O Servidor Web Apache

O proxecto de servidor HTTP Apache é un software de desenrolo colaborativo que ten como único obxectivo a creación dunha implementación dun servidor HTTP (Web) robusto, de grado comercial, e de acceso libre ao seu código fonte.

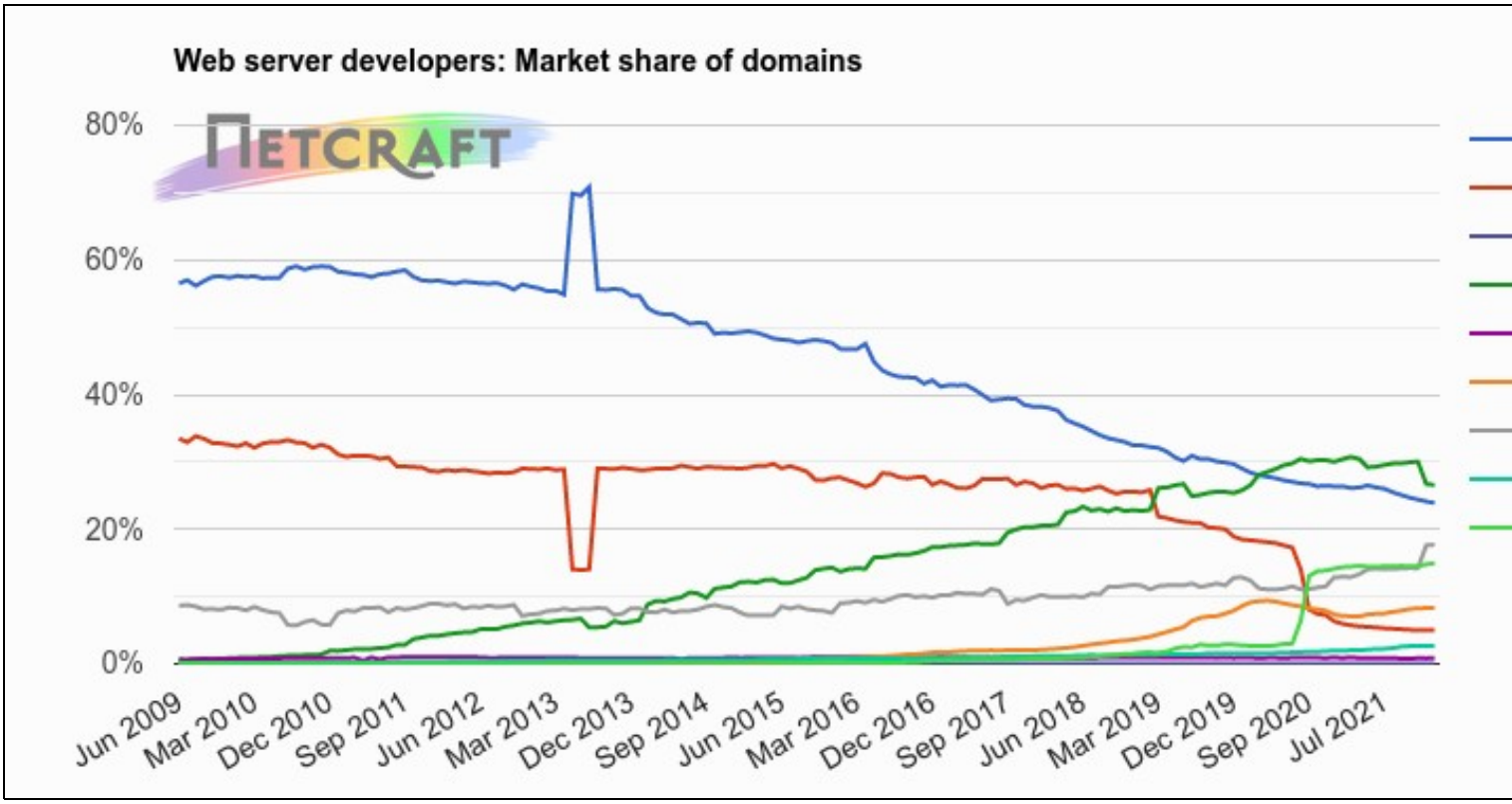
Este proxecto é mantido por un grupo de voluntarios de moitos lugares do mundo que se comunican a través de Internet e da Web e que planifican e desenvolven o servidor e a súa documentación de uso. O proxecto forma parte da Apache Software Foundation. Ademais centos de usuarios teñen aportado ideas, código e a documentación para o pro-xecto.

Sumario

Introdución

Como comezou Apache? Pois no ano 1995, o software máis empregado na web era o demo HTTP desenrolado por Rob McCool no NCSA (National Center for Supercomputing Applications, University of Illinois). Debido a que esta persoa deixou o proxecto, moitos programadores desenrolaron as súas extensións e corrixiron os erros que tiña a súa maneira. A finais de febreiro de 1995, un grupo destes desenroladores coordináronse entre si e formaron o *Grupo Apache*, e xa no mes de abril dese mesmo ano, tomando como base a versión 1.3 do servidor da NCSA publicaron a primeira versión (0.6.2) de proba do servidor Apache, sendo no mes de decembro cando publicaron a definitiva.

Menos dun ano despois, Apache xa era o servidor web máis empregado no mundo, posto que acaba de perder a día de hoxe (Xaneiro de 2021).



Cota de mercado de Servidores web atendendo ao número de dominios. Fonte NixCraft
No ano 1999 membros do grupo Apache, formaron a Apache Software Foundation para fornecer soporte organizacional, legal e financeiro para o servidor HTTP Apache. Esta fundación colocou o software sobre unha base sólida para o futuro desenrolo e ampliou en gran medida o número de proxectos de código aberto que están baixo o paraugas da fundación.

Entre as principais características de Apache, podemos atopar as seguintes:

- Compatible cos últimos protocolos (HTTP/2), e tamén con HTTP/1.1
- Altamente configurable e extensible con módulos de terceiros (e tamén propios)
- Pode ser personalizado con módulos desenrolados coa API Apache
- Provese o seu código fonte con licenza sen ningún tipo de restrición
- Pode correr en multitude de sistemas operativos (Windows, Unix, Linux, ...)
- Está sendo desenvolvido de forma activa
- Implementa moitas características necesitadas por programadores

É moi común atopar o servidor web Apache acompañado de outro software coma por exemplo, en sistemas LAMP (Linux + Apache + MySQL + PHP), sendo este un dos seus máis grandes desafíos, no que ao rendemento se refire.

Pola contra, Apache foi moi criticado por non ter unha interface gráfica para a súa configuración. Toda a súa configuración faise en ficheiros de texto.

Instalación

Apache ten un número grande de versións, sendo actualmente as máis empregadas a 2.2 e a 2.4. Este dato é importante telo en conta porque non todo o que se configura para unha é valido para a outra. No noso caso, centrarémonos na versión 2.4.

Para instalar o servidor web Apache, podemos facelo ou ben a partires do seu código fonte (que podemos obter no sitio web do servidor web Apache), que deberemos compilar, ou a partires de arquivos binarios xa previamente compilados.

No sitio web do servidor web Apache, non atopamos ningún ficheiro binario para descargar e poder instalar. Deberemos facelo no caso de Linux e Unix desde os paquetes que prové a distribución instalada, e no caso de Windows, paquetes de terceiros que o soen empaquetar xunto con algún outro software (*ApacheHaus*, *Apache Lounge*, *BitNami WAMP Stack*, *WampServer* ou *XAMP*)

Nun entorno Linux/Debian, hai que instalar o paquete apache2

```
apt-get install apache2
```

Cada vez que se faga un cambio na súa configuración é necesario reiniciar o servidor, ou volver a cargar os ficheiros de configuración

Inicio (varias posibilidades)

```
service apache2 start
systemctl start apache2.service
apachectl start
```

Parada (varias posibilidades)

```
service apache2 stop
systemctl stop apache2.service
apachectl stop
```

Reinicio (varias posibilidades)

```
service apache2 restart
systemctl restart apache2.service
apachectl restart
```

Recarga de ficheiros de configuración

```
service apache2 reload
```

Estrutura dos ficheiros de configuración

A partires da versión 2 de Apache, deixouse de establecer todas as directivas de configuración de Apache nun so ficheiro e pasouse a facelo en varios que se inclúen uns a outros, ademais de establecer toda a configuración baseada no uso de sitios virtuais.

Estrutura dos ficheiros de configuración en Linux Debian/Ubuntu

Nos equipos Linux en xeral o ficheiro de configuración non soe ser único. En Ubuntu/Debian, todos os ficheiros de configuración están dentro do directorio **/etc/apache2**. O ficheiro de configuración é **/etc/apache2/apache2.conf**. Ten unha estrutura de ficheiro autoexplicativo, e nel atopamos un gráfico en modo texto, que nos explica como se inclúen os ficheiros de configuración:

```
# /etc/apache2/
# |-- apache2.conf
# | `-- ports.conf
# |-- mods-enabled
# | |-- *.load
# | `-- *.conf
# |-- conf-enabled
# | `-- *.conf
# `-- sites-enabled
#   `-- *.conf
```

A explicación de cada ficheiro incluído no directorio /etc/apache2 é a seguinte:

- **apache2.conf**: O ficheiro principal de configuración de Apache2. Conten toda a configuración global.
- **conf-available**: este directorio conten ficheiros de configuración dispoñibles . Todos os ficheiros que en versións anteriores estaban en */etc/apache2/conf.d* deberían moverse a */etc/apache2/conf-available*.
- **conf-enabled**: mantén ligazóns simbólicas aos ficheiros en */etc/apache2/conf-available*. Cando un ficheiro de configuración é enlazado mediante unha ligazón simbólica estará activo a seguinte vez que se reinicie Apache.
- **envvars**: ficheiro con variables de contorna e os seus valores.
- **mods-available**: este directorio contén ficheiros de configuración para cargar módulos e configuralos. Sen embargo, non todos os módulos teñen ficheiros de configuración específicos.
- **mods-enabled**: mantén ligazóns simbólicas aos ficheiros en */etc/apache2/mods-available*. Cando un ficheiro de configuración de módulo é enlazado mediante un ligazón simbólica será activado despois do seguinte reinicio de Apache.
- **ports.conf**: Aloxa as directivas que determinan os portos TCP nos que Apache escoita e atende peticións.

