APACHE 2

Documento original

“Manual de Supervivencia del Administrador de Apache”

Autor: Miguel Jaque Barbero (10 de noviembre de 2006)

Actualizado: Antonio Boronat Pérez (noviembre de 2017 noviembre de 2021)

SMX2 -Servicios en Red

Índice

Introducción

El servidor web Apache es el más popular en Internet y por tanto el que usaremos para implantar este servicio en un servidor de prueba. La documentación de guía la encontrarás en:

http://www.ite.educacion.es/formacion/materiales/85/cd/linux/m3/index.html

o en el documento Servidor-web-Apache.pdf

Manual de referencia de Apache2 https://httpd.apache.org/docs/2.4/mod/quickreference.html

Puede ser interesante la instalación de Apache junto con PHP y MySQL para crear webs dinámicas. Una forma muy sencilla es mediante un empaquetado XAMPP, y otra puede ser ejecutando las órdenes "apt-get" que se muestran en el enlace recomendado para instalar PowerAdmin.

Instalación

Con Debian / Ubuntu, la instalación de Apache es muy sencilla. Basta con ejecutar la orden:

```
# apt-get install apache2
```

Y tendrás el servidor web instalado.

Configuración

La configuración de Debian / Ubuntu, por defecto, incluye:

Los archivos de configuración de Apache en `/etc/apache2`

El directorio raíz para los documentos en `/var/www`

Arranque y Parada

Apache se ejecuta como un servicio en segundo plano (daemon). Para arrancarlo basta con ejecutar una de las órdenes:

```
# service apache2 start   o   # systemctl start apache2
```

Y para detenerlo: `# service apache2 stop` o `# systemctl stop apache2`

Echándole un vistazo al script (/etc/init.d/apache2) verás que el comando que realmente arranca y detiene Apache es apache2ctl. Consultando la página de manual (man apache2ctl) verás que tiene muchas más opciones que start y stop.

También podemos reiniciar el servicio si ya está en funcionamiento y deseamos que cargue nuevos módulos por ejemplo:

```
# service apache2 restart   o   # systemctl restart apache2
```

Si hemos habilitado nuevos contenidos para que sean accesibles podemos recargarlos con:

```
# service apache2 reload   o   # systemctl reload apache2
```

Finalmente podemos ver si el servidor está en funcionamiento con:

```
# service apache2 status   o   # systemctl status apache2
```

Podemos validar los ficheros de configuración con: `# apachectl configtest`

Procesos

Aunque Apache se ejecuta inicialmente como *root*, pues en un sistema Linux sólo *root* está autorizado a abrir sockets en puertos por debajo del 1000, y Apache escucha, por defecto, en el tradicional **puerto 80 del protocolo HTTP**.

Al arrancar Apache el proceso de *root* arrancará otros procesos hijos que ya se ejecutan con otro usuario y grupo menos peligroso que *root*. Por defecto estos procesos hijos se ejecutan con el usuario ***www-data*** y el grupo ***www-data***. Asegúrate de que este usuario tiene permiso para leer los documentos que debe servir Apache.

El proceso inicial no atenderá ninguna petición de cliente para evitar así problemas de seguridad. En su lugar, serán los procesos hijos quienes se encargan de servir las páginas.

Puedes comprobar todo esto con la orden *ps -aux* para ver los procesos en ejecución en tu sistema.

Servidor Básico

El fichero de configuración de Apache que Debian / Ubuntu instala por defecto (*/etc/apache2/apache2.conf*), aunque es muy bueno, es también muy complicado y no nos sirve para explicar los conceptos más elementales de Apache, además en las últimas versiones algunos parámetros de la configuración básica han pasado a establecerse en ficheros de configuración, en la configuración de los módulos que las aplican o en los ficheros de configuración de las Web.

Así que vamos a cambiarlo (guardando antes una copia de seguridad) por este otro:

```
#Fichero de Configuración SENCILLO (/etc/apache2/apache2.conf)
#Directorio con los ficheros de configuración de Apache
ServerRoot "/etc/apache2"
#Directorio raíz de los documentos publicados
DocumentRoot "/var/www"
#Fichero en el que se guarda el número del proceso Apache
PidFile /var/run/apache2/apache2.pid
#Usuario y grupo con los que se ejecutará Apache
User www-data
Group www-data
#Fichero de log para los errores
ErrorLog /var/log/apache2/error.log
#Incluye los módulos habilitados en nuestro servidor
IncludeOptional mods-enabled/*.load
IncludeOptional mods-enabled/*.conf
IncludeOptional conf-enabled/*.conf
#Incluye la config. de puertos en los que escucha el servidor
Include ports.conf
#Incluye las Web  habilitadas en nuestro servidor
IncludeOptional sites-enabled/*.conf
Archivo /etc/apache2/mods-enabled/dir.conf:
#Lista de ficheros que pueden servir como índices de directorio
DirectoryIndex index.html index.htm
mods-enabled/mime.conf:
```

```
#Fichero con la lista de tipos mime
TypesConfig /etc/mime.types
```

Archivo /etc/apache2/sites-available/nombre_web.conf:

```
#Nombre con el que el Servidor se conoce a sí mismo
ServerName www.depinfo.iesjc
#Directorio raíz de los documentos publicados en esta web
DocumentRoot /var/www/depinfo
ports.conf:
#Puerto en el que escuchará Apache
Listen 80
```

En el contenido anterior:

1. Los ficheros de configuración son de texto plano.
2. Cada línea contiene una directiva de configuración.
3. Las líneas que empiezan por # son comentarios.

