Apunts d'Apache

Juan Aguilera

2 de juny de 2016

Índex

1 Instal·lació del paquet apache **3**

2 Configuració d'apache **3**
2.1 fitxer de configuració principal 3
2.2 fitxer de configuració de ports 3

3 Principals directives **4**
3.1 Context 4
3.2 Taula de directives 4
3.3 Taula de contextes 8

4 Site per usuari **8**

5 VirtualHost **9**
5.1 Directives associades 9
5.1.1 VirtualHost 9
5.1.2 ServerName 10
5.1.3 ServerAlias 10
5.1.4 DocumentRoot 10
5.2 VirtualHosting basat en ip 10
5.3 VirtualHosting basat en nom 10
5.4 VirtualHosting basat en port 10

6 Control de l'accés als directoris **10**

7 Crear regles en Apache **11**
7.1 Operadors 11

8 Apache com a proxy **11**
8.1 Forward Proxy 11
8.1.1 Acl's en Apache 12
8.1.2 Proxy com a servidor cau 13

8.1.3 Proxy com a servidor cau 15
8.2 Reverse Proxy o gateway 19

A FAQ i Definicicions generals **20**

B VirtualHost FAQ **21**

1 Instal·lació del paquet apache

Versió d'apache

```
[^O%]juan@cilo:~$ sudo apache2 -v
[sudo] password for juan:
Server version: Apache/2.4.10 (Debian)
Server built:   Aug 28 2015 16:28:08
```

d'entre les novetats de la versió apache 2.4 respecte les versions anteriors cap destacar que la directiva *NameVirtualHost* ja no és útil i és *deprecated*. Per més info, veure les new features d'apache a la pàgina oficial[1].

2 Configuració d'apache

2.1 fitxer de configuració principal

jugant amb l'index.html Traient l'index.html de /var/www podem veure els directoris que es troben a www. Això s perque deu haver-hi per defecte un "DirectoryIndex index.html" i un "Option Indexes" si esborres el fitxer es mostra tot. Això és degut a que en apache2.conf ho han definides tres directives *Directory* per configurar com s'ha de comportar el servei quan s'intenta accedir a aquesta ruta:

```
<Directory />
        Options FollowSymLinks
        AllowOverride None
        Require all denied
</Directory>

<Directory /usr/share>
        AllowOverride None
        Require all granted
</Directory>

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

#<Directory /srv/>
#       Options Indexes FollowSymLinks
#       AllowOverride None
#       Require all granted
#</Directory>
```

2.2 fitxer de configuració de ports

És on especifiquem els ports on ha d'escoltar el dimoni.

3 Principals directives

Tota la documentació sobre qualsevol directiva es pot trobar en la pàgina de referència de directives d'apache

Per entendre les directives a apache és important saber en quins contextos es poden utilitzar. Hi han els següents:

3.1 Context

Indica en quin àmbit o àmbits s'aplica la directiva:

server config This means that the directive may be used in the server configuration files (e.g., apache2.conf), but not within any `<VirtualHost>` or `<Directory>` containers. It is not allowed in .htaccess files at all.

virtual host This context means that the directive may appear inside `<VirtualHost>` containers in the server configuration files.

directory A directive marked as being valid in this context may be used inside `<Directory>`, `<Location>`, `<Files>`, and `<Proxy>` containers in the server configuration files, subject to the restrictions outlined in Configuration Sections.

.htaccess If a directive is valid in this context, it means that it can appear inside per-directory .htaccess files. It may not be processed, though depending upon the overrides currently active.

