El servidor HTTP Apache (II): configuració d'accés a directoris i fitxers

Configuració d'accés a directoris

Dins de la configuració d'un "virtual host" (¡o dins de l'arxiu "apache2.conf/httpd.conf" i/o els inclosos dins de la carpeta "conf-available/conf.d"...afectant llavors a tots els "virtual hosts"!) es pot especificar la manera com volem que el servidor Apache gestioni l'accés a determinades carpetes existents en el disc dur. Això es fa amb diferents directives escrites dins d'una secció (una per cada carpeta) que s'obre amb l'etiqueta *<Directory "/ruta/carpeta">* i es tanca amb *</Directory>*. Aspectes a tenir en compte:

*Les directives allà indicades s'apliquen a la carpeta indicada (mitjançant la seva ruta absoluta dins del sistema de fitxers, no des del DocumentRoot!) i automàticament a totes les seves subcarpetes.

La ruta indicada pot contenir els comodins "" (equivalent a qualsevol número de caràcters -que no siguin el caràcter "/"-) i "?" (equivalent a un caràcter qualsevol, però un només)

*Si hi ha vàries seccions *<Directory>* vinculades a diferents carpetes que estan una dins d'una altra, les directives s'apliquen en l'ordre següent: primer les corresponents a la subcarpeta amb la ruta més curta, i així anar "cap a dins" fins a la carpeta amb una ruta més llarga (la més "interna". L'ordre d'aplicació de les directives és molt important perquè sempre -i només- s'aplica l'última de les que hi hagi repetides. En el cas de seccions *<Directory>* diferents que afectin a la mateixa carpeta , s'aplicarà la que estigui escrita als arxius de configuració la darrera de totes.

Més versàtil que la directiva <Directory> és, però, la directiva *<DirectoryMatch "/ruta/carpeta">...</DirectoryMatch>*. La única diferència està en que a *<Directory>* cal especificar la ruta exacta de la carpeta en qüestió (o bé utilitzar comodins), però a *<DirectoryMatch>* es poden utilitzar expressions regulars, de manera que es poden encabir moltes rutes diferents de cop si aquestes tenen un nom que segueix un mateix patró. Per exemple, *<DirectoryMatch /var/[a-d].*>* indica qualsevol subcarpeta dins la carpeta "/var" del sistema el nom de la qual comenci per "a","b","c" o "d").

NOTA: Per saber més sobre expressions regulars aconsello consultar l'estupenda web http://www.regular-expressions.info o també https://en.wikipedia.org/wiki/Regular_expression#Standards. Un resum introductori també es pot trobar a https://www.digitalocean.com/community/tutorials/an-introduction-to-regular-expressions

NOTA: Escriure el símbol ~ després de "Directory" a l'etiqueta *<Directory>* és igual q usar l'etiqueta *<DirectoryMatch>*. És a dir, *<Directory ~ "/\.svn">* és equivalent a *<DirectoryMatch "/\.svn">*

Directiva *AllowOverride*

Una directiva que es pot escriure dins d'una secció *<Directory>* (però no en *<DirectoryMatch>*) que cal explicar és la directiva *AllowOverride*. Aquesta directiva s'utilitza per decidir si dins de la carpeta (i subcarpetes) indicada a la secció *<Directory>* on està escrita es tindran en compte -i si sí, de quina manera- els "arxius d'accés" (habitualment anomenats "**.htaccess**" gràcies a la directiva *AccessFileName* dins de "apache2.conf/httpd.conf"). Un "arxiu 'd'accés" és simplement un arxiu de text ubicat dins d'una carpeta que conté qualsevol de les directives que podriem trobar igualment dins d'una secció *<Directory>*. D'aquesta manera, si existís (a "apache2.conf/httpd.conf", a l'arxiu d'un "virtual host", etc.) una secció *<Directory>* associada a una determinada carpeta i dins d'aquesta mateixa carpeta es trobés un arxiu d'accés, les directives escrites en aquest sobreescriurien les directives presents a la secció *<Directory>* corresponent.

La utilitat dels arxius d'accés està en donar la possibilitat de poder configurar l'accés a una carpeta determinada per part d'usuaris que no tenen permisos al sistema per modificar els arxius de configuració generals (és a dir, l'arxiu "apache2.conf/httpd.conf", els ubicats dins de "conf-available.d/conf.d" o els corresponents als "virtualhosts" -ubicats dins de "sites-available.d/conf.d"-, etc). No obstant, si s'utilitzen (en principi estan desactivats via *AllowOverride*, com veurem més endavant) obliguen a l'Apache a buscar-los i llegir-los a cada subcarpeta que hi hagi sota la carpeta arrel, fet que penalitza molt el seu rendiment (per exemple, si un usuari demana la pàgina http://exemple.com/una/ruta/pagina.html , l'Apache llegirà -per ordre- els eventuals fitxers d'accés presents a "/var/www/html/una" i "/var/www/html/una/ruta", i això amb cada nova petició. Un altre tema a més és l'inconvenient de què les directives escrites d'aquesta forma (és a dir, de forma descentralitzada), puguin entrar en contradicció amb el que mana la configuració general...