Directivas Básicas

ServerName

Establece el nombre con el que el servidor se conoce a sí mismo. Esta directiva se utiliza para las redirecciones. Es decir, cuando Apache le tiene que indicar al cliente (el navegador) otra dirección a la que tiene que ir.

Es importante que ese nombre se pueda resolver por DNS. De lo contrario el cliente no podrá acceder a la página redireccionada.

ServerRoot

Establece el directorio en el que se encuentran los ficheros de configuración de Apache. Y si Apache no conoce el directorio de configuración, ¿cómo puede acceder a éste fichero de configuración? La respuesta a esta paradoja es que Apache puede arrancarse pasándole como parámetro un fichero de configuración. Pero puede ocurrir que en el fichero de configuración hagamos referencia a otros ficheros que deben incluirse.

DocumentRoot

Establece el directorio en el que se encuentran los ficheros que Apache servirá a los clientes (páginas HTML, scripts PHP, CGIs, etc.). Cada servidor virtual tiene el su directorio definido con esta directiva.

PidFile

Establece el fichero en el que se guardará el número del proceso de Apache. Este fichero es el que se lee cuando hay que parar/matar el proceso.

User

Establece el usuario con el que se ejecutará Apache. Bueno, realmente el proceso Apache se ejecuta como root, porque normalmente tiene que abrir un socket de escucha para el puerto 80 y, en POSIX, sólo root puede abrir puertos por debajo del 1000. Sin embargo, tras arrancar ese primer proceso como root, Apache crea varios procesos hijos que se ejecutan con el usuario establecido en esta directiva (www-data en Debian). Estos procesos serán quienes realmente atenderán las peticiones de los usuarios.

Group

Establece el grupo con el que se ejecutará Apache (sus procesos de escucha).

ErrorLog

Establece el fichero de log de errores de Apache. En este fichero se registrarán los fallos de acceso, intentos de acceso a recursos sin autorización, páginas no encontradas, etc.

Listen

Establece la dirección IP y el puerto en el que escuchará Apache. Por defecto, Apache escuchará en todas las direcciones IP habilitadas en la máquina. Se define en el fichero ports.conf.

DirectoryIndex

Establece los nombres de ficheros que servirán como índices al acceder a un directorio sin indicar ningún recurso concreto. Así, si un usuario solicita www.miweb.org/dir1/ Apache buscará en ese directorio ficheros con el nombre indicado en esta directiva para servirlos.

TypesConfig

Establece el fichero con la lista de tipos Mime. Los tipos Mime constituyen un estándar que relaciona tipos de ficheros con sus extensiones y le permiten a Apache informar al navegador del tipo de fichero que le está entregando. Así, el navegador decide cómo presentarlo (mostrando una página web, ejecutando un plugin, guardándolo en disco...)

Activar espacio web para usuarios del sistema

Apache dispone de un módulo que permite a los usuarios del sistema tener su página web en el servidor, pero con la ventaja que el espacio disponible para albergar esta página se encuentra dentro del directorio personal de cada usuario, de forma que cada uno puede administrarse su propia página sin tener que tener más privilegios.

Para habilitar el módulo se tiene que ejecutar:

```
$sudo a2enmod userdir
```

Una vez el módulo está habilitado, cualquier usuario puede poner su página web dentro del directorio public_html en su directorio personal (/home/nombre_usuario/public_html).

Normalmente esta carpeta no estará creada, pero puede crearla el propio usuario, sólo tiene que tener en cuenta el darle permisos para que sea accesible. Lo mejor sería darle los permisos 755 para que el usuario pudiera hacer modificaciones y el resto sólo pudiera acceder para ver los objetos almacenados.

Los objetos que hay dentro de esta carpeta y que formarán la página web (documentos html, imágenes, vídeos...) deberían tener los permisos 644 porqué, a diferencia del directorio que los contiene, no necesitan el permiso de ejecución. Evidentemente si se tienen subdirectorios también necesitarán este permiso x.

Como el resto de ubicaciones, Apache intentará mostrar la página llamada index, así que si esta pagina existe, para mostrarla, sólo se tiene que poner en el navegador la URL:

```
http://direccion_servidor/~nombre_usuario
```

Servidor Configurado por Directorios

En el servidor sencillo que acabamos de ver, la configuración es la misma para todos los documentos que estemos publicando. En algunos casos puede interesarnos tener configuraciones diferentes para distintos directorios e incluso para distintos ficheros. Para eso, podemos utilizar un fichero de configuración como el siguiente:

```
#Fichero de Configuración POR BLOQUES
ServerName "www.depinfo.iesjc"
ServerRoot "/etc/apache2"
DocumentRoot "/var/www"
PidFile /var/run/apache2/apache2.pid
User www-data
```

```
Group www-data
ErrorLog /var/log/apache2/error.log
Listen 80
TypesConfig /etc/mime.types

#Establecemos la configuración de cada directorio
<Directory "/var/www">
      #No permitimos los índices automáticos en ningún sitio (salvo
      #los que permitamos explícitamente)
      Options -Indexes
</Directory>
<Directory "/var/www/descargas">
      #En este directorio, permitimos índices automáticos, pero no
      #permitimos enlaces simbólicos
      Options +Indexes -FollowSymLinks
      #No permitimos a nadie acceder a los ficheros .htaccess de este
      #directorio
      <Files .htaccess>
            order allow,deny
            deny from all
      </Files>
</Directory>
<Location /server-status>
      #En esta dirección, mostramos información sobre el estado del
      #servidor
      SetHandler server-status
</Location>
```

Directivas para la Configuración por Bloques

Al aplicar configuraciones diferentes por directorio, por fichero o por localización, decimos que estamos aplicando “Configuración por Bloques”. Hemos utilizado las siguientes directivas:

<Directory>

Indica que el bloque de configuración que abarca (entre <Directory> y </Directory>) se aplica al directorio indicado y a sus subdirectorios. También hay una directiva <DirectoryMatch> que permite utilizar expresiones regulares. Así, un mismo bloque de configuración puede aplicarse a varios directorios.