3.2 Taula de directives

Directiva	**Context**	**Descripció**
AllowOverride	directory	Què directives es poden posar en el fitxer *.htacces*. Per defecte el seu valor és none.
DirectoryIndex	server config, virtual host, directory, .htaccess	Diu què mostrar quan l'usuari demana una URL que fa referència a un directori i acabant amb una /
Options	server config, virtual host, directory, .htaccess	Configura qué característica està disponible en un directory en un àmbit en particular.
NameVirtualHost		**DEPRECATED en apache 2.4**

DocumentRoot	Server config; VirtualHost	Les adreces demanades pel client seràn relatives a aquesta ruta. Httpd servirà fitxers relatius a la ruta.
ServerRoot	server config	indica el directoi base per la configuració d'apache.
ServerName	server config; Virtualhost	Nom de servidor i port que el servei utilitza per identificarse a si mateix.
ServerAlias	Virtualhost	Alias per el nom del servidor.
VirtualHost	server config	context per configurar un virtualhost.
Directory	server config; VirtualHost	Enclose a group of directives that apply only to the named file-system directory, sub-directories, and their contents
Location	server config; VirtualHost	Applies the enclosed directives only to matching URLs.
Files	server config; VirtualHost; Directory; .htaccess	Contains directives that apply to matched filenames. Exemple: `<Files ~ ''\.(gif|png)$''>`.

`<Limit>`	directory, .htaccess	Serveix per restringir l'efecte del control d'accés als mètodes que definim en aquesta directiva. Exemple:
		`<Limit POST PUT DELETE>`
		`  Require valid-user`
		`</Limit>`
		estem restringir l'accés als usuaris que s'autentifiquin per els mètodes `POST`, `PUT` i `DELETE`. Mirar també LimitExcept que fa el mateix però exceptuant més que restringint mètodes.

Allow, Deny, Order	Directory; .htaccess	**DEPRECATED**. Substituit pels *Require*, *RequireAny—All—None*

Require	directory, .htaccess	Requeriment per a que s'acompleixi una directiva d'autorització. Mirar els exemples de la documentació d'Apache.
RequireAll	Directory; .htaccess	totes les directives Require dintre d'aquesta s'han de complir.
RequireAny	Directory; .htaccess	Alguna de les directives *Require* dintre d'aquesta s'han de complir. Si hi han directives *Require* que no estan dintre de les *Require** aleshores s'interpreten com a que es troben en el *RequireAny*

Options Pot ser *None* o més d'entre molts valors d'entre els que destaquen:

- *FollowSymlinks*: Es permeten les peticions a enllaços simbòlics que hi hagin en el directori.
- *Indexes*: Si demanem una URL corresponent a un directori, mostra el que haguem escollit amb la directiva *DirectoryIndex* (en cas que no hi hagi DirectoryIndex).
- *All*: Tots.

Nota: Amb els signes + o - davant dels diferents valors que pot prendre *Opions* voldrà dir que afegim o traiem llurs valors (si n'hi ha algun amb signe o han d'estar tots).

DirectoryIndex Lista de recursos que el cliente puede encontrar en un directorio dado cuando el cliente solicita la URL de ese directorio acabada en /. En la lista podemos poner ficheros que estan en el directorio o rutas relativas que estan dentro del directorio. Al cliente se le devolverá el primero de los recursos que se encuentre en esa lista.
Está en estos contextos: server config, virtual host, directory, .htaccess.
Exemple:

DirectoryIndex index.html
then a request for
`http://example.com/docs/`
would return
`http://example.com/docs/index.html`

if it exists, or would list the directory if it did not.
Exemple (Note that the documents do not need to be relative to the directory):

`DirectoryIndex index.html index.txt /cgi-bin/index.pl`

would cause the CGI script

`/cgi-bin/index.pl`

to be executed if neither index.html or index.txt existed in a directory.
Si no encuentra ninguno de esos archivos, y está establecida la opción Options Indexes para ese directorio, el servidor generará y devolverá una lista, en formato HTML, de los subdirectorios y archivos del directorio. El valor predeterminado, almacenado en `/etc/apache2/apache2.conf`, es ≪index.html index.cgi index.pl index.php index.xhtml≫. Por tanto, si Apache2 encuentra un archivo en un directorio solicitado que se ajusta a alguno de esos nombres, se mostrará el primero de todos.

3.3 Taula de contextes

Context	Directives que podem posar-hi
Server config	Directives sobre el dimoni i el servei; ports d'escolta; Includes d'altres fitxers; Directoris generals; Què fer amb determinats fitxers genèrics.
.htaccess	Directives de tota mena que apliquen al directori on es troba el fitxer.

