El servidor HTTP Apache (III): ús de mòduls diversos (Part 2)

*Directives definides al mòdul "rewrite"

Les directives d'aquest mòdul (https://httpd.apache.org/docs/current/mod/mod_rewrite.html) serveixen per redireccionar al client, conduïnt-lo a pàgines noves des de links antics. Aquestes directives poden actuar a tota la URL, des del protocol fins el nom de domini i port passant per les eventuals rutes i/o "querystring". Així doncs, les poden servir, per exemple, per convertir URLs molt llargues i complicades en URLs més senzilles per l'usuari, o per moure contingut a una altra estructura de subcarpetes o fins i tot a un altre domini, etc, etc.

> NOTA: Això mateix ja hem vist que ho fa la directiva *Redirect* del mòdul "alias", però les directives del mòdul "rewrite" són molt més completes, flexibles i complexes; de fet, una gran diferència entre ambdós mètodes és que el mòdul "rewrite" retorna directament al client una resposta diferent de la corresponent a la seva petició mentre que una redirecció simplement retorna un codi d'estat 3xx per tal què el client en faci llavors una nova petició "correcta". A la pràctica això vol dir que si es fa servir el mòdul "rewrite", el contingut de la barra d'adreces del navegador no canvia mentre que si es fa una redirecció sí.

Tot i que les directives d'aquest mòdul es poden d'escriure a la configuració general del servidor o també dins d'una secció *<Directory>* (o en l'arxiu ".htaccess" equivalent), el més habitual amb diferència és escriure-les dins d'una secció *<VirtualHost>* per tal que afectin a tot un "virtualhost" (i només a ell); així ho suposarem al llarg dels següents paràgrafs. En tot cas, per poder fer servir aquest mòdul, obligatòriament sempre hem d'indicar primer la línia *RewriteEngine On* per tal d'activar el mòdul "rewrite" per aquell "virtualhost" en concret (suposant que el mòdul ja estigui carregat, que a sistemes Ubuntu i Fedora ja ho està per defecte). A partir d'aquí, concretament estem parlant de dues directives fonamentals:

> *RewriteRule exprRegRutaDeLaPeticioClient DestíOnEsVa [opcio1,opcio2]* : Fa el redireccionament. L'expressió regular indicada és validada contra la part de la URL que representa el "path" (la ruta) del recurs demanat a la petició (sense la querystring i sense la barra inicial); és a dir, en una petició com https://www.elmeudomini.com/una/ruta/pagina.html?a=1&b=2 l'expressió regular es validaria contra "**una/ruta/pagina.html**". D'altra banda, el destí indicat substituirà a aquesta ruta i pot tenir alguna de les següents formes:
>
> * La ruta absoluta en disc d'un determinat recurs (sempre i quan Apache vegi que existeix)
> * La ruta de la nova URL (en el cas que Apache vegi que no es correspon a una ruta de disc)
> * Una nova URL completa. En el cas que el protocol,nom de domini i port coincideixin amb el del "virtualhost" en qüestió, aquesta part de la URL s'obviarà i la redirecció es realitzarà com si només s'hagués indicat la ruta de la nova URL desitjada (és a dir, el cas anterior). En cas contrari, es fa una redirecció completa a la URL indicada.
> * Un guió ("-"). Indica que no es realitzarà cap redireccionament (la ruta de la petició no es canvia, doncs). Aquesta opció és útil combinant-la amb algunes opcions indicades entre claudàtors, tal com veurem
>
> NOTA: Opcions interessants que es poden afegir entre claudàtors a la directiva *RewriteRule* són:
> *[NC]* : Fa que l'expressió regular s'avalui de forma "case-insensitive" (és a dir, no importa que estigui en majúscules o minúscules (per defecte sí que importa)
> *[F]* : Genera una resposta 403 (per tant, a la pràctica, prohibeix l'accés a la petició del client afectada)
> *[R=nº]* : Genera una resposta 302 (o si s'indica el codi explícitament, una resposta 3xx concreta; en el cas d'indicar el codi 301, cal saber que la URL del nou destí serà mostrada al navegador)
> *[L]* : Indica que la regla *RewriteRule* on aparegui serà la darrera regla que es llegirà (no es continuarà llegint més regles *RewriteRule*, per tant). L'opció contrària (que és la per defecte, és *[N]*)
> *[QSA]* : Manté a la URL redireccionada la "querystring" que l'usuari pugui haver escrit a la URL original. L'opció contrària és *[QSD]*
> En tot cas, teniu una explicació més completa i detallada a https://httpd.apache.org/docs/2.4/rewrite/flags.html
>
> NOTA: Si s'escriuen diverses línies *RewriteRule*, l'ordre en què apareixen escrites és important perquè aquest serà l'ordre en què seran aplicades en temps real quan es detecti una petició. Per tant, si alguna de les regles tingués l'opció *[L]* indicada, això farà que les regles que hi puguin haver escrites després no s'apliquin; si no, s'aplicarà, per un mateix patró trobat, la darrera regla coincidient

NOTA: Si s'utilitza la directiva *RewriteRule* dins d'una secció *<Directory>* (o en un arxiu ".htaccess", que ve a ser el mateix però si no tenim permisos d'administrador per canviar la configuració general de l'Apache) hem de tenir en compte que l'expressió regular es validarà contra només una part de la ruta de la petició, no tota ella: es validarà contra la part de la ruta posterior a la ruta del directori en sí. Per canviar aquest comportament i fer que l'expressió regular s'avalui per tota la URL (és a dir, des del "DocumentRoot"), caldrà indicar abans de qualsevol directiva RewriteRule la directiva *RewriteBase / * En general, la ruta escrita com a valor de la directiva *RewriteBase* representa l'inici de la ruta de la URL a partir del qual es validaran les expressions regulars indicades a *RewriteRule* (independentment de en quina carpeta -mitjançant la secció *<Directory>* o l'arxiu ".htaccess"- estiguin definides).

RewriteCond caracteristica exprRegCorresponentAValorAComprovar [opcio1,opcio2] : Opcional. Serveix per definir el valor que es vol comprovar d'una determinada característica en cada petició rebuda per, si es detectés concordància, realitzar el redireccionament establert en la primera directiva *RewriteRule* que es trobi a continuació. Si n'hi hagués més d'una característica a comprovar, es poden escriure múltiples línies *RewriteCond*, una per cadascuna d'aquestes característiques.

NOTA: Algunes de les possibles característiques el valor de les quals es pot comprobar amb *RewriteCond* són (entre moltes altres):
%{HTTP_REFERER} : Url de la pàgina anterior d'on ve el visitant
%{REMOTE_ADDR} : IP de la màquina client
%{HTTP_USER_AGENT} : Nom del client web (navegador) emprat
%{HTTP_HOST} : IP o nom DNS del servidor
%{SERVER_PORT} : Port del servidor al que es dirigeix la petició del client
%{SCRIPT_FILENAME} : Ruta del fitxer sol·licitat pel client (sense incloure el nom del servidor)
%{QUERY_STRING} : Cua de variables posterior al fitxer sol.licitat (?var1=valor1&var2=valor2...)
%{REQUEST_URI} : Concatenació de SCRIPT_FILENAME i QUERY_STRING
%{TIME_HOUR} : Hora quan és realitzada la petició. També hi ha *%{TIME_MIN}* , *%{TIME_DAY}* , etc

NOTA: A més d'indicar una expressió regular, es poden indicar el següents símbols especials per especificar condicions concretes com a valor a comprovar:
Símbol "*!*" : S'indica just al davant del valor a comprovar concret especificat , i significa "no". Per tant, serveix per indicar que la concordància es vol que ocorri quan la característica no tingui el valor indicat
Símbol "=" : S'indica just al davant del valor a comprovar concret especificat , i significa que la comparació es vol fer amb aquest valor de forma literal (és a dir, que aquest no és una expressió reg. sinó una cadena tal qual)
-f : Significa "l'element indicat com a característica és un fitxer"
-s: Significa "l'element indicat com a característica és un fitxer amb una mida més gran de 0 bytes"
-d : Significa "l'element indicat com a característica és un directori"
-l : Significa "l'element indicat com a característica és un enllaç"

NOTA: Si hi hagués escrita una o més línies *RewriteCond* només si la petició del client concorda amb totes les característiques indicades en aquestes línies el redireccionament indicat es farà efectiu. Si es vol fer el redireccionament si la petició concorda amb almenys una d'elles, caldrà afegir l'opció *[OR]* en totes les línies *RewriteCond* implicades. D'altra banda, una altra opció important de la directiva *RewriteCond* és *[NC]*, amb el mateix significat que el que té a la directiva *RewriteRule*

Tant la directiva *RewriteCond* (a l'hora d'especificar el valor a comprovar) com la directiva *RewriteRule* (a l'hora d'especificar l'eventual petició del client a processar) solen fer un ús extens d'expressions regulars. Per aprofundir en el seu coneixement (que, de fet, ja vam conèixer en parlar de les directives *<FilesMatch>* i similars), recomano consultar http://httpd.apache.org/docs/current/rewrite (i, en especial, la introducció bàsica https://httpd.apache.org/docs/current/rewrite/intro.html).