Options

Modifica las opciones que se aplican a un directorio. Las opciones se añaden (con +) o se quitan (con -) respecto a las que en ese momento se aplican sobre el directorio.

Mediante esta directiva pueden añadirse/quitarse las siguientes opciones:

All, ExecCGI, FollowSymLinks, Includes, IncludesNOEXEC, Indexes, Multiviews y SymLinksIfOwnerMatch.

<File>

Indica que el bloque de configuración que abarca (entre <File> y </File>) se aplicará a los ficheros cuyo nombre coincida con el indicado.

También hay una directiva <FileMatch> que permite utilizar expresiones regulares. Así, un mismo bloque de configuración puede aplicarse a varios ficheros.

<Location>

Indica que el bloque de configuración que abarca (entre <Location> y </Location>) se aplicará a las localizaciones que coincidan con la indicada. La “localización” es la dirección que solicita el cliente. Fíjate que no es lo mismo que el sistema de ficheros (sobre el que trabajan <Directory> y <File>). Utilizando redirecciones y alias es posible que un cliente solicite una determinada “localización” y que la página que se le entregue tenga un *path* completamente distinto al que él especificó. También hay una directiva <LocationMatch> que permite utilizar expresiones regulares. Así, un mismo bloque de configuración puede aplicarse a varias localizaciones.

SetHandler

Establece el manejador que se utilizará para atender las peticiones a un directorio, a un tipo de ficheros o a una localización. Un "handler" es una representación interna de Apache de una acción que se va a ejecutar cuando hay una llamada a un fichero. Generalmente, los ficheros tienen handlers implícitos, basados en el tipo de fichero de que se trata. Normalmente, todos los ficheros son simplemente servidos por el servidor, pero algunos tipos de ficheros se tratan de forma diferente. Los posibles handlers son: default-handler, send-as-is, cgi-script, imap-file, server-info, server-status y type-map.

Servidores Virtuales

Tener todo un servidor Apache para atender sólo un sitio web es una pérdida de recursos. Apache es capaz de atender, desde una sola máquina a todo un conjunto de sitios web. Es decir, podemos servir al mismo tiempo peticiones para www.depinfo.iesjc , www.depadm.iesjc, profes.iesjc , alumnes.iesjc...
Esto se hace utilizando “Servidores virtuales”.

Opciones de Arquitectura

Para atender varios sitios web primero debemos conseguir que las peticiones de los clientes para esas URLs lleguen hasta nuestro servidor. Este es un problema de la configuración de DNS que, como administradores de Apache, no nos corresponde. Pero, si lo que quieres es hacer pruebas, puedes modificar el fichero hosts de tus clientes (/etc/hosts en Debian /Ubuntu) y mapear en él los sitios web con las direcciones IP que escuchen tu/s tarjetas de red.
A continuación se muestra un ejemplo de /etc/hosts:
127.0.0.1 localhost
192.168.4.140 www.smx.net
192.168.4.140 www.servicios.org
Otro opción, sería que el profesor configure el servicio DHCP + DNS del servidor de aula de forma que las MV en modo puente puedan configurarse automáticamente y su nombre se registre en el servicio de DNS, así se podrá acceder a las Web de los compañeros de clase.
En el caso de DNSMASQ, se deberá configurar /etc/dnsmasq.conf:

- Descomentar la línea de la parte DHCP: *dhcp-ignore-names*, de forma que toma los nombres de los sistemas cliente y los usa para añadirlos a DNS.
- En segundo lugar si deseamos que los sistemas cliente disponga de más de un nombre, debemos añadir también en /etc/dnsmasq.conf las sentencias CNAME para cada cliente de forma que nos permita crear al menos dos servidores virtuales de Apache usando el nombre de host. La sintaxis a emplear por cada sistema cliente será: *cname = nom_alias,nom_sistema* sólo un álias por línea . Por ejemplo, *cname=toniserver2,toniserver*

Una vez hecho esto, tendremos varias opciones para configurar nuestros hosts virtuales. En primer lugar, podemos establecer varias direcciones IP y asignar una a cada host virtual. Esto lo llamaremos “servidores virtuales por IP”.

En segundo lugar, podemos establecer distintos puertos de escucha para sitio web. Sí, esto es algo complicado porque implica decirle al cliente a qué puerto debe dirigirse. Pero puede ser útil para redes internas. Esto lo llamaremos “Servidores virtuales por Puerto”.

Y, en tercer lugar, el protocolo HTTP 1.1 permite al cliente indicarnos, mediante una cabecera, el nombre del sitio web al que quiere acceder. Como ya apenas quedan navegadores que no soporten el protocolo HTTP 1.1, está es la opción más utilizada.