VirtualHost	`<VirtualHost *:80>`
	`DirectoryIndex index.html index.txt`
	`Option Indexes`
	`DocumentRoot /var/www/Bash`
	`ServerName www.bash.aguileraj.net`
	`ServerAlias www.aguileraj.net`
	`ServerPath error.html`
	`<Directory ps1>`
	`. . .`
	`</Directory>`
	`</VirtualHost>`

Directory	`<Directory ps1>`
	`  Order Allow,Deny`
	`Allow from 192.168.199.0/24`
	`Deny from 192.168.199.1`
	`DirectoryIndex index.html`
	`Option Indexes`
	`<Files ~ ‘‘\.(gif|jpe?g|png)$’’>`
	`. . .`
	`  </Files>`
	`</Directory>`

4 Site per usuari

Amb el mòdul d'*Apache mod_userdir* podem fer que els usuaris del sistema on està instal·lat el servei tinguin un espai web personal dintre del directori de la seva *home* `~/public_html`. Per veure el contingut des de la web, la *url* és

`http://ip_o_fqdn_d_apache/~aqui_posem_el_nom_de_lusuari`.

Com he dit abans, cal que el mòdul *mod_userdir* estigui habilitat:

```
$ sudo a2enmod userdir
```

Aquesta comanda ens crea dos enllaços al directori
`/etc/apache2/mods_enabled` de dos fitxers del directori
`/etc/apache2/mods_available`: `userdir.conf` i `userdir.load`.
Cal que el directori `public_html` dels usuaris desitjats estigui creat.

Fitxer `userdir.conf`:

```
<IfModule mod_userdir.c>
  # establim quin és el nom del directori
  UserDir public_html
  # no permetem que l'usuari root en tingui
  UserDir disabled root

  # restriccions sobre el directori
  <Directory /home/*/public_html>
    # Permetem els següents grups de directives en
    # .htaccess
    AllowOverride FileInfo AuthConfig Limit Indexes

    Options MultiViews Indexes
    SymLinksIfOwnerMatch IncludesNoExec
    <Limit GET POST OPTIONS>
      Require all granted
    </Limit>
    <LimitExcept GET POST OPTIONS>
      Rquire all denied
    </LimitExcept>
  </Directory>
</IfModule>
```

5 VirtualHost

Com crear llocs web virtuals i quines directives hi intervenen?

Amb l'*Apache* com amb molts altres servidors web tenim l'opció de tenir i servir múltiples web que responen a *url* de *fqdn* diferents com si provenguessin de diferents servidors. De cadascuna d'aquestes en diem *Virtualhost*.

5.1 Directives associades

5.1.1 VirtualHost

Directiva principal d'un VirtualHost. Conté directives associades al VirtualHost. Té com a paràmetre la *ip* o *fqdn* destí de la petició *HTTP*.

5.1.2 ServerName

Nom que ha d'aparèixer en la *url* de la petició.

5.1.3 ServerAlias

Utilitzat per configurar un alias per a un virtualhost basat en nom. Context: *VirtualHost*. Exemple:

```
<VirtualHost *:80>
  ServerName server.domain.com
  ServerAlias server server2.domain.com server2
  ServerAlias *.example.com
  UseCanonicalName Off
  # ...
</VirtualHost>
```

5.1.4 DocumentRoot

Directori arrel del contingut *html* del lloc web virtual.

5.2 VirtualHosting basat en ip

Depenent de la *ip* per on li arriba la petició el servidor respon amb un lloc web virtual o altre.

5.3 VirtualHosting basat en nom

Depenent del nom del servidor en la *url* el servidor respon amb un lloc un un altre. Aquí és clau el valor que donem en la directiva *Servername* que és el nom del servidor per el qual ha de respondre el lloc web virtual.

5.4 VirtualHosting basat en port

Depenent del port per on li arriba la petició el servidor respon amb un lloc web virtual o altre.

