El servidor HTTP Apache

Un servidor "web" és una aplicació que normalment roman escoltant en segon pla per tal d'oferir en el moment que els clients -normalment remots- ho demanin no només pàgines web (d'aquí el nom) sinó molts altres recursos de diferents tipus que sol tenir emmagatzemats al seu disc dur (com ara imatges, vídeos, fitxers de text, etc, etc). Els navegadors són els clients "web" més habituals. Tant els servidors web (per rebre i interpretar correctament la petició) com els navegadors (per enviar la petició i interpretar correctament la resposta) han de poder establir comunicació entre ells mitjançant un determinat protocol que ambdós entenguin. Aquest protocol és el HTTP (https://http2.github.io)

El servidor "web" Apache (http://httpd.apache.org) és el més utilitzat en el món, tal com es pot veure a les "Web Server Survey" mensuals disponibles a http://www.netcraft.com (on es pot anar seguint la popularitat dels diferents servidors més importants). A més a més, és lliure, multiplataforma i gratuït. Aquest document explicarà les característiques, configuracions i usos més habituals d'aquest programa, però si és necessària qualsevol ampliació d'informació, tot el que necessitareu serà consultar la excel.lent documentació oficial, disponible a http://httpd.apache.org/docs (i més concretament, la corresponent a la versió de l'Apache amb la què treballarem: http://httpd.apache.org/docs/current).

Instal.lació de l'Apache i posada en marxa

Ubuntu 19.10 : Un cop està tot el sistema actualitzat (sudo apt update && sudo apt upgrade) procedirem a instal.lar l'Apache simplement executant: sudo apt install apache2

> NOTA:Si anéssim a l'apartat de descàrregues de la web de l'Apache veuríem que allà hi ha una versió més moderna que la que es s'instal.la des dels repositoris de l'Ubuntu, però només està disponible en forma de codi font (és a dir, l'hauríem de compilar per tenir l'executable) i el procés, tot i que no és complicat, no ens val la pena ja que els canvis entre la versió més moderna i la que ofereix Ubuntu són mínims

Un cop instal.lat, el servidor es posarà en marxa automàticament (escoltant al port 80) sense haver de fer res especial. Per comprovar-ho un mateix, es pot obrir un navegador qualsevol i accedir a la direcció http://127.0.0.1 (o bé http://nom , on "nom" és el nom associat a la ip 127.0.0.1 dins del nostre arxiu "/etc/hosts" -normalment aquest nom ja hi és i és "localhost"-) : si tot ha anat bé s'hauria de veure una pàgina web explicativa del programa. Si hom volgués conectar-se al servidor Apache d'un company, llavors al navegador hauria d'escriure http://IP.lan.del.company (o bé http://nom, on "nom" és el nom associat a la IP del servidor del company dins del nostre arxiu "/etc/hosts").

Per iniciar, aturar o reiniciar manualment el servei s'ha de procedir com qualsevol altre servei Systemd del sistema, respectivament: *sudo systemctl start apache2 ; sudo systemctl stop apache2 ; sudo systemctl restart apache2* També està la comanda *systemctl status apache2* per veure si el servei està encés o no. Per fer que s'inicïi sempre automàticament en arrencar el sistema, s'ha d'executar *sudo systemctl enable apache2* (i per deshabilitar-lo permanentment -fins que no es torni a habilitar-, *sudo systemctl disable apache2).*

Fedora 30 : Un cop està tot el sistema actualitzat *(sudo dnf upgrade)* procedirem a instal.lar l'Apache simplement executant: *sudo dnf install httpd* Un cop instal.lat, caldrà posar en marxa el servidor (amb *sudo systemctl start httpd)* i/o habilitar-lo (amb *sudo systemctl enable httpd).* La resta d'accions Systemd també estan disponibles, òbviament: "stop", "restart", "disable", "status", etc. Un cop posat en marxa, es pot comprovar que, efectivament, el servidor està escoltant al port 80 de la mateixa manera que s'ha descrit a l'apartat anterior: obrint un navegador qualsevol i escrivint a la barra de direccions la URL http://127.0.0.1 (per tal d'obtenir la pàgina web de mostra del programa).

> NOTA: Una altra manera de comprovar que el servidor web està escoltant és veure si el port nº80 de la màquina on aquest servidor hauria d'estar funcionant està, efectivament, obert. Això es pot fer observant la sortida de la comanda *ss -tln*

> NOTA: Si no es pot accedir a la pàgina de mostra des del navegador, comproveu que no sigui el tallafocs del Fedora qui ho estigui impedint. Per veure-ho ràpid, podeu aturar-lo (*sudo systemctl stop firewalld*) i tornar-ho a intentar

Estructura global de carpetes i fitxers de configuració

Ubuntu 19.10 : Dins de la carpeta "/etc/apache2" es troben la majoria de subcarpetes i fitxers de configuració de l'Apache (que no deixen de ser simples fitxers de text pla amb un determinat contingut). Concretament, els fitxers més importants són:

*"**apache2.conf"** : Fitxer de configuració principal del programa. Quasi tota la configuració general es pot establir dins d'aquest fitxer, encara que es recomana (per simplicitat) utilitzar fitxers separats específics (ubicats a les subcarpetes "mods-enabled/", "conf-enabled/" i "sites-enabled/"), els quals de seguida estudiarem. Per tant, quasi bé mai caldrà canviar res d'aquest fitxer en concret.

*"**ports.conf**" : Fitxer (referenciat des de "apache2.conf") on s'especifica el/s port/s per on escoltara/n els diferents "virtual hosts" que haguem definit (no hi ha cap problema en què tots ells escoltin per un únic port i una mateixa IP). Un "virtual host" representa un lloc web que funciona (i està configurat) de forma totalment independent respecte a un altre lloc web -és a dir, un altre "virtual host"-, però que està definit juntament amb la resta de "virtual hosts" dins un únic servidor Apache (és a dir, l'únic servidor "real", d'aquí el nom). Bàsicament, el que ens permeten els "virtual hosts" és tenir un lloc anomenat, per exemple, "www.hola.com" i un altre anomenat "www.adeu.com" gestionats pel mateix servidor Apache. Aquest fitxer "ports.conf" és important tenir-lo en compte sobre tot a l'hora de configurar servidors HTTPS (és a dir, servidors HTTP combinats amb la capa de seguretat i encriptació TLS), tal com estudiarem més endavant.

*"**envvars**" : Fitxer on es defineixen les variables d'entorn que es necessiten tenir establertes per a què l'Apache pugui funciona correctament (ja que el seu valor s'utilitza al llarg de la resta de fitxers de configuració). Per exemple, APACHE_RUN_USER i APACHE_RUN_GROUP (que serveixen per establir l'usuari i grup sota els que s'executarà el servidor -i per tant, els que definiran els permisos que té el programa-; ambdues variables valen "www-data", un usuari "virtual" específic que es crea en instal.lar l'Apache) o APACHE_PID_FILE (que serveix per establir la ruta del fitxer temporal que conté el PID del procés Apache en aquest moment; val "/var/run/apache2/apache2.pid") o APACHE_LOG_DIR (que serveix per establir la ruta de la carpeta on s'emmagatzemaran els diferents arxius de registres -els "logs"- del servidor; val "/var/log/apache2"), entre altres. Nosaltres no necessitarem modificar aquest fitxer mai.