- **sites-available**: Este directorio ten os ficheiros de configuración dos *Host Virtuais* de Apache. Os Host virtuais permiten que Apache sexa configurado para múltiples sitios con configuracións separadas.
- **sites-enabled**: Ao igual que *mods-enabled*, *sites-enabled* contén ligazóns simbólicas a ficheiros dentro de */etc/apache2/sites-available*. De xeito similar, cando se crea unha ligazón simbólica o sitio virtual estará activo unha vez se reinicie Apache.
- **magic**: instrucións para determinar os tipos MIME baseándose nos primeiros bytes de cada ficheiro.

Para estes tres directorios nos que habitualmente se crean ligazóns simbólicas aos ficheiros que existen no directorio *available* correspondente, hai certas aplicacións utilidade para executar desde a liña de comandos. Estas utilidades son **a2ensite**, **a2dissite**, **a2enmod**, **a2dismod**, **a2enconf** e **a2disconf** para habilitar e deshabilitar sitios virtuais, módulos e fragmentos de configuración. A maiores, outros ficheiros de configuración poden ser engadidos usando a directiva **Include**, que tamén poden ser incluídos empregando comodíns para incluír varios nunha soa directiva. Calquera directiva pode ser introducida nestes ficheiros de configuración. Os cambios so serán recoñecidos unha vez que se reinicie o servidor Apache.

Estrutura dos ficheiros de configuración en Windows

Na plataforma Microsoft Windows, xa dependemos de paquetes creados por terceiros, e podemos atoparnos con diferentes formas de chamarlle ao ficheiros de configuración e diferentes formas, con e sen, ficheiros incluídos similar a Linux Debian/Ubuntu. O máis normal é que o ficheiro de configuración principal se chame *httpd.conf* e como moito un directorio chamado extra con configuración opcional incluída con sentencias *Include*.

Sintaxe dos ficheiros de configuración

Os ficheiros de configuración conteñen unha directiva por liña. O carácter de barra invertida (backslash) "\" pode ser usado como derradeiro carácter dunha liña para indicar que a directiva continúa na seguinte liña. Nese caso non debe haber ningún outro carácter ou espazo en branco entre a barra invertida e o fin da liña.

Os argumentos das directivas sepáranse por espazos en branco. Se un argumento conten espazos, deberase encerrar entre comiñas dobres.

As directivas nos ficheiros de configuración non son sensibles a maiúsculas, pero os argumentos con frecuencia si que son sensibles.

As liñas que comezan co carácter #considéranse comentarios e son ignorados. Os comentarios non deberían estar incluídos na mesma liña que outras directivas de configuración. Tamén os espazos en branco que preceden a calquera directiva son ignorados, e por iso, pódense indentar as liñas para maior claridade. Tamén as liñas completas en branco son ignoradas.

Os valores definidos coa directiva *Define* ou variables de shell pódense empregar nos ficheiros de configuración coa sintaxe *${VAR}*. Se "VAR" é o nome dunha variable válida o valor desa variable é substituído en calquera liña do ficheiro de configuración, e o procesado continúa como se o seu contido estivese directamente no ficheiro de configuración. Se a variable "VAR" non se atopa, os caracteres *${VAR}* déixanse sen cambios e reportase unha advertencia (warning). Os nomes das variables non deberían conter o carácter ?:?.

A lonxitude máxima nos ficheiros de configuración normales, despois da substitución de variable e unión de liñas é aproximadamente de 16 MB. Nos ficheiros *.htaccess* a lonxitude máxima é de 8190 caracteres. Pódese comprobar a sintaxe dos ficheiros de configuración antes de iniciar o servidor mediante a execución do comando **apachectl configtest** ou coa opción **-t**.

Configuración básica. Directivas

Portos de escoita e enderezos

Cando se inicia o demo http, este enlazase a algún porto e enderezo IP do equipo local e espera por peticións. Por defecto, escoita en todos os enderezos do equipo. Sen embargo, pode ser necesario dicirlle que escoite en portos específicos ou so en algún enderezo ou unha combinación de ambos. Con frecuencia todo isto combínase coa propiedade de Host Virtual, que determina como httpd responde aos diferentes enderezos nomes de equipo e portos.

A directiva `Listen` dille ao servidor que acepte peticións so en algún(s) porto(s) específico(s) ou combinacións de enderezos e portos. Se unicamente se especifica unha directiva `Listen` o servidor escoita peticións nese porto en todos os seus enderezos. Se se especifica un enderezo IP xunto co porto, atenderá peticións nese porto e interface. Pódense poñer múltiples directivas `Listen` xunto con portos e enderezos. O servidor atenderá peticións en calquera deles.

Por exemplo, para facer que o servidor acepte peticións nos portos *80* e *8000* en todos os seus interfaces de rede usaremos:

```
Listen 80
Listen 8080
```

Para facer que o servidor acepte conexións no porto *80* para unha interface e o *8000* para outro, empregaremos

```
Listen 192.0.2.1:80
 Listen 192.0.2.5:8000
```

Os enderezos IPv6 débense poñer encerrados entre corchetes coma no seguinte exemplo:

```
 Listen [2001:db8::a00:20ff:fea7:ccea]:80
```

A directiva *Listen* non implementa Virtual Hosts ? so lle di ao servidor principal en que enderezos e portos se atenden peticións. Se non hai ningunha directiva `<VirtualHost>` o servidor atende peticións do mesmo xeito. Sen embargo, `<VirtualHost>` pode ser empregado para especificar un comportamento diferente en un ou máis enderezos e portos. Para implementar un Host Virtual, o servidor debe antes saber en que enderezos e portos se atenden peticións. Despois unha sección `<VirtualHost>` deberá ser creada para o enderezo e porto especificado para establecer o comportamento dese Host Virtual. Nótese que se se define a sección `<VirtualHost>` nun enderezo e porto no que o servidor non atende peticións non poderá ser accedido.

Se se emprega cifrado SSL e o módulo ssl está activo, debermos tamén activar o porto 443:

```
<IfModule ssl_module>
    Listen 443
</IfModule>
```

Virtual Hosts

O termo Virtual Host refírese a práctica de executar máis dun sito (coma *company1.exemplo.com* e *company2.exemplo.com*) nunha mesma. Os Virtual Hosts poden ser *baseados en IP*, querendo dicir que temos enderezos IP diferentes para sitio, ou *baseados en nome*, que quere dicir que temos múltiples sitios web no mesmo enderezo. O feito de que se executen na mesma máquina física non é apreciable polo usuario final.

Apache foi un dos primeiros servidores en soportar servidores virtuais baseados en enderezos IP, xa desde versións moi preliminares. A partires da versión 1.1 soporta tanto host virtuais baseados en nomes coma en enderezos

Host Virtuais baseados en nomes

Cos host virtuais baseados en nomes, o servidor depende de que o cliente reporte o nome do host como parte das súas cabeceiras HTTP. Desta maneira varios host diferentes poden compartir o mesmo enderezo IP.

Configurar host virtuais baseados en nomes, é máis doado, xa que é necesario configurar o servidor DNS para que cada nome apunte ao enderezo correcto, e logo configurar o servidor web Apache para que recoñeza os diferentes nomes. O hosting virtual baseado en nomes, tamén é unha axuda á escaseza de enderezos IP. Así que deberase empregar sempre o hosting virtual baseado en nomes a non ser que se requira o hosting virtual baseado en enderezos IP de forma explícita.

É importante recoñecer que o primeiro paso para a resolución no hosting virtual baseado en nomes é a resolución de enderezos IP. Esta resolución baseada en nomes, so escolle o host virtual máis axeitado unha vez reducidos os candidatos baseados no enderezo IP encaixado. Empregando un (*) para o enderezo IP en todas as directivas `<VirtualHost>` fai ese mapeado baseado no enderezo IP irrelevante.

Cando chega unha petición o servidor busca a directiva `<VirtualHost>`que mellor encaixe (a máis específica) baseado no enderezo IP e no porto especificado na petición. Se hai máis de un virtual host que encaixe coa combinación de enderezo IP e porto, Apache2 compara as directivas `ServerName` e `ServerAlias` co nome de servidor presente na petición.

Se non se atopa ningunha directiva `ServerName` ou `ServerAlias` no conxunto de host virtuais que conteñen a combinación de enderezo IP e número de porto que mellor en caixa, entón *o primeiro host virtual en ser listado que encaixe* será empregado [1].

O primeiro paso para crear un bloque `<VirtualHost>` para cada host diferente que se quere servir. Dentro de cada bloque `<VirtualHost>` necesítase polo menos unha directiva `ServerName` para designar que host virtual se sirve, e tamén unha directiva `DocumentRoot` que indique en que lugar do sistema de ficheiros reside o contido para ese virtual host. Exemplo:

```
<VirtualHost *:80>
    ServerName www.exemplo1.com
    ServerAlias exemplo1.com
    DocumentRoot "/www/exemplo1"
</VirtualHost>
<VirtualHost *:80>
    ServerName www.exemplo2.com
```

```
    ServerAlias exemplo2.com
    DocumentRoot "/www/exemplo2"
</VirtualHost>
```

Host Virtuais baseados en enderezos IP

O virtual hosting baseado en enderezos IP é un método para aplicar diferentes directivas baseadas no enderezo IP e porto no que se recibe a petición. Desta maneira sérvense diferentes sitios e en diferentes portos e interfaces de rede.