6 Control de l'accés als directoris

When multiple Require directives are used in a single configuration section and are not contained in another authorization directive like ¡RequireAll¿, they are implicitly contained within a ¡RequireAny¿ directive. Thus the first one to authorize a user authorizes the entire request, and subsequent Require directives are ignored.

7 Crear regles en Apache

La idea és crear regles les quals estàn formades per condicions. Les directives associades són:

RewriteEngine que ha d'estar a *on*. *RewriteCond* per crear una condició. *RewriteRule*

7.1 Operadors

[NX], [R,NC], [F], [OR], [R,L], [P].

8 Apache com a proxy

Apache tant pot desempènyer la funcionalitat de proxy directe o proxy (*Forward proxy*) com la de proxy invers o gateway (*Reverse proxy*).

Principals mòduls

Per el mínim de funcionalitat:

- `mod_proxy`: mòdul necessari per a fer que Apache sigui un proxy.
- `mod_proxy_balancer`: per a que faci de balancejador de càrrega.
- `mod_cache` si el que volem és un proxy que faci també de servidor cau. Amb aquest mòdul farà falta també el mòdul `mod_cache_disk` i/o el mòdul `mod_file_cache`.

Protocols que pot implementar com a proxy:

8.1 Forward Proxy

Principals directives

- `ProxyRequests`
  ha d'estar a On. si el proxy ha de ser un proxy per llocs http o llocs ftp, necessitarem els mòduls `mod_proxy_http` i el `mod_proxy_ftp`. Si els llocs són HTTPS, a més a més necessitem habilitat el `mod_proxy_connect`.
- `<Proxy>`
  Per aplicar directives relatives al contingut que es vol accedir a través del proxy i que està indicat com a paràmetre al costat de la directiva. Per exemple: en la directiva `<Proxy http://cnn.com/*>` configurarem què s'ha de fer per accedir a cnn.com a través del proxy. Un altre exemple:

```
<Proxy ''*''>
  Require host internal.example.com
</Proxy>
```

Per accedir a qualsevol lloc a través del proxy cal ser del domini internal.exemple.com. Fixeu-vos que després, en el firewall, pot haver-hi una regla que només deixi sortir les consultes HTTP que surten del proxy.

8.1.1 Acl's en Apache

Amb les directives *Proxy* i *Require* podem fer regles acl per filtrar el tràfic HTTP. La sintaxi[1]:

```
<Proxy URL>
  regles acl
</Proxy>
```

Exemple de regles

1. **(Squid) acl localnet src 192.168.13.0/24**

```
<Proxy *>
  Require ip 192.168.13
  # Require not ip 192.168.13
</Proxy>
```

2. **(Squid) srcdomain**

```
<Proxy *>
  Require host <domini>
  # Require not host <domini>
</Proxy>
```

3. **(Squid) dstdomain**

```
<Proxy domini_destinacio>
  Require ...
</Proxy>
```

4. **(Squid) acl de tipus time**

[1] la *url* de la directiva *Proxy* pot ser una expressió regular

```
acl pati MTWHF 9:00-21:00
http_access allow pati
```

Hem d'utilitzar expressions regulars. Per fer-ho, usem la directiva `Require expr`. També utilitzem variables d'entorn que es criden amb la següent sintaxi: `%{NOM_VARIABLE}`

```
<Proxy *>
  Require expr %{TIME_HOUR} -ge 9 &&
    %{TIME_HOUR} -le 9
</Proxy>
```

Sobre expressions regulars teniu més info a [4].

5. **(Squid) acl referrer_regex** Aquí podem usar la variable

```
<Proxy *>
  Require expr "%{HTTP_REFERRER} >= '\.google'"
</Proxy>
```

6. **(Squid) acl browser** Aquí podem usar la variable
7. **Llistes**.

```
<Proxy *>
  Include ruta_del_fitxer_amb_la_llista
</Proxy>
```

8.1.2 Proxy com a servidor cau

Els objectius d'un servidor cau són els d'evitar total o parcialment les peticions que fan els clients als servidors web per a disminuir el tràfic i augmentar l'ample de banda. Si no millora significativament el rendiment un servidor cau no serveix de res. Hem de tenir clar que muntar un servidor cau és costós. És necessari que aquest tingui un molt bon accés a internet per a respondre amb celeritat a les peticions dels clients.