NOTA: A continuació es llisten els símbols més comuns que podem es expressions regulars, a mode de "xuleta" bàsica:
Símbol "^" : Indica que tot allò escrit a continuació ha d'aparèixer al principi de la cadena en qüestió
Símbol "*$*" : Indica que tot allò escrit a continuació ha d'aparèixer al final de la cadena en qüestió
Símbol "." : Equival a qualsevol caràcter
Símbol "*[ ]*" : Equival a qualsevol caràcter dels que apareguin indicats entre els claudàtors
Símbol "*[^]*" : Equival a qualsevol caràcter dels que no apareguin indicats entre els claudàtors
Símbol "+" : Indica que el símbol (o grup de símbols) anterior pot aparèixer una vegada o més
Símbol "*" : Indica que el símbol (o grup de símbols) anterior pot no aparèixer, aparèixer una vegada o més
Símbol "*?*" : Indica que el símbol (o grup de símbols) anterior és opcional (pot no aparèixer o aparèixer una vegada)
Símbol "*{nº}*" : Indica que el símbol (o grup de símbols) anterior ha d'aparèixer el nombre de vegades indicat entre les claus
Símbol "*(|)*" : Indica diferents opcions per un determinat valor. En el cas de la directiva *RewriteRule*, el valor concret triat per la petició actual d'entre els d'un grup encerclat entre parèntesis es podrà reutilitzar dins de l'especificació del destí mitjançant l'expressió *$1* (pel primer grup que aparegui indicat a l'expressió regular) o *$2* (pel segon, si n'hi hagués), etc
Símbol "\" : Escrit davant de qualsevol dels símbols anteriors fa que aquests perdin el seu significat especial (per exemple, "\." fa que el punt no representi qualsevol caràcter sinó que sigui un punt literal)

A continuació presentem, de totes formes, alguns exemples senzills d'ús:

*Exemple que permet escriure com a ruta d'una URL el valor "/about" (o, opcionalment "/about/", però en tot cas sense res més per davant ni per darrera) en lloc del nom real de l'arxiu ("/about.html"). Així, la URL http://www.elmeudomini.com/about es transforma internament http://www.elmeudomini.com/about.html, que és la direcció real de la pàgina a la què es vol accedir (però que l'usuari no veurà).

```
RewriteEngine on
RewriteRule ^about/?$ about.html [NC]
```

*Exemple que permet escriure com a ruta d'una URL un valor curt en lloc d'un valor llarg. En aquest cas la URL http://www.elmeudomini.com/camisa/estiu equival a http://www.elmeudomini.com/results.php?item=camisa&estacio=estiu, per exemple. D'aquesta forma, l'usuari podrà escriure la primera i anirà a parar automàticament al recurs corresponent a la segona (*recordeu el significat dels símbols (...|...|...) en les expressions regulars i també del símbol $1*)

```
RewriteEngine on
RewriteRule ^camisa/(estiu|hivern|tardor|prima) results.php?item=camisa&estacio=$1
```

Si volguéssim especificar no només camises sinò qualsevol altre tipus d'item, podríem anar un pas més enllà i redactar les següents línies *(fixeu-vos com hem fet servir una expressió regular que engloba qualsevol combinació de símbols alfanumèrics i fem servir el símbol $2 ja que ara tenim dos grups -que s'identifiquen pels parèntesis- de possibles valors a triar entre varis possibles)*

```
RewriteEngine on
RewriteRule ^([A-Za-z0-9]+)/(estiu|hivern|tardor|prima) results.php?item=$1&estacio=$2 [QSA]
```

Exemple que redirecciona qualsevol pàgina inexistent a la pàgina inicial del lloc, anomenada "home.html" (noteu que l'expressió "." equival a qualsevol ruta, però el filtre s'ha realitzat a la directiva *RewriteCond*)

```
RewriteEngine on
RewriteCond %{REQUEST_URI} !-f
RewriteRule . home.html
```

NOTA: Noteu que a la regla no ha calgut escriure una expressió regular com ".*" sinó que amb tan sols "." ja aconseguim l'objectiu d'indicar "qualsevol ruta en la URL". Això és perquè indicant només un punt estem dient que volem que coincideixi qualsevol ruta que tingui almenys un caràcter qualsevol, i aquí la clau està en el "almenys" perquè no hem delimitat el punt ni per davant ("^") ni per darrera ("$"), així que amb només un caràcter que tingui en qualsevol posició ja hi haurà concordància. Aquest detall és important perquè per l'Apache és molt més lleuger avaluar l'expressió simple "." que no pas ".*" (on ha de fer un recorregut per tota la ruta fins el final).

*Exemple que denega (gràcies a l'opció *[F]*) l'accés al "virtualhost" (o a una carpeta concreta si aquestes línies estiguessin escrites en una secció *<Directory>*) des de totes les IPs de clients menys la 12.34.56.78

```
RewriteEngine on
RewriteCond %{REMOTE_ADDR} !^(12\.34\.56\.78)$
RewriteRule . - [F,L]
```

*Directives definides al mòdul "headers"

Les directives d'aquest mòdul (https://httpd.apache.org/docs/current/mod/mod_headers.html) serveixen per definir (si no ho estan ja de forma predeterminada) i modificar (és a dir, aglutinar, substituir o fins i tot eliminar) capçaleres HTTP, tant en les peticions rebudes com, sobre tot, en les respostes a donar. Aquest mòdul està activat per defecte a sistemes Fedora (això es pot comprovar fàcilment fent *grep headers /etc/httpd/conf.modules.d/**) però a Ubuntu cal activar-lo (amb *sudo a2enmod headers*); en tot cas, no hi ha cap configuració predeterminada associada, així que l'hauriem d'escriure manualment, ja sigui a nivell de configuració general del servidor, a nivell de "virtualhost" o fins i tot de seccions *<Directory>*.

NOTA: Tenint en compte l'ordre d'aplicació estàndard de les directives segons si estan escrites en la configuració general, en la d'un "virtualhost" o en la d'una secció *<Directory>*, l'ordre amb què aquestes estiguin escrites dins d'una mateixa secció també és important perquè la darrera sempre guanya

Concretament, les directives més interessants d'aquest mòdul són:

Header set nomCapçalera valor : Assigna el valor indicat (una cadena) a la capçalera HTTP de resposta indicada (creant-la si no existís prèviament,o substituint el seu valor si sí)

NOTA: Al valor indicat es poden incloure determinats modificadors, com ara *%{nomVariable}e* (per indicar el valor d'una determinada variable d'entorn), entre altres

Header edit nomCapçalera exprRegvalorAntic valorNou : Si existeix prèviament la capçalera HTTP de resposta indicada, canvia per un nou valor la part del seu valor actual que concordi amb l'expressió regular indicada.

Header merge nomCapçalera valor : Si existeix prèviament la capçalera HTTP de resposta indicada, afegeix el valor indicat al final del valor actual (separat per una coma)

Header unset nomCapçalera : Elimina la capçalera HTTP de resposta indicada

NOTA: En cap de les directives anteriors es distingeix entre majúscula i minúscules en els nom i valors de les capçaleres HTTP indicades

NOTA: També existeix la directiva *RequestHeader* , que funciona igual que la directiva *Header* (és a dir, també disposa de les accions *set*, *edit*, *merge* i *unset*, entre d'altres) però referint-se a capçaleres HTTP de la petició rebuda

NOTA: Al final de qualsevol de les accions anteriors (tant de la directiva *Header* com de *RequestHeader*) es poden afegir les següents expressions:

env=nomVariable: L'acció en qüestió es realitzarà només si la variable indicada existeix
env=!nomVariable: L'acció en qüestió es realitzarà només si la variable indicada **no** existeix
expr="unaExpressió": L'acció en qüestió es realitzarà només si l'expressió avaluada retorna *true* La sintaxi de les expressions és la mateix que vam veure quan vam estudiar la directiva *SetEnvIf* Un exemple:
Header set CustomHeader my-value "expr=%{REQUEST_URI} =~ m#^special_path.php$#"

*Directives definides al mòdul "proxy"

Apache pot configurar-se tant com servidor "proxy directe" o com servidor "proxy invers". Per això cal establir certes directives del mòdul "proxy" (https://httpd.apache.org/docs/current/mod/mod_proxy.html).