En tot cas, els valors més habituals que pot tenir la directiva *AllowOverride* per gestionar l'activació (parcial o total) dels arxius d'accés són:

AllowOverride AuthConfig : Permet que es puguin emprar només les directives *AuthName, AuthType, AuthUserFile i Require* (entre poques més) a l'eventual arxiu d'accés existent dins de la carpeta associada a la secció *<Directory>* on es trobi aquesta línia. Aquestes directives tenen a veure amb la protecció amb contrasenya per accedir via HTTP a la carpeta en qüestió (en parlarem en un apartat posterior)

AllowOverride FileInfo : Permet que es puguin emprar només les directives (moltes d'elles estudiades més endavant) *ErrorDocument, SetHandler, Header, RequestHeader, SetEnvIf, RewriteEngine, RewriteOptions, RewriteBase, RewriteCond, RewriteRule* i *Redirect* (entre poques més) a l'eventual arxiu d'accés existent dins de la carpeta associada la secció *<Directory>* on es trobi aquesta línia

AllowOverride Indexes : Permet que es puguin emprar només les directives *AddDescription, AddIcon AddIconByType, AddIconByEncoding, DefaultIcon, DirectoryIndex, FancyIndexing, HeaderName, IndexIgnore, IndexOptions* i *ReadmeName* (moltes d'elles pròpies dels mòduls "autoindex" o "dir", estudiats més endavant) a l'eventual arxiu d'accés existent dins de la carpeta associada a la secció *<Directory>* on es trobi aquesta línia.

AllowOverride Options : Permet que es puguin sobreescriure els valors de la directiva *Options* (llegir més avall). Si només es vol que es puguin sobreescriure certs valors d'aquesta directiva, es pot escriure *AllowOverride Options=unvalor,unaltre,... (així: AllowOverride Options=Indexes, Includes* per exemple)

AllowOverride None : No permet que es pugui emprar cap directiva a l'eventual arxiu d'accés existent dins de la carpeta associada a la secció *<Directory>* on es trobi aquesta línia. En altres paraules, desactiva els fitxers d'accés en aquella carpeta (i subcarpetes), millorant així el rendiment.

AllowOverride All : Permet que es puguin emprar totes les directives possibles a l'eventual arxiu d'accés existent dins de la carpeta associada a la secció *<Directory>* on es trobi aquesta línia. Es desaconsella el seu ús per seguretat.

Es poden combinar varis valors en una mateixa línia, p. ex així: *AllowOverride AuthConfig Options*

Directiva *Options*

Una altra directiva bàsica que es també habitual dins d'una secció *<Directory>* és *Options*, la qual igualment té diferents valors com per exemple:

Options [+|-] Indexes : Indica que si a la carpeta associada a la secció *<Directory>* actual no hi ha cap arxiu per defecte (és a dir, un arxiu anomenat "index.html" o, més exactament, algun dels indicats a la directiva *DirectoryIndex* -pertanyent a la configuració del mòdul "dir", del qual parlarem més endavant-), retorna la llista del contingut d'aquesta carpeta. Útil per mostrar llistats de fitxers (a descarregar) en comptes de pàgines web.

Options [+|-] FollowSymLinks : Indica que es poden seguir els enllaços ("accesos directes")

Options [+|-] MultiViews : Permet negociar la mostra del contingut segons la configuració del client (llenguatge, codificació). Es fa servir per saber quin idioma té el client, per exemple.

Options [+|-] ExecCGI : Es poden executar scripts CGI que hi hagin (veure + endavant)
Options [+|-] Includes : Es poden executar scripts SSI que hi hagin dins d'aquesta carpeta
Options [+|-] IncludeNoExec : Es poden executar scripts SSI excepte l´"exec"

Options [+|-] All : Totes les anteriors

Si hi ha un + ó -, s'apliquen les directives a sobre de les que ja s´hagin heretat. Si no hi ha res, es reseteja i es torna a començar. Es poden combinar varis valors en una mateixa línia, així: *Options FollowSymLinks Indexes*, per exemple.

Directiva *Require*

Una altra directiva bàsica que es també habitual dins d'una secció *<Directory>* és *Require xxx valor ...*, la qual permet l'accés de les peticions de clients a la carpeta associada només si aquestes tenen un valor determinat de la característica indicada a "xxx". Aquí poden haver molts casos particulars, així que llistarem els més habituals:

Require all granted : Es permet l'accés completament.
Require all denied : Es prohibeix l'accés completament

Require ip 192.168.1.34 192.168.2 172.20 10 : Es permet l'accés només des dels clients que tinguin (en aquest cas concret) la ip 192.168.1.34 o què pertanyin a la xarxa 192.168.2.0 (classe C), 172.20.0.0 (classe B) o 10.0.0.0 (classe A).

NOTA: L'anterior exemple també es podria haver escrit *Require ip 192.168.1.34 192.168.2.0/24 172.20.0.0/16 10.0.0.0/8 O Require ip 192.168.1.34 192.168.2.0/255.255.255.0 172.20.0.0/255.255.0.0 10.0.0.0/255.0.0.0*
NOTA : Si volguéssim escriure *Require ip 127.0.0.1* per només donar accés a l'ordinador local a un determinat recurs, una alternativa equivalent és escriure *Require local*
NOTA : També existeix la directiva *Require host nomClient unAltre ...* , la qual permet l'accés només des dels clients amb els noms llistats (que poden ser DNS o bé reconeguts via "/etc/hosts")
NOTA : Tant *Require ip* com *Require host* funcionen si el mòdul *authz_host* està activat (per defecte ja ho està)

Require user nomUsuari unAltre ... : Es permet l'accés només als usuaris indicats (que no són usuaris del sistema sinò uns propis de l'Apache...per més informació veure l'apartat posterior que parla sobre la protecció de recursos amb contrasenya). També existeix la directiva *Require group nomGrup unAltre ...*, la qual és similar però per a grups d'usuaris i *Require valid-user*, la qual permet l'accés a tots els usuaris que estiguin definits dins l'Apache sense distinció).

NOTA : Aquesta funcionalitat funciona si el mòdul *authz_user* està activat (per defecte ja ho està)
NOTA: En la majoria dels casos (tal com veurem a l'apartat corresponent), per realitzar el procés d'autenticació correctament les directives *Require user, Require group o Require valid-user* han de venir acompanyades d'altres directives auxiliars, concretament: *AuthName*, *AuthType*, *AuthBasicProvider* (o *AuthDigestProvider*), *AuthUserFile i AuthGroupFile*.

