fichiers

- **Liens symboliques** pour suivre les liens symboliques spécifiés dans les répertoires. Avec l'`Option FollowSymLinks`
- **Les répertoires des utilisateurs** C'est la directive `UserDir` qui permet d'associer un chemin d'accès à un répertoire dédié aux documents publics à l'intérieur du home directory d'un utilisateur. Ainsi, généralement,

  `UserDir public_html`

  permet de faire correspondre

  `http ://www.mon-asso.org/ parrain` à

  `parrain/public_html/`
- Mais aussi :
  - la redirection d'URLs (en cas de déménagement des documents), la directive `Redirect`
  - la réécriture d'URLS...

Gestion des performances

Apache peut contrôler les ressources qu'il alloue.

- Les directives `LimitRequestBody`, `LimitRequestFields`, `LimitRequestFieldSize`, `LimitRequestLine` permettent de limiter les ressources que va utiliser Apache pour lire une requête d'un client.
- Les directives `RLimitCPU`, `RLimitMEM` et `RLimitNPROC` permettent de limiter les ressources alloués aux process fils d'apache comme les scripts CGI ou les SSI.

La sécurité

Elle se situe à plusieurs niveaux :

- le contrôle des accès aux documents dans les fichiers de configuration principaux
- et via les fichiers `.htaccess`
- l'accès par authentification, avec la restriction sur les domaines ou adresses des machines clientes ;
- l'utilisation de connexions sécurisées, la gestion des certificats, etc. . .via une interface vers la librairie OpenSSL.

Autorisation ou interdiction d'accès

Pour autoriser ou interdire l'accès de certaines pages Web à des adresses IP, à des noms d'hôtes,... :

- les commandes `allow from ...` (pour autoriser) et `deny from ...` (pour interdire). Par exemple :
  - `allow from all` : autorise l'accès à tout le monde,
  - `allow from 12.23` : autorise l'accès aux adresses IP commençant par 12.23,
  - `deny from .fr` : interdit l'accès à tous les noms d'hôtes du domaine .fr,
  - `deny from belgique.iut-lens.univ-artois.fr` : interdit l'accès à l'hôte belgique.iut-lens.univ-artois.fr.

Autorisation ou interdiction d'accès

- la directive `Order` permet de spécifier quelle est la propriété la plus prioritaire entre `allow` et `deny` :

  `Order deny, allow`

  spécifiera que `allow` est prioritaire (il a le dernier mot) par rapport à `deny`.

  `Order allow, deny`

  spécifiera que `deny` est prioritaire par rapport à `allow`.

Autorisation ou interdiction d'accès

Exemples :

```
<Directory rep1>
  Order allow, deny
  Allow from all
  Deny from 123.34.56.45
</Directory>


<Directory rep2>
  Order deny, allow
  Allow from all
  Deny from 123.34.56.45
</Directory>
```

Authentification par mot de passe

- `AuthType` pour indiquer le type d'accès utilisé (`Basic` ou `Digest`),
- `AuthName` pour indiquer le domaine dans lequel les noms et les mots de passe doivent être valides.
- `AuthUserFile` indique le fichier contenant les noms d'utilisateurs et leur mots de passe,
- `AuthGroupFile` indique un fichier dans lequel est défini un ensemble de groupes d'utilisateurs.
- `Require` permet d'indiquer quels sont les utilisateurs qui pourront accéder aux documents du répertoire après authentification.

Authentification par mot de passe

Exemple :

```
<Directory rep>
  AuthType Basic
  AuthName secret
  AuthUserFile /home/jean/.htpasswd
  Require valid-user
</Directory>
```

Les scripts CGI

La directive `ScriptAlias` permet d'indiquer au serveur où se trouvent les scripts CGI. La syntaxe est

```
ScriptAlias cheminCGI répertoireCGI
```

où `cheminCGI` indique le chemin utilisé dans l'url pour atteindre les scripts CGI et `répertoireCGI` indique le répertoire où se trouvent les scripts CGI. Ainsi, la requête `http://serveur/cheminCGI/script` fera s'exécuter le script `répertoireCGI/script`.
L'option `ExecCGI` permet d'autoriser l'exécution de scripts CGI.

Les server-side scripts

- Il est possible de mettre du code dans les différents documents publiés qui sera interprété par le serveur.
- Ce code doit être filtré par des interpréteurs avant que le serveur ne le retourne au client.
- Il peut être également filtré lorsqu'il est envoyé par le client (requête `POST` ou `PUT`) et reçu par le serveur.

Les server-side scripts

- On peut placer dans une section `Files` ces filtres-interpréteurs. Par exemple, la directive suivante :

```
<Files *.php>
  SetOutputFilter PHP
  SetInputFilter PHP
</Files>
```

  permet d'indiquer que les fichiers portant l'extension `.php` doivent être filtrés par l'interpréte PHP.
- L'option `Includes` permet l'exécution du code côté serveur.