Un "proxy directe" és un servidor intermediari que se situa entre el client (normalment serà un navegador però, en qualsevol cas, haurà d'estar prèviament configurat per tal d'emprar el servidor "proxy directe" concret desitjat) i el servidor de destí; per tal d'obtenir contingut provinent d'aquest servidor de destí, el client enviarà la petició al servidor "proxy directe" que tindrà configurat, i és aquest qui reenviarà la petició al servidor de destí; el servidor "proxy directe" llavors recull la resposta del servidor de destí i la retorna al client.

Un ús típic de servidors "proxy directes" és proporcionar accés a Internet a clients interns d'una xarxa local que d'una altra forma estarien restringits per un tallafocs. També es poden fer servir com a servidors de cau per millorar l'ample de banda disponible i, per tant, la velocitat en obtenir els clients el contingut demanat (ja que, en guardar-se per primer cop en la memòria cau del servidor proxy un determinat contingut prèviament recollit d'Internet -pàgines HTML, CSS, Javascript, imatges, etc-, les següents peticions que demanin per aquest mateix contingut fetes des de qualsevol altre client de la LAN que utilitzi el servidor proxy obtindran el contingut en qüestió directament d'aquest servidor proxy sense haver de tornar a "sortir a Internet", amb les avantatges, per tant, de rebre la resposta directament en velocitat de LAN i de no "malgastar" l'ample de banda del "router" en aquesta petició i així poder aprofitar-lo per una altra; en aquest sentit, cal saber que Apache incorpora un altre mòdul específic per gestionar memòries cau anomenat "cache", que estudiarem més endavant). Els "servidors proxy" també es poden fer servir, a més, per controlar i censurar l'accés a Internet (mitjançant l'aplicació de diferents regles que restringeixin o permetin el tràfic -

segons si les peticions van dirigides a determinats dominis, o provenen de determinats clients, etc-), així com també per obtenir estadístiques d'aquest tràfic (concretament sobre les característiques de les peticions i respostes processades) o bé en temps real o bé a partir de valors històrics que van essent guardats en algun registre del sistema.

> NOTA: Existeix una manera de configurar una xarxa LAN per a què no calgui indicar explícitament a cadascun dels clients (navegadors) allà existents que facin servir un determinat "servidor proxy" present a la xarxa local sinó que, sense configurar-hi res, es pot fer que dirigeixin les peticions directament al "servidor proxy". Això és el que s'anomena "**proxy transparent**" i per implementar-lo, però, cal fer que la porta d'enllaç d'aquests clients sigui la que funcioni com a "servidor proxy" (en lloc de què sigui una màquina separada). Això involucra una sèrie de passos més tècnics (on intervenen, per exemple, regles de tallafocs) on no hi entrarem. En tot cas, en els següents paràgrafs suposarem que, d'una manera o una altra, els clients fan les peticions al "servidor proxy" directe que tindrem en marxa gràcies a l'Apache funcionant en aquest mode.

> NOTA: Més enllà del que veurem amb l'Apache, un programa específicament dissenyat per actuar com a "proxy directe" HTTP/HTTPS (ja sigui explícit o transparent) amb memòria cau és **Squid** (http://www.squid- cache.org); aquest programa, a més de permetre gestionar el contingut "catxejat" de peticions prèvies de múltiples clients, pot censurar destins -via la inspecció de dominis DNS/URLs/IPs/ports/... de destí, o altres tècniques- i pot registrar els events detectats (peticions, respostes, errors, etc) en logs per fer-ne estadístiques, entre moltes altres funcionalitats. Un altre software que pot funcionar com a servidor proxy-cau HTTP és **TrafficServer** (https://trafficserver.apache.org)

El servidor destí pot ser de diferents tipus: pot ser un altre servidor HTTP o HTTPS, pot ser un servidor FTP, un servidor WebSockets, etc. És per això que el mòdul "proxy" genèric de l'Apache ha de venir acompanyat sempre d'un altre mòdul suplementari específic pel tipus de servidor destí utilitzat. En aquest sentit es necessitaria tenir activats, a més del mòdul "proxy", els diferents mòduls específics, (respectivament: "**proxy_http**", "**proxy_connect**", "**proxy_ftp**" o "**proxy_wstunnel**"... per veure'n més consulteu https://httpd.apache.org/docs/current/mod). A Fedora això ja està fet per defecte gràcies a l'existència de l'arxiu "/etc/httpd/conf.modules.d/00-proxy.conf" (el qual activa tots els mòduls de tipus "proxy" existents) però a Ubuntu haurem de fer l'activació explícitament (concretament, per exemple, si volem fer servir Apache com a servidor "proxy directe" d'un altre servidor web HTTP/S, haurem d'executar llavors (un cop) la comanda *sudo a2enmod proxy proxy_http proxy_connect*)

En tot cas, un cop activats els mòduls adients, per implementar la funcionalitat de "proxy directe" en l'Apache només cal establir la següent directiva i ja està:

> *ProxyRequests {on|off}* : Si el seu valor és "on", activa la funcionalitat de "proxy directe" de l'Apache. Aquesta directiva es pot escriure o bé a la configuració general (de fet l'Ubuntu ja té preparat un arxiu específic per indicar-hi allà aquesta directiva, l'arxiu "/etc/apache2/mods-available/proxy.conf") però també dins de la configuració d'un "virtual host" concret.

No obstant, com els proxies directes permeten a qualsevol tipus de clients accedir a llocs externs arbitraris a través d'ells (i ocultar així el seu origen), és molt recomanable assegurar el servidor proxy per tal de què només el puguin utilitzar clients autoritzats. Això es pot aconseguir mitjançant la directiva *Require* adient (o similars) indicada dins d'una secció *<Proxy "urlDesti">* , a escriure sota on estigui la directiva *ProxyRequests* Per exemple, les següents línies restringeixen l'ús del "proxy directe" als ordinadors clients pertanyents a la xarxa local 192.168.1.0/24 (els quals podran navegar a qualsevol destí, "*"):

```
<Proxy "*">
  Require ip 192.168.1
  ProxySet connectiontimeout=5 timeout=30
</Proxy>
```

> NOTA: Com ja s'ha vist, l'asterisc indica que les directives a l'interior de la secció *<Proxy>* s'aplicaran a tots els servidors destins demanats. Es podria haver restringit a una/es Url/s concreta/es indicant-la/es (fent servir comodins si són més d'una) però gairebé mai es fa això

> NOTA: La directiva *ProxySet* és opcional i serveix per especificar detalls interns de les connexions, ja que l'Apache funcionant en mode "proxy" per defecte no manté les connexions TCP "keep-alive" ni reutilitza connexions (és a dir, les connexions TCP al destí són obertes i tancades cada cop). Si no es vol això, cal indicar les característiques concretes desitjades respecte la gestió de les connexions com a parelles nom<->valor de la directiva *ProxySet*

Un "proxy invers" és un servidor intermediari (normalment accessible des d'Internet, a diferència de com sol passar amb els "proxys directes") que es mostra davant dels clients com qualsevol altre servidor web ordinari (per tant, en aquest cas no és necessària cap configuració especial als navegadors). En aquest escenari, el client realitza la petició d'un determinat contingut al servidor "proxy invers" però aquest no emmagatzema (o genera) pas aquest contingut sinó que decideix el destí on reenviarà la petició (destí que serà un servidor web concret, anomenat servidor "backend", que sí que normament estarà "ocult a ulls d'Internet" perquè pertanyerà a la nostra xarxa interna) per tal d'obtenir-ne la resposta (el contingut demanat) d'allà i llavors retornar-lo finalment al client. D'aquesta manera, aquest contingut a "ulls del client" sempre és generat pel servidor proxy mateix, ja que la infraestructura "backend" li resulta invisible.