Require expr expressió (on expressió pot ser qualsevol de les indicades a la documentació oficial en http://httpd.apache.org/docs/current/expr.html) : Es permet l'accés només si l'expressió és certa.

NOTA: Tal com es detalla a l'enllaç anterior, una expressió senzilla sol estar composta per tres elements: el nom (escrit dins dels símbols "*%{ ... }*") d'una determinada variable ja predefinida pel propi Apache i que normalment conté algun valor relacionat amb la petició en qüestió, un operador (que, segons el tipus, permet comparar cadenes o números sencers) i el valor concret a contrastar. Es poden combinar també diferents expressions amb connectors com "||" (OR) o "&&" (AND), i també es poden utilitzar diverses funcions predefinides, entre altres capacitats. A continuació es mostren alguns exemples :

Require expr %{HTTP_USER_AGENT} != 'BadBot' : S'accepten peticions de tots els clients excepte de 'BadBot'
Require expr "%{TIME_HOUR} -ge 9 && %{TIME_HOUR} -le 17" : S'accepten peticions només entre 9h i 17h
Require expr "!(%{QUERY_STRING}=~/xx/) && %{REQUEST_URI} in {'/a, '/b'}" : S'accepten peticions que no continguin dins la seva "querystring" la cadena (o millor dit, l'expressió regular) "xx" i que, a més, demanin un recurs la url del qual no contingui les cadenes "/a" o "/b"

Require method GET POST ... : Es permet l'accés només als metodes HTTP indicats
Require env variableEntorn unaAltraVariable ... : Es permet l'accés només si la/es variable/s d'entorn indicada/es està/an definida/es (és a dir, té/tenen valor). Aquestes variables poden ser del sistema operatiu o bé internes de l'Apache; s'estudiaran en detall més endavant en posteriors documents quan veiem les directives *SetEnv/SetEnvIf* i *PassEnv*

Totes les directives *Require xxx* anteriors admeten la possibilitat de ser negades mitjançant la paraula "not" així *Require not xxx valor*. És a dir, per exemple, per permetre l'accés a qualsevol client que NO tingui la IP 192.168.1.34 hauríem d'escriure *Require not ip 192.168.1.34*

Si es volen indicar més d'una línia *Require xxx* per una secció *<Directory>* determinada (o un fitxer d'accés determinat, que ve a ser el mateix), és recomanable encabir-les totes dins d'una secció iniciada per l'etiqueta *<RequireAll>* i acabada per *</RequireAll>*. En aquest cas, cal tenir present que l'ordre en què estiguin escrites les diferents línies *Require xxx* dins d'aquesta secció és important perquè en cas de contradicció sempre guanya la darrera escrita; per això, en general, s'acostuma a escriure com a última línea la directiva *Require all denied :* per assegurar-se de què més enllà de les directives anteriors que permeten l'accés no hi ha cap més. També existeixen les etiquetes *<RequireAny>...</RequireAny>* i *<RequireNone>...</RequireNone>*

> NOTA: D'altra banda, les directives *Alias* i *ScriptAlias* també solen acompanyar a seccions *<Directory>*, però les veurem més endavant, en veure el mòdul "alias"

Exemples

A l'arxiu "apache2.conf/httpd.conf" hi ha per defecte un parell de seccions *<Directory>* interessants, com per exemple la que s'aplica a la carpeta arrel ("/") del sistema de fitxers. Aquí està (com es pot veure, denega l'accés a tot el contingut sota "/" -excepte el que s'especifiqui en carpetes subseqüents- i desactiva els arxius d'accés):

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

Una altra secció *<Directory>* que es troba definida per defecte a "apache2.conf/httpd.conf" és la que correspon a la carpeta "/var/www", contenidora dels documents del "virtual host" que ve predefinit en una instal·lació d'Apache per defecte. Aquí està (es pot veure com aquí la línia *Require* sobrescriu el que s'havia indicat a la secció Directory de la carpeta arrel):

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

Configuració d'accés a fitxers

Si el que es necessita és controlar fitxers concrets (en comptes de carpetes) es pot usar la secció *<Files "nomFitxer">...</Files>*. Cal tenir en compte, però, que en aquesta secció no s'especifica la ruta del fitxer sinò només el seu nom: això vol dir que si s'escrivís dins de "apache2.conf/httpd.conf" (o similar), aquesta secció afectaria a tots els arxius amb el nom indicat que existissin a tot el sistema de fitxers del servidor; si s'escrivís dins de l'arxiu de configuració d'un "virtual host" afectaria a tots els arxius amb el nom indicat dins de la seva carpeta arrel (i subcarpetes); si volguéssim, en canvi, restringir la recerca d'aquests fitxers només a una carpeta (i subcarpetes) determinada, hauríem d'incloure la secció *<Files>* dins de la secció *<Directory>* adient.

Més útil que la directiva *<Files>* és, però, la directiva *<FilesMatch "nomFitxer">...</FilesMatch>*. La única diferència està en que a *<Files>* cal especificar el nom exacte del fitxer en qüestió (o bé utilitzar comodins), però a *<FilesMatch>* es poden utilitzar expressions regulars, de manera que es poden encabir molts fitxers diferents de cop si aquests tenen un nom que segueix un mateix patró.