Y por último, podemos hacer una mezcla con todas estas opciones.

Servidores Virtuales por IP

Supongamos en primer lugar, que nuestro servidor atiende dos direcciones IP y que asignamos cada una de ellas a un sitio web.

El fichero de configuración sería el siguiente y estaría ubicado en */etc/apache2/sites-available/nombre_web.conf* :

```
#Fichero de Configuración para HOSTS VIRTUALES POR IP
ServerRoot "/etc/apache2"
DocumentRoot "/var/www"
PidFile /var/run/apache2/apache2.pid
User www-data
Group www-data
ErrorLog /var/log/apache2/error.log
TypesConfig /etc/mime.types

#Establecemos la configuración para cada Host Virtual

<VirtualHost 192.168.2.166>
      #Cada uno tiene su nombre
      ServerName www.depinfo.iesjc
      DocumentRoot /var/www/depinfo
      #Dividimos los ficheros de log
```

```
        ErrorLog /tmp/www1_ERROR.log
        TransferLog /tmp/www1_ACCESS.log
</VirtualHost>

<VirtualHost 192.168.2.200>
        ServerName www2.depinfo.iesjc
        DocumentRoot /var/www/depinfo2
        ErrorLog /tmp/www2_ERROR.log
        TransferLog /tmp/www2_ACCESS.log
</VirtualHost>
```

Servidores Virtuales por Puerto

Hagamos lo mismo, pero en una de las direcciones IP, utilicemos dos puertos TCP/IP distintos para atender dos sitios diferentes.

En el fichero */etc/apache2/ports.conf* debemos añadir: `Listen 8080`

para que el servidor también escuche en este nuevo puerto.

El fichero de configuración sería el siguiente, con la misma ubicación que en el caso anterior:

```
#Fichero de Configuración para HOSTS VIRTUALES POR IP Y PUERTO
ServerRoot "/etc/apache2"
DocumentRoot "/var/www"
PidFile /var/run/apache2/apache2.pid
User www-data
Group www-data
ErrorLog /var/log/apache2/error.log
TypesConfig /etc/mime.types

#Indicamos el puerto de cada sitio web
<VirtualHost 192.168.2.166:80>
        ServerName www.depinfo.iesjc
        DocumentRoot /var/www/depinfo
        ErrorLog /tmp/www1_ERROR.log
```

```
        TransferLog /tmp/www1_ACCESS.log
</VirtualHost>

<VirtualHost 192.168.2.166:8080>
        ServerName profes.depinfo.iesjc
        DocumentRoot /var/www/profes
        ErrorLog /tmp/profes_ERROR.log
        TransferLog /tmp/profes_ACCESS.log
</VirtualHost>

<VirtualHost 192.168.2.200>
        ServerName www2.depinfo.iesjc
        DocumentRoot /var/www/depinfo2
        ErrorLog /tmp/www2_ERROR.log
        TransferLog /tmp/www2_ACCESS.log
</VirtualHost>
```

Servidores Virtuales por Nombre

Y ahora, configuremos el servidor utilizando los nombres de cada sitio web:

El fichero de configuración sería el siguiente, con la misma ubicación que en el caso anterior:

```
#Fichero de Configuración para HOSTS VIRTUALES POR NOMBRE
ServerRoot "/etc/apache2"
DocumentRoot "/var/www"
PidFile /var/run/apache2/apache2.pid
User www-data
Group www-data
ErrorLog /var/log/apache2/error.log
TypesConfig /etc/mime.types

#La configuración de hosts virtuales por nombre se aplica a las
#peticiones recibidas en esta IP
NameVirtualHost 192.168.2.166
#Configuramos haciendo referencia a cada sitio por su nombre
<VirtualHost www.depinfo.iesjc>
     ServerName www.depinfo.iesjc
     DocumentRoot/var/www/depinfo
     ErrorLog /tmp/www1_ERROR.log
     TransferLog /tmp/www1_ACCESS.log
</VirtualHost>

<VirtualHost www2.depinfo.iesjc>
     ServerName www2.depinfo.iesjc
     DocumentRoot /var/www/depinfo2
     ErrorLog /tmp/www2_ERROR.log
     TransferLog /tmp/www2_ACCESS.log
</VirtualHost>
```

Servidores Virtuales por Nombre, IP y Puerto

Para completar el ejemplo, configuramos el servidor utilizando todas las opciones:

```
#Fichero de Configuración para Hosts Virtuales
ServerRoot "/etc/apache2"
DocumentRoot "/var/www"
PidFile /var/run/apache2/apache2.pid
User www-data
Group www-data
ErrorLog /var/log/apache2/error.log
TypesConfig /etc/mime.types
NameVirtualHost 192.168.2.166
<VirtualHost www.depinfp.iesjc>
     ServerName www.depinfo.iesjc
     DocumentRoot /var/www/depinfo
     ErrorLog /tmp/depinfo_ERROR.log
     TransferLog /tmp/depinfo_ACCESS.log
</VirtualHost>

<VirtualHost www2.depinfo.iesjc>
     ServerName www2.depinfo.iesjc
     DocumentRoot /var/www/depinfo2
     ErrorLog /tmp/depinfo2_ERROR.log
     TransferLog /tmp/depinfo2_ACCESS.log
</VirtualHost>

<VirtualHost 192.168.2.200:80>
     ServerName www2.depinfo.iesjc
     DocumentRoot /var/www/depinfo2
     ErrorLog /tmp/depinfo2_ERROR.log
     TransferLog /tmp/depinfo2_ACCESS.log
</VirtualHost>
```

```
<VirtualHost 192.168.2.200:8080>
     ServerName profes.depinfo.iesjc
     DocumentRoot /var/www/profes
     ErrorLog /tmp/profes_intranet_ERROR.log
     TransferLog /tmp/profes_intranet_ACCESS.log
</VirtualHost>
```

Servidores Virtuales Dinámicos

Pero... ¿qué pasa si tenemos que configurar decenas, cientos o incluso miles de sitios web tal y como ocurre en un ISP? Para eso tenemos la opción de Hosts Virtuales Dinámicos.