I les subcarpetes més importants són:

*"**sites-available/**" : Subcarpeta on s'ubiquen els fitxers de configuració corresponents a cada "virtual host" definit. Per defecte Ubuntu ofereix dos "virtual hosts" ja llestos per fer servir: un de tipus HTTP (arxiu "000-default.conf") i un altre de tipus HTTPS (arxiu "default-ssl.conf").

> NOTA: La raó de què el nom del fitxer .conf del "virtual host" HTTP comenci amb "000" és perquè la lectura dels diferents fitxers .conf ubicats en aquesta subcarpeta es realitza en ordre alfabètic: d'aquesta manera, "000-default.conf" sempre serà el primer en ser llegit.

*"**sites-enabled/**" : Subcarpeta que conté enllaços directes apuntant a fitxers ubicats dins de la subcarpeta "sites-available/". La existència d'un enllaç concret implica l'activació del "virtual host" associat (o dit d'una altra manera, per desactivar un "virtual host" és suficient amb esborrar l'enllaç d'aquesta carpeta i per tornar-lo a activar només cal refer l'enllaç amb una simple comanda *ln -s*). Per defecte només el "virtual host" HTTP està activat.

*"**mods-available/**" : Subcarpeta on s'ubiquen els fitxers corresponents a la gestió dels diferents mòduls ("plugins") que el servidor Apache té disponibles per utilitzar. Els mòduls de l'Apache són "trossos" de programa que, en general, augmenten la funcionalitat del programa base (encara que entre sí són força diferents en termes d'objectius i funcionament). Dins d'aquesta carpeta cada mòdul sol tenir associat un arxiu amb extensió ".load" (imprescindible) i un altre amb extensió ".conf" (opcional), ambdós amb el mateix nom (el del propi mòdul). Els arxius .load contenen sols una línia de text com *LoadModule nomModul_module /usr/lib/apache2/modules/nomFitxer.so*, la qual s'encarrega de carregar efectivament el mòdul real (que és el fitxer binari amb extensió ".so"

-concretament, es tracta d'una llibreria dinàmica-) a la memòria RAM de la màquina. Els arxius .conf contenen la configuració del mòdul pertinent, la qual pot ser molt diferent de mòdul a mòdul.

*"**mods-enabled/**" : Subcarpeta que conté enllaços directes apuntant a fitxers ubicats dins de la subcarpeta "mods-available/". La existència d'un enllaç concret implica l'activació del mòdul associat i, per tant, la possibilitat d'utilitzar la funcionalitat que aquest mòdul aporta (o dit d'una altra manera, per evitar que es pugui fer servir aquesta funcionalitat extra -és a dir, per "desactivar" el mòdul- és suficient amb esborrar l'enllaç d'aquesta carpeta, i per tornar-la a activar només cal refer l'enllaç amb una simple comanda *ln -s*).

*"**conf-available/**" : Subcarpeta on s'ubiquen fitxers amb trossos de configuració extra (generalment relatius a aspectes molt específics) que són interpretats com si fossin part de l'"apache2.conf". La gràcia està en què si la configuració es troba en algun tros dins d'aquesta carpeta no cal tocar el fitxer de configuració principal.

*"**conf-enabled/**" : Subcarpeta que conté enllaços directes apuntant a fitxers ubicats dins de la subcarpeta "conf-available/". La existència d'un enllaç concret implica l'activació de la configuració extra associada (o dit d'una altra manera, per no tenir en compte aquest tros de configuració -és a dir, per "desactivar-lo"- és suficient amb esborrar l'enllaç d'aquesta carpeta, i per tornar-lo a tenir en compte només cal refer l'enllaç amb una simple comanda *ln -s*).

Cal tenir present que qualsevol canvi que es guardi en algun dels fitxers i subcarpetes anteriors no serà efectiu fins reiniciar el servidor (o bé, de forma alternativa, executar la comanda *systemclt reload apache2*, la qual obliga al servidor a llegir de nou tots els arxius de configuració sense que calgui reiniciar).

En comptes d'haver d'utilitzar manualment la comanda *ln -s* cada cop que volguem activar o desactivar mòduls (i/o "virtual hosts" i/o configuracions extra) podem aconseguir el mateix (és a dir, crear o destruir els enllaços directes pertinents) fent servir el següent conjunt de comandes (instal.lades automàticament amb el servidor) molt senzilles:

a2enmod nomModul : activa el mòdul indicat
a2dismod nomModul : desactiva el mòdul indicat
a2ensite nomarxiuVH : activa el "virtual host" indicat
a2dissite nomarxiuVH : desactiva el "virtual host indicat
a2enconf nomarxiuConf : activa el tros de configuració indicada
a2disconf nomarxiuConf : desactiva el tros de configuració indicada

Després d'executar qualsevol d'aquestes comandes, caldrà reiniciar (o "recarregar") el servidor per a què els canvis es tinguin en compte. També és interessant la comanda a2query , la qual amb el paràmetre -m mostra la llista de mòduls activats en aquest moment, amb el paràmetre -s mostra la llista de "virtual hosts" activats en aquest moment i amb el paràmetre -c mostra la llista de trossos de configuració extra activats en aquest moment. A Fedora la comanda que ens permet saber els mòduls actualment carregats és httpd -M

Fedora 30 : A sistemes Fedora/CentOS l'estructura de fitxers i carpetes de configuració és diferent. L'arxiu de configuració principal passa a anomenar-se "**httpd.conf**" i es troba dins de la carpeta "/etc/httpd/conf". Aquest arxiu inclou el que seria l'"apache2.conf", el "ports.conf" i el "000-default.conf"; si es volguessin afegir més "virtual hosts", llavors caldria fer-ho en un fitxer ".conf" separat, ubicat dins de la carpeta "**/etc/httpd/conf.d**". El contingut d'aquesta carpeta, de fet, és equivalent al de la carpeta "/etc/apache2/sites-enabled" però no només: dins de "/etc/httpd/conf.d" també s'ubiquen els fitxers equivalents als ".conf" que hi havia a "/etc/apache2/mods-enabled" i també els equivalents als que hi havia a "/etc/apache2/conf-enabled". Finalment, el contingut de la carpeta "/etc/httpd/conf.modules.d" és equivalent al dels fitxers ".load" de dins de la carpeta "/etc/apache2/mods-enabled" (essent la carpeta "/usr/lib64/httpd/modules" la ubicació on es troben els mòduls pròpiament dits).

Directives de configuració generals

Al fitxer "apache2.conf" (a Ubuntu) hi ha unes quantes directives globals interessants que poden ser modificades si cal. Unes quantes són de tipus general...:

ServerRoot /etc/apache2 : Aquesta línia no caldrà canviar-la quasi mai. Indica la ruta de la carpeta sota la qual es troben els arxius de configuració, registre i error

PidFile, User, Group : Aquestes línies apliquen diversos valors especificats a l'arxiu "envvars". La més important per nosaltres serà la directiva *User*, la qual, recordem, fa que, a pesar de què el servidor arrenqui sempre com a root (perquè necessita vincular-se al port nº80, port que, com tots els ports menors de 1024, és privilegiat i només el pot obrir l'usuari root) tots els processos fill del servidor adquireixin els permisos de l'usuari allà indicat. Aquest usuari, per tant, ha de poder llegir els directoris que volguem publicar i ha de ser el propietari del directori de logs.