> NOTA: Escriure el símbol ~ després de "Files " a l'etiqueta *<Files>* és equivalent a fer servir l'etiqueta *<FilesMatch>*. És a dir, *<Files ~ "^\.ht">* és equivalent a *<FilesMatch "^\.ht">*

La directiva més habitual que se sol escriure dins d'una secció *<Files>* o *<FilesMatch>* és la *Require xxx* explicada als paràgrafs anteriors (però aplicada en aquest cas a fitxers en lloc de carpetes, òbviament). Així, per exemple, trobem que a "apache2.conf/httpd.conf" hi ha respectivament, les línies,...

```
<FilesMatch “^\.ht”>
        Require all denied
</FilesMatch>
```

```
<Files “.ht*”>
        Require all denied
</Files>
```

...les quals deneguen l'accés (i per tant, la lectura) a tots els arxius del sistema de fitxers del servidor que el seu nom comenci amb ".ht" (recordem que en una expressió regular com la de l'esquerra, el símbol ^ indica "començament" i el símbol \ indica que el següent caràcter -el punt- deixa de tenir el significat especial que en una expressió regular habitualment té). És a dir, protegeix el contingut dels possibles arxius ".htaccess" (entre altres) de la vista dels visitants.

En tot cas, si es vol limitar l'abast d'aplicació d'una secció *<Files>* o *<FilesMatch>* als fitxers ubicats dins d'una determinada carpeta, sempre es pot escriure dins d'una secció *<Directory>* corresponent a aquesta carpeta, per exemple així (en aquest cas, es denega l'accés al fitxer anomenat "private.html" només si aquest està ubicat dins de la carpeta "/var/web/dir1" o en les subcarpetes que hi pugui haver al seu interior):

```
<Directory "/var/web/dir1">
        <Files "private.html">
                Require all denied
        </Files>
</Directory>
```

Configuració d'accés a "locations"

A més de les seccions *<Directory>* i *<Files>* (que controlen el comportament relacionat amb ubicacions dins del sistema de fitxers del servidor), també existeix la secció *<Location "/una/urlpath">... </Location>* -i la seva corresponent *<LocationMatch "/una/urlpath"> ... </LocationMatch>*-, que serveix per controlar el comportament relacionat amb la part de la URL rebuda del client posterior al nom DNS del servidor

Per exemple, si s'escriu a la barra del navegador la URL http://www.exemple.com/webmail/inbox, (i suposant que tenim un "virtual host" vinculat al nom "www.exemple.com") l'Apache en principi mirarà en la carpeta "**/webmail/inbox**" sota el DocumentRoot definit. Podríem, per tant, definir una secció en aquest "virtual host" tal com *<Location /webmail/inbox>* per gestionar les peticions vinculades a aquesta ruta determinada.

La gràcia, però, de fer servir la secció *<Location>* en lloc de *<Directory>* és que la ruta indicada a la primera pot no tenir res a veure amb la jerarquia de carpetes reals (tal com veurem quan veiem l'apartat sobre els àlies). És a dir, que en l'exemple anterior hem suposat que "/webmail/inbox" és una carpeta real sota el DocumentRoot definit però no té perquè ser així, ja que l'Apache incorpora mecanismes de "mapeig" (els mencionats "àlies") que fan que una ruta concreta en una URL (una "location") es correspongui a la ruta d'una determinada carpeta del disc dur (un "directory") que no tingui res a veure. D'aquesta manera, s'aconsegueix que l'usuari, en fer peticions a "locations", no sàiga quina és la jerarquia real de carpetes internes en el servidor.

> NOTA: Remarcar que un valor tal com "/una/urlpath" indicat en una secció *<Location>* farà que s'apliquin les directives del seu interior quan es rebin peticions dirigides a rutes tal com "/una/urlpath", "/una/urlpath/" (fixeu-vos en la barra del fintal) i també "/una/urlpath/qualsevolcosa" però no pas "/una/urlpathqualsevolcosa". En el cas de les seccions *<LocationMatch>* ja dependrà de l'expressió regular concreta emprada.

Directiva *SetHandler*

Aquesta directiva pot anar escrita dins d'una secció *<Directory>* (o *<DirectoryMatch>*) però és més habitual trobar-la dins de les seccions *<Files>* (o *<FilesMatch>*) o *<Location>* (o *<LocationMatch>*) Bàsicament pot tenir dos valors:

SetHandler nomHandler : En rebre's una petició demanant qualsevol fitxer que es trobi dins de la carpeta corresponent a la secció *<Directory>* o *<Location>* on aquesta directiva estigui escrita (o que es correspongui amb algun dels fitxers indicats a la secció *<Files>* on ella estigui escrita), aquesta directiva fa que la petició sigui derivada i tractada pel "handler" indicat. Un "handler" és un miniprograma independent que ja ve preincrustat dins del propi Apache però que no forma part de l'executable principal. A http://httpd.apache.org/docs/current/handler.html es troben els "handlers" oferts pel propi Apache per defecte (tot i que els únics que estudiarem -més endavant- són els "handlers" anomenats "server-info" i "server-status").

SetHandler "rutaOUrlPrograma" : En rebre's una petició demanant qualsevol fitxer que es trobi dins de la carpeta corresponent a la secció *<Directory>* o *<Location>* on aquesta directiva estigui escrita (o que es correspongui amb algun dels fitxers indicats a la secció *<Files>* on ella estigui escrita), aquesta directiva fa que la petició sigui derivada i tractada per un executable de tercers (com pot ser, per exemple, l'intèrpret PHP-FPM), ja sigui invocant-lo directament a través de la seva ruta o bé a través d'una URL si fos un executable remot. Un exemple força comú justament és associar les peticions d'arxius amb extensió ".php" a un "handler" que sigui l'intèrpret PHP-FPM (el significat del valor concret usat aquí ho estudiarem més endavant), així:

```
<FilesMatch \.php$>
        SetHandler "proxy:unix:/run/php/php8.1-fpm.sock|fcgi://localhost/"
</FilesMatch>
```