A tenir en compte en el tràfic web:

- Round trips (anades i tornades) entre client i servidor web
- Ample de banda
- latència: desfase o temps que triga el servidor en respondre o en processar una petició HTTP un cop l'ha rebuda.

Mòduls i directives

Per Apache com a servidor cau, el mòdul que hem de fer servir és `mod_cache`, `mod_cache_disk` i/o `mod_file_cache` (servidor cau enmagatzema el contingut a la memòria per servir-lo més ràpidament) junt amb els mòduls que necessita com a proxy. Principals directives del mòdul `mod_cache`:

- `CacheRoot`: directori arrel on estarà el cau
- `CacheEnable`: quin contingut guardem al cau i a on.
- `CacheDisable`: quin contingut no hem de guardar.
- `CacheIgnoreCacheControl`: amb valor a *on*, ignora la capçalera *Cahe-Control:no-cache* o la capçalera *Pragma: no-cache* de la petició HTTP que serveix per demanar que el contingut de la resposta no sigui servit de la cache.
- `CacheIgnoreHeaders`: No guardem les capçaleres de les respostes al cau.
- `CacheIgnoreNoLastMod`: Normalment, les respostes sense la capçalera Last Modified no són guardades al cau. Amb aquesta directiva a *on* ignorem aquest fet.
- `CacheIgnoreQueryString`: Normalment, les respostes amb diferent query string es consideren diferents. Amb aquesta directiva a *on* ignorem aquest fet i donem de resposta qualsevol contingut encara que difereixi de la query string del contingut guardat.
- `CacheMaxExpire`: el máxim temps en segons que guardem un document.
- `CacheMinExpire`: el mínim temps en segons que guardem un document.
- `CacheStorePrivate|NoStore|Expired`: guardem a la cache contingut encara que tingui capçaleres privades (per defecte no es guarden), no store o respostes en les que hagi expirat el contingut.

I aquí enumerem algunes de `mod_cache_disk`:

- `CacheDirLevels`: Nombre de nivell de subdirectoris.
- `CacheDirLength`: Longitut del nom dels directoris.
- `CacheMax|MinFileSize`: tamany màxim — mínim del fitxer.
- `CacheRoot`: Ruta del directori de la cache.

8.1.3 Proxy com a servidor cau

Els objectius d'un servidor cau són els d'evitar total o parcialment les peticions que fan els clients als servidors web per a disminuir el tràfic i augmentar l'ample de banda. Si no millora significativament el rendiment un servidor cau no serveix de res. Hem de tenir clar que muntar un servidor cau és costós. És necessari que aquest tingui un molt bon accés a internet per a respondre amb celeritat a les peticions dels clients.

A tenir en compte en el tràfic web:

- Round trips (anades i tornades) entre client i servidor web
- Ample de banda
- latència: desfase o temps que triga el servidor en respondre o en processar una petició HTTP un cop l'ha rebuda.

Mòduls i directives

Per Apache com a servidor cau, el mòdul que hem de fer servir és `mod_cache`, `mod_cache_disk` i/o `mod_file_cache` (servidor cau enmagatzema el contingut a la memòria per servir-lo més ràpidament) junt amb els mòduls que necessita com a proxy. Principals directives del mòdul `mod_cache`:

- `CacheRoot`: directori arrel on estarà el cau
- `CacheEnable`: quin contingut guardem al cau i a on.
- `CacheDisable`: quin contingut no hem de guardar.
- `CacheIgnoreCacheControl`: amb valor a *on*, ignora la capçalera *Cahe-Control:no-cache* o la capçalera *Pragma: no-cache* de la petició HTTP que serveix per demanar que el contingut de la resposta no sigui servit de la cache.
- `CacheIgnoreHeaders`: No guardem les capçaleres de les respostes al cau.
- `CacheIgnoreNoLastMod`: Normalment, les respostes sense la capçalera Last Modified no són guardades al cau. Amb aquesta directiva a *on* ignorem aquest fet.
- `CacheIgnoreQueryString`: Normalment, les respostes amb diferent query string es consideren diferents. Amb aquesta directiva a *on* ignorem aquest fet i donem de resposta qualsevol contingut encara que difereixi de la query string del contingut guardat.