En moitos casos, o uso de virtual hosts baseados en nome é máis convinte, porque permite a moitos host virtuais compartir un mesmo enderezo/porto.

A frase *baseado en enderezos IP* indica que o servidor debe ter diferentes combinacións enderezo IP/porto para cada host. Isto pode conseguirse nunha máquina tendo varias interfaces físicas ou empregando interfaces de rede virtuais, que son soportados polos sistemas operativos modernos, e/ou empregando múltiples números de porto.

Exemplo:

```
<VirtualHost 172.20.30.40:80>
    ServerAdmin webmaster@www1.exemplo.com
    DocumentRoot "/www/vhosts/www1"
</VirtualHost>
<VirtualHost 172.20.30.50:80>
    ServerAdmin webmaster@www2.exemplo.org
    DocumentRoot "/www/vhosts/www2"
</VirtualHost>
```

Host virtual por defecto de Apache

Apache2 ven de serie con unha configuración de host virtual por defecto amigable. Isto é, configurado cun único host virtual por defecto (empregando a directiva VirtualHost), que pode ser modificada ou usada como se houbese so un único sitio, ou tamén como modelo para host virtuais adicionais se se desexan configurar. Se se deixa so, este host virtual, funcionará coma sitio por defecto, ou tamén se a URL que introduce o usuario non encaixa con ningunha directiva ServerName de ningún outro host virtual. Para modificar o host virtual, edítase o ficheiro `/etc/apache2/sites-available/000-default.conf`.

As directivas establecidas para un host virtual, so se aplican a ese host virtual en particular. Se a directiva é de ámbito de servidor e non se define dentro do host virtual, úsase un valor por defecto. Por exemplo, pódese definir o enderezo de correo electrónico do Webmaster no ámbito do servidor, e non definilo dentro de cada host virtual de forma individual.

Se se quere configura un novo sitio ou host virtual pódese copiar ese ficheiro dentro doe mesmo directorio cun nome a escoller por parte do usuario.

Contextos

Os contextos indican onde se permite establecer unha directiva. É unha lista separada por comas dos seguintes valores:

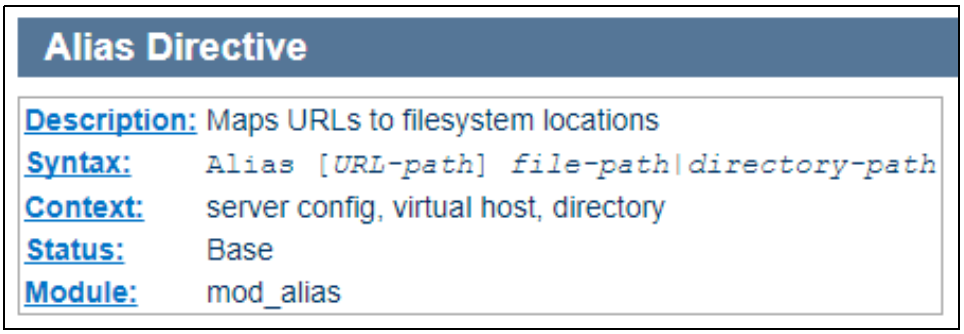
- **server config**: Isto significa que a directiva pode ser empregada nos ficheiros de configuración de Apache (por exemplo, *apache2.conf*), pero non dentro dos contedores `<VirtualHost>` ou `<Directory>`. Tampouco se permite nos ficheiros `.htaccess`.
- **virtual host**: Este contexto significa que unha directiva pode aparecer dentro dun contexto `<VirtualHost>` que se atope en calquera ficheiro de configuración.
- **directory**: Unha directiva válida neste contexto, pode ser usada dentro de contedores `<Directory>`, `<Location>`, `<Files>`, `<If>`, e `<Proxy>` nos ficheiros de configuración suxeitos a restricións descritas na sección de configuración.
- **.htaccess**: Se unha directiva é válida neste contexto significa que pode aparecer nun ficheiro de configuración de directorio con nome `.htaccess`. Estes poden non ser procesados dependendo das directivas de anulación activas.

Cada directiva soamente é valida dentro do seu contexto. Se se intenta empregar fora do contexto designado, aparecerá un erro de configuración que pode previr ao servidor de manexar peticións nese contexto de forma incorrecta, ou impedir que o servidor opere, (por exemplo impedindo que se inicie).

As localizacións dunha directiva son resultado unha operación booleana de OU de todos os contextos válidos. En outras palabras, se unha directiva está marcada como válida dentro de *server config*, *.htaccess* pode ser empregada dentro do ficheiro *apache2.conf* e tamén en ficheiros `.htaccess`, pero non dentro de `<Directory>` ou `<VirtualHost>`.

Directivas

Apache ten centos de directivas. Nesta guía non as imos amosar todas, soamente as máis empregadas e necesarias para configurar un servidor web. Se consultamos as directivas na Documentación oficial do Servidor Web Apache, podemos ver unha descrición breve, a súa sintaxe, o contexto onde se pode empregar e o modulo que a prové (para poder activalo). É unha boa práctica consultar cada directiva na documentación oficial andes de empregala. Vexámolo nun exemplo:



Alias Directive

Description:	Maps URLs to filesystem locations
Syntax:	`Alias [URL-path] file-path|directory-path`
Context:	server config, virtual host, directory
Status:	Base
Module:	mod_alias

Exemplo de directiva
ServerName, ServerAlias e ServerAdmin

A directiva `ServerName` establece o nome do host (hostname) que o servidor (ou VirtualHost) usa para identificarse a si mesmo. A directiva `ServerName` úsase (posiblemente xunto con `ServerAlias`) para identificar de forma única un host virtual cando se empregan host virtuais baseados en nomes.

A directiva `ServerName` pode aparecer en calquera lugar dentro da definición dun servidor. Sen embargo, cada aparición anula a anterior aparición (dentro do servidor). Se empregamos host virtuais baseados en nomes, a directiva `ServerName` dentro dunha sección `<VirtualHost>` especifica que nome de host debe aparecer na cabeceira *"Host:"* dunha petición.

A directiva `ServerAlias` establece nomes alternativos para un host para empregar host virtuais baseados en nomes. A directiva `ServerAlias` pode incluír comodíns se é necesario.

Para o proceso de encaixe do mellor conxunto de bloques `<Virtualhost>` nos host virtuais baseados en nomes, estes son procesados en orde de aparición na configuración. Úsase o primeiro ServerName ou ServerAlias sen precedencia de comodíns.

A directiva `ServerAdmin` establece o enderezo de contacto que o servidor inclúe nas mensaxes de erro que se devolven ao cliente.

Exemplo:

```
<VirtualHost *:80>
    ServerName server.exemplo.com
    ServerAdmin www-admin@exemplo.com
    ServerAlias server server2.exemplo.com server2
    ServerAlias *.exemplo.com
            # ...
</VirtualHost>
```

DocumentRoot

Esta directiva establece o directorio desde o cal o demo httpd sirve os ficheiros. A non ser que unha URL encaixe con algunha directiva como `Alias`, o servidor, concaténalle ao camiño especificado na raíz de documentos a URL referenciada para construír o camiño ao documento.

Exemplo:

```
<VirtualHost *:80>
    ServerName www.exemplo1.com
    ServerAlias exemplo1.com
    DocumentRoot "/www/exemplo1/web"
</VirtualHost>
```

Así un acceso a `http://www.exemplo.com/index.html` refírese a `/www/exemplo1/web/index.html`. Se a ruta ao directorio raíz non é absoluta, asúmese que é relatica ao contido da directiva `ServerRoot`.

 O `DocumentRoot` debería estar especificado sen a barra final.

Alias

A directiva `Alias` permite a documentos almacenados no sistema de ficheiros local nalgún lugar diferente a `DocumentRoot`. URLs (unha vez descodificado %) que comeza cunha ruta de URL serán mapeadas a ficheiros locais que comezan cunha ruta de directorio. A ruta de URL é sensible a maiúsculas, incluso en sistemas de ficheiros non sensibles a elas.

```
Alias "/image" "/ftp/pub/image"
```

Unha petición para `http://exemplo.com/image/foo.gif` causará que o servidor devolva o ficheiro `/ftp/pub/image/foo.gif`. So se emparellan anacos de rutas completas, polo que o exemplo anterior non emparellaría coa URL `http://exemplo.com/imagefoo.gif`.

 Hai que ter coidado con incluír a / ao final da URL, xa que requirira que exista tamén ao final da ruta no sistema de ficheiros. Así se se usa

```
Alias "/icons/" "/usr/local/apache/icons/"
```

Entón a URL `/icons` non se poderá acceder a través do Alias, xa que falta o / final. Do mesmo xeito, se o omitimos, tamén o debemos omitir na ruta do sistema de ficheiros. É de notar que pode necesitarse especificar seccións `<Directory>` adicionais relacionadas co destino dos alias. Particularmente , se creamos un Alias a un directorio fora do `DocumentRoot`, seguramente necesitemos permitir o acceso de forma explícita.

```
Alias "/image" "/ftp/pub/image"
<Directory "/ftp/pub/image">
    Require all granted
</Directory>
```

DirectoryIndex

A directiva `DirectoryIndex` establece a lista de recursos a atopar cando, cando o cliente solicita un índice do directorio especificando unha / ao final do nome do directorio. As URL locais son as URL (codificadas con %) dun documento relativo ao directorio solicitado. É habitual que conteñan o nome dun ficheiro no directorio. Pódense poñer varias URLs, pero devolverase a primeira que se atope. Se ningunha existe, e está activada a directiva `Indexes` o servidor xerará un listado do contido do directorio Exemplo:

```
DirectoryIndex index.html
```

Así, unha petición para `http://exemplo.com/docs/` devolverá `http://exemplo.com/docs/index.html` se este existe, ou un listado do directorio se está activa a opción `Indexes` ou unha mensaxe de erro.