AccessFileName .htaccess : Indica el nom predeterminat (en aquest cas, ".htaccess", fixeu-vos en el punt inicial -és un axiu ocult, per tant-) de tots els possibles "arxius d'accés". Sobre els arxius d'accés en parlarem a l'apartat d'aquest document que explica les seccions <Directory>.

...unes altres estan relacionades amb la gestió dels "logs" (explicades més endavant, en l'apartat sobre "virtual hosts") ...i també tenim les directives *Include* adequades per tal d'afegir dins de l'arxiu "apache2.conf", tal com ja hem comentat abans, les configuracions que hi ha repartides per la resta de fitxers (el "ports.conf" i tots els ".conf" ubicats dins de les carpetes "mods-enabled", "sites-enabled" i "conf-enabled").

D'altra banda, el fitxer "ports.conf" conté la directiva:

Listen [direccioIP:]port : Estableix la tarja de xarxa (amb la IP indicada) i port pel qual s'escoltaran peticions. Es poden escriure vàries línies *Listen*, però llavors caldrà tenir en compte la configuració dels diferents "virtual hosts" (veure més avall). Fixeu-vos que també poden aparèixer línies *Listen* dins de condicionals que comproven si un determinat mòdul està activat o no: això permet obrir un port només si el mòdul adequat (normalment relacionat amb les conexions segures TLS) està activat.

Als arxius que hi ha dins de la subcarpeta "conf-available" també podem trobar algunes directives interessants, com ara:

AddDefaultCharset UTF-8 : Estableix la codificació predefinida de les pàgines web a servir a UTF-8 (una evolució ampliada i millorada del vell estàndar ASCII) sobrescrivint llavors la que estigui especificada dins d'elles (amb l'etiqueta *<meta charset...>*, per exemple). Si aquesta línia està comentada, llavors prevaldrà la codificació indicada dins de la pròpia pàgina web.

ErrorDocument nºerror {"text error" | /etc/apache2/fitxer | http://ruta/remota/resultat/dun/script } : Estableix el missatge de text que s'enviarà al client si el servidor detecta un error HTTP indicat. En comptes del missatge de text també es pot establir la ruta d'un fitxer (llavors s'enviaria al client el seu contingut) o la URL d'un script -PHP per exemple- (llavors s'enviaria al client el resultat d'haver executat aquest script). Per exemple, si el nº d'error fos el 404, el document a retornar hauria de ser la pàgina mostrant "pàgina no trobada".

Ordre de preferència de les directives

Algunes de les directives anteriors (i moltes més que veurem més endavant) poden aparèixer escrites en diferents arxius de configuració a la vegada. Segons en quin arxiu hi siguin, tindran una afectació més o menys global. Concretament:

*Les directives escrites dins l'arxiu "apache2.conf" (o, indistintament dins d'algun arxiu ".conf" ubicat dins de les carpetes "conf-available" i/o "mods-available") afecten a tots els "virtual hosts" sense distinció (els mòduls es carreguen globalment).

*Les directives escrites dins d'algun arxiu ".conf" ubicat dins de la carpeta "sites-available" només afecten al "virtual host" en qüestió

*Les directives escrites dins de les seccions <Directory> o <File> (o similars) presents en qualsevol dels arxius anteriors (i que de seguida estudiarem) només afecten a aquella carpeta o fitxer en qüestió concret respectivament

Si es donés el cas que la mateixa directiva es troba en diferents llocs amb valors diferents, sempre guanya la més concreta; és a dir, que si trobem una directiva a l'arxiu "apache2.conf" i després dins una secció <Directory>, s'aplicarà aquesta última. Cal tenir present aquest fet, per exemple, a l'hora d'establir directives de registre (*ErrorLog*, *CustomLog*, ...) , entre altres casos.

NOTA: En el cas de Fedora, si apareix la mateixa directiva a l'arxiu "httpd.conf" i en un altre arxiu ubicat dins de la carpeta "/etc/httpd/conf.d", s'aplicarà sempre aquest darrer. En el cas de què apareixi la mateixa directiva en arxius ubicats tots dins de la carpeta "/etc/httpd/conf.d", s'aplicarà sempre la directiva que estigui escrita al darrer arxiu, ordenant-se aquests alfabèticament a partir del seu nom.

Per trobar d'un cop totes les línies que continguin una directiva concreta al llarg dels diferents arxius on poden aparèixer en Ubuntu es pot executar la comanda: *grep -ri "nomDirectiva" /etc/apache2/**

Configuració de "Virtual hosts"

La configuració d'un "virtual host" ha d'estar envoltada de l'etiqueta *<VirtualHost ip:port>* al principi i una etiqueta *</VirtualHost>* al final. Aquí "ip:port" indica, respectivament, la direcció IP concreta per on estarà accessible aquest "virtual host" (o bé el símbol "*" si volem que ho estigui a través de qualsevol IP que tingui el nostre servidor) i el port per on estarà escoltant (el qual sol ser el nº80 o bé el nº443 si és un servidor segur però, en tot cas, ha d'estar prèviament definit a l'arxiu "ports.conf").

Aquesta configuració la escriurem dins d'un arxiu (anomenat com volguem però amb extensió .conf) dins de la carpeta "/etc/apache2/sites-available" (a Ubuntu) o "/etc/httpd/conf.d" (a Fedora) i, en el cas d'Ubuntu, activarem a més el "virtual host" en qüestió amb la comanda *a2ensite* ; en qualsevol cas, finalment, caldrà reiniciar el servei Apache. Entre les etiquetes *<VirtualHost ip:port>...</VirtualHost>* les directives imprescindibles que cal escriure per definir un "virtual host" són:

ServerName nom : Indica el nom DNS del "virtual host". Aquesta dada és la que distingueix un "virtual host" d'un altre que hi estigui definit dins del mateix servidor Apache.

NOTA: El valor d'aquesta directiva es compararà, cada cop que es rebi una petició d'un client, amb el valor d'una capçalera que incorpora aquesta anomenada "Host", la qual precisament serveix per saber a quin "virtual host" va dirigida la petició en qüestió

NOTA: Si no disposem d'un servidor DNS a la nostra infrastructura de xarxa (o bé no tenim contractat cap servidor DNS extern que apunti a la IP de la màquina on s'està executant l'Apache) no podrem tenir accessible el nostre servidor Apache a Internet. No obstant, sí que el podrem fer servir dins de la nostra LAN: per això hauríem, per exemple, de configurar convenientment l'arxiu "/etc/hosts" a cadascun dels ordinadors (clients i servidors) presents a la nostra xarxa de forma que reconeguin mútuament els seus respectius noms.

ServerAlias unNomDiferent unAltre...: Indica un nom o noms alternatiu/s a l'indicat a *ServerName* . D'aquesta manera, els clients podran accedir al "virtual host" fent servir diferents noms. És molt típic trobar que *ServerName* tingui el valor *www.domini.com* i *ServerAlias* tingui el valor *domini.com*, per exemple. En qualsevol cas, el servidor DNS hauria d'estar configurat per apuntar al servidor Apache també amb aquests noms alternatius. Aquesta directiva no és obligatòria.