- `CacheMaxExpire`: el máxim temps en segons que guardem un document.
- `CacheMinExpire`: el mínim temps en segons que guardem un document.
- `CacheStorePrivate|NoStore|Expired`: guardem a la cache contingut encara que tingui capçaleres privades (per defecte no es guarden), no store o respostes en les que hagi expirat el contingut.

I aquí enumerem algunes de `mod_cache_disk`:

- `CacheDirLevels`: Nombre de nivell de subdirectoris.
- `CacheDirLength`: Longitut del nom dels directoris.
- `CacheMax|MinFileSize`: tamany màxim — mínim del fitxer.
- `CacheRoot`: Ruta del directori de la cache.

Exemples de configuració

Configuració de proxy.conf. Per accedir a qualsevol lloc a través del proxy s'ha de tenir com a ip la 192.168.1.0:

```
ProxyRequests On
<Proxy *>
  AddDefaultCharset off
  Require ip 192.168.1
  Require all denied
</Proxy>
```

HTTP Headers que intervenen en un servidor cau

- `Cache-control`: Sobreescriu els algoritmes de cau sobre un contingut en particular. Pot estar en el client com en el servidor. Pot portar molts valors o directives. Entre elles, la directiva *max-age*.
- `Age`: Capçalera que inclou un servidor cau en la seva resposta si el contingut el serveix ell informant de l'edat d'aquest. O sigui, si en una resposta apareix aquesta capçalera, vol dir que el contingut no ve de primera mà. NO obstant això, aquesta capçalera només la inclouen els servidors que acompleixen el protocol HTTP/1.1, per lo que l'absència d'aquesta capçalera també pot voler dir que el contingut ha sigut donat per un servidor cau HTTP/1.0.
- `Date`: Des de la resposta, és el temps en el que aquesta fou generada.
- `Expires`: Data en la qual el contingut es considera obsolet.

Com decidir si un contingut és encara fresc o si és obsolet

Hem de saber l'edat, *Age* d'un element i si aquesta edat supera el temps de vida en el que es considera que un contingut és fresc, *freshness lifetime*.

Com calcular l'edat d'un contingut

El servidor original envia la capçalera Date en les seves respostes donant el temps en el que la resposta fou generada. Per tant, l'edat és el temps en el que porta el contingut a la cache des de la resposta del servidor original.

Com calcular freshness lifetime

La directiva *max-age* té prioritat sobre la capçalera *Header*. Per lo que si aquesta directiva existeix:

```
freshness_lifetime = max_age_value
```

En canvi, si la capçalera *Expires* es troba en la resposta i *max-age* no:

```
freshness_lifetime = expire - date
```

Si la directiva ni la capçalera existeixen aleshores el servidor cau calcularà mitjançant un càlcul heurístic, un valor per el *freshness lifetime*.

Finalment, determinem si un confingut és obsolet o no amb la sergüent comprovació:

```
response_is_fresh = (freshness_lifetime > current_age)
```

Què pot ser contingut susceptible d'estar en cau

1. Caching must be enabled for this URL. See the CacheEnable and CacheDisable directives.
2. The response must have a HTTP status code of 200, 203, 300, 301 or 410.
3. The request must be a HTTP GET request.
4. If the response contains an *Authorization:* header, it must also contain an *s-maxage*, *must-revalidate* or *public* option in the *Cache-Control:* header, or it won't be cached.