Ordre d'aplicació de les directives presents dins de seccions *<Directory>*,*<Files>* i *<Location>*

És possible que tinguem, en un o diversos fitxers ".conf", diferents seccions *<Directory>* (o *<DirectoryMatch>*), *<Files>* (o *<FilesMatch>*) o *<Location>* (o *<LocationMatch>*) que afectin al mateix recurs. En aquest cas, ¿quin és l'ordre de preferència de cadascuna d'aquestes directives per saber quina s'aplica finalment? Doncs és el següent (tenint en compte que la darrera directiva trobada és la que s'aplica):

1.-Directives dins de seccions *<Directory>*. Entre elles se segueix l'ordre de les rutes indicades (és a dir, les directives associades a cada subcarpeta interior s'avaluen després de les indicades a la carpeta on està ubicada aquesta subcarpeta). En el cas que diverses seccions *<Directory>* afectessin a la mateixa carpeta, llavors se segueix l'ordre amb que apareguin dins dels fitxers de configuració.
2.-Directives dins d'arxius ".htaccess". Se segueix el mateix criteri per l'ordre que en el punt anterior
3.-Directives dins de seccions *<DirectoryMatch>* (i *<Directory "~">*). Entre elles se segueix l'ordre amb que apareguin dins dels fitxers de configuració
4.-Directives dins de seccions *<Files>* i *<FilesMatch>*. Entre elles se segueix l'ordre amb que apareguin dins dels fitxers de configuració
5.-Directives dins de seccions *<Location>* i *<LocationMatch>*. Entre elles se segueix l'ordre amb que apareguin dins dels fitxers de configuració

Per tant, com a resum i simplificant, podem dir que les directives indicades en seccions *<Location>* tenen preferència sobre les indicades en seccions *<Files>* i aquestes sobre les indicades en seccions *<Directory>*

NOTA: De fet, per exemple, una secció com *<Location "/">* seria una manera senzilla d'aplicar una determinada configuració sío o sí a tots els recursos oferits pel servidor

D'altra banda, en el cas d'haver seccions del mateix nivell que afectin al mateix recurs tant en la configuració general com dins de la configuració d'un "virtualhost", les directives indicades en aquestes darreres seran les que s'apliquin (és a dir, la configuració dels "virtualhosts" té preferència sobre la general).

TRUC: Ús de l'arxiu ".htaccess" per protegir carpetes amb contrasenya

Els passos a seguir són els següents:

NOTA: Atenció, s'ha de tenir el mòdul "**auth_basic**" (ó "**auth_digest**" segons el cas) prèviament activat per a que aquesta característica funcioni. Afortunadament, en una instal·lació per defecte de l'Apache això ja és així. És cert també que existeixen altres mòduls (del tipus "auth**n**_*" -que només autentiquen- , "auth**z**_*" -que només autoritzen- i/o "auth_*" -que fan les dues coses-) que són més sofisticats i que poden fer-se servir pel mateix objectiu , però no els veurem en ser més farragosos de configurar.

1.-Canviar la configuració original de l'etiqueta *<Directory>* corresponent a la carpeta que es vol protegir per a què posi *AllowOverride AuthConfig*.

2.-Generar un fitxer -ocult- que contingui la contrasenya de l´usuari (que pot no ser del sistema!) al que volem permetre l´accés. Això es fa amb la comanda *htpasswd -c /ruta/.fitxer nomusuariinventat* (preguntarà la contrasenya que volem). La ruta del fitxer creat NO ha de poder ser accesible a través de cap client web, lògicament. El paràmetre *-c* és per crear l´arxiu: si no el posem afegirem un nou usuari a l´arxiu existent.

NOTA: Altres paràmetres que poden ser interessants de la comanda *htpasswd* són *-b* (per poder introduir la contrasenya com a últim valor de la comanda en comptes d'interactivament), *-s* o *-m* (per emmagatzemar la contrasenya encriptada mitjançant, respectivament, l'algoritme SHA o MD5, en comptes de l'usat per defecte -el *crypt()*, que és menys segur-), o *-n* (per no guardar el resultat en el fitxer sinò treure'l per pantalla), entre altres
NOTA: Aquesta comanda ve dins del paquet "apache2-utils" (a Ubuntu) i "httpd-tools" (a Fedora), el qual , en tot cas, en haver instal.lat el servidor Apache s'haurà instal.lat com a dependència.

3.-Generar l'arxiu d'accés (".htaccess") dins la carpeta que es vol restringir amb aquest contingut:

AuthType Basic
AuthName "Missatge a mostrar"
AuthUserFile /ruta/arxiu/creat/pas/anterior
#Permet l'accés a tots els usuaris vàlids. Per usuaris concrets: *Require user usu1 usu2*
Require valid-user

NOTA: Aquestes línies, com ja sabem, es poden escriure perfectament dins d'una secció *<Directory /ruta/carpeta/a/protegir>* pertanyent al "virtualhost" adient. L'elecció de fer servir un arxiu ".htaccess" ve motivada per la circumstància (bastant habitual en serveis de "hosting", com ja hem dit) de no poder modificar directament els arxius de configuració de l'Apache.