DocumentRoot /ruta/carpeta : Indica la ruta de la carpeta arrel des d'on pengen les pàgines web (i la resta de documents) que serveix aquest "virtual host". Al "virtual host" definit per defecte en una instal.lació d'Apache (és a dir, a "000-default.conf" en Ubuntu) aquesta carpeta és "/var/www/html". Això vol dir que la pàgina web que apareix quan accedim a http://127.0.0.1 (o http://ip.dun.altre.servidor) només havent instal.lat l'Apache es troba allà (concretament és la pàgina anomenada "index.html"). El mateix passaria si escrivíssim http://127.0.0.1/index.html (o http://ip.dun.altre.servidor/index.html) ja que "index.html" és el nom de pàgina web a mostrar que Apache pren per defecte si a la petició no s'especifica cap. Si volguéssim publicar qualsevol altre pàgina o fitxer (per exemple, un document anomenat "hola.pdf") només caldria copiar-lo dins d'aquesta carpeta: anant llavors a http://ip.del.servidor.web/hola.pdf ja el tindríem disponible.

NOTA: Dins de la carpeta arrel es poden crear tantes subcarpetes com es vulgui, les quals també s'hauran d'indicar convenientment a la URL demanada; per exemple, si creem una carpeta "/var/www/html/fotos" i allà copiem l'arxiu "platja1.png", per accedir a aquesta foto via navegador hauríem d'escriure a la seva barra de direccions la següent URL: http://ip.del.servidor.web/fotos/platja1.png". Com es pot comprovar, tot el que queda fora de "/var/www/html" és invisible al visitant: per això s'anomena "carpeta arrel".

NOTA:Per a què tot funcioni correctament, els permisos de la carpeta arrel i el de les seves subcarpetes han de permetre l'accés a l'usuari propi de l'Apache ("www-data" a Ubuntu, "apache" a Fedora). És a dir, han de tenir el permís d'execució -"x"- pel grup "d'altres" (ja que per defecte el propietari i el grup d'aquestes carpetes és l'usuari "root", tant a Ubuntu com a Fedora). D'altra banda, els permisos dels fitxers ubicats dins de la carpeta arrel i dins de les seves subcarpetes el que han de permetre és la lectura a l'usuari propi de l'Apache. És a dir, han de tenir el permís de lectura -"r"- pel grup d'altres. Aquestes configuracions ja estan ben establertes per defecte, però, en tot cas, recomano repassar les comandes **chmod** i **chown** per si fos necessari el seu ús en algun altre moment

Directives de registres

Altres directives molt comunes definides per cada "virtual host" són les relacionades amb els missatges de registre d'erros...:

ErrorLog /ruta/error.log : Indica la ruta de l'arxiu que guardarà els missatges dels errors que puguin ocórrer. La configuració per defecte fa servir el fitxer "/var/log/apache2/error.log" (a Ubuntu) o "/var/log/httpd/error.log" (a Fedora).

NOTA: Per evitar que l'arxiu error.log vagi creixent de forma incontrolada a mesura que es van emmagatzemant més i més missatges de registre, hi ha la possibilitat d'indicar a *ErrorLog* que es vol utilitzar un programa especial de rotació de logs. La idea és anar omplint el fitxer fins un límit (donat per temps o per tamany de fitxer), moment en el qual aquest fitxer es guarda comprimit i s'obre un nou fitxer on es comença un altre cop a emmagatzemar novesdades. Per evitar acabar amb un munt de fitxers comprimits, s'estableix un màxim d'aquests -per exemple, 5-, de forma que en arribar al sisè, se sobrescriurà el fitxer comprimit més antic, i així succesivament; per això es diu "rotació de logs". Encara que es poden fer servir altres (com LogRotate: https://github.com/logrotate/logrotate), l'Apache ja incorpora un programa propi de rotació de logs anomenat *rotatelogs* (http://httpd.apache.org/docs/current/programs/rotatelogs.html) el qual es pot invocar així :

ErrorLog "|bin/rotatelogs -n 5 /ruta/error.log 86400" on -n 5 indica que hi haurà 5 fitxers (l'actual i 4 comprimits) i 86400 indica l'interval de temps (en segons) cada quan es farà la compressió del fitxer actual -sobrescrivint el comprimit més antic en aquell moment- i l'obertura d'un nou fitxer log blanc.

ErrorLog "|bin/rotatelogs -n 5 /ruta/error.log 10M" on 10M indica el tamany que haurà d'aconseguir el fitxer de log actual per ser comprimit i obrir-ne un de nou. Es poden escriure els sufixes B -Bytes-, K -Kilobytes-, M -Megabytes- i G -Gigabytes-.

NOTA: A la directiva ErrorLog també es podria indicar, en comptes de la ruta d'un fitxer, la paraula "journald" (d'aquesta manera, es delega al sistema de logs de Systemd la recopilació i gestió dels missatges d'error) però això només és possible si s'usa el mòdul "mod_journald" , disponible només a partir de la versió 2.5 d'Apache (http://httpd.apache.org/docs/trunk/mod/mod_journald.html)

LogLevel nivell : Indica el nivell mínim d'importància que els errors han de tenir per ser guardats a l'arxiu especificat a *ErrorLog*. Els nivells són (de menys a més importants): "debug", "info", "notice", "warn", "error", "crit", "alert" i "emerg". També es pot especificar un nivell mínim només per errors relacionats amb un determinat mòdul (o més) diferents del nivell mínim per defecte; això es fa indicant el nom del mòdul seguit de dos punts i el nivell, així per exemple: *LogLevel notice rewrite:info* (on s'estableix el nivell "notice" per tots els errors excepte els relacionats amb el mòdul "rewrite" -que ja estudiarem-, els quals han de tenir el nivell "info").

ErrorLogFormat "format" : Aquesta línia defineix quina informació es vol guardar a l'arxiu indicat a *ErrorLog* seguint el format especificat. Els indicadors de format possibles estan documentat a http://httpd.apache.org/docs/current/mod/core.html#errorlogformat . No obstant, aquesta directiva no se sol utilitzar gaire (de fet, per defecte no apareix escrita) perquè l'Apache ja incorpora un format predefinit pels missatges d'error que sol ser adequat a la majoria d'ocasions; concretament, consta dels següents camps delimitats per claudàtors: la data i hora del missatge, el nom del mòdul que produeix el missatge i el seu nivell d'importància (veure *LogLevel*), el PID del procés que ha experimentat l'error, la direcció del client que ha fet la petició i una frase detallada explicant l'error.