5. If the URL included a query string (e.g. from a HTML form GET method) it will not be cached unless the response specifies an explicit expiration by including an *Expires:* header or the *max-age* or *s-maxage* directive of the *Cache-Control:* header, as per RFC2616 sections 13.9 and 13.2.1. [2]

6. si en *Cache-control* hi ha *private*, en principi no és *cachejable* a no ser que tingui Apache la directiva *CacheStorePrivate*.

7. si en *Cache-control* hi ha *no-store*, en principi no és *cachejable* a no ser que tingui Apache la directiva *CacheStoreNoStore*.

8. A response will not be stored if it includes a *Vary:* header containing the match-all "*".

En el cau s'ha de tenir en compte també si el contingut és *Multiviews*. Per exemple, si té diferents idiomes. Amb la cpaçalera *Vary* es determina quines són les capçaleres que s'han de tenir en compte. Per exemple:

```
Vary: negotiate,accept-language,accept-charset
```

Cau de tres estats vs cau de dos estats

En un servidor cau de dos estats tenim contingut fresc o obsolet i que, per tant, s'esborra del disc. En un servidor de tres estats un contingut, en canvi, pot ser:

- **Fresh**: EL servidor cau el pot servir directament del disc perquè és suficientment nou.
- **Stale**: Contingut massa vell. El servidor cau ha de connectar amb el servidor original i comprovar si el contingut és encara fresc abans de servir-lo al client. El servidor original respondrà amb el contingut nou o amb un missatge dient que el contingut que té la cache es pot servir.
- **Non-existent**: S'ha de demanar el contingut al servidor original.

Com s'enmagatzema el contingut en l'Apache Es pot decidir si guardar el contingut en disc o en memòria. No obstant això, encara que haguem decidit guardar el contingut en disc, els continguts més sol·licitats estaràn temporalment en memòria per a un ràpid accés.

Per Una URL guardada a disc es crea un directory amb un hash de la url que inclou també els valors depenents de la capçalera *Vary*.

CacheDirLevels indica els nivells de subdirectoris i el *CacheDirLength* indica la longitut dels noms dels directoris.

[2] *s-maxage* és com el max-age però el sobreescriu si hi és.

htcacheclean comanda per control·lar l'espai del contingut al cau. Es pot execuar com a comanda o com a dimoni. Si s'usa com a comanda, pot executar-se periòdicament amb cron o posant-la en el fitxer `/etc/rc.local`.

Evitar massives peticions al servidor original per un mateix contingut obsolet

Amb Apache això es pot evitar marcant un contingut com a *Lock* i tenir-lo en un directori per a aquests tipus de continguts. Aleshores, si arriba una altra petició per el mateix contingut ja no es torna a enviar una nova petició al servidor demanant per una actualització d'aquest i així només es fa una vegada. Les peticions que arriben després de la primere esperen aleshores a que arribi el refresc i es donen aleshores del cau. D'això se'n diu *Thundering Herd*. Directives associades:

- `CacheLock`: Ha d'estar a on.
- `CacheLockPath`: Ruta del directori on es deixa el contingut bloquejat.
- `CacheLockMaxAge`: Temps màxim en segons en el que el contingut pot estar bloquejat.

registres del servidor cau

```
CustomLog ‘‘cached-requests.log’’ common env=cache-hit
CustomLog ‘‘uncached-requests.log’’ common env=cache-miss
CustomLog ‘‘revalidated-requests.log’’ common env=cache-revalidate
CustomLog ‘‘invalidated-requests.log’’ common env=cache-invalidate
```

Problemes amb el servidor cau

Com hem vist abans, les decisions que pren el servidor cau per saber si un contingut és fresc o no és a través de les capçaleres de resposta. O sigui, que si aquestes capçaleres no estàn ben informades el rendiment del servidor cau no serà òptim. És més, podria ser que fins i tot relanteixi el servei a contingut per internet.

En aques cas és necessari tenir la capacitat de poder reescriure les capçaleres de les respostes, i això amb Apache es pot fer amb els mòduls `mod_headers` i `mod_expires`.