A més de protegir així els servidors "backend" de l'accés directe dels clients (i, per tant, de possibles atacs de tipus DoS, per exemple), els "proxies inversos" permeten implementar mecanismes de balanceig de càrrega i alta disponibilitat fent de punts centralitzats d'entrada de peticions (a vegades amb mecanismes d'autorització implementats) a partir dels quals les peticions es redistribueixen entre un conjunt de servidors "backend" indistingibles entre sí (entenent aquests tant com màquines com aplicacions), repartint així la feina a cadascun segons les seves possibilitats i capacitats. Altres possibles usos d'un servidor proxy invers són, per exemple, el permetre als clients accedir des d'Internet a servidors web que estan darrera d'un tallafocs, o també fer de servidors de memòria cau de les respostes més sol·licitades, o també fer de "terminadors TLS" (és a dir, fer deservidors que realitzen el (des)xifratge TLS, descarregant d'aquesta tasca als servidors interns, els quals podran ser llavors simples servidors HTTP plans), etc.

> NOTA: Programes que poden actuar com a "proxy invers HTTP/S" (a més de l'Apache si té el mòdul "proxy" activat) són **Nginx** (https://www.nginx.com), **HAProxy** (http://www.haproxy.org , el qual també pot actuar com a proxy invers TCP genèric), **Varnish** (https://varnish-cache.org), el qual es pot fer servir com a servidor de memòria cau del contingut més demanat pels clients (a https://www.linode.com/docs/websites/varnish/getting-started-with-varnish-cache en podem trobar una guia) o **TrafficServer** (https://trafficserver.apache.org) , entre altres.

> NOTA: Una manera molt fàcil de distingir un proxy de tipus "invers" d'un proxy que sigui "directe" és fixar-se en qui tria el servidor HTTP/S final: si és el propi proxy, llavors aquest és "invers", si és el client web, llavors és "directe". És per això que, per exemple, per anar a un servidor web fent servir un proxy invers només ens cal especificar aquest -perquè serà ell qui s'hi connecti al servidor web en qüestió sense poder intervenir-hi des del client- (*curl https://reverse.example.com*) però si fem servir un proxy directe hem d'especificar tant aquest com el servidor web remot concret al qual volem connectar (*curl -x https://forward.example.com https://web.server.com*).Teniu més informació a https://kinsta.com/blog/reverse-proxy Una altra manera de veure-ho és pensar que un "proxy directe" serveix a un conjunt de clients generalment concrets i determinats per a què aquests vagin a parar a un conjunt de servidors de destí finals generalment indeterminat mentre que un "proxy invers" serveix a un conjunt de clients generalment indeterminats per a què aquests vagin a parar a un conjunt de servidors de destí finals generalment concrets i determinats.

Igual que passava amb els "proxies directe", si fem servir un "proxy invers", el servidor destí (en aquest cas, anomenat "backend") pot ser de diferents tipus: pot ser un altre servidor HTTP o HTTPS, pot ser un servidor FTP, un servidor WebSockets, etc o fins i tot un servidor FastCGI (que és el cas més habitual si volem servir pàgines PHP amb l'Apache, ja que l'intèrpret PHP està dissenyat per actuar com a servidor "backend" de tipus FastCGI si es configura en l'anomenat mode "PHP-FPM"). És per això que el mòdul "proxy" genèric de l'Apache ha de venir acompanyat sempre d'un altre mòdul suplementari específic pel tipus de servidor "backend" utilitzat. En aquest sentit es necessitaria tenir activats, a més del mòdul "proxy", els diferents mòduls específics ("**proxy_http**", "**proxy_connect**", "**proxy_ftp**", "**proxy_wstunnel**" o "**proxy_fcgi**", respectivament... per veure'n més consulteu https://httpd.apache.org/docs/current/mod). A Fedora això ja està fet per defecte gràcies a l'existència de l'arxiu "/etc/httpd/conf.modules.d/00-proxy.conf" (el qual activa tots els mòduls de tipus "proxy" existents) però a Ubuntu haurem de fer l'activació explícitament (concretament, per exemple, si volem fer servir Apache com a servidor "proxy invers" d'un altre servidor web HTTP/S, haurem d'executar llavors (un cop) la comanda *sudo a2enmod proxy proxy_http proxy_connect*), o si volem fer servir Apache com a servidor "proxy invers" de l'intèrpret PHP en mode FastCGI (l'anomenat "PHP-FPM"), haurem d'executar la comanda: *sudo a2enmod proxy proxy_fcgi*

En tot cas, un cop activats els mòduls adients, per implementar la funcionalitat de "proxy invers" en l'Apache només cal establir la següent directiva i ja està:

ProxyPass "/" "fcgi://1.2.3.4:9999" : Activa la funcionalitat de "proxy invers" de l'Apache especificant com a servidor "backend" en aquest cas un servidor FastCGI (com ara un intèrpret PHP-FPM) amb una determinada IP escoltant a un determinat port. El valor "/" indica la URL (i totes les que hi penjin d'ella) que es redireccionaran al "backend" (en aquest cas, com que és l'arrel -el *DocumentRoot*- es redireccionaran totes les peticions). Aquesta directiva es pot escriure a la configuració general del servidor però també dins de la configuració d'un "virtual host" concret.

Aclarim que no només es poden redireccionar peticions dirigides al DocumentRoot de l'Apache que actua com a servidor "proxy invers": qualsevol URL que aquest rebi que tingui associada una directiva *ProxyPass* (o que hi estigui inclosa) serà reenviada convenientment. Per exemple, aquí tindríem diverses redireccions a diferents carpetes del servidor "backend" (que en aquest cas es pot veure que és de tipus HTTP, així que caldrà que s'hagi activat prèviament el mòdul específic pertinent -que en aquest cas seria "proxy_http"-) a partir de sengles rutes d'URL diferents:

ProxyPass "/guies" "http://backend.server.com/examples"
ProxyPass "/documents" "http://backend.server.com/docs"
ProxyPass "/principal" "http://backend.server.com"

NOTA: La directiva *ProxyPass* admet els mateixos paràmetres (opcionals) que admet la directiva *ProxySet* (que vam veure en el cas dels servidors "proxy directes" però que també es podria utilitzar en el cas dels "proxies inversos"). S'escriuen al final de tot, per exemple: P*roxyPass "/docs" "http://backend.server.com/docs" connectiontimeout=5ms timeout=30* (on, en aquest cas el primer paràmetre indica el timeout -en "ms"- per esperar a crear una connexió amb el servidor "backend" i el segon indica el timeout -en "s-" per esperar una resposta del servidor "backend" a una petició allà dirigida prèviament-. Destacarem a més paràmetres com *max=nº* (per indicar el màxim de connexions simultànies permeses cap al servidor "backend"), *keepalive=on* (per enviar de forma periòdica un paquet TCP "keepalive" al servidor "backend" de manera que l'eventual tallafocs que hi hagi entre ell i l'Apache no talli la connexió entre aquests degut a passar massa temps d'inactivitat) o *retry=nº* (per indicar els segons que s'esperarà l'Apache a tornar a intentar connectar amb un servidor "backend" si aquest prèviament ha donat error), entre molts més.

NOTA: És possible no haver d'escriure el primer valor de la directiva *ProxyPass* (o *ProxyPassMatch)* si aquesta s'indica a l'interior d'una secció *<Location>* (o *<LocationMatch>*, respectivament), ja que aquesta secció faria la mateixa funció:

```
<Location "/docs">
  ProxyPass "http://backend.server.com/docs"
</Location>
```

NOTA: És possible evitar l'accés a determinades carpetes del servidor "backend" utilitzant (a més de possibles seccions *<Location>* amb les directives *Require* pertinents), el símbol "!" a la directiva *ProxyPass*, així:

```
ProxyPass "/docs/restringits" !
ProxyPass "/docs" "http://backend.server.com/docs"
```

NOTA: És molt recomanable afegir abans de les directives ProxyPass que s'indiquin, la directiva *ProxyRequests off* per assegurar-se de no estar usant l'Apache com a "proxy directe": els dos modes són incompatibles en un mateix "virtualhost".