...i també amb els de registre d'events en general:

LogFormat "format" nomFormat : Defineix un format i li assigna un nom per poder-lo referenciar més endavant en una línia *CustomLog* (que és la que realment s'encarrega de guardar la informació en un arxiu de registre personalitzat, amb el format referenciat). Aquest format està documentat a http://httpd.apache.org/docs/current/mod/mod_log_config.html#formats (i podem trobar exemples a http://httpd.apache.org/docs/current/logs.html#accesslog) però tot seguit mostrem els valors típics:

%a : direcció IP del client
%h : nom del client
%H : el protocol de la petició (HTTP, HTTPS,...)
%m : el mètode de la petició (GET, POST,...)
%U : la url sencera demanada, sense querystring
%q : la querystring (el que va després del ?)
%r : primera línia de la petició (és a dir, la línia *<METODE><URI><VERSIO>*)
%u : usuari remot (si es requereix protecció per contrasenya; veure més endavant)
%{NomCapçaleraClient}i : valor de la capçalera de client (petició) indicada
%s : estat de la petició
%B : tamany del cos de la resposta. També pot servir %b
%{NomCapçaleraServidor}o : valor de la capçalera de servidor (resposta) indicada
%t : l´hora en la què es va rebre la petició (en format anglès)
%{format}t: l´hora en la què es va rebre la petició (en format strftime: http://strftime.org)
%D : temps trigat en servir la petició, en microsegons
%T : temps trigat en servir la petició, en segons

Un exemple podria ser: *LogFormat "%h %u %t |"%r|" %>s %b %{User-agent}i" unformat* . Poden existir múltiples línies *LogFormat*

NOTA: Els modificadors < i > s'usen en peticions que poden ser diferents entre l'original i la final; per exemple, %>s es pot usar per guardar l'estat final de la petició

CustomLog /ruta/fitxer "format" ó *CustomLog /ruta/fitxer nomFormat* : Registra informació escollida amb un determinat format (definit directament fent servir les mateixes opcions de la llista anterior -%r, %s, ...- o bé indicant el seu nom si ja ha sigut prèviament definit i batejat en alguna líniea *LogFormat* anterior). Pot haver-hi múltiples línies *CustomLog* . Per defecte l'Apache té definit una línia *CustomLog* que genera un arxiu anomenat "/var/log/apache2/access.log" (a Ubuntu) o "/var/log/httpd/access_log (a Fedora) amb la informació més típica que podem esperar d'un arxiu de registre de peticions i respostes HTTP; aquest fitxer serà el que més sovint consultem quan volguem saber l'activitat del nostre servidor quina és.

NOTA: A la directiva CustomLog també es podria indicar, en comptes de la ruta d'un fitxer:
a) La invocació a un programa extern de gestió de logs, com ara *rotatelogs*, així: "*|bin/rotatelogs -n nº /ruta/fitxer xxxx"*.
b) La paraula "journald"; d'aquesta manera es delegaria al sistema de logs de Systemd la recopilació i gestió dels missatges. Això, però, només és possible si s'usa el mòdul "mod_journald" , disponible només a partir de la versió 2.5 d'Apache (http://httpd.apache.org/docs/trunk/mod/mod_journald.html)

NOTA: Existeixen múltiples programes que permeten visualitzar d'una forma gràfica i entenedora les dades emmagatzemades en els fitxers de registre que se li indiquin (com ara l'"access.log"). Alguns exemples són: Awstats (https://github.com/eldy/awstats), Analog (https://github.com/c-amie/analog-ce) o GoAccess (https://github.com/allinurl/goaccess) , entre altres.

NOTA: Una manera ràpida de veure en temps real els nous missatges que es van emmagatzemant a un determinat arxiu log (això també val per l'arxiu error.log) és executar en un terminal la comanda *tail -f /ruta/fitxer*

Exemple

A continuació es presenta un exemple d'arxiu de configuració d'un servidor Apache (amb una única IP i port) on es defineixen dos "virtual hosts", cadascun d'ells identificat per un nom DNS ("www.hola.com" i "www.hola.org"), noms convenientment establerts prèviament per a què apuntin, en ambdós casos, a la mateixa màquina (IP):

```
# Això és un comentari: cal assegurar-se que existeixi una línia Listen 80 (a "ports.conf", normalment)
<VirtualHost *:80>
        ServerName www.hola.com
        DocumentRoot "/var/www/html/web1"
        #Altres directives, com ara ErrorLog, CustomLog o seccions <Directory>, <Files>, etc
</VirtualHost>
<VirtualHost *:80>
        ServerName www.hola.org
        DocumentRoot "/var/www/html/web2"
       #Altres directives, com ara ErrorLog, CustomLog o seccions <Directory>, <Files>, etc
</VirtualHost>
```

Configuració d'accés a directoris

Dins de la configuració d'un "virtual host" (¡o dins de l'arxiu "apache2.conf" i/o els inclosos dins de la carpeta "conf-available"...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 especificada (mitjançant la seva ruta absoluta dins del sistema de fitxers) i automàticament a totes les seves subcarpetes.

La ruta indicada pot contenir els comodins "" (equivalent a qualsevol número de caràcters -que no sigui "/"-) i "?" (equivalent a un caràcter qualsevol 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 més interna, i així anar "cap amunt" fins a la carpeta "pare". L'ordre d'aplicació de les directives és molt important perquè sempre -i només- s'aplica l'última de les que hi hagi repetides.

Més versàtil que la directiva <Directory> és, però, la directiva *<DirectoryMatch nomFitxer>...</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. El tema de les expressions regulars és apassionant però se'n va una mica dels objectius d'aquest curs; per saber-ne més, aconsello consultar l'estupenda web http://www.regular-expressions.info o també https://en.wikipedia.org/wiki/Regular_expression#Standards . Un resum 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 equivalent a fer servir 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> que cal explicar prèviament 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"). 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", 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 poden modificar els arxius de configuració generals ("apache2.conf", els dels "virtual hosts",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/llarga/pagina.html , l'Apache llegirà -per ordre- els eventuals fitxers d'accés presents a "/var/www/html/una", "/var/www/html/una/ruta"i "/var/www/html/una/ruta/llarga", i això amb totes les peticions. Un altre tema a més és l'inconvenient de què les directives que hi hagi escrites puguin entrar en contradicció amb el que mana la configuració general...

A partir d'aquí, 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 d'altres) 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, Redirect* i família (entre d'altres) a l'eventual arxiu d'accés existent dins de la carpeta associada 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).

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 : Es poden seguir els accesos directes ("enllaços simbòlics)

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 tenen cadascuna 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 també *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*

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. A continuació es proposen 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 les 9:00h i les 17:00h

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"

Totes les directives *Require xxx* anteriors admeten la possibilitat de ser negades mitjançant la paraula "not" així *Require xxx not 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 ip not 192.168.1.34* .

Exemples

A l'arxiu "apache2.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" és la que correspon a la carpeta "/var/www", contenidora dels documents del "virtual host" definit a "000-default.conf". 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" (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> adequada.

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 important que pot escriure's dins d'una secció <Files> o <FilesMatch> és la *Require xxx* explicada als paràgrafs anteriors, però aplicada a fitxers en comptes de carpetes. Així, per exemple, trobem que a "apache2.conf" tenim unes línies com...:

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

...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" (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.

Un altre exemple (aquest cas no inclòs per defecte) seria aquest, on es defineix un "virtual host" que no permet l'accés a les carpetes anomenades "privat" que hi hagin al llarg de qualsevol URL (http://www.hola.org/privat, http://www.hola.org/xxx/privat, http://www.hola.org/xxx/yyy/privat , etc (el símbol . indica un caràcter qualsevol i el símbol + indica que aquest caràcter qualsevol es pot repetir una o més vegades, sent en cada repetició un altre caràcter qualsevol):

```
<VirtualHost *:80>
    ServerName www.hola.org
    DocumentRoot "/var/www/html/web"
    <FilesMatch "^.+/privat">
        Require all denied
    </FilesMatch>
</VirtualHost>
```

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>* -i la seva corresponent *<LocationMatch>*-, la qual controla el comportament relacionat amb la part de la URL rebuda del client posterior al nom DNS del servidor, la qual pot no tenir res a veure amb la jerarquia de carpetes reals (veure l'apartat següent que parla dels àlies). És a dir, si l'usuari escriu, per exemple, a la barra del navegador la direcció http://www.exemple.com/webmail/inbox, (i suposant que tenim un "virtual host" vinculat al nom "www.exemple.com") l'Apache mirarà en la carpeta "/webmail/inbox" sota el DocumentRoot definit, el qual podria ser, per exemple, "/var/www/html/servidorCorreu" (essent, per tant, "/var/www/html/servidorCorreu/webmail/inbox" la carpeta real a accedir). Podríem, per tant, definir una secció en aquest "virtual host" tal com <Location /webmail/inbox> per gestionar aquesta petició vinculada a una carpeta determinada. Un ús comú d'aquesta secció és permetre a un script (PHP, per exemple) gestionar peticions fetes a una determinada URL on l'usuari no té perquè saber quina és la carpeta real associada.