8.2 Reverse Proxy o gateway

mira wiki en español

Principals directives

- `ProxyPass` mapeja les peticions a una ruta local (del gateway) cap a una url d'un servidor remot. Més detall a apache
- `ProxyRequests` ha d'estar a Off. Deshabilitem apache com a forward proxy.
- `BalancerMember` afegeix un membre més al balancejador definit dintre d'una directiva del tipus:

```
# definim el balancejador amb balancer://<i el nom que li volguem donar per ide
<Proxy balancer://\ldots>
  BalancerMember <url del balancejador 1>
\ldots
BalancerMember <url del balancejador n>
\ldots
</proxy>
```

- `ProxyPassReverse` no és obligatòria per fer el balanceig de càrrega ni el proxy invers i el que fa és modificar les caçaleres `Location`, `Content-location` i `Uri` de la resposta HTTP. O sigui, les respostes dels servidors remots seràn sobreescrits. és útil per exemple per a ficar en les capçaleres de les respostes HTTP les dades del servidor proxy i amagar la dels membresi amagar la dels membres.
- `ProxySet` afegeix paràmetres al balancejador. Amb aquesta directiva podem donar valor al paràmetre lbmethod, el qual indica l'algoritme de balanceig que s'ha de fer servir.

A FAQ i Definicicions generals

Els nombres màgics El fitxer `/etc/apache2/magic` és el llistat de nombres màgics que té apache per identificar tipus de fitxers. Un nombre màgic és un nombre posat en un lloc al començament d'un fitxer i que serveix per identificar el tipus de fitxer.

En instal·lar el paquet Apache no està escoltant peticions ipv4 És el que passa en el cas de Debian i versions d'Apache 2.2 i 2.4 aparentment, perquè realment sí que està rebent peticions de ipv4 (es mapejen les direccions ipv4 en format ipv6).

Si el que vull és explícitament que apache escolti per un determinat port ipv4, farè:

`Listen 0.0.0.0:80`

Veure la documentació d'apache referent al binding[1]

URI vs URL LEs URL són un tipus de URI. Les URL defineixen un recurs d'internet mitjançant la seva localització. Per exemple, tota URI que vagi precedida per "http:" s una URL perquè denotarà una localització d'un recurs a internet. Un altre tipus de URI són les URN. Aquestes defineixen el nom d'un recurs a internet. Per exemple, l'isbn a internet podria estar representat per una URN del tipus, per exemple, isbn:n-nn-nnnnnn-n. Això no denota cap lloc, sinó un nom. Verure més a aquest article de w3c[2].

Cada petició HTTP d'un client genera una nova connexió Aquesta característica és pròpia de HTTP1.1.

Diferències entre forward proxy i reverse proxy La diferència principal és que mentre el forward proxy s'encarrega de fer d'intermediari dels clients i de protegir la identitat d'aquests, el reverse proxy s'encarrega de fer d'intermediari d'un o d'un conjunt de servidors i de protegir la identitat d'aquests.

En un forward proxy és el proxy el que envia la petició al servidor i en l'invers els clients envien les peticions al proxy en comptes de fer-ho directament als servidors. El proxy, per la seva banda, té la capacitat de seguir el fluxe de peticions i respostes des dels clients cap als aervidors i al revés.

Dir també que amb un proxy reverse podem fer un balancejador de càrrega entre servidors que poden oferir idèntic servei.

Veure més a aquest article de Jscape[3].

B VirtualHost FAQ

Se puede hacer un virtualhost como www.bash.net y otro como www.asix.net o ha de ser www.bash.aguileraj.net y www.asix.aguileraj.net? Sí. El que el Virtualhost estigui en un domini totalment diferent o no que el mateix servidor no afecta al funcionament. El procès és el següent:

1. preguntamos a dns por FQDN y este busca por dominio
2. Nos dan una ip
3. Vamos por ip
4. Apache mira url y entonces determina qué virtualhost es

Referències

[1] `http://httpd.apache.org/docs/`

[2] `http://www.w3.org`

[3] http://www.jscape.com/blog

[4] https://httpd.apache.org/docs/2.4/expr.html