NOTA: Cal saber que, en el cas de tenir un servidor "backend" de tipus HTTP/HTTPS és molt recomanable, a més de fer servir la directiva *ProxyPass* (o *ProxyPassMatch*), afegir la directiva *ProxyPassReverse*, la qual té els mateixos valors (URL a redireccionar i servidor destí, és a dir, per exemple: *ProxyPassReverse "/docs" "http://backend.server.com/docs*"). Aquesta directiva permet realitzar els canvis necessaris en les capçaleres de resposta provinents del "backend" per tal de què en el cas que aquesta resposta redireccioni a un altre recurs ofert pel mateix "backend" (mitjançant algun codi 3xx), el client faci la nova petició al servidor proxy i no pas directament al servidor "backend" (que hauria de continuar estant ocult); la idea és que el client no "notarà" que hi ha cap servidor "backend" gràcies a la modificació pertinent de les capçaleres *Location:* i *Content-Location:* realitzada per la directiva *ProxyPassReverse*

NOTA: Tenint en compte tot l'explicat a la nota anterior, cal ser conscient, però, que la directiva *ProxyPassReverse* només modifica les capçaleres HTTP de resposta allà comentades; si al codi HTML d'alguna pàgina web servida pel "backend" hi hagués indicats enllaços absoluts del tipus *<a href="http://backend.server.com/fitxer.html">*, aquests no són modificats i, per tant, quan l'usuari, des del client extern, cliqui sobre ell, rebrà un error de "Pàgina no encontrada" en haver-se intentat saltar el proxy. La solució és o bé escriure els enllaços de forma relativa (així, *<a href="fitxer.html">*, per exemple) o bé utilitzar, dins de la configuració del nostre "virtualhost", la directiva *ProxyHTMLURLMap* (com per exemple així: *ProxyHTMLURLMap "http://backend.server.com/docs" "/docs"* -o també, alternativament, així: *ProxyHTMLURLMap "http://backend.server.com/docs" "http://www.eldominipublicdelmeuApache.com/docs"*-) del mòdul "**proxy_html**" (https://httpd.apache.org/docs/current/mod/mod_proxy_html.html) Aquesta directiva modifica els enllaços de les pàgines servides per l'Apache en mode "proxy invers" per tal que si hi apareix, en aquest cas d'exemple, la cadena "http://backend.server.com/docs" sigui substituïda per la cadena "/docs")

NOTA: Existeix una directiva similar a *ProxyPass* anomenada *ProxyPassMatch* la qual permet indicar expressions regulars al seu primer valor (el que indica les URLs a considerar per redireccionar)

Si hi ha la possibilitat de tenir múltiples servidors "backend", cal implementar algun mecanisme de balanceig de càrrega. Per això cal activar -i configurar adientment- el mòdul "proxy_balancer" (documentat a https://httpd.apache.org/docs/current/mod/mod_proxy_balancer.html), com es mostra en aquest exemple:

```
# Defineixo els servidors "backends" que formaran part del balancejador anomenat "micluster"
<Proxy balancer://micluster>
   BalancerMember http://1.2.3.4:8080
   BalancerMember http://1.2.3.4:8081
</Proxy>
<VirtualHost *:*>
   ...
   # Configuro ProxyPass per usar "micluster" com a balancejar la càrrega de les peticions rebudes
   ProxyPass / balancer://micluster
</VirtualHost>
```

NOTA: Al final de la directiva *BalancerMember* es poden afegir diferents paràmetres que permeten definir amb més detall el comportament de cada membre del cluster per separat, com per exemple *loadfactor=nº* (on el número indicat poer ser un valor decimal entre 1.0 -per defecte- i 100.0 que defineix el pes a aplicar a cada membre del "cluster" en qüestió) o qualsevol dels paràmetres de la directiva *ProxyPass* (comentats a la primera nota d'aquesta pàgina), entre els quals, en aquest cas, convé destacar el paràmetre *status=+H* (el qual indica -gràcies al valor concret "H"- que només s'utilitzarà el membre del cluster en qüestió quan no n'hi ha cap altre membre funcional; és a dir, és un membre de reserva global) o *status=+R*, el qual indica que el membre en qüestió funcionarà només quan un altre membre individual qualsevol del clúster deixi de funcionar (és a dir, és un membre de reserva un-a-un)

NOTA: El balancejador de càrrega com a tal també pot admetre diversos paràmetres de configuració "globals" mitjançant la directiva *ProxySet* (ja vista) indicada dins de la secció *<Proxy>* (com ja vam veure, de fet, en explicar els "proxies directes"). Concretament, un d'aquests paràmetres és *lbmethod=* , que serveix per indicar el mètode establert per balancejar la càrrega, el qual pot ser "*byrequests*" (per nombre de peticions -valor per defecte-) o "*bytraffic*" (per quantitat de bytes rebuts), entre altres.

*Directives definides al mòdul "cache" (i família)

Per "catxejar" s'entén un procediment que busca millorar el rendiment d'un servidor web i que consisteix en emmagatzemar en la RAM el contingut més demanat pels clients, de manera que se'ls permeti un accés més ràpid. La idea de "catxejar" un determinat contingut és evitar la repetició d'operacions intensives que consumeixen recursos del servidor (principalment cicles de CPU i temps d'I/O a disc) i que allarguen els temps de resposta, de manera que s'acceleri el processament de la resposta i el seu enviament.

A més d'activar "**cache_module**" (https://httpd.apache.org/docs/current/mod/mod_cache.html), el mòdul que implementa la infraestructura bàsica per poder "catxejar" (tant a Ubuntu com a Fedora per defecte ja ho està), el servidor Apache disposa de diferents mètodes de "catxejament" (cadascun associat a un o més mòduls accessoris depenents de l'anterior) que convé conèixer-los per poder escollir en cada moment i circumstància quin d'aquests mètodes és el més adeqüat. Concretament:

- D'entre les directives més interessants del mòdul general "cache" (les quals podran especificar-se a nivell de configuració general o bé de "virtualhost") podem destacar:

  CacheQuickHandler {on|off} : Si aquesta directiva val "on" -valor per defecte-, el contingut de la memòria cau és confrontat amb allò demanat en cada petició en una fase molt temprana dins del procés de gestió de la petició per part de l'Apache; això vol dir que si hi ha èxit en la confrontació, la resposta serà servida molt ràpidament sense realitzar cap altre gestió (com seria una eventual redirecció, o una autenticació/autorització del contingut -compte amb això!-, o fins i tot qualsevol evaluació de seccions *<Location>* o *<Directory>* etc). En canvi, si val "off", la confrontació amb el contingut de la memòria cau es realitzarà significativament més tard en el procés de gestió de la petició per part de l'Apache, permetent així que les directives convencionals tinguin efecte.

CacheEnable {disk | socache} {rutaURLLocalAPartirDOnEsCatxeja | URLServidorRemot} : Activa el mètode de catxejament especificat com a primer paràmetre (el qual pot ser o bé "*disk*" o bé "*socache*"...per veure'n la diferència llegiu les notes següents) per a tot el contingut penjant a partir de la ruta de la URL servida pel propi Apache local especificada com a segon paràmetre (o bé, si el servidor Apache actua com a proxy, pel contingut obtingut d'algun servidor remot, indicat en aquest cas amb la URL sencera)

NOTA: Aquesta directiva es pot indicar també dins d'una secció *<Location>* o *<LocationMatch>*; si és el cas, s'ha d'obviar llavors el seu segon paràmetre
NOTA: També existeix la directiva *CacheDisable {rutaURLLocal | URLServidorRemot}* , amb significat evident

CacheDefaultExpire nº i *CacheMaxExpire nº* : La primera directiva especifica el temps de vida (TTL) -en segons- que el contingut "catxejat" romandrà a la memòria cau abans de considerar-se caducat (si no hi ha cap altre factor que indiqui algun altre valor, com ara les capçaleres HTTP de petició *Expires:nº* o *Cache-Control: max-age=nº* en el cas del mètode de "catxejament" estàndard HTTP). La segona especifica el temps de vida màxim -en segons també- que, en tot cas, pot estar un contingut a la memòria cau abans de considerar-se caducat.

NOTA: Per saber-ne més sobre les capçaleres HTTP anteriors, es pot consultar respectivament https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Expires i https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control El servidor Apache té un mòdul anomenat "**expires_module**" (https://httpd.apache.org/docs/current/mod/mod_expires.html), que és específic per controlar el valor de la capçalera *Expires:* en respostes HTTP (i també el de l'opció *max-age=* de la capçalera *Cache-Control:*) per tal de sobreescriure la configuració establerta a la directiva *CacheDefaultExpire* i també definir el comportament adient a la memòria cau del navegador. Si es volgués controlar altres valors de la capçalera *Cache-Control:*, caldrà fer ús del mòdul genèric "headers_module", ja estudiat.