Options

A directiva Options controla que características do servidor están activas nun directorio en particular. Pode tomar o valor de `None`, o cal indica que ningunha opción está habilitada ou unha ou máis das seguintes opcións:

- **`All`**: Todas as opcións excepto `MultiViews`. Este é o valor por defecto.
- **`FollowSymLinks`**: O servidor seguirá todos as ligazóns simbólicas deste directorio. Anque o servidor siga a ligazón simbólica non se cambiará o nome da ruta emparellada dentro da sección `<Directory>`.
- **`Includes`**: Permítense as inclusións do lado do servidor fornecidas polo módulo `mod_include`.
- **`Indexes`**: Se se solicita unha URL que se mapea a un directorio que non existe ningún dos ficheiros listados coa directiva `DirectoryIndex` (por exemplo, index.html) nese directorio, entón o módulo `mod_autoindex` devolverá un listado formatado do contido do directorio.
- **`SymLinksIfOwnerMatch`**: O servidor so seguirá a ligazón simbólica se os propietarios do enlace e do destino coinciden.

Hai máis opcións que se poden consultar na documentación oficial, pero son menos usadas que as que listamos aquí.

Normalmente, se se aplican múltiples `Options` a un directorio, entón so son de aplicación as máis específicas e as outras son ignoradas e non se mesturan. Sen embargo, se todas as opcións expresadas na directiva `Options` están precedidas por un símbolo + ou -, entón mestúranse as opcións. Toda opción precedida por un + engádense a lista de opcións actuais, e calqueras precedidas por un ? réstase da lista. Non se permite mesturar `Options` con un + ou un ? con outras sen símbolos, e é causa de resultados impredecibles.

Por exemplo, sen símbolos + e - :

```
<Directory /web/docs>
   Options Indexes FollowSymLinks
```

```
</Directory>
<Directory /web/docs/spec>
   Options Includes
</Directory>
```

entón soamente se establece `Includes` para o directorio `/web/docs/spec`. Sen embargo, se a segunda directiva `Options` tivese símbolos + e - :

```
<Directory /web/docs>
   Options Indexes FollowSymLinks
</Directory>
<Directory /web/docs/spec>
   Options +Includes -Indexes
</Directory>
```

entón establécense as opcións `FollowSymLinks` e `Includes` para o directorio `/web/docs/spec`.

Seccións de Configuración

O bloque `<VirtualHost>` dálle aos administradores a capacidade de modificar o comportamento do servidor web a nivel de host ou de dominio. Calquera opción especificada nun `<VirtualHost>` aplícase ao dominio enteiro. Sen embargo, non prové a capacidade de especificar opcións a nivel de directorio. Afortunadamente, Apache dispón de outras posibilidades para especificar configuración.

Os contedores máis empregados son aqueles que permiten cambiar algunha configuración en determinados lugares do sistema de ficheiros ou do espazo web. Primeiro é necesario comprender a diferencia entre eses dous.

O sistema de ficheiros é a vista dos discos locais vistos polo sistema operativo. Por exemplo, nunha instalación por defecto os documentos que amosa Apache residen en `/var/www/html` nos sistemas de ficheiros baseados en Unix ou en `C:/Program Files/Apache Group/Apache2` en sistemas de ficheiros Windows. (Nótese que as barras de separación de directorios sempre son en dirección cara adiante, incluso para Windows.)

Polo contrario, o espazo web é a vista do sitio ofrecida polo servidor web tal e como o ve o cliente. Así a ruta `/dir/` no espazo web corresponde coa ruta `/var/www/html/dir/` no sistema de ficheiros dunha instalación por defecto de Apache en Unix/Linux. O espazo web non se mapea de forma directa ao sistema de ficheiros, xa que hai moitas páxinas que se xeneran de forma dinámica desde bases de datos ou outras localizacións.

Contedores do sistema de ficheiros

As directivas `<Directory>` e `<Files>`, xunto coas súas correspondentes directivas contrapartes con expresións regulares, aplícanse directamente a partes do sistemas de ficheiros. As directivas incluídas nunha sección `<Directory>` aplícase á porción de sistema de ficheiros correspondente ao directorio especificado, e a todos os seus subdirectorios dentro dese directorio (así coma tamén a todos os ficheiros neses directorios). O mesmo efecto pode obterse empregando ficheiros `.htaccess`. Por exemplo, na seguinte configuración habilítase a opción `Indexes` para o directorio `/var/web/dir1` e todos os seus subdirectorios.

```
<Directory "/var/web/dir1">
    Options +Indexes
</Directory>
```

Adicionalmente hai que ter en conta, con respecto aos bloques <Directory>:

- Os bloques `Directory` non se poden aniñar uns dentro doutros.
- Os bloques `Directory` poden aniñarse dentro de bloques `<VirtualHost>`.
- A ruta contida dentro dun bloque `Directory`, pode conter comodíns. O asterisco (*) emparellarase con calquera serie de caracteres mentres que o interrogante, (?) so se emparella cun único carácter. Isto pode ser útil se necesitamos controlar algunha opción para o `DocumentRoot` de moitos host virtuais. Pódese empregar un bloque `<Directory>` coa seguinte liña:

```
<Directory /srv/www/*/public_html>
```

As directivas englobadas nunha sección `<Files>` so se aplican a calquera ficheiro cun nome especificado, tendo en conta en que directorio se atopa. Así, por exemplo, as seguintes directivas de configuración ubicadas no ficheiro de configuración global, denegan o acceso a calquera ficheiro chamado *private.html* independentemente de onde se atope.

```
<Files "private.html">
    Require all denied
</Files>
```

Para referirse a ficheiros que se atopan en lugares particulares do sistema de ficheiros, as seccións `<Files>` e `<Directory>` poden estar combinadas. Por exemplo, a seguinte configuración denegará o acceso aos ficheiros `/var/web/dir1/private.html`, `/var/web/dir1/subdir2/private.html`, `/var/web/dir1/subdir3/private.html`, e calquera outra instancia de `private.html` que se atope debaixo do directorio `/var/web/dir1/`.

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

Se aínda así se necesita un emparellamento máis flexible, cada contedor ten a súa contraparte, `<DirectoryMatch>` e `<FilesMatch>` que periten o uso de expresións regulares perl-compatibles. Empregado expresións regulares, podemos denegar o acceso a moitos tipos de imaxe da seguinte maneira:

```
<FilesMatch "\.(bmp|gif|jpe?g|png)$">
    Require all denied
</FilesMatch>
```

Contedores do espazo web

A directiva `<Location>` e a súa contraparte que usa expresións regulares `<LocationMatch>`, cambian a configuración para contido situado no espazo web.

Por exemplo, a seguinte configuración prevén o acceso a calquera ruta de URL que comece con `/private`, e aplícase a peticións tales coma `http://yoursite.exemplo.com/private`, `http://yoursite.exemplo.com/private123`, e `http://yoursite.exemplo.com/private/dir/file.html` así coma a calquera outra petición que comece coa `/private`.

```
<LocationMatch "^/private">
    Require all denied
</LocationMatch>
```

Escoller entre contedores do sistema de ficheiros e do espazo web, é moi doado. Cando queremos aplicar directivas a obxectos que residen no sistema de ficheiros sempre se empregan as directivas `<Directory>` e/ou `<Files>`. Cando aplicamos directivas a obxectos que non residen no sistema de ficheiros (como pode ser unha páxina web xerada desde unha base de datos, empregaremos `<Location>`.

Na web [2], podemos atopar máis información sobre as seccións de configuración

Ficheiros `.htaccess`

Os ficheiros `.htaccess`[2] (ou "ficheiros de configuración distribuídos") proven unha vía para facer cambios na configuración a nivel de directorio. Un ficheiro que conten unha ou máis directivas de configuración colócase nun directorio en particular, e as directivas aplícanse a ese directorio e a todos os seus subdirectorios

Así descentralizase a administración do servidor web. Calquera opción especificada unha sección `<Directory>` pode especificarse, como norma xeral, dentro dun ficheiro `.htaccess`. Os ficheiros `.htaccess` son moi útiles en casos no que o operador do sitio web ten acceso para editar ficheiros no directorio publico do sitio web, pero non nos ficheiros de configuración de Apache.

O que poñemos neses ficheiros ven determinado polo valor da directiva `AllowOverride`. Esta directiva especifica, en categorías, que directivas se lles permite aparecer nos ficheiros `.htaccess`. Cando o servidor atopa un ficheiro `.htaccess`, necesita saber que directivas declaradas nese ficheiro poden anular ou cambiar outras declaradas previamente.

A directiva `AllowOverride` so é valida en seccións `<Directory>` especificadas sen expresións regulares, e non están permitidas en seccións `<Location>`, `<DirectoryMatch>` ou `<Files>`.

Cando esta directiva ten o valor `None` e a directiva `AllowOverrideList` tamén ten o valor `None`, os ficheiros `.htaccess` ignóranse completamente. Nese caso, o servidor nin sequera os intenta ler do sistema de ficheiros. Cando toma o valor `All`, entón permítese calquera directiva válida no contexto `.htaccess` nos ficheiros `.htaccess`.