Debemos cargar el módulo: `# a2enmod vhost_alias`

Veamos un ejemplo:

```
#Fichero de Configuración para Hosts Virtuales Dinámicos
ServerName "iesjc"
ServerRoot "/etc/apache2"
DocumentRoot "/var/www"
PidFile /var/run/apache2/apache2.pid
User www-data
Group www-data
ErrorLog /var/log/apache2/error.log
TypesConfig /etc/mime.types
ErrorLog /tmp/iesjc_ERROR.log
#Le indicamos a Apache que utilice como nombre (ServerName) el de la petición recibida
UseCanonicalName Off
#Establecemos el DocumentRoot para cada sitio web que atendemos.
VirtualDocumentRoot /var/www/%1.0
```

Pero, si tenemos muchos sitios web, también puede ser útil que cada uno de ellos tenga su propio fichero de configuración (ya le daremos permiso a sus administradores para que lo gestionen ellos

mismos). Esto podemos conseguirlo cargando todos los ficheros de configuración de los sitios virtuales desde un directorio.

Podríamos hacerlo utilizando la siguiente línea en el fichero de configuración de Apache:

```
Include /etc/apache2/sites-enabled/[^.#]*
```

Esta directiva incluye en el fichero de configuración todos los ficheros que encuentre en el directorio indicado.

Directivas para la Configuración de Servidores Virtuales

Veamos las directivas que hemos utilizado:

<VirtualHost>

Indica que el bloque de configuración que abarca (entre <VirtualHost>
y </VirtualHost>) se aplica al sitio web indicado.

Cada host virtual se puede identificar por IP, IP:Puerto o por nombre.

NameVirtualHost

Establece la dirección IP sobre la que se configurarán hosts virtuales.

VirtualDocumentRoot

Establece dinámicamente la raíz de los documentos (DocumentRoot) para los hosts virtuales. Para establecer el nombre del directorio usa % que hace referencia al nombre que estamos para acceder a nuestro servidor y el número que va con el % indica que parte de este nombre se usa para identificar su directorio:

%0 se usa todo el nombre, por tanto si nuestra web se identifica por www.depinfo.iesjc, el directorio raíz será /var/www/www.depinfo.iesjc/

%1 toma la primera parte del nombre, www y el directorio raíz estaría en /var/www/www

%2 toma la segunda parte.

La lista siguiente muestra el funcionamiento ya que se pueden realizar combinaciones:

%% inserta un %

%p inserta el número de port del servidor virtual

%N.M donde N indica la parte del nombre y M indica cuantos caracteres se toman de él.

El valor para N:

0 el nombre completo → %0

1 la primera parte → %1

2 la segunda parte → %2

-1 la última parte → %-1

-2 la penúltima parte → %-2

2+ la segunda y todas las partes subsecuentes → %2+

-2+ la penúltima y todas las partes precedentes → %-2+

1+ and -1+ lo mismo que cero 0

UseCanonicalName

Establece de qué forma conocerá Apache su propio nombre (para las redirecciones).

TransferLog

Establece el fichero de log de acceso de Apache.En este fichero se registrarán los accesos a las páginas servidas por Apache.

Acceso seguro con OpenSSL (HTTPS)

Las conexiones seguras se consiguen usando el protocolo **https**. Este protocolo es similar a http, pero utiliza algoritmos de cifrado SSL (en nuestro caso los proporcionados por el paquete OpenSSL).

Para conseguir que Apache nos envíe páginas cifradas se tiene que hacer una sencilla configuración, que a grandes rasgos consiste en generar el certificado que se presentará al cliente cuanto se conecte (en un servidor "profesional" este certificado se pediría a una agencia certificadora) y crear un servidor virtual que escuche en el puerto 443 (por defecto del protocolo https) y que presente el certificado.

Generación de certificados

En primer lugar se tiene que crear el directorio donde se guardarán los certificados. Este directorio se creará donde esté el resto de configuración de Apache, aunque también es habitual que se almacenen en los directorios que disponen las distribuciones de Linux (En Debian / Ubuntu están en */etc/ssl/certs/* para los certificados y */etc/ssl/private/* para las claves):

```
$sudo mkdir /etc/apache2/ssl
```

A continuación se generará el certificado con la orden:

```
#/usr/sbin/make-ssl-cert /usr/share/ssl-cert/ssleay.cnf /etc/apache2/ssl/apache.pem
```

el certificado estará en el fichero `/etc/apache2/ssl/apache.pem`. Este fichero contiene las claves necesarias para el intercambio de información cifrada, así que sólo falta la configuración del servidor virtual.