➢ D'entre els mètodes concrets de "catxejament" que podem implementar (cadascun amb un/s determinat/s submòdul/s específic/s associat/s amb sengles directives particulars) podem destacar:

- "*Key-value caching*" : Principalment emprat en processos d'autenticació i/o de xifratge (com el TLS), guarda en RAM ítems que costen de computar repetidament en forma d'"objecte compartit". Per tant, no serveix per catxejar contingut en sí sinó les operacions involucrades que permeten establir l'accés del client a aquest contingut (com els detalls de l'autenticació usada -on, segons el mètode emprat, si aquest és complex -via LDAP, via SGBDs, etc- el servidor podria patir un impacte de rendiment per cada intent d'autenticació no catxejada; i/o els detalls de les sessions TLS -on el "handshake" que s'ha d'(re)establir a cada sessió també penalitza el rendiment si no es catxeja-, etc)

NOTA: Existeixen diverses implementacions de l'"objecte compartit" (***shared object***) a la memòria RAM ; tots ells permeten utilitzar el mètode de catxejament de tipus "key-value" però cadascun té les seves pròpies característiques i necessita tenir activat un mòdul diferent. En concret, podem destacar els mòduls "**socache_dbm_module**" (que implementa una base de dades simple de tipus "DBM", la qual és un magatzem clau-valor basat en fitxers que fa ús de hashes), "**socache_shmcb_module**" (que és una millora de l'anterior, on s'implementa un anell que automàticament elimina les entrades més antigues si s'omple -la seva mida és configurable-) "**socache_memcache_module**" (que fa ús d'un programa extern anomenat "Memcached" (https://memcached.org), especialitzat en gestionar bases de dades de tipus "key-value" distribuïdes) o "**socache_redis_module**" (que fa ús d'un altre programa extern, en aquest cas anomenat "Redis" (https://redis.io), també especialitzat en gestionar bases de dades de tipus "key-value" distribuïdes). Es pot trobar una llista més exhaustiva a https://httpd.apache.org/docs/current/socache.html En tot cas, és imprescindible, per poder triar quina d'aquestes implementacions anteriors es vol fer servir (a més de per establir configuracions generals de l'objecte compartit), emprar certes directives només disponibles si es té activat (també) el mòdul genèric "**cache_socache_module**" (https://httpd.apache.org/docs/current/mod/mod_cache_socache.html), (el qual, al seu torn, ja sabem que necessita tenir activat també el mòdul general "cache_module"). Per tant, en resum, calen els mòduls "cache", "cache_socache" i alguns dels quatre mòduls d'implementació "socache_*" específics

NOTA: Depenent del tipus d'elements a emmagatzemar en l´"objecte compartit" a la RAM, a més dels mòduls anteriors també caldrà tenir activat algun altre mòdul més específic, com ara "**ssl_module**" per si es volen catxejar sessions TLS (aquest mòdul proporciona directives com *SSLSessionCache*, *SSLStaplingCache*, etc que estudiarem més endavant) o "**authn_socache_module**" per si es vol emprar la directiva *AuthnCacheSOCache*, entre altres. Podeu llegir https://www.digitalocean.com/community/tutorials/how-to-configure-apache-content-caching-on-ubuntu-14-04 per més informació

- "*Standard HTTP caching*" : El mecanisme més comú; emmagatzema en RAM respostes HTTP i les valida quan expiren. Es pot configurar en detall per prioritzar rendiment o flexibilitat. A més, ofereix la possibilitat que el propi client pugui controlar fins a cert punt si vol obtenir respostes catxejades (o no) mitjançant la presència de determinades capçaleres HTTP de petició específiques.

  NOTA: En aquest cas, podem tornar a poder utilitzar alguna de les implementacions basades en l'arquitectura "shared object" a RAM ja vistes (com ara "**socache_dbm_module**", "**socache_shmcb_module**", "**socache_memcache_module**" o "**socache_redis_module**", totes elles depenents del mòdul-arrel "**cache_socache_module**", basat al seu torn en el mòdul principal "cache_module") o bé usar (a més del sempre present "cache_module"), "**cache_disk_module**" (https://httpd.apache.org/docs/2.4/mod/mod_cache_disk.html) -que a Fedora ja ve activat de sèrie però a Ubuntu caldrà fer *sudo a2enmod cache_disk*-, el qual basa el seu funcionament en una memòria cau implementada en disc. En aquesta memòria, les capçaleres HTTP (i també els cossos) de les respostes són emmagatzemades dins del disc en una estructura de carpetes derivada a partir del hash MD5 de cada URL catxejada en qüestió. Utilitzar aquest mecanisme de "catxejament" és molt interessant quan es vol fer de proxy amb contingut obtingut de servidors remots, o generat des d'un procés dinàmic (com poden ser els resultats dels scripts PHP) o simplement quan es vol accelerar les respostes agafant els recursos catxejats d'un disc més ràpid del disc principal on normalment hi són.

  NOTA: És important aclarir que en fer servir el mòdul "cache_disk", tot i tenir els fitxers "catxejats" en disc, l'Apache aprofita els mecanismes integrats dins del sistema operatiu per mantenir el seu accés en memòria RAM.

  NOTA: Cal tenir en compte que, quan es fa servir el mòdul "cache_disk", la memòria cau en disc no s'esborra automàticament, així que cal invocar a una comanda específica que fa aquesta tasca: *htcacheclean* (la qual té paràmetres per llistar les URL catxejades, per esborrar les URL explícitament indicades, per mantenir la memòria cau en disc dins d'una determinada mida o dins d'un determinat límit de nombre d'inodes, etc) o bé sota demanda o bé de forma periòdica (per així mantenir una monitorització contínua; en aquest darrer cas, això es pot fer com a dimoni Systemctl -el qual a Ubuntu s'anomena *apache-htcacheclean* i pertany al paquet "apache2-utils", i a Fedora s'anomena directament *htcacheclean* i s'instal·la juntament amb el propi paquet de l'Apache) o bé com a tasca programada puntual via Cron o timers)

  NOTA: La memòria cau HTTP d'Apache és una memòria de "tres estats" (a diferència de la memòria basada en "objecte compartit", que és de dos). Això significa que el contingut emmagatzemat pot estar en un de tres estats: pot ser fresc (és a dir, es pot servir als clients sense més verificació), pot estar obsolet (és a dir, que el TTL del contingut ha caducat; aquest és l'estat que no incorpora el "shared object"...ja que en aquest cas si el contingut ja no és fresc directament és destruït), o pot ser inexistent (si el contingut no es troba a la memòria cau). Si el contingut es torna obsolet, a la següent sol·licitud, la memòria cau pot revalidar-lo comprovant el contingut a l'origen; si no ha canviat, restablirà la data de frescor i publicarà el contingut actual; en cas contrari, obtindrà el contingut canviat i l'emmagatzemarà durant el període de temps (TTL) que indiqui la política de memòria cau establerta.

  D'entre les directives més interessants del mòdul general "cache_disk" que s'han mencionat a les notes anteriors (les quals poden especificar-se tant a nivell de configuració general -de fet, a Ubuntu ja existeix un arxiu específic preparat per indicar-les, "/etc/apache2/mods-available/cache_disk.conf"- com a nivell de "virtualhost") podem destacar les següents:

  CacheRoot /ruta/carpeta : Indica l'ubicació en disc de la memòria cau

  CacheDirLevels nº i *CacheDirLength nº* : Ambdues directives (obligatòries) serveixen per definir com es construirà l'estructura del directoris de la memòria cau. Aquesta estructura estarà formada per fitxers que guarden el contingut catxejat i que tenen un nom que serà el hash MD5 de la URL associada a aquest contingut (per tant, serà un nom unívoc que es farà servir com a clau per identificar les dades corresponents a cada URL). Més en concret, les dades s'organitzen dins de la memòria cau en directoris el nom dels quals es deriva dels primers caràcters de cada hash, on *CacheDirLevels* indica el nombre de subdirectoris que hi haurà i *CacheDirLength* indica quants caràcters s'utilitzaran com a nom a cada directori. Per exemple, si tenim un hash com "b1946ac92492d2347c6235b4d2611184" i les directives *CacheDirLevels 2* i *CacheDirLength* 1, el contingut en qüestió es guardaria dins de la cau en una estructura de directoris com "b/1/946ac92492d2347c6235b4d2611184".

NOTA: Altres directives que es poden definir són *CacheMaxFileSize nº* i *CacheMinFileSize nº* , que estableixen el rang de mida dels fitxers (en bytes) que Apache guardarà en la memòria cau. També són interessants *CacheReadSize nº* i *CacheReadTime nº*, que permeten esperar i "buffejar" contingut abans d'enviar-lo al client (això és útil si el contingut resideix en algun altre lloc diferent del servidor Apache)

NOTA: Quan un "virtualhost" té diversos noms "www.elmeudomini.com", "elmeudomini.com", etc), utilitzar la directiva *UseCanonicalName on* pot millorar dràsticament els èxits de confrontació en la memòria cau perquè així "virtualhosts" amb diversos noms no produeixen entitats diferents a la memòria cau sinó que el contingut s'emmagatzema només amb el nom canònic i prou.