O valor desa directiva pode ser un dos seguintes para os grupos de directivas indicados:

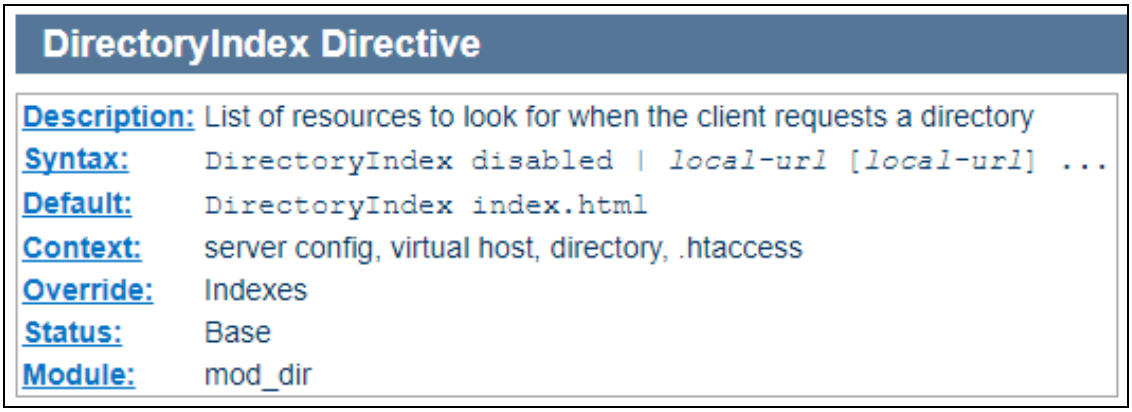
- **`AuthConfig`**: Permite o uso de directivas de autorización (`AuthGroupFile`, `AuthName`, `AuthType`, `AuthUserFile`, `Require`, etc.).
- **`FileInfo`**: Permite o uso de directivas que controlan tipos de documentos, directivas do módulo `mod_rewrite`, do módulo `mod_alias`, e a directiva `Action` do módulo `mod_actions`.

- **`Indexes`**: Permite o uso de directivas que controlan o indexado e listado de directorios (`AddDescription`, `AddIcon`, `DefaultIcon`, `DirectoryIndex`, etc.).
- **`Limit`**: Permite o uso de directivas que controlan o acceso de hosts (`Allow`, `Deny` e `Order`).
- **`Nonfatal=[Override|Unknown|All]`**: Este valor fai que ante valores non permitidos pola directiva `AllowOverride` os erros que aparezan nos ficheiros `.htaccess` se traten coma non fatais. No lugar de causar un Erro Interno do Servidor, as directivas non permitidas ou desactivadas ignóranse e amosase un mensaxe de advertencia (warning) nos ficheiros de log:
    - **`Nonfatal=Override`** trata directivas non permitidas por `AllowOverride` coma non fatais.
    - **`Nonfatal=Unknown`** trata directivas non coñecidas coma non fatais. Isto tamén cubre directivas e tipos implementados por módulos non presentes.
    - **`Nonfatal=All`** trata os dous anteriores como non fatais.
- **`Options[=Option,...]`**: Permite o uso de directivas que controlan opcións a nivel de directorio (Directiva `Options`). Un signo igual seguido dunha lista de opcións separada por comas sen espazos indica que opcións en particular se poden establecer coa directiva `Options`.

Exemplo:

```
AllowOverride AuthConfig Indexes
```

Se se permite unha directiva nos ficheiros `.htaccess` a documentación oficial desa directiva terá unha sección **Override** especificando que valor ten que estar na directiva `AllowOverride` para poder ver se está permitida. Na seguinte imaxe, pódese ver un exemplo coa directiva `DirectoryIndex`:



Os ficheiros `.htaccess` procésanse para cada petición e aplícanse para cada un dos ficheiros no directorio actual. As directivas tamén son de aplicación nos directorios que hai dentro do directorio procesado na xerarquía do sistema de ficheiros. A pesar da potencia e flexibilidade que dan os ficheiro `.htaccess` tamén hai unha serie de desvantaxes para usalos.

- Se está habilitado o uso de ficheiros `.htaccess` Apache debe comprobar e procesar todas as directivas que hai nos ficheiros `.htaccess` en cada petición. Ademais, Apache debe consultar todos os ficheiros `.htaccess` que hai nos directorios de nivel superior no sistema de ficheiros. Isto pode causar unha perda de rendemento xa que todas as comprobacións hai que facelas anque os ficheiros `.htaccess` non se usen.
- As opcións establecidas en `.htaccess` poden anular opcións establecidas nos bloques `<Directory>`, o cal pode causar confusión e comportamentos non desexados. Toda opción establecida nun ficheiro `.htaccess` pode tamén establecerse nun bloque `<Directory>`.
- Ao permitirlla a usuarios sen privilexios modificar a configuración do servidor web, pode ser un potencial fallo de seguridade. Sen embargo, os valores de `AllowOverride` poden mitigar este risco de xeito considerable.

Tamén se pode cambiar o valor da directiva `AccessFileName` para especificar outro nome no que Apache busque estas opcións de configuración. Se se cambia esta opción hai que establecer un bloque `<Files>` para previr accesos non intencionados a eses ficheiros. Todo isto non é recomendable por razóns de seguridade obvias. Se botamos un ollo ao ficheiro principal de configuración de Apache2 (`apache2.conf`) podemos atopar un anaco del coma isto:

```
AccessFileName .htaccess
<Files ~ "^\.ht">
    Require all denied
</Files>
```

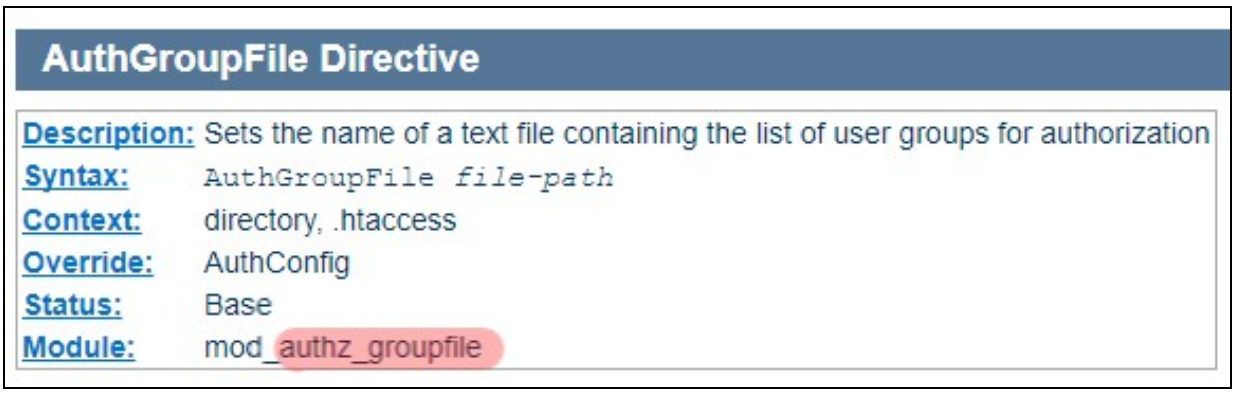
A primeira liña indícalle a Apache que busque ficheiros `.htaccess` nos directorios de acceso público. A segunda, o bloque `<Files ~ "^\.ht">` dille a Apache que denegue todas as peticións explícitas de calquera solicitude de ficheiros cuxo nome comece cos caracteres `.ht`. Isto prevén aos visitantes externos de ter acceso a opcións de configuración. Desta maneira, podemos previr aos usuarios do sitio web de acceder a ficheiros non autorizados, unicamente facendo que o seu nome comece por `.ht` (por exemplo `.htpasswd`)

Módulos

A inclusión de módulos para ampliar a funcionalidade de Apache, faise coa directiva `LoadModule`seguida do nome do módulo e o nome de ficheiro (ou libraría) que o contén. Serán necesarias tantas directivas `LoadModule` coma módulos que vaian a ser empregados. En Debian/Ubuntu os módulos poden ser habilitados ou deshabilitados cos comandos **a2enmod** e **a2dismod**. O nome do módulo encaixa co nome do ficheiro que o contén (sen a extensión .conf). Exemplo:

```
a2enmod usertrack
a2dismod usertrack
```

Sempre é obrigatorio, ao habilitar ou deshabilitar módulos reiniciar o servizo Apache2. Pero, a pregunta do millón, é, que módulos se deben habilitar? Non todos se deben habilitar, porque o servidor malgastará demasiada memoria e tempo de procesador. A resposta é cargar so os xustamente necesarios e ningún máis. Se consultamos calquera directiva na Documentación oficial de Apache, podemos ver que módulo prové esa directiva e se é necesario activalo para poder usala. Velaí vai un exemplo:



Tamén podemos comprobar que módulos están cargados co comando

```
apachectl -t -D DUMP_MODULES
```

Control de Acceso

O control de acceso pode facerse a través de varios módulos. Os máis importantes deles son `mod_authz_core` e `mod_authz_host`. Se se quere restrinxir o acceso a porcións dun sitio web baseados no enderezo IP do visitante, a maneira máis sinxela é empregar o módulo `mod_authz_host`. A directiva `Require` ten unha variedade grande de maneiras de permitir ou denegar acceso a recursos. En conxunción coas directivas de agrupamento `RequireAll`, `RequireAny`, e `RequireNone`, estes requirimentos pódense combinar de innumerables maneiras, para conseguir a política de acceso que se desexa. A maneira de usar estas directivas é a seguinte:

```
Require host nome
Require ip ip.address
```

Na primeira delas, o enderezo é un nome totalmente cualificado (FQDN) ou un nome parcial. Tamén se poden poñer múltiples nomes ou enderezos, se é necesario. Na segunda maneira, ip.address é un enderezo IP, un enderezo IP parcial, unha parella enderezo de rede/máscara ou un bloque CIDR. Tamén se poden empregar bloques CIDR.

Exemplos:

```
Require ip 10.1.2.3
Require ip 192.168.1.104 192.168.1.205
Require ip 10.1
Require ip 10.1.0.0/255.255.0.0
Require ip 10.1.0.0/16
Require host exemplo.org
Require host .net exemplo.edu
```

Na documentación de `mod_authz_host` hai máis exemplos desta sintaxe.