4.-La diferència entre el mètode Basic i el Digest (tots dos, per cert, pertanyents al propi protocol HTTP) és que el primer usa un arxiu de contrasenyes (amb format *user:password*) que, tot i que dins el fitxer s'emmagatzemen encriptades, es transmeten de client a servidor en text pla, mentre que el segon transmet les contrasenyes en MD5, fet que és una mica més segur. Si fem servir el mètode Digest, cal tenir en compte dues diferències, però:

*Per crear el fitxer de contrasenyes s'ha d'emprar la comanda *htdigest* en lloc de *htpasswd* El seu funcionament és idèntic excepte en què ara a més s'ha d'indicar com a paràmetre el valor d'AuthName, així: *htdigest -c /ruta/.fitxer "Missatge a mostrar" nomusuariinventat*

*Les directives a incloure dins de l'arxiu ".htaccess" (o a la secció *<Directory>* adient) són:
AuthType Digest
AuthName "Missatge a mostrar"
AuthDigestFile /ruta/arxiu/creat/pas/anterior
Require valid-user

NOTA: Recordeu que si es troben diferents arxius ".htaccess" al llarg de l´arbre de directoris, s´aniran tenint en compte a mesura que ens anem endinsant: per tant, tindrà preferència l´últim arxiu ".htaccess" que es trobi.
NOTA: Tal com ja s'ha dit, existeixen altres mètodes d'autenticació, com ara fer que l'Apache consulti directament els usuaris en una base de dades (amb el mòdul "**auth_pgsql**") o en un servidor Ldap (amb el mòdul "**auth_ldap**"), entre d'altres, però això ara no ho veurem. Es pot consultar, en general: https://httpd.apache.org/docs/2.4/howto/auth.html

EXERCICIS:

1.-a) Crea un arxiu anomenat "lilili.conf" dins de la carpeta "sites-available/" (a Ubuntu) o "conf.d" (a Fedora) amb el següent contingut,...

```
<VirtualHost *:80>
    ServerName www.hola.org
    DocumentRoot "/var/www/lilili"
</VirtualHost>
```

... i tot seguit crea la carpeta "/var/www/lilili" i allà dins crea la subcarpeta "lolol o". Tant dins de la carpeta "lilili" com dins de la carpeta "lolol o" crea un arxiu que s'anomeni igual, "privat", i que tingui el mateix contingut HTML: *<html><body>Atenció, perill!</body></html>*. Després de reiniciar el servei Apache (no sense abans, a Ubuntu, executar *sudo a2ensite lilili*) i d'afegir, a la màquina client, la línia *ip.maq.serv.web www.hola.org* al seu arxiu "/etc/hosts", comprova que pots veure sense problema aquestes pàgines web anant respectivament a les URL http://www.hola.org/privat i http://www.hola.org/lolol o/privat

b) Afegeix la següent secció de configuració dins de la configuració del "virtualhost" anterior (és a dir, entre les etiquetes *<VirtualHost *:80>* i *</VirtualHost>*) i, després de reiniciar el servei Apache, comprova si ara pots veure les pàgines webs anteriors. ¿Per què? (observa l'expressió regular utilitzada). ¿Quin codi d'error obtens en la resposta (pots inspeccionar-la amb *curl -v* o amb les eines de desenvolupador del navegador).

```
<FilesMatch "[pP]rivat.*">
    Require all denied
</FilesMatch>
```

c) Substitueix ara la secció *<FilesMatch>* introduïda a l'apartat anterior per aquesta altra secció i, després de reiniciar el servei Apache, comprova si ara pots veure les pàgines webs anteriors. ¿Per què? (observa l'expressió regular utilitzada). ¿Quin codi d'error obtens en la resposta (pots inspeccionar-la amb *curl -v* o amb les eines de desenvolupador del navegador).

```
<LocationMatch "^.*/privat">
    Require all denied
</LocationMatch>
```

cII) Afegeix dins de la secció *<LocationMatch>* anterior, la línia *ErrorDocument 403 "Fora d'aquí"* i reinicia el servei Apache. Torna a intentar accedir a les mateixes webs que abans. ¿Quin missatge veus ara? ¿Per què?

d) Substitueix ara la secció *<LocationMatch>* introduïda a l'apartat anterior per aquesta altra secció i, després de reiniciar el servei Apache, visita la URL http://www.hola.org/lolol o i digues què veus. ¿Per què?

```
<Directory "/var/www/lilili/lolol o">
    Options +Indexes
</Directory>
```

dII) ¿Què veus si tornes a visitar la mateixa URL anterior però havent canviat abans la línia *Options +Indexes* anterior per aquesta altra línia: *Options -Indexes* (i havent reiniciant de nou el servei Apache)? D'altra banda, respon: ¿quina diferència hi ha entre escriure la directiva *Options +Indexes* (o *Options -Indexes*) i escriure *Options Indexes* ?

e) Crea un arxiu ".htaccess" dins de "/var/www/lilili/lolol o" amb el següent contingut i tot seguit (no cal reiniciar el servei Apache, ni tampoc esborrar la línia *Options -Indexes* establerta a l'apartat anterior) torna a visitar la URL http://www.hola.org/lolol o i raona per què no veus cap canvi respecte l'observat a l'apartat anterior. En aquest sentit, ¿què veus si visites la URL http://www.hola.org/lolol o/blablebli i per què?

```
Options +Indexes
ErrorDocument 404 "T'has equivocat, jajajaja!"
```

eII) Afegeix ara la línia *AllowOverride FileInfo Options* dins de la secció *<Directory>* establerta a l'apartat *d)* i, després de reiniciar el servei Apache (ara sí és necessari perquè hem modificat un arxiu "*.conf", no pas el ".htaccess", on no cal reiniciar res), torna a visitar la URL http://www.hola.org/lololo i digues què veus ara i per què. En aquest sentit, ¿què veus ara si visites la URL http://www.hola.org/lololo/blablebli i per què?

f) Crea dins de la carpeta "/var/www/lilili" un arxiu anomenat "pepe.html" amb el següent contingut HTML: *<html><body> Hola, sóc Pepe </body></html>* i tot seguit crea dins de la carpeta "/var/www/lilili/lololo" un enllaç a aquest arxiu anomenat "pepelink.html" (per exemple, amb la comanda *ln -s*). Vés a continuació a la URL http://www.hola.org/lololo/pepelink.html i digues què veus. Finalment, edita l'arxiu ".htaccess" ubicat dins de la carpeta "/var/www/lilili/lololo" per a què tingui una línia tal com aquesta: *Options +Indexes* ***-FollowSymLinks*** i torna a intentar accedir a la mateixa URL anterior. ¿Què veus ara i per què?

fII) Esborra l'arxiu "/var/www/lilili/lololo/.htaccess" i tot seguit comprova què veus en accedir a la URL http://www.hola.org/lololo/pepelink.html (i raona per què).