El tema del "catxejament" és certament complex ("què s'ha de catxejar?", "durant quan de temps?", "sota quines circumstàncies?",...). Una bona guia per tenir-ne criteri (específicament del "catxejament" de tipus estàndard HTTP) és https://www.digitalocean.com/community/tutorials/web-caching-basics-terminology-http-headers-and-caching-strategies i també https://www.mnot.net/cache_docs . Guies més específiques sobre configuracions particulars del servidor Apache que també poden ser d'ajuda són https://www.digitalocean.com/community/tutorials/how-to-configure-apache-content-caching-on-ubuntu-14-04#standard-http-caching o https://httpd.apache.org/docs/current/caching.html

*Directives definides al mòdul "dav"

WebDAV ("Web-based Distributed Authoring and Versioning"), es una serie d'extensions del protocol HTTP que permet als usuaris editar i administrar els recursos disponibles en un servidor web (normalment de forma remota). És a dir, el que ens permet WebDAV (mitjançant peticions HTTP) és gestionar totalment els fitxers emmagatzemats en un servidor web, ja sigui per moure'ls d'ubicació, per pujar-hi fitxers nous o eliminar-los, per modificar-ne les propietats, nom o nivells d'accés/seguretat dels fitxers existents, per editar-ne el seu contingut, etc. De fet, WebDAV permet que diversos usuaris estiguin treballant amb els fitxers emmagatzemats al servidor web (els quals podran ser qualsevol tipus de arxiu: documents, imatges, etc...) de forma concurrent sense problemes. A més, WebDAV també permet portar un control de les versions d'un mateix fitxer, podent recuperar així, en cas de fallada, qualsevol de les versions anteriors. En tot cas, tot seguit es mostren els passos per configurar un servidor Apache com a servidor WebDAV fent ús del mòdul "**dav_module**" (https://httpd.apache.org/docs/current/mod/mod_dav.html):

1.-Crear la carpeta (normalment ho farem sota el "DocumentRoot" pertinent) que serà el magatzem remot, i assignar-li els permisos adients (que, en aquest cas, són poder ser accedida, llegida i escrita per l'usuari de l'Apache: "www-data" a Ubuntu, "apache" a Fedora)

sudo mkdir /var/www/html/webdav
A Ubuntu: *sudo chown -R www-data:www-data /var/www/html/webdav*
A Fedora: *sudo chown -R apache:apache /var/www/html/webdav*

NOTA: Tingueu en compte que l'únic usuari real que tindrà accés serà l'usuari del sistema propi de l'Apache (és a dir, que tots els usuaris que fem servir a directives *Auth** seran "mapejat" a aquest usuari. És a dir, no hi ha un control de permisos individualitzat

1BIS.- Crear una altra carpeta (ara, en aquest cas, millor fora del "DocumentRoot") i assignar-li també els permissos adients (que també són poder ser accedida, llegida i escrita per l'usuari de l'Apache: "www-data" a Ubuntu, "apache" a Fedora).

sudo mkdir -p /opt/apache
A Ubuntu: *sudo chown -R www-data:www-data /opt/apache*
A Fedora: *sudo chown -R apache:apache /opt/apache*

Aquesta carpeta representa la ubicació dins de la qual hi haurà el fitxer de "lock" (el qual serà indicat concretament per la directiva *DavLockDB*, veure més avall). Un fitxer de "lock" és el mecanisme intern de l'Apache (concretament, una base de dades en format SDBM) que serveix per gestionar les eventuals escriptures concurrents que diversos usuaris puguin voler fer en un mateix moment a un mateix contingut ofert pel servidor WebDAV.

2.-Crear un "virtualhost" específic per tal de compartir la carpeta-magatzem a la url http://www.miservidor.com/webdav. Per exemple, podríem crear un fitxer anomenat "webdav.conf" (a Ubuntu seria dins de la carpeta "/etc/apache2/sites-available" -i recordem que després és necessària la corresponent activació mitjançant *sudo a2ensite webdav*- i a Fedora dins de la carpeta "/etc/httpd/conf.d") amb un contingut similar al següent (la directiva *Dav On* també podria estar indicada dins d'una secció *<Directory>* o globalment a tota la configuració del "virtualhost"):

```
<VirtualHost *:80>
        ServerName www.miservidor.com
        DocumentRoot /var/www/html
        DavLockDB /opt/apache/lol
        Alias /webdav /var/www/html/webdav
        <Location /webdav>
                Dav On
        </Location>
</VirtualHost>
```

NOTA: La directiva *DavLockDB* indica la ruta absoluta del fitxer SDBM (incloent el seu nom però no pas la seva extensió!) que el mòdul "dav" farà servir en aquest "virtualhost". Si no s'indiqués una ruta absoluta en aquesta directiva (és a dir, només el nom del fitxer i ja està), s'entendrà quela seva ruta és relativa a "ServerRoot".

NOTA: El fet de tenir la carpeta "WebDav" dins del "DocumentRoot" ens estalvia haver d'indicar una secció *<Directory>* amb la directiva *Require ...* al seu interior (com passava amb els àlias) per tal de donar permís al seu accés (l'arxiu de "lock", tot i estar fora del "DocumentRoot", com que és accedit internament pel procés Apache i no pas des dels usuaris clients no la necessita tampoc). No obstant, si tenim el framework SELinux activat, caldrà informar-lo de què l'Apache volem que hi tingui permís d'escriptura (més enllà dels permisos estàndard, que ja estan correctament assignats). Per tant, caldrà executar les següents comandes abans de fer el següent punt:
chcon -R -t httpd_sys_rw_content_t /var/www/html/webdav && chcon -t httpd_sys_rw_content_t /opt/apache/lol

NOTA: Opcionalment, es pot limitar, mitjançant la configuració d'algun mecanisme d'autenticació, l'accés de determinats usuaris a la carpeta-magatzem (encara que tots tinguin, com ja s'ha dit, un cop autenticats, els mateixos permisos que té l'usuari del sistema de l'Apache).. Per exemple, si escrivim les següents línies dins de les etiquetes *<Location /webdav>...</Location>* anteriors estarem usant el sistema d'autenticació bàsic ja conegut (recordeu que caldrà crear l'arxiu "/etc/apache2/users.password" amb la comanda *htpasswd -c...* suposarem en tot cas que allí existeix un usuari anomenat "pepe"):

```
AuthType Basic
AuthName WebDAV
AuthUserFile /etc/apache2/users.password
Require valid-user
```

També podríem fer-ho amb l'autenticació "digest" si tenim el mòdul "auth_digest" activat (que per defecte sol ser el cas). En aquest cas caldrà crear l'arxiu "/etc/apache2/users.password" amb la comanda *htdigest -c...* (suposarem en tot cas que allí existeix un usuari anomenat "pepe"):

```
AuthType Digest
AuthName WebDAV    #Aquest nom caldrà indicar-ho a la comanda htdigest, recordeu
AuthUserFile /etc/apache2/users.password
Require valid-user
```

3.-A Ubuntu cal activar els mòduls necessaris per implmentar la funcionalitat DAV (a Fedora ja ho estan, d'activats), concretament així: *sudo a2enmod dav && sudo a2enmod dav_fs* En tot cas, sigui com sigui caldrà finalment reiniciar el servei.

La manera més habitual d'interaccionar des d'un client amb un servidor WebDAV és mitjançant l'ús de programes-clients natius del sistema que puguin realitzar sobre els fitxers emmagatzemats en el servidor les accions típiques que realitzaríem sobre fitxers locals (i que l'extensió del protocol HTTP WebDAV ofereix, com ara afegir fitxers, copiar-los, eliminar-los, etc) , cosa que un navegador estàndard no pot.

NOTA: Una altra manera, que sí que ens permetria utilitzar els navegadors per realitzar totes aqueses accions WebDAV des d'un client seria tenir programada una pàgina web dinàmica (amb el llenguatge PHP, per exemple, entre altres) allotjada al servidor WebDAV que fes tota la "feina bruta" de gestió de l'emmagatzematge dels fitxers. Això és el que fan aplicacions web com NextCloud (https://nextcloud.com) , per exemple. Però això ja s'escapa de l'objectiu d'aquest text

Un d'aquests programes-client natiu del sistema és el gestor de fitxers estàndard de l'escriptori Gnome, el Nautilus. Escrivint a la barra de direccions del Nautilus (o d'algun altre gestor de fitxers gràfics...la majoria són equivalents) o bé, alternativament, al seu apartat "Altres ubicacions"→"Connectar a un servidor", el següent: *dav://pepe:contrasenya@www.miservidor.com/webdav* accedirem al contingut de la carpeta remota indicada (i hi podrem treballar amb ell) tal com si fos el contingut d'una carpeta local més.