Com a síntesi, la següent llista proporciona una orientació sobre l'ordre que l'Apache segueix per gestionar les diferents opcions de configuració. Les opcions de més al final sobrescriuren les més al principi:

1.-Es llegeixen les seccions <Directory> presents a l'arxiu "apache2.conf". Si hi ha vàries seccions cadascuna associades a vàries subcarpetes dins del mateix arbre, la primera secció en llegir-se és la que correspon a la ruta més curta i la darrera és la que correspon a la subcarpeta amb la ruta més llarga.

2.-Es llegeixen els fitxers d'accés (".htaccess), sobrescrivint les seccions <Directory> anteriors si això està permés amb l'opció *AllowOverride*

3.-Es llegeixen les seccions <DirectoryMatch> i <Directory ~> presents a l'arxiu "apache2.conf".

4.-Es llegeixen les seccions <Files> i <FilesMatch> presents a l'arxiu "apache2.conf".

5.-Es llegeixen les seccions <Location> i <LocationMatch> presents a l'arxiu "apache2.conf".

6.-Es tornen a repetir els passos anteriors per totes les directives que hi hagi dins tots els fitxers inclosos un rera l'altre via *Include* dins de l'arxiu "apache2.conf" (és a dir, tots els que hi ha dins de "conf-enabled/", "sites-enabled/", "mods-enabled/"...). És molt important, doncs, que l'ordre d'inclusió d'aquests fitxer sigui el correcte, perquè sempre guanya l'últim, recordem.

Ús de diferents mòduls (per defecte activats)

Les directives següents, definides per un determinat mòdul, es poden deixar escrites dins del seu propi arxiu .conf ubicat dins de la carpeta "mods-available" (i d'aquesta manera, afectarien a tots els "virtual hosts" de forma global) però també es poden escriure, si el mòdul està activat, dins de l'arxiu de configuració d'un "virtual host" en particular (dins de la secció <VirtualHost *:80> pertinent), afectant llavors només a aquest "virtual host".

*Directives definides al mòdul "dir" (dir.conf)

DirectoryIndex index.html ... : Indica el nom del fitxer -normalment de tipus HTML- que es mostrarà al client si aquest no especifica cap nom en particular quan fa la petició. Per exemple, si un usuari escriu la direcció http://www.exemple.com i la carpeta arrel és "/var/www/html", ¿quina és la pàgina ubicada dins d'aquesta carpeta que s'haurà de mostrar? La que marqui *DirectoryIndex* (per defecte, "index.html"). Així doncs, haver escrit http://www.exemple.com/index.html seria equivalent (però més llarg). Igualment, si l'usuari escriu http://www.exemple.com/unasubcarpeta , ¿quina és la pàgina ubicada dins d'aquesta subcarpeta que s'haurà de mostrar? Doncs la que allà s'anomeni "index.html" també. Si la directiva DirectoryIndex indiqués més d'un nom, l'Apache buscarà a la carpeta pertinent els fitxers per ordre,des del primer nom indicat fins l'últim, mostrant el primer que coincideixi i aturant-se allà. Si no en troba cap, no hi haurà fitxer per defecte i com que la configuració de l'Apache per la carpeta "/var/www/html" (DocumentRoot per defecte) és *Options Indexes*, el que fa llavors és mostrar un llistat de tots els fitxers ubicats a la carpeta demanada, per a què el client esculli el fitxer concret que vol.

*Directives definides al mòdul "alias" (alias.conf)

Alias /àlies "/ruta/carpeta/real" : Permet que el client pugui accedir al contingut d'una carpeta (¡la ruta real de la qual estarà fora del DocumentRoot de qualsevol "virtual host"!) simplement escrivint una paraula ("l'àlies") al final de la URL demanada. Per exemple, si tenim un servidor Apache accesible a través de la URL http://www.exemple.com i volem que un usuari pugui accedir a la carpeta real "/home/pepe" (la qual està fora de la carpeta arrel de qualsevol "virtual host") escrivint a la barra de direccions del navegador una URL tal com http://www.exemple.com/manolo , hauríem d'afegir una línia tal com *Alias /manolo "/home/pepe"* (¡hauríem a més de vigilar amb els permisos d'accés i lectura d'aquesta carpeta!).

NOTA: La directiva *Alias* normalment apareix escrita just per sobre d'una secció <Directory> (associada a la carpeta real de la qual són àlies) que conté les directives *Require xxx* pertinents.

NOTA: També existeixen la directiva *AliasMatch*, que permet l'ús d'expressions regulars per indicar l'àlies i la directiva *ScriptAlias*, que permet definir àlies apuntant a carpetes que contenen fitxers executables (normalment scripts de tipus CGI...en parlarem d'això més endavant).

Redirect /ruta/relativa/desde/DocumentRoot/de/carpeta/vella {http://urlnova | /ruta/relativa/nova}: Informa al client que una carpeta determinada ha sigut canviat de lloc, redireccionant-lo al lloc nou. Aquesta directiva fa que l'Apache envïi una resposta amb codi 301 (redirecció temporal); si es vol fer que envïi una resposta 302 (redirecció permanent), es pot escriure la paraula "permanent" entre *Redirect* i la ruta de la carpeta vella. També es pot enviar una resposta amb codi 303 si s'escriu la paraula "seeother" i una resposta amb codi 410 si s'escriu la paraula "gone" (en aquest cas, la URL nova no existeix i per tant, no s'ha d'escriure res al seu lloc) Aquesta directiva es podria escriure, per exemple, dins de la secció <VirtualHost *:80> pertinent per tal de redirigir tot el web ("/") a un domini nou.

NOTA: També existeix la directiva *RedirectMatch* que permet l'ús d'expressions regulars per indicar vàries rutes de carpetes antigues de cop.

Altres mòduls interessants activats per defecte són "setenvif", "env", "deflate", ...

Ús de diferents mòduls (per defecte desactivats)

*Directives definides al mòdul "userdir" (userdir.conf)

Si un usuari dins la seva carpeta personal té una carpeta anomenada "public_html", el seu contingut es podrà veure a través de http://ipservidor/~nomusuari, sempre i quan aquest mòdul estigui activat i existeixi la directiva *UserDir public_html*. Si volguéssim canviar el nom d´aquesta carpeta per a què fos un altre diferent de "public_html" simplement caldria substituir la directiva anterior per *UserDir nouNom*. També es podria escriure, en comptes d'un nom nou, la ruta absoluta (des de l´arrel del sistema) d´una carpeta a partir de la qual els usuaris col.locarien els seus continguts (cadascun en una subcarpeta concreta).