Servidor Virtual Seguro

Para el correcto funcionamiento sólo nos falta configurar un servidor virtual que escuche en el puerto 443 y presente el certificado que hemos generado para el cifrado de la conexión. Para eso sólo se tiene que seguir los mismos pasos que se han seguido antes para crear el servidor virtual y añadir algunas líneas:

```
<VirtualHost *:443>
ServerAdmin webmaster@localhost
ServerName www.smx.org
SSLEngine On
SSLCertificateFile /etc/apache2/ssl/apache.pem
DocumentRoot /var/www
<Directory />
     Options FollowSymLinks
     AllowOverride None
</Directory>
<Directory /var/www/>
     Options Indexes FollowSymLinks MultiViews
     AllowOverride None
</Directory>
ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
<Directory "/usr/lib/cgi-bin">
     AllowOverride None
     Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
</Directory>
ErrorLog /var/log/apache2/error.log
#Posible valores: debug, info, notice, warn, erro, alert, emerg.
LogLevel warn
CustomLog /var/log/apache2/access.log combined
Alias /doc/ "/usr/share/doc/"
(...)
</VirtualHost>
```

Básicamente, la única diferencia con los otros servidores virtuales que se ha creado es el uso del puerto 443 (primera línea en rojo), que activa el motor SSL y se indica donde está el fichero con el certificado (segunda y tercera) y finalmente que al directorio se pide el uso de SSL.

En el fichero `/etc/apache2/ports.conf` se debe tener la directiva `Listen 443` para que funcione correctamente, pero en circunstancias normales esta directiva ya estará puesta desde la instalación de Apache y activada en el momento que se carga el módulo SSL.

Activando el Acceso Seguro

Cuando ya está todo el resto de configuración hecha sólo faltará activar el módulo SSL, por eso se ejecutará la orden:

```
$sudo a2enmod ssl
```

Y se reinicia el Apache para que cargue toda la configuración nueva:

```
$sudo service apache2 restart
```

Como comprobaremos al cargar nuestra Web con su certificado en el navegador, nos indica que el sitio Web no es seguro, que no se confía en el certificado asociado. Esto se debe a que el certificado no ha sido emitido por una Entidad de Certificación reconocida, que el certificado raíz de la Entidad de Certificación no está instalado en el navegador y por tanto no debemos confiar en él. Veamos a continuación otra forma de generar certificados a partir de un Certificado de Autoridad (CA), el cual podremos instalar en los navegadores cliente y comprobaremos que ahora sí que confia en los certificados firmados con niesror CA.

Sigue las instrucciones siguientes:

```
Clave para el certificado CA con contraseña:
#openssl genrsa -out clave_privada_ca.pem -des3 2048
Operación genrsa --> Genera una clave privada de tipo RSA
-des3 2048 --> Cifrado Triple-DES con 2048 bits de longitud, podemos usar cifreado Blowfish en su lugar.


Crea el certificado de la Autoridad de Certificación:
#openssl req -new -x509 -key clave_privada_ca.pem -out cacert.pem -days 365
Operación req → gestiones para las solicitudes de firma de certificados x509
      $ man req: (lo que pone el en manual)
      -new --> Genera una nueva solicitud de certificado
      -x509 --> Gestión de certificados x509
```

-key --> -Clave a usar

-out --> Fichero de salida con el certificado

-days --> Periodo de val¡dez del certificado, en este caso 365 dias.

Clave para el certificado de nuestro servidor:

#**openssl genrsa -out clave_privada_cert_serv.pem 2048**

Crea la clave, pero como no tiene la opción de cifrado, no pide contraseña.

Petición de certificado de servidor sin firmar y sin contraseña para que no se pare el servicio al arrancar en espera de la clave.

#**openssl req -new -nodes -key clave_privada_cert_serv.pem -out peticion_cert_serv.pem -days 365**

Con -nodes → Con esta opción se indica que si se crea una clave privada ésta no debe encriptarse.

Crear este fichero: **config_cert.txt**:

basicConstraints=critical,CA:FALSE

extendedKeyUsage=serverAuth,emailProtection

La primera línea indica que no se pueden firmar otros certificados con éste. La segunda que lo usaremos como certificado de servidor.

Firma del certificado de servidor con el certificado CA:

#**openssl x509 -CA cacert.pem -CAkey clave_privada_ca.pem -req -in peticion_cert_serv.pem -days 365 -sha512 -CAcreateserial -out cert_serv_firmado.pem -extfile config_cert.txt**

Operación x509 → Gestión de datos de certificados x509

$ man x509: (lo que pone en el manual)

-CA → Especifica el certificado de autoridad que se usar para firmar.

-CAkey → Especifica la clave privada usada para firmar el certificado

-req → Indica que se va a usar una petición de certificado en lugar de un certificado

`-in → Indica el fichero de entrada con el certificado a tratar`

`-sha512 → Función hash usada para crear el resumen del certificado para comprobar su integridad.`

`-CAcreateserial → Crea un fichero con el número de serie para CA, es necesario si se usa la opción -CA.`

`-extfile → Especifica un fichero con extensiones usadas en el certificado`

Ahora el fichero con el certificado CA (cacert.pem) lo copiamos a los sistemas cliente y en el navegador en la configuración lo añadiremos en el apartado de Certificados en la zona de "Entidades Emisoras" en Chrome o "Autoridades" en Firefox. Si ahora cargamos nuestra Web la conexión aparecerá como segura ya que hemos indicado que confíe en este CA.