2.-a) Afegeix la línia següent dins de la secció *<Directory>* establerta a l'apartat *d)* de l'exercici anterior...

Require expr %{HTTP_USER_AGENT} !~ /curl./*

...i, després de reiniciar el servei Apache, vés a la URL http://www.hola.org/lololo/pepelink.html des d'un navegador i també via *curl*; ¿què passa en cadascun dels casos i per què?

NOTA: Les expressions regulars (tal com la que estem comparant a l'exemple anterior amb l'operador "!~") no s'han d'escriure entre cometes sinò, tal com es pot veure, entre barres ("/"). Si es vol que l'expressió regular sigui "case-insensitive", llavors cal escriure-la entre barres i just després de la barra del final afegir el caràcter "i", així: "/.../i"

b) Crea un arxiu anomenat "a.html" dins de la carpeta "/var/www/lilili/lololo" amb el següent contingut: *<html><body>Visca</body></html>* i tot seguit fes que la secció *<Directory>* establerta a l'apartat *d)* de l'exercici anterior només consti de la següent línia...

Require expr "%{QUERY_STRING} =~ /xx/"

...i, després de reiniciar el servei Apache, vés (amb *curl* o mab el navegador,tant és) a les URLs http://www.hola.org/lololo/a.html, http://www.hola.org/lololo/a.html?xa i http://www.hola.org/lololo/a.html?xx i raona el perquè de cada resposta obtinguda.

c) ¿Què passaria si afegíssim en l´arxiu de configuració d'un virtualhost hipotètic les següents línies?

```
<Files ~ "/var/www/^hola[a-z]+$">
      Require expr "%{TIME_HOUR} -ge 9 && %{TIME_HOUR} -le 17"
</Files>
```

NOTA: Els operadors "-gt" (més gran),"-ge" (més gran o igual),"-eq" (igual),"-ne" (no igual), "-le" (més petit o igual) i "-lt" (més petit) serveixen per comparar nombres sencers entre sí. Els operadors ">", ">=","==","!=","<=","<" també existeixen però es reserven per comparar cadenes i "=~" i "!~", tal com ja hem vist, serveixen per comparar expressions regulars.

NOTA: També existeix l'operador "-ipmatch", que serveix per comparar IPs (o bé individuals o bé de xarxa amb la forma IPXarxa/mascara). De fet, equivalent a *Require ip* seria l'expressió *Require expr "%{REMOTE_ADDR} -ipmatch 192.168.1.0/24"* (i també es podria haver escrit el mateix però de forma més eficient així: *Require expr -R 192.168.1.0/24*)

NOTA: D'altra banda, tal com s'explica a la documentació oficial (a https://httpd.apache.org/docs/2.4/expr.html#functions) és possible també indicar determinades funcions en les expressions a comparar que poden ser útils per fer càlculs estàndard, com obtenir el hash d'alguna cadena d'entrada, passar a majúscules o minúscules alguna cadena d'entrada, obtenir totes les capçaleres de les peticions, llegir el contingut d'un fitxer, obtenir el valor de variables d'entorn del sistema, etc

NOTA: Una altra secció de configuració interessant és *<Limit protocol>...</Limit>* on "protocol" pot ser GET, POST, etc. Aquesta secció serveix per gestionar l'accés al servidor (sobre tot mitjançant directives *Require xxx*) pels protocols HTTP indicats. Similar a *<Limit>* existeix la directiva *<LimitExcept>*, que funciona de forma oposada

3.-a) Protegeix amb contrasenya (fent servir el mètode Basic) la carpeta "/var/www/lilili/lololo" (publicada pel servidor Apache a l'exercici anterior) mitjançant l'ús d'un arxiu ".htaccess" amb el contingut adient. Prova-ho accedint a la URL http://www.hola.org/lololo/a.html

NOTA: Recordeu que si es troben diferents arxius ".htaccess" al llarg de l´arbre de directoris, s´aniran tenint en compte a mesura que ens anem endinsant: per tant, tindrà preferència l´últim arxiu ".htaccess" que es trobi. D'altra banda, l'arxiu es pot anomenar d´una altra manera amb la directiva *AccessFileName nomarxiu* de l´arxiu "apache2.conf/httpd.conf"

b) Elimina l'arxiu ".htaccess" anterior. Ara protegirem la mateixa carpeta fent servir el mateix mètode "Basic" però usant usuaris existents no en un fitxer sinó en una base de dades. Els passos per assolir-ho són:

NOTA: Per a què l'autenticació via fitxer de text (és a dir, fent servir l'AuthBasicProvider "file") funcioni cal que estigui activat el mòdul "authn_file", mentre que per a què l'autenticació via base de dades funcioni cal que estigui activat el mòdul "authn_dbd". No obstant, en una instal·lació estàndard d'Apache tots dos estan activats per defecte, així que no ens haurem de preocupar per aquest detall, en principi.