Pódese insertar a palabra `not` para negar un requirimento particular. Nese caso, como not é a negación dun valor, por si so non permite ou denega unha petición xa que non verdadeiro, non implica falso. Dese xeito, para negarlle a visita usando unha negación, o bloque debe ter polo menos un elemento que se avalíe a verdadeiro ou falso. Por exemplo, se temos alguén atacando ao servidor, e queremos denegarlle o acceso, podemos facelo da seguinte maneira:

```
<RequireAll>
    Require all granted
    Require not ip 10.252.46.165
</RequireAll>
```

Os visitantes que accedan desde o enderezo (10.252.46.165) non serán capaces de ver o contido protexido por estas directivas.

Contedores de Autorización

A directiva `Require` comproba se un determinado usuario pode acceder de acordo cun provedor de autorización e unhas restricións especificadas. O módulo `mod_authz_core` (que ven Apache) proporciona varios provedores de autorización dos máis empregados son

- **`Require all granted`**: Acceso incondicional permitido.
- **`Require all denied`**: Acceso incondicional denegado.

Ámbolos dous, habitualmente están dentro de seccións `<Directory>`. `<RequireAll>` e `</RequireAll>` tamén se usan para englobar un grupo de directivas de autorización das que ningunha pode fallar e polo menos unha debe de cumprirse para que o grupo se avalíe como exitoso.

Se ningunha das directivas dentro de `<RequireAll>` falla, e polo menos unha se avalía de forma afirmativa, entón o bloque avaliase coma exitoso. Se ningunha se avalía de forma positiva e ningunha falla, o bloque avaliase coma neutral. En tódolos demais casos coma non exitoso. `<RequireAny>` e `</RequireAny>` empréganse para englobar un grupo de directivas de autorización das que polo menos unha ten que avaliarse de xeito afirmativo para que a directiva `<RequireAny>` sexa exitosa.

Se unha ou máis directivas contidas dentro de `<RequireAny>` se avalía de xeito afirmativo, entón, a directiva `<RequireAny>` é exitosa. Se ningunha é exitosa pero ningunha falla, entón devolvese un resultado neutral. Nos outros casos a avaliación é negativa.

Se se poñen varias directivas `Require` sen ningún tipo de agrupamento, suponse que o agrupamento que as engloba `<RequireAny>`. `<RequireNone>` e `</RequireNone>` úsanse para englobar grupos de directivas de autorización das cales ningunha debe avaliarse de forma positiva para que a directiva `<RequireNone>` non falle.

Se unha ou máis directivas contidas dentro de `<RequireNone>` ten éxito, a directiva `<RequireNone>` falla. En todos os demais casos o resultado é neutral. Así do mesmo xeito que a outra directiva para negar autorización `Require not`, non pode autorizar unha petición de forma independente, xa que nunca devolve un resultado exitoso. Sen embargo, si que se pode empregar para restrinxir un conxunto de usuarios que si terían acceso a un recurso.

Os contedores de autorización `<RequireAll>`, `<RequireAny>` e `<RequireNone>` pódense combinar uns con outros e coa directiva `Require` para construír unha lóxica de autorización complexa.

Na documentación de `mod_auth_core` pódese atopar máis información e exemplos.

Autenticación e autorización

Autenticación[3] é calquera proceso no cal se verifica que alguén é quen realmente di que é. Autorización é calquera proceso polo que a calquera usuario se lle permite estar onde quere estar ou de obter a información que quere obter.

Hai tres tipos de módulos involucrados nos procesos de autenticación e autorización. É necesario escoller polo menos un módulo de cada grupo.

- Tipos de Autenticación (`mod_auth_basic` e `mod_auth_digest`)
- Provedor de autenticación (`mod_authn_anon`, `mod_authn_dbd`, `mod_authn_dbm`, `mod_authn_file`, `mod_authnz_ldap` e `mod_authn_socache`)
- Autorización (`mod_authnz_ldap`, `mod_authz_dbd`, `mod_authz_dbm`, `mod_authz_groupfile`, `mod_authz_host`, `mod_authz_owne` e`mod_authz_user`)

De xeito adicional a estes módulos, tamén están os módulos `mod_authn_core` e `mod_authz_core`, que implementen directivas propias comúns a todos os módulos de autenticación.

Tamén o módulo `mod_authnz_ldap` ofrece á vez autenticación e o provedor de autorización. O módulo `mod_authz_host` ofrece autorización e control de acceso baseado no nome do equipo, enderezo IP ou características da petición, pero non é parte do sistema de provedores de autenticación.

As directivas tratadas aquí necesitan estar dentro dalgunha sección de configuración do servidor principal, tipicamente nunha sección `<Directory>` ou nun ficheiro de configuración de directorio `.htaccess`. Se se pretende empregar ficheiros `.htaccess` hai que poñer na directiva `AllowOverride` un valor que permita directivas de autenticación neses ficheiros. Os valores deben ser `AuthConfig` ou `All`. Finalmente, a directiva `Require` de forma illada ou agrupada con `<RequireAll>`, `<RequireAny>` ou `<RequireNone>` indican a que usuarios se lles permite ver o contido do directorio.

Autenticación Basic

O módulo `mod_auth_basic` permite o uso da autenticación HTTP Basic para restrinxir o acceso comprobando os usuarios no provedor indicado. Este módulo debe ser combinado cun módulo de autenticación, como pode ser `mod_authn_file` e un modulo de autorización como pode ser `mod_authz_user`. Hai que ter en conta que o contrasinal envíase ao servidor codificado co algoritmo **base64**, o cal ten un algoritmo inverso que permite unha descodificación sinxela.

Para escoller este módulo é necesario poñer o valor `Basic` na directiva `AuthType`. Esta directiva establece o nome do dominio (*realm*) de autorización para un directorio. Este nome de dominio amósaselle ao cliente para que saiba que nome de usuario e contrasinal debe introducir. Se este nome de dominio conten espazos debe ser encerrado entre comiñas dobres. A cadea introducida na directiva `AuthName` coincide coa mensaxe que lle aparece no cadro de diálogo de autenticación en moitos navegadores. Así, por exemplo, se un usuario de autenticou no nome de dominio ?Área restrinxida?, automaticamente empregarase o mesmo nome de usuario e contrasinal sempre que se indique ese mesmo dominio. Así impídese que se lle pregunte o nome de usuario e contrasinal ao usuario máis dunha vez no mesmo dominio. Por suposto, se cambia o nome de equipo do cliente ou o seu enderezo IP, debese introducir de novo a información de login.

Segundo, necesítase un provedor de Autenticación. Se se usa `mod_auth_basic` como tipo de Autenticación, necesítase poñer na directiva `AuthBasicProvider` un dos seguintes valores: `file` para usar o módulo `mod_authn_file`, `dbm` para usar o módulo `mod_authn_dbm` ou `dbd` para o módulo `mod_authn_dbd`. Un dos valores máis típicos é `file`. Nese caso necesítase un ficheiro de texto cos nomes de usuario e contrasinais que sexa lexible polo demo de execución de Apache2. O ficheiro crease co comando `htpasswd` que ven de serie coa instalación de Apache. Para establecer cal é ese ficheiro empregase a directiva `AuthUserFile`.

Exemplo:

```
#First time -c option to create the file.
htpasswd -c /usr/local/apache/passwd/passwords rbowen
#Next time we add new users to file.
htpasswd  /usr/local/apache/passwd/passwords joesmith
```

Finalmente, a directiva `Require` soa ou agrupada con `<RequireAll>`, `<RequireAny>` ou `<RequireNone>` indica que usuarios se lles permite ver o contido do directorio.

No directorio en cuestión, a configuración pode ser tal que así:

```
AuthType Basic
AuthName "Restricted Files"
AuthBasicProvider file
AuthUserFile "/usr/local/apache/passwd/passwords"
Require user rbowen
```

Autenticación Digest

O módulo `mod_auth_digest` implemente a Autenticación HTTP a cal usa o algoritmo MD5 para codificar passwords, e é unha alternativa a `mod_auth_basic` no que o contrasinal é transmitido en texto plano. Sen embargo, isto non é ningunha mellora de seguridade sobre a autenticación basic. Por outro lado, o almacenamento do contrasinal no servidor é menos seguro con autenticación digest que con autenticación basic. Desa maneira é moito mellor alternativa empregar autenticación basic xunto co módulo `mod_ssl`. Para escoller este módulo é necesario escoller o valor `Digest` na directiva `AuthType`. Do mesmo xeito que `mod_auth_basic`, a directiva `AuthName` establece o nome do dominio de autenticación (*realm*) dun directorio. Este nome de dominio é o que se lle da ao cliente para que o usuario saiba que nome de usuario e contrasinal debe poñer. Pero a maiores, tamén se emprega co comando `htdigest` para crear os contrasinais dos usuarios.

De maneira similar ao módulo `mod_auth_basic` é necesario un provedor de autenticación. A directiva `AuthDigestProvider` debe especificar un dos tres seguintes valores `file`, `dbm` ou `dbd` dependendo de que módulo se vai empregar.