En activar el mòdul automàticament tots els usuaris del sistema poden gaudir de la possibilitat de publicar documents a través de la seva carpeta "public_html". Si es vol que algun usuari del sistema no ho pugui fer, caldrà especificar-ho explícitament amb la directiva *UserDir disable usuari1 usuari2 ...*

La secció <Directory> que és present a l'arxiu .conf del mòdul configura per defecte la ruta /home/*/public_html (on * equival al nom de l'usuari la carpeta de la qual es vulgui accedir en aquell moment per aquella petició en concret) amb una sèrie d'opcions *AllowOverride*, *Options*, *<Limits>*, etc. que en principi no caldrà modificar.

*Directives definides al mòdul "rewrite" (no té rewrite.conf)

Aquestes directives redireccionen al client, conduïnt-lo a pàgines noves des de links antics. 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: també poden servir, per exemple, per convertir URLs molt llargues i complicades en URLs més senzilles per l'usuari o per denegar l'accés a determinades URLs depenent de segons quines circumstàncies (qui és el client o d'on ve, quina és l'hora, etc), etc, etc.

Aquestes directives s'han d'escriure dins de la secció <Directory> (o en l'arxiu ".htaccess" equivalent) associada a la carpeta corresponent al link antic, i obligatòriament sempre després de la línia *RewriteEngine On* (encarregada de posar en marxa el mòdul "rewrite" per aquella carpeta en concret). A partir d'aquí, concretament estem parlant de dues directives fonamentals:

RewriteCond caracteristica valorAComprovar [opcion1,opcio2] : Opcional, serveix per definir el valor que es vol comprovar d'una determinada característica en cada petició rebuda (si n'hi ha més d'una característica a comprovar, es poden escriure múltiples línies *RewriteCond*, una per cadascuna d'aquestes característiques). 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

RewriteRule exprRegEscritaPelClientASubstituir urlRealOnEsVa [opcion1,opcio2] : Serveix per realitzar el redireccionament pròpiament dit, el qual pot efectuar-se sempre o només si la petició concorda amb totes les característices comprovada/es prèviament (o, si s'especifica el l'opció [OR], amb almenys una d'elles)

Tant la directiva *RewriteCond* (a l'hora d'especificar el valor a comprovar) com la directiva *RewriteRule* solen fer un ús extens d'expressions regulars i condicionals, cosa que fa la seva utilització més complicades del que és habitual respecte altres directives d'altres mòduls. Per aprofundir en el coneixement de les expressions regulars (que ja vam conèixer en parlar de les directives *<FilesMatch>* i similars),

recomano consultar http://httpd.apache.org/docs/current/rewrite. També és aconsellable la "xuleta" disponible a http://www.cheatography.com/davechild/cheat-sheets/mod-rewrite o el lloc interactiu de proves https://regex101.com. A continuació presentem, de totes formes, alguns exemples senzills d'ús:

*Exemple que permet escriure "about" al final de la URL en comptes del nom real de l'arxiu sol.licitat ("about.html"). És a dir, permet no haver d'escriure l'extensió (i així fer més còmoda i maca la URL). El símbol ^ indica el començament de la part de la URL que no conté el nom DNS del servidor (és a dir, si la URL completa és http://www.exemple.com/una/url, la part de la URL que processa *RewriteRule* és només "una/url") i el símbol $ indica el final de la URL (sense querystring) Això vol dir que ^about$ representa URLs del tipus http://www.exemple.com/about . El que fa *RewriteRule*, quan detecta una petició amb una URL d'aquest tipus, canviar la paraula "about" per "about.html", de tal manera que la nova URL quedi així: http://www.exemple.com/about.html, que és la direcció real de la pàgina a la què es vol accedir. El símbol [NC] indica que les lletres minúscules i majúscules es consideren iguals.

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

*Exemple per convertir una URL llarga en una URL curta (per exemple, del tipus http://www.exemple.com/results.php?item=camisa&estacio=estiu en una altra del tipus http://www.exemple.com/camisa/estiu) de manera que l'usuari pugui escriure aquesta darrera per anar a la primera. El símbol "|" significa "o" entre tots els elements que hi ha entre parèntesis. El símbol $1 representa l'element seleccionat d'aquest grup (primer -d'aquí el "1"- i únic grup d'elements -els grups s'identifiquen per la presència de parèntesis- que hi ha).

```
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. Entre claudàtors s'indiquen els caracters acceptats per ocupar la posició d'un caracter però seguidament hi ha el símbol "+" que indica que poden haver un número indeterminat de caracters (d'entre els llistats dins dels claudàtors). L'opció [QSA] serveix a més, per mantenir a la URL redireccionada la querystring que l'usuari pugui haver escrit a la URL original.

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

*Exemple per redireccionar qualsevol pàgina inexistent a la pàgina inicial del lloc, anomenada "home.html" (encara que seria més adient utilitzar la directiva *ErrorDocument 404 /ruta/error.html* per això, però és simplement com a demostració). El símbol "!" significa "no", i el símbol "-f" significa és un fitxer" (altres són, per exemple "-d" per dir "és un directori" o "-l" per dir "és un enllaç", entre d'altres). El símbol "." significa "qualsevol caràcter" i el "*" significa que aquest "qualsevol caràcter" pot aparèixer infinits cops (per tant, una expressió com ^(.*)$ en realitat equival a qualsevol URL.En aquest exemple es pot veure que el patró general d'una directiva RewriteCond és *RewriteCond*