Pero al realizar este paso en navegadores como *Brave* o *Chrome*, nos muestran un error de certificado en el que dicen que CN no es válido. Esto se debe a que esperan un certificado en el que los nombres del servidor aparezcan en una lista de nombres alternativos y que aunque sean completamente diferentes hacen referencia al mismo dominio, como una lista de alias. Para conseguir esto usaremos un certificado con soporte SAN (Subject Alternative Name – Nombres Alternativos del objetivo del certificado). Los pasos son básicamente los que ya hemos visto cambiando solo las órdenes para generar la petición de certificado del servidor y la firma de este para obtener el certificado propiamente dicho. Veamos los pasos.

En primer lugar creamos un fichero con toda la configuración y datos del certificado a generar. Aquí pondremos la información relativa al servidor para el que creamos el certificado, los datos que en el ejemplo anterior introducíamos desde la línea de órdenes, también ponemos la configuración del certificado, que no podremos firmar otros certificados con él y que lo usaremos para autenticar servidores y finalmente también añadiremos los nombres alternativos que podemos usar en la URL de los navegadores para acceder a este servidor. Veamos un ejemplo de este fichero, llamado *san.txt*:

```
[req]
distinguished_name = req_distinguished_name
req_extensions = v3_req
prompt = no

[req_distinguished_name]
C = ES
```

```
ST = Castellon
L = Benicarlo
O = iesjc
OU = depinfo
CN = ubnt2004

[v3_req]
basicConstraints=critical,CA:FALSE
extendedKeyUsage = serverAuth
subjectAltName = @alt_names

[alt_names]
DNS.1 = ubnt2004
DNS.2 = ubnt2004.depinfo
DNS.3 = ubnt2004.depinfo.iesjc
```

Ahora las órdenes en las que usamos el fichero anterior para generar la petición de certificado primero y la firma de la petición con el certificado CA para obtener el certificado a continuación son las siguientes, las órdenes previas a estas son las mismas que hemos visto anteriormente:

```
# openssl req -sha512 -new -out peticion_cert_serv.csr -key
clave_privada_cert_serv.pem -config san.txt

# openssl x509 -days 365 -req -sha512 -in peticion_cert_serv.csr -
CAcreateserial -CA cacert.pem -CAkey clave_privada_ca.pem -out
cert_serv_firmado.crt -extensions v3_req -extfile san.txt
```

Una vez tenemos el certificado cert_serv_firmado.crt y su clave los ponemos en la configuración del servidor virtual como hemos visto antes

Autenticación

En muchas ocasiones querremos restringir el acceso a algunos recursos (páginas). Con Apache podemos establecer mecanismos de usuario y contraseña, para limitar el acceso. Además, los usuarios pueden incluirse en grupos y establecer permisos para estos últimos. ¡PERO CUIDADO! la transmisión de información tiene una encriptación muy débil. No montes ninguno de los mecanismos que veremos a continuación si no utilizas también SSL. Cualquier sniffer podrá fácilmente robarte las contraseñas.

¿Cómo funciona?

Una vez que establecemos que un determinado recurso de nuestro servidor requiere autenticación, al intentar acceder a él Apache devuelve al cliente un mensaje 401 (Authentication Required). Normalmente, en los navegadores actuales, si existe alguna pareja usuario/clave guardada para el recurso la enviarán al servidor de forma automática. Si no es así, preguntarán al usuario. Desde el lado del servidor, hay dos mecanismos de autenticación.

Autenticación Básica

Con este mecanismo la contraseña es enviada en claro por la red, o como mucho, con una codificación en base 64 (fácil, fácil...).

Debemos habilitar el módulo: **a2enmod auth_basic**

Veamos un ejemplo:

```
<VirtualHost profes.depinfo.iesc>
     ServerName profes.depinfo.iesjc
     DocumentRoot /var/www/profes
     ErrorLog /tmp/profes_ERROR.log
     TransferLog /tmp/profes_ACCESS.log
<Directory /var/www/profes>
     #Establecemos el tipo de Control de Acceso que se utilizará
     #para acceder a este directorio
     AuthType Basic
```

```
    #Le damos un nombre al entorno privado que protegemos
    AuthName "Información Profes"
    #Fichero con la información de los usuarios autorizados y
    #sus contraseñas (encriptadas).
    AuthUserFile /etc/apache2/passwd/passwd_profes
    #Fichero con la información de los grupos y sus usuarios
    #miembros.
    AuthGroupFile /etc/apache2/passwd/grupo_profes
    #Establecemos el nivel de seguridad para este directorio.
    require valid-user
    #Otros valores posibles son:
    # require user usuario1 usuario2 ... lista con los
    # usuarios autorizados
    # require group grupo1 grupo2 ... lista con los
    # grupos autorizados
    # require user usu1 usu2 group gr1 gr2 ... una
    # combinación de ambos
</Directory>
</VirtualHost>
```

Ficheros de Usuarios y Grupos

La información con los usuarios y las contraseñas se guarda en el fichero señalado por la directiva AuthUserFile. Este fichero tendrá algo parecido a:

usu1:$apr1$Ym2fv...$kGAt2g1y6dOSt404Xfka2.

usu2:$apr1$0D0nG/..$cv8eAB0Fykj/PnoIP/0x6.

Y, naturalmente, debe estar donde ningún usuario del servidor web pueda llegar a leerlo.

Como poca gente es capaz de escribir directamente la contraseña encriptada, Apache incluye una aplicación para gestionar este fichero. Se trata de htpasswd.