Se se escolle o valor `file`, coa utilidade `htdigest` pódese crear a lista de usuarios. A forma de usala é similar a `htpasswd` vista anteriormente, pero a maiores debe ser especificado o nome de dominio no momento da creación, e debe coincidir co introducido na directiva `AuthName`. Vémolo nun exemplo

```
#First time -c option to create the file.
htdigest -c /usr/local/apache/passwd/passwords myrealm rbowen
#Next time we add new users to file.
htdigest  /usr/local/apache/passwd/passwords myrealm joesmith
```

Finalmente, a directiva `Require` soa ou agrupada con `<RequireAll>`, `<RequireAny>` ou `<RequireNone>` indica que usuarios se lles permite ver o contido do directorio. Un exemplo de configuración pode ser este:

```
AuthType Digest
AuthName myrealm
AuthDigestProvider file
AuthUserFile "/usr/local/apache/passwd/passwords"
Require user rbowen
```

Permitirlle o acceso a máis dunha persoa

As directivas anteriores so lle permiten a unha persoa (ou máis de unha se especificamos unha lista de usuarios) acceder ao directorio. En moitos casos, queremos permitir que poida acceder máis dunha persoa. Aquí e onde entra en xogo a directiva `AuthGroupFile`. Se queremos que máis dunha persoa poida acceder, necesitarase crear un ficheiro con grupos de usuarios que asocia nomes de grupos con listas de usuarios de cada grupo. O formato dese ficheiro é ben simple, e pódese crear con calquera editor. O contido pode ser coma este: GroupName: rbowen dpitts sungo rshersey Soamente consiste nunha lista de usuarios separada por espazos. Así soamente nos queda modificar os ficheiros `.htaccess` ou os bloques `<Directory>` de xeito similar ao seguinte:

```
AuthType Basic
AuthName "By Invitation Only"
AuthBasicProvider file
AuthUserFile "/usr/local/apache/passwd/passwords"
AuthGroupFile "/usr/local/apache/passwd/groups"
Require group GroupName
```

Agora, calquera que apareza no listado *GroupName*, e apareza no ficheiro de password, poderá acceder se introduce o contrasinal correcto.

Tamén hai outra maneira de permitir que múltiples usuarios poidan acceder, anque é menos específica. En vez de crear un grupo de usuarios, podemos usar a seguinte directiva:

```
Require valid-user
```

Usando isto, calquera que apareza no ficheiro de contrasinais e introduza o contrasinal correcto, poderá acceder.

Autenticación contra un directorio LDAP

Mediante o módulo **mod_authnz_ldap** podemos autenticar os usuarios contra un servidor LDAP, como pode ser un controlador de dominio de Active Directory. Imos necesitar sempre, un usuario co que facer as procuras na árbore de LDAP

```
# Parámetros básicos de configuración para a autenticación contra LDAP
AuthBasicProvider ldap
AuthType Basic

# Se falla a autenticación LDAP o usuario no se poderá autenticar con outros métodos.
AuthLDAPBindAuthoritative off
AuthName "Acceso restrinxido"

# Fase I: AUTENTICACIÓN contra LDAP
#Este usuario é o que vai a buscar aos demáis usuarios na árbore LDAP.
AuthLDAPBindDN "CN=consultas,CN=Users,DC=proba,DC=lan"
AuthLDAPBindPassword abc123..
#Indicamos o enderezo IP ou nome do servidor de LDAP/Active Directory, e a OU na que imos a buscar aos usuarios.
AuthLDAPURL "ldap://192.168.10.30/OU=usuarios,DC=proba,DC=lan?sAMAccountName?sub?(objectClass=*)"

# Fase II: AUTORIZACIÓN contra LDAP
AuthLDAPGroupAttributeIsDN on
# Autorización a calquera usuario válido
require valid-user

#Autorización aos membros do grupo "Grupo1"
#require ldap-group CN=Grupo1,OU=Grupos,DC=midominio,DC=local
```

HTTPS

TLS, ou transport layer security, e o seu predecesor SSL, secure sockets layer, son os protocolos seguros creados para envolver o tráfico normal nunha capa de seguridade. Estes protocolos premiten que o tráfico se transmita de forma segura entre as partes remotas sen a posibilidade de que sexa interceptado e lido por calquera que estea no medio. Son tamén instrumentos para validar a identidade e de dominios e servidores a través de internet establecendo canais seguros e auténticos mediante unha autoridade certificadora.

O soporte de SSL ven de serie co paquete de Apache2 de Ubuntu/Debian. So é necesario habilitar o módulo `ssl` para empregar todas as avantaxes que prové SSL no noso sistema.

Neste momento podemos pensar en host virtuais que soporten conexións con protocolos http e https, pero non é aconsellable. É mellor crear un host virtual novo que soporte SSL.

O esencial que hai que facer para que HTTPS funcione é ter unha parella de chave privada e certificado dixital. A maneira de obter chave privada e certificado non é relevante. É posible obter un certificado dixital autoasinado co seguinte comando:

```
openssl req -new -x509 -days 365 -nodes -out \
/etc/ssl/certs/meusitio.pem -keyout  /etc/ssl/private/meusitio.key
```

Se facemos iso, temos que ter en conta que calquera navegador pode non recoñecer o certificado e será necesario unha excepción.

Nesta guía podemos obter un certificado válido para empregar con Apache2.

Finalmente, o novo host virtual que emprega o protocolo https, pode ser coma este:

```
<VirtualHost *:443>
    ServerName www.exemplo.com

    SSLEngine on
    SSLCertificateFile "/etc/ssl/certs/meusitio.pem"
    SSLCertificateKeyFile "/etc/ssl/private/meusitio.key"
    ...
</VirtualHost>
```

Tamén podemos desactivar as versións 2 e 3 de SSL por estar obsoleto

```
SSLProtocol all -SSLv3
```

Máis información na documentación oficial de apache sobre encriptación.

Logs

Para manexar de xeito efectivo un servidor web, é necesario ter feedback sobre a actividade e o rendemento do servidor, así coma dos problemas que poden ocorrer. O servidor HTTP Apache, fornece unha variedade de diferentes mecanismos para rexistrar todo o que ocorre no servidor, desde a petición inicial pasando polo mapeo da URL ata a resolución final da conexión incluíndo os erros que poden ocorrer no proceso. Adicionalmente módulos de terceiros, poden fornecer capacidade de rexistro, ou inxectar entidades nos ficheiros de log, e tamén aplicacións coma programas CGI, scripts PHP ou outros poden mandar mensaxes ao log do servidor.

Calquera que escriba no directorio onde o demo httpd de Apache escribe un ficheiro de log, pode gañar acceso ao uid co que se iniciou o servidor, que normalmente non é root. NON se lle deben dar acceso aos directorio onde están almacenados os ficheiros de log sen saber cales poden ser as consecuencias.

Log de erros

O rexistro de log de erros do servidor, cuxo nome e localización se establece coa directiva `ErrorLog` é o ficheiro de log máis importante. Aquí é onde Apache manda información de diagnóstico e rexistra calquera erro que se atope no procesamento das peticións. É o primeiro lugar onde se debe mirar cando se produce un erro co inicio do servidor, ou coa operación do servidor, xa que frecuentemente contén detalles do que está mal e como poder solucionalo. O log de erros, habitualmente escríbese nun ficheiro (tipicamente error.log en sistemas Unix/Linux). En sistemas Unix é posible facer que o servidor mande erros ao *syslog* ou retransmitilos a outro programa. En Ubuntu/Debian o ficheiro máis típico é `/var/log/apache2/error.log`.

```
ErrorLog /var/log/apache2/myvirtualhost/error.log
```

Se a ruta ao ficheiro non é absoluta, entón asúmese que é relativa ao valor da directiva `ServerRoot`. O formato do log de erros defínese coa directiva `ErrorLogFormat`, que nos permite personalizar que valores se rexistran. Un formato por defecto defínese se non se establece ningún.

```
ErrorLogFormat "[%t] [%l] [pid %P] %F: %E: [client %a] %M"
```

Unha mensaxe típica é esta:

```
[Fri Sep 09 10:42:29.902022 2011] [core:error] [pid 35708:tid 4328636416] [client 72.15.99.187] File does not exist: /usr/local/apac
```

O primeiro elemento na entrada do log é a data e hora da mensaxe. A seguinte, o módulo que produciu a mensaxe (`core`, neste caso) e o nivel de severidade da mensaxe. A continuación sígueo o ID do proceso, se é aplicable o ID do fío que provocou a condición. Despois temos o enderezo que fixo a petición, e finalmente a mensaxe detallada de erro, que neste case indica que hai un ficheiro que non existe.

Poden aparecer unha gran variedade de mensaxes diferentes no log de erros. Moitos deles son similares ao exemplo anterior. O log de erros tamén pode ter información de depuración de scripts CGI. Calquera información escrita no *stderr* por un script CGI copiase directamente ao ficheiro de log de erros.

Durante a testaxe, e común monitorizar de xeito continuo o log de erros para buscar problemas. En sistemas Ubuntu/Debian, pode facerse empregando:

```
tail -f /var/log/apache2/error_log
```

Consulta nadocumentación de Apache2 por máis formatos de log de erros.

Logs de Acceso

O ficheiro de log de accesos rexistra todas as peticións procesadas polo servidor. A localización e contido do ficheiro de log de acceso, contrólase coa directiva `CustomLog`. A directiva `LogFormat` pode ser usada para simplificar a selección de contidos dos logs. Aquí descríbese como como configurar o servidor para gravar información no log de accesos.

Por suposto, almacenar información no ficheiro de log de accesos, é so o inicio da xestión de logs. O seguinte paso é analizala para obter estatísticas produtivas.

A directiva `CustomLog`, ademais do ficheiro onde se garda a información de log, pode tamén especificar un formato explícito ou un alcume ou nickname definido por unha directiva `LogFormat` anterior. Se a ruta ao ficheiro non é absoluta, asúmese que é relativa á directiva `ServerRoot`.

Exemplo:

```
# CustomLog with format nickname
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog "logs/access_log" common
# CustomLog with explicit format string
CustomLog "logs/access_log" "%h %l %u %t \"%r\" %>s %b"
```

O formato dos logs de acceso é altamente configurable. Especifícase cunha cadea que se parece ao formado expresado na sentencia `printf()` da linguaxe de programación C.

Para unha lista completa de cadeas de formatos, de `mod_log_config` consultar aquí.

Múltiples logs de acceso, poden ser creados simplemente especificando múltiples `CustomLog` nos ficheiros de configuración. Por exemplo, as seguintes directivas crean tres logs de acceso. O primeiro contén información información básica do formato de logs, mentres que o segundo e terceiro rexistran a páxina desde a que se chegou aquí, e tamén información do navegador do usuario.