*) Crear una base de dades formada per una taula amb dues columnes, les quals contindran, respectivament, els noms dels usuaris i les seves respectives contrasenyes. Aquesta base de dades pot ser de tipus MySQL, PostgreSQL, etc però per simplicitat, en aquest exercici farem servir una de tipus SQLite3, la qual té la particularitat d'emmagatzemar cada base de dades en un simple fitxer (fàcilment transportable) sense haver de mantenir cap servei en marxa . Concretament, les passes per tenir aquesta base de dades funcionant (després d'haver instal·lat el paquet "sqlite3" al sistema) són:

+ Generar (i xifrar) la contrasenya que associarem a cada usuari que volguem autenticar. Això ho farem executant la comanda *htpasswd -bns nomUsuari contrasenya*
Per exemple, fent *htpasswd -bns pepe 1234* obtindrem el valor "pepe:{SHA}cRDtpNCeBiql5KOQsKVyrA0sAiA=". Haurem d'anotar el valor que ve després dels dos punts perquè és la contrasenya xifrada que haurem d'insertar a la BD.

+ Executar la comanda *sudo sqlite3 /var/mibd* per crear (si no ho està ja) una base de dades anomenada "mibd" (ubicada dins de la carpeta "/var"), i entrar-hi a treballar a dins seu (en un shell propi de SQLite; allà és on executarem, de fet, les comandes SQL detallades tot seguit)

+ Executar, dins del shell propi de SQLite, les comandes SQL següents:

CREATE TABLE usuaris (nom TEXT, contra TEXT);
INSERT INTO usuaris VALUES ("pepe","{SHA}cRDtpNCeBiql5KOQsKVyrA0sAiA=");
...(repetir la comanda INSERT tants cops com usuaris es vulguin afegir a la taula)

+ Executar la comanda *.quit* per sortir del shell propi de SQLite i tornar al shell Bash.

NOTA: L'algoritme d'encriptació triat per guardar les contrasenyes a la taula de la base de dades depèn del tipus d'autenticació HTTP que es vulgui que empri el servidor Apache ("Basic" o "Digest"). En concret, en el cas de fer servir el tipus "Basic" (que és el que farem), les contrasenyes es guardaran sense xifrar. Per més informació, consulteu https://httpd.apache.org/docs/2.4/misc/password_encryptions.html
NOTA: Per conéixer la funcionalitat bàsica d'un servidor SQLite3, consulteu https://www.sqlitetutorial.net

*) Configurar el servidor Apache per a què faci ús de la base de dades "mibd" (veure "NOTA IMPORTANT de sota). Concretament, farem que l'arxiu "/etc/apache2/sites-available/lilili.conf" (a Ubuntu) o "/etc/httpd/conf.d/lilili.conf" (a Fedora) treballat als exercicis anteriors tingui ara el següent contingut (on les directives en negreta són les novetats) i reiniciarem el servei:

```
<VirtualHost *:80>
    ServerName www.hola.org
    DocumentRoot "/var/www/lilili"
    DBDriver sqlite3              #Altres opcions poden ser "mysql" o "pgsql"
    DBDParams "/var/midb"         #Els valors possibles per altres SGBDs s'indiquen a una nota
    <Directory "/var/www/lilili/lololo">
        Options +Indexes
```

```
            AuthType Basic
            AuthName "Missatge a mostrar"
            AuthBasicProvider dbd  #L'AuthBasicProvider per defecte és "file"
            AuthDBDUserPWQuery "SELECT contra FROM usuaris WHERE nom = %s"
            Require valid-user
        </Directory>
</VirtualHost>
```

NOTA IMPORTANT: Per a què la configuració anterior funcioni, prèviament s'haurà hagut d'instal·lar el paquet "apr-util- sqlite" (a Fedora) o "libaprutil1-dbd-sqlite3" (a Ubuntu), el qual conté el "driver" binari efectiu invocat a la configuració en qüestió. En el cas de fer servir un altre driver, haurà calgut instal·lar el paquet pertinent ("apr-util-mysql"/"libaprutil1-dbd-mysql", "apr-util-pgsql"/"libaprutil1-dbd-pgsql", etc)

NOTA: En el cas de voler connectar amb un servidor MySQL, el valor de la directiva *DBDParams* hauria de ser "*host=x.x.x.x,port=n,dbname=midb,user=nom,pass=1234*". En el cas de voler connectar amb un servidor PostgreSQL, el valor de la directiva *DBDParams* hauria de ser "*host=x.x.x.x port=n dbname=midb user=nom password=1234"*. En tot, per més informació consulteu https://httpd.apache.org/docs/2.4/mod/mod_dbd.html *i* https://httpd.apache.org/docs/2.4/mod/mod_authn_dbd.html

NOTA: Existeixen altres orígens on el servidor Apache pot anar a buscar els usuaris i contrasenyes a validar. Per exemple, fent servir el mòdul https://www.adelton.com/apache/mod_authnz_pam (disponible en forma de paquet anomenat "mod_authnz_pam" -a Fedora- o "libapache2-mod-authnz-pam" -a Ubuntu-) es poden utilitzar els usuaris i contrasenyes del propi sistema (emmagatzemats als arxius "/etc/passwd" i "/etc/shadow") o bé qualsevol altre mecanisme que ofereixi el subsistema PAM d'autenticació de Linux (tal com es mostra, per exemple, en aquest tutorial: https://www.server-world.info/en/note?os=CentOS_Stream_8&p=httpd&f=9) . D'altra banda, fent servir el mòdul https://github.com/phokz/mod-auth-external (disponible en forma de paquets anomenats "mod_authnz_external" i "pwauth" -a Fedora- o "libapache2-mod-authnz-external" i "pwauth" -a Ubuntu-) es pot utilitzar qualsevol aplicació externa de tercers (un shell script Bash, un script Php, la pròpia utilitat *pwauth*,...) per autenticar l'usuari i contrasenya proporcionat.