```
RewriteEngine on
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRulev ^(.*)$ home.html
```

*Exemple per denegar l'accés a la carpeta des de totes les IPs de clients excepte la 12.34.56.78. L'opció [F] és la que prohibeix l'accés (generant una resposta 403) i l'opció [L] indica que no es continuarà llegint més regles Rewrite (és la darrera, per tant).

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

*Directives definides al mòdul "proxy" (proxy.conf)

Apache pot configurar-se tant com un servidor "proxy directe" o com un servidor "proxy invers".

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 destí; per tal d'obtenir contingut provinent d'aquest servidor destí, el client envia la petició al servidor "proxy directe" que té configurat i és aquest qui reenvia la petició al servidor destí; el servidor "proxy directe" llavors recull al resposta del servidor 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 ser servir els "proxies directes" com a servidors de catxé per millorar l'ample de banda disponible (en aquest sentit, Apache incorpora un altre mòdul específic per gestionar la catxé, anomenat "cache", que no estudiarem.

NOTA: Un programa específicament dissenyat per actuar com a "proxy directe" amb catxé és Squid (http://www.squid-cache.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 activar, a més del mòdul "proxy", els diferents mòduls específics, respectivament: "proxy_http", "proxy_connect", "proxy_ftp" o "proxy_wstunnel".

Per exemple, si volem fer servir Apache com a servidor "proxy directe" d'un altre servidor web (HTTPS), primer haurem d'executar (a Ubuntu) la comanda: *a2enmod proxy proxy_http proxy_connect* i tot seguit, configurar-lo amb la directiva *ProxyRequests*, així:

ProxyRequests {on|off} : Aquesta directiva activa (si el seu valor és "on") la funcionalitat de "proxy directe" de l'Apache. Es pot escriure o bé a la configuració general (és a dir, dins de l'"apache2.conf" o dins de l'arxiu "mods- available/proxy.conf") però també dins de la configuració d'un "virtual host" concret.

Amb la directiva anterior posada a "on" ja tindríem l'Apache funcionant com a "proxy directe"; no obstant, com aquest tipus de proxies 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:

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

NOTA: 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 que es mostra davant del client com qualsevol altre servidor web ordinari (per tant, 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 on reenviarà la petició (un servidor destí concret, anomenat servidor "backend") 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, resultant invisible la infraestructura "backend". A més de protegir així els servidors "backend" de l'accés directe dels clients, 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" (entenent aquests tant com màquines com aplicacions), repartint així la feina a cadascun segons les seves possibilitats i capacitats. Un altre ús és permetre als clients accedir des d'Internet a servidors web que estan darrera d'un tallafocs o també fer de servidors catxé.

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 activar, a més del mòdul "proxy", els diferents mòduls específics, respectivament: "proxy_http", "proxy_connect", "proxy_ftp" , "proxy_wstunnel" o "proxy_fcgi".

Per exemple, si volem fer servir Apache com a servidor "proxy invers" de l'intèrpret PHP en mode FastCGI (l'anomenat "PHP-FPM"), primer haurem d'executar (a Ubuntu) la comanda: *a2enmod proxy proxy_fcgi* i tot seguit, configurar-lo amb la directiva *ProxyPass*, així per exemple:

> *ProxyPass "/" "fcgi://1.2.3.4:9999"* : Aquesta directiva activa la funcionalitat de "proxy invers" de l'Apache especificant com a servidor "backend" un servidor FastCGI (com ara un intèrpret PHP-FPM) amb una determinada IP escoltant a un determinat port. El valor "/" indica la URL (i tots 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 (és a dir, dins de l'"apache2.conf" o bé dins de l'arxiu "mods-available/proxy.conf") però també dins de la configuració d'un "virtual host" concret.

NOTA: Si el servidor "backend" fos d'un altre tipus qualsevol (per exemple, HTTP), caldria activar llavors el mòdul específic pertinent (en aquest as, "proxy_http") i, conseqüentment, el segon valor indicat a la directiva *ProxyPass* tindria una URL del tipus "http://..." Igualment, aquest segon valor pot ser no només una IP sinó també un nom DNS.

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)

NOTA: 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 dues redireccions a diferents carpetes del servidor "backend" (podrien ser la mateixa, de fet) a partir de dues URLs diferents:

```
ProxyPass "/examples" "http://backend.server.com/examples"
ProxyPass "/docs" "http://backend.server.com/docs"
```

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: La directiva *ProxyPass* admet els mateixos paràmetres (opcionals) que admetia la directiva *ProxySet* en el cas dels servidors "proxy directes. S'escriuen al final de tot, per exemple: P*roxyPass "/docs" "http://backend.server.com/docs" connectiontimeout=5 timeout=30*

NOTA: 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: *ProxyPassReverse "/docs" "http://backend.server.com/docs*"). Aquesta directiva permet realitzar canvis dinàmics en les capçaleres de la petició del client i les eventuals redireccions de manera que el client no "noti" que hi ha cap servidor "backend"

NOTA: Una manera alternativa d'activar el "proxy invers" a l'Apache és, en comptes de fer servir la directiva *ProxyPass*, fer servir l'opició [P] de la directiva *RewriteRule* (del mòdul "rewrite").

NOTA: Si es tenen múltiples servidors "backend", cal implementar algun mecanisme de balanceig de càrrega. Per fer això cal activar el mòdul específic "proxy_balancer" i configurar-lo convenientment. Teniu molta informació sobre com implementar diferents tipus de "proxies inversos" a https://httpd.apache.org/docs/2.4/howto/reverse_proxy.html però a continuació es mostra un exemple il.lustratiu:

```
<Proxy balancer://mycluster>
   # Define two back-end servers
   BalancerMember http://1.2.3.4:8080/
   BalancerMember http://1.2.3.4:8081/
</Proxy>
<VirtualHost *:*>
   # Apply VH settings as desired but configure ProxyPass argument to use "mycluster" to balance the load
   ProxyPass / balancer://mycluster
</VirtualHost>
```

NOTA: Existeixen mòduls de tipus proxy més sofisticats (és a dir, especialitzats en tasques més específiques) que no estudiarem. Per obtenir la llista sencera, aneu a http://httpd.apache.org/docs/current/mod.

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 dins de "mods-available", però els valors per defecte ja són prou bons.

"Benchmarking"

Per mesurar el rendiment del servidor (i comparar-lo activant, per exemple, diferents mòduls MPM), podem fer servir eines especialitzades. L'Apache, per exemple, incorpora la comanda ab, que estressa el servidor amb múltiples peticions. Un exemple típic d'una prova realitzada amb aquesta comanda seria:

ab -n 1000 -c 5 http://www.exemple.com

on *-n* és el número de peticions totals i *-c* és el número de peticions concurrents. També està *-t* que diu el número de segons màxim que durarà la prova (per defecte no hi ha límit). Una altra eina similar és "Siege" (https://github.com/JoeDog/siege). Una altra és "Wrk" (https://github.com/wg/wrk)

En qualsevol cas, per conèixer en un moment determinat la llista de procesos que estan consumint més memòria RAM en aquest moment, podem executar per exemple aquesta canonada de comandes:

echo [PID] [MEM] [PATH] && ps aux | awk '{print $2, $4, $11}' | sort -k2rn | head -n 20

Per conèixer quins processos estan monopolitzant més estona la CPU, podem executar:

ps -eo pcpu,pid,user,args | sort -k 1 -r | head -20

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

S'ha de tenir el mòdul "auth_basic" (ó "auth_digest" segons el cas) activat. Existeixen altres mòduls (del tipus "auth_*", "authn_*" i/o "authz_*") que són més sofisticats i que poden fer-se servir pel mateix objectiu , però no els veurem en ser més farragosos de configurar. Els passos a seguir són els següents:

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 (per emmagatzemar la contrasenya encriptada mitjançant l'algoritme SHA en comptes de l'usat per defecte -el crypt(), que és menys segur-), -m (per emmagatzemar la contrasenya encriptada mitjançant l'algoritme 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)

NOTA: Aquesta comanda ve dins del paquet "apache2-utils", el qual en instal.lar 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
AuthUserFile /ruta/arxiu/creat/pas/anterior
AuthName "Missatge a mostrar"
Require valid-user #Permet l'accés a tots els usuaris vàlids. Per usuaris concrets: *Require user usu1 usu2*

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

4.-La diferència entre el mètode Basic i el Digest és que el primer utilitza un arxiu de contrasenyes (amb format user:password) les quals, encara que dins el fitxer s'emmagatzemen encriptades es transmeten de client a servidor en text pla, però el segon transmet les contrasenyes en MD5, fet que aporta (una miiiica) més de seguretat. Si fem servir, de totes maneres, el mètode Digest, cal tenir en compte dues diferències:

*Per crear el fitxer de contrasenyes s'ha d'emprar la comanda htdigest en comptes 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) seran:
AuthType Digest
AuthDigestFile /ruta/arxiu/creat/pas/anterior
AuthName "Missatge a mostrar"
Require valid-user

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.

Tal com ja s'ha dit, existeixen altres mètodes d'autenticació,com ara fent que l'Apache consulti directament els usuaris en una base de dades ("auth_pgsql") o en un servidor Ldap ("auth_ldap"), entre d'altres, però això no ho veurem.