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log common
CustomLog logs/referer_log "%{Referer}i -> %U"
CustomLog logs/agent_log "%{User-agent}i"
```

Virtual Hosts

Cando se executa un servidor con varios host virtuais, hai varias opcións para manexar ficheiros de log. Primeiro e posible empregar logs do mesmo xeito que un servidor individual. Simplemente poñendo as directivas de log fora das seccións `<VirtualHost>` no contexto do servidor principal, é posible facer un rexistro de todas as peticións no mesmo log se acceso. Esta técnica non permite facer unha colección de estatísticas por servidor virtual sinxela. Se as directivas `CustomLog` ou `ErrorLog` se establecen dentro de bloques `<VirtualHost>` todas as peticións ou erros dese host virtual en particular rexístranse un ficheiro específico dese host virtual. Calquera host virtual que non teña directivas de log, terá sempre o rexistro de peticións do servidor principal. Esta técnica é moi útil para un número pequeno de host virtuais, pero se o número de hosts é moi grande pode ser difícil de manexar, xa que pode crear problemas de insuficiencia de descritores de ficheiros. Para o rexistro no log de acceso, hai unha solución de compromiso. Engadindo información no host virtual á cadea do formato de logs é posible facer un log de todos os host no mesmo ficheiro de log, e posteriormente dividilos en ficheiros individuais. Por exemplo, considerando as seguintes directivas:

```
LogFormat "%v %l %u %t \"%r\" %>s %b" comonvhost
CustomLog logs/access_log comonvhost
```

O `%v` emprégase para rexistrar o nome do virtual host que se está servindo a unha petición.

Mensaxes de erro personalizadas

Anque o servidor HTTP Apache dispón de respostas xenéricas aos eventos dos códigos de estado HTTP 4xx e 5xx, estas respostas son as veces un pouco concisas, pouco informativas, e que poden intimidar aos usuarios do sitio.

Pódense fornecer respostas personalizadas, que son máis amigables, noutro idioma diferente ao inglés ou máis acorde co deseñado sitio web.

As respostas personalizadas pódense definir para cada código de estado HTTP designado coma unha condición de erro, é dicir, para códigos de estado 4xx e 5xx.

Tamén se dispón dun conxunto de valores para que os documentos coas mensaxes de erro poidan ser personalizados baseándose nos valores desas variables, empregando inclusións do lado do servidor ("Server Side Includes"). Ou tamén podemos ter condicións de erro manexada por un programa cgi ou calquera outro manexador (PHP, `mod_perl`, etc) que faga uso desas variables.

Os documentos de erros configúranse coa directiva `ErrorDocument`, que se pode empregar en contexto global, `virtualhost`, `directorio`, ou tamén en ficheiros .htaccess se a directiva `AllowOverride` ten o valor `FileInfo`.

```
ErrorDocument 500 "Sorry, our script crashed. Oh dear"
ErrorDocument 500 /cgi-bin/crash-recover
ErrorDocument 500 http://error.exemplo.com/server_error.html
ErrorDocument 404 /errors/not_found.html
ErrorDocument 401 /subscription/how_to_subscribe.html
```

A sintaxe da directiva ErrorDocument é:

```
ErrorDocument <3-digit-code> <action>
```

onde se trata a acción coma:

- Unha URL local á que redireccionarse (se a acción comeza con unha "/").
- Unha URL externa á que redireccionarse (se a acción é unha URL válida).
- Texto a amosar (en calquera dos outros casos). O texto debe estar entre comiñas dobres (") se consiste en máis dunha palabra.

mod_rewrite

O módulo **mod_rewrite** é un dos módulos máis empregados en Apache. Sirve para manipular as URL das peticións dos clientes, dándolle ao usuario a saída da páxina resultante. Por exemplo, podemos transformar URL's coma *http://www.somesite.com/widgets/blue/* en *http://www.somesite.com/widgets.php?colour=blue*. Todo isto ocorre no servidor sen que o cliente se percate.

mod_rewrite traballa da seguinte maneira: As URL's son chequeadas contra unha serie de regras. Estas regras conteñen expresións regulares para detectar un patrón determinado. Se o patrón é atopado, e se cumpren as condicións que impoña a regra, o patrón é substituído por outra cadea de texto. Este proceso continúa ata que non quedan máis regras ou se para explicitamente o proceso de búsqueda.

O uso máis común de *mod_rewrite* é na transformación de URL's "sucias" en URL's "limpas" ou amigables. O motivo é que os buscadores, non indexan páxinas con URL's sucias. O outro uso común e a redirección dunha páxina a outra.

A seguinte é unha URL considerada "sucia"

```
http://www.exemplo.com/index.php?cat=1&prod=200701
```

e a seguinte é moito máis amigable

```
http://www.exemplo.com/categoria/1/producto/200701
```

Un exemplo básico de reescritura pode ser a seguinte redirección (reenvia todas as páxinas do directorio chamadas *alice.html* á páxina *bob.html*):

```
RewriteEngine on
RewriteRule ^alice.html$ bob.html
```

O formato das regras é o seguinte:

```
RewriteEngine on
RewriteRule Patrón Substitución [Flags opcionais]
```

sendo

- *Patrón*: Unha expresión regular que será aplicada á URL actual. Se xa se aplicaron antes outras regras de reescritura, tómase a URL modificada.
- *Substitución*: Indica a cadea pola que se substitúe a URL.
- *Flags opcionais*: Expresanse entre corchetes separados por comas, e poden significar:

    ◊ *F*:Forbidden. O usuario recibirá un erro 403
    ◊ *L*:Last Rule. Non se procesarán máis regras se se procesa esta.
    ◊ *NC*: No Case. Non se distingue entre maiúsculas e minúsculas na regra.
    ◊ *R[=code]:Redirect. Rediríxese a navegación a nova URL. O usuario verá a nova URL no navegador.*

Para ver máis flags opcionais, consultar a páxina de manual do apache referente aos flags.

Moidas das veces, que se escriben regras, resulta interesante, usar parte dos elementos incluídos na expresión regular do patrón, para introducilos literalmente na cadea de substitución. Vémolo cun exemplo:

```
RewriteEngine On
RewriteRule ^([a-zA-Z]+)/([a-zA-Z]+)/([a-zA-Z0-9]+)$ /aplicacion_roupa.php?tipo=$1&sexo=$2&talla=$3 [L,NC]
```

Cada grupo encerrado entre () pode ser recordado na cadea de substitución xunto co caracter $ e seguido do número de orde que ocupe no patrón. Así no último exemplo, $1 equivale a **[a-zA-Z]+**, $2 equivale a **[a-zA-Z]+** e $3 equivale a **[a-zA-Z0-9]+**

Isto é usado con frecuencia nos xestores de contidos (CMS) máis habituais, para crear URL's limpas.

Un exemplo, pode ser o seguinte en drupal:

```
RewriteEngine On
RewriteRule ^/(.*)$ /index.php?q=$1 [L,QSA]
```

Tamén podemos poñer determinadas condicións as regras. Para facer iso, debemos poñer condicións coa directiva *RewriteCond* precedendo a *RewriteRule*. Pode haber varias condicións precedendo a unha regra. Tamén se pode recordar o texto das expresións regulares especificadas nas condicións co caracter %.

Por exemplo:

```
RewriteCond %{HTTP_HOST} (.*)
RewriteRule ^/(.*) /sites/%1/$1
```

se o nome do sitio é exemplo.com e a ruta /doc, a URL resultante será exemplo.com/sites/exemplo.com/doc

Exemplos:

- Redirixir unha URL vella a outra nova. O cliente, verá sempre a URL vella, pero cargarase a nova.

```
RewriteEngine  on
RewriteRule    ^/foo\.html$  /bar.html [PT]
```

- Redirixir unha URL vella a outra nova. O cliente, verá sempre a URL nova.

```
RewriteEngine  on
RewriteRule    ^/foo\.html$  /bar.html [R]
```

Este exemplo tamén se poderia facer coa directiva **Redirect**

```
Redirect /foo.html /bar.html
```

- Recurso movido a outro servidor. Seguimos mantendo o nome vello, pero rediriximos ao novo.

```
#con mod_rewrite
RewriteEngine on
RewriteRule   ^/docs/(.+)  http://new.example.com/docs/$1  [R,L]

#Con Redirect
Redirect /docs/ http://new.example.com/docs/
```

- Redirixir un nome de dominio a outro.

```
<VirtualHost *:80>
  ServerName undesired.example.com
  ServerAlias example.com notthis.example.com

  Redirect / http://www.example.com/
</VirtualHost>

<VirtualHost *:80>
  ServerName www.example.com
  ...
</VirtualHost>
```

- Redirixir unha porción a HTTPS

```
<If "%{SERVER_PROTOCOL} != 'HTTPS'">
Redirect /admin/ https://www.example.com/admin/
</If>
```

- Redirixir a outra URL dependendo do idioma

```
RewriteCond %{HTTP:Accept-Language} ^en [NC]
RewriteRule ^$ http://mysite.com/en/ [L,R=301]

RewriteCond %{HTTP:Accept-Language} ^de [NC]
RewriteRule ^$ http://mysite.com/de/ [L,R=301]
```

- Redirixir a outro sitio dependendo do idioma

```
RewriteCond %{HTTP:Accept-Language} ^en [NC]
RewriteRule ^$ http://en.mysite.com [L,R=301]

RewriteCond %{HTTP:Accept-Language} ^de [NC]
RewriteRule ^$ http://de.mysite.com/ [L,R=301]
```

Referencias externas

1. ? O nome do ficheiro que contén o host virtual por defecto, é 000-default.conf por esta mesma razón, para ser servido no caso de que ningún outro encaixe.
2. ? Posibles valores da directiva `AllowOverride`[1]
3. ? Autenticación