Un altre programa natiu és la llibreria "davfs2" (https://savannah.nongnu.org/projects/davfs2), la qual permet al sistema client reconèixer el sistema de fitxers "dav" com un sistema de fitxers local vàlid (i per tant, poder muntar, com si fos un sistema de fitxers local més, el magatzem WebDAV ofert pel servidor remot que haguem indicat). Després d'instal·lar a la màquina client aquesta llibreria, el muntatge de la carpeta WebDAV remota en una carpeta local (suposarem que aquesta es diu "/mnt/dav") es pot ver de vàries maneres:

*Executant la següent comanda com a "root" (si s'ha implementat autenticació HTTP "Basic/Digest", es preguntarà interactivament l'usuari i la contrasenya per accedir-hi):

mount -t davfs -o uid=usuari,gid=usuari http://www.miservidor.com/webdav /mnt/dav

NOTA: Les opcions *uid=* i *gid=* representen el nom (o bé l'identificador numèric) de l'usuari i grup local que serà el propietari del punt de muntatge (interessa, doncs, que sigui el mateix usuari que l'actual; si no es posa, el propietari serà el "root"). Es poden consultar més opcions fent *man mount.davfs* o bé llegint l'arxiu de configuració "/etc/davfs2/davfs2.conf"
NOTA: Depenent de la distribució és possible que calgui afegir l'usuari actual al grup "davfs2", així: *sudo usermod -a -G davfs2 usuari*
NOTA:Si el servidor WebDav demanés usuari i contrasenya i no volguéssim indicar-la interactivament, podem especificar-ho llavors en un fitxer apart anomenat "/etc/davfs2/secrets" ja existent, (únic per tots els usuaris locals del client) o bé crear-ne un de nou amb el nom "~/.dav2fs/secrets" i amb permisos 600, (propi per cada usuari local), amb el següent contingut: */mnt/dav pepe contrasenya* Una altra opció és utilitzar certificats TLS

Alternativament, es pot afegir una línia com la següent a l'arxiu "/etc/fstab" (l'avantatge d'aquesta solució envers el muntatge manual és que només en aquesta el muntatge roman després de reiniciar la màquina client):

http://www.miservidor.com/webdav /mnt/dav davfs x-systemd.automount,noauto,user,rw 0 0

NOTA: Les tres notes indicades en parlar del muntatge manual aquí també són d'aplicació
NOTA: Les opcions de muntatge *x-systemd.automount,noauto* fan que la carpeta en qüestió no es munti automàticament a l'arranc de l'ordinador sinò en voler-hi accedir per primer cop, un cop ja iniciada la sessió de l'usuari i, per tant, establerta la configuració de xarxa. Si no es fa així, es corre el risc de voler muntar la carpeta remota abans de què les tarjes de xarxa estiguin configurades bé. L'opció *user* permet que un usuari no administrador pugui també muntar la carpeta i l'opció r*w* indica que el muntatge es farà en mode lectura-escriptura

*Utilitzant el client WebDAV de consola "cadaver" (https://github.com/notroj/cadaver, serà necessari instal·lar-lo primer al sistema client perquè no sol venir de sèrie), tal com es mostra a continuació. Aquesta comanda permet administrar interactivament el contingut de la carpeta remota mitjançant comandes d'un shell propi molt semblants a les clàssiques d'un client FTP: *ls, lls, pwd, lpwd, cd, lcd, mkdir, rmdir, mput, mget, mdelete, quit*, així:

cadaver http://pepe:contraseña@www.miservidor.com/webdav

*Els mòduls MPM

Apache ofereix diferents mòduls per gestionar les conexions que li arriben dels clients (els quals s'anomenen genèricament MPM, de "multi-processing modules"). No obstant, només un mòdul MPM pot estar activat en un determinat moment: si es vol canviar de mòdul MPM cal desactivar l'actual i activar llavors el desitjat. Els mòduls MPM més importants són tres:

"*Prefork*" (mpm_prefork): Aquest mòdul crea un conjunt de processos fill en arrencar el servidor (visibles amb eines com top o ps ax), els quals seran els responsables d'escoltar i atendre les peticions que rebin. Cada procés fill gestiona només un fil d'execució, i això implica que només pot gestionar una petició a cada moment. Degut a això, la velocitat de resposta baixa força quan hi ha moltes peticions concurrents, havent- se d'esperar algunes d'elles en una cua. Per evitar aquest problema, el servidor Apache pot incrementar el número de processos fill a crear, però això té l'inconvenient d'augmentar l'ús de RAM. És el recomanat si es vol utilitzar l'intèrpret PHP en forma de mòdul integrat dins l'Apache (paquet "libapache2-mod-php" a Ubuntu) però cal dir que aquesta implementació PHP no és la més òptima. A més, no funciona amb HTTP/2

"*Worker*" (mpm_worker): Aquest mòdul crea un conjunt de processos fill en arrencar el servidor (visibles amb eines com top o ps ax) i cada procés fill al seu torn crea un conjunt de fils d'execució, cadascun dels quals està llest per rebre una petició diferent. Això permet gestionar més peticions concurrents i és més eficient en l'ús de la RAM, però és menys robust i no pot usar-se si s'empren altres mòduls que no són "thread-safe" (i el "libapache2-mod-php" no ho és).

"*Event*" (mpm_event) : Mòdul activat per defecte. A l'igual que el mòdul "worker", aquest mòdul crea un conjunt de processos fills en arrencar el servidor i cada procés fill al seu torn crea un conjunt de fils d'execució, però ara un d'aquests fils està associat a una conexió de tipus "keepAlive", la qual serveix per rebre les peticions i despatxar-les a algun altre fil d'execució que estigui lliure en aquell moment. Això permet gestionar més peticions concurrents encara. Si volem utilitzar l'intèrpret PHP fent servir aquest mòdul, però, hem d'emprar la implementació FastCGI , la qual s'anomena "php-fpm" (i a més, això vol dir, entre altres coses, que cal fer ús del mòdul "proxy_fcgi" de l'Apache).

> NOTA: A Ubuntu existeix un paquet anomenat "libapache2-mod-fastcgi" (no instal.lat per defecte) que representa una implementació diferent del mòdul "proxy_fcgi" més antiga que no farem servir

Per activar o desactivar un mòdul MPM cal fer el mateix que amb qualsevol altre mòdul: a Ubuntu podem executar simplement la comanda *sudo a2enmod mpm_worker*, per exemple, per activar el mòdul "worker". Cal tenir en compte, no obstant, que un mòdul MPM és incompatible amb qualsevol altre, així que l'activació d'un comportarà la desactivació automàtica de l'actual.

Cada mòdul MPM té el seu propi fitxer de configuració (a Ubuntu dins de "mods-available"), però els valors per defecte ja són prou bons. Com exemple, a continuació es presenta un contingut típic del fitxer "mpm_event.conf" (en principi no haurem de canviar els valors per defecte que allà hi són presents perquè són prou bons per la gran majoria de circumstàncies):

```
<IfModule mpm_event_module>
        #Nº de processos fill que s'inicien amb el pare (després són gestionats dinàm. segons la càrrega)
        StartServers 2
        #Nº mínim de fils d'execució ociosos (per no haver de crear-los en el moment de la petició)
        MinSpareThreads 75
        #Nº màxim de fils d'execució ociosos (no cal abusar: consumeixen memòria i CPU)
        MaxSpareThreads 250
        #Nº màxim absolut de fils d'execució possibles en total
        ThreadLimit 400
        #Nº màxim absolut de fils d'execució possibles per connexió
        ThreadsPerChild 25
        #Nº màxim de clients conectats simultàniament permesos (els qui superin el límit esperaran en
        #una cua. Si aquest número és massa baix,les conexions a la cua poden perdre's de tant esperar;
        #si és massa alt, pot provocar esgotament de memòria, degradant severament el rendiment
        MaxRequestWorkers 400
        #Nº màxim de peticions que pot atendre un procés fill durant la seva vida. Una vegada aquest
        #límit ha sigut assolit, el procés fill mor. Si s'estableix a 0, el fill mai expira.
        MaxConnectionsPerChild 10000
</IfModule>
```