Para crear un fichero como el del ejemplo, tendremos que ejecutar las siguientes órdenes:

```
# htpasswd -cm /etc/apache2/passwd/passwd_profes    usu1
# htpasswd -m /etc/apache2/passwd/passwd_profes      usu2
```

El primero crea el fichero y le añade el usuario usu1. La opción **c** indica que se debe crear un nuevo fichero y la opción **m** que la contraseña se encriptará con MD5.
La segunda orden simplemente añade el usuario us2 al fichero ya creado.
En ambos casos se nos preguntará por la contraseña.
El fichero de grupos contiene la información de estos, indicando qué usuarios pertenecen a cada grupo. No requiere ninguna orden, pues se trata de un simple fichero de texto, con una estructura como la siguiente:
profes: usu1 usu2
admin: adm1 adm2
Fíjate que los usuarios de cada grupo no se separan por comas (como se hace en /etc/group), sino por espacios en blanco.

Autenticación con Digest (cifrado)

Podemos mejorar algo la seguridad utilizando Digest. En lugar de enviar la contraseña en claro (o con codificación en base 64) por la red, con Digest las contraseñas no viajan por la red. En lugar de eso, al pedir un recurso protegido, el servidor envía al cliente un número (noonce). Utilizando ese número “único”, la URI del recurso solicitado y la contraseña, el navegador realiza un digest. Es decir, un cálculo basado en MD5 que le da un número muy raro y difícil de obtener sin la contraseña (muy difícil). El servidor comprueba el digest utilizando para ello la información almacenada. Por peculiaridades matemáticas de este tipo de operaciones, el servidor no necesita guardar la contraseña (ni siquiera encriptada). Le basta con guardar otro digest basado en la misma contraseña para poder comprobarlo.
Ciertamente, este mecanismo reduce el riesgo de captura de contraseñas. Veamos un ejemplo:

```
#Fichero de Configuración con Autenticación con Digest
#Cargamos el módulo que necesitamos con a2enmod auth_digest
<VirtualHost profes.depinfo.iesjc>
     ServerName profes.depinfo.iesjc
     ServerAdmin administrador@depinfo.iesjc
     DocumentRoot /var/www/profes
     ErrorLog /tmp/profes_ERROR.log
     TransferLog /tmp/profes_ACCESS.log
     <Directory /var/www/profes>
```

```
            AuthType Digest
            AuthName profes
            # Indica el encargado de suministrar el Digest
            # por defecto file. O bien, usando los módulos
            # apropiados, usar bases de datos donde se almacenan
            # los usuarios y contraseñas
            AuthDigestProvider file
            #Fichero con los Digest de los usuarios autorizados
            AuthUserFile /etc/apache2/passwd/digest_profes
            require valid-user
      </Directory>
</VirtualHost>
```

Fichero de Digest

La gestión del fichero de digest en el servidor se realiza con la aplicación **htdigest**. Para añadir los mismos usuarios que teníamos, debemos realizar las siguientes órdenes:

```
# htdigest -c /etc/apache2/passwd/digest_profes profes usu1
# htdigest /etc/apache2/passwd/digest_profes profes usu2
```

De nuevo, la opción **c** en el primer comando creará el fichero.

Fíjate que es necesario pasarle al comando el nombre del entorno protegido (profes), este campo se denomina *realm* y su valor se corresponde al que hemos usado en **`AuthName`**.

Autenticación por Redes, Dominios y Hosts

Pero en ocasiones necesitamos establecer permisos no solo por usuario, también por dirección IP de origen, subred, etc. Para eso disponemos de la directivas **Require ip** o **Require host** que suministra el módulo *mod_authz_host* que suele estar habilitado por defecto.

Veamos un ejemplo:

```
#Fichero de Configuración con Autenticación Básica y Allow
ServerName "iesjc"
ServerRoot "/etc/apache2"
DocumentRoot "/var/www"
```

```
PidFile /var/run/apache2/apache2.pid
User www-data
Group www-data
ErrorLog /var/log/apache2/error.log
TypesConfig /etc/mime.types
ErrorLog /tmp/iesjc_ERROR.log
NameVirtualHost 192.168.0.51

<VirtualHost profes.depinfo.iesjc>
     ServerName profes.depinfo.iesjc
     DocumentRoot "/var/www/profes"
     ErrorLog /tmp/profes_ERROR.log
     TransferLog /tmp/profes_ACCESS.log
     <Directory /var/www/profes>
          AuthType Basic
          AuthName "Información profes"
          AuthUserFile/etc/apache2/passwd/passwd_profes
          AuthGroupFile /etc/apache2/passwd/grupo_profes
          require valid-user
     </Directory>

     <Directory /var/www/alum_smx>
          # Sólo permite el acceso desde esta subred
          Require ip 192.168.100
     </Directory>
</VirtualHost>
```

El funcionamiento de Require puede establerse mediante IP's o mediante subdominios y nombres de host, por ejemplo:

`Require host iesjc.org aula18.edif_a.net toniserver` Permite el acceso a sistemas que pertenezcan a estos dominios y al sistema toniserver.

`Require ip 10.2  192.168.100.100` Permite el acceso a sistemas que estén en esta subred y al que tenga esa IP.

También se puede usar con negación mediante :

```
Require not host  iesjc.org   aula18.edif_a.net   toniserver
Require not ip 10.2   192.168.100.100
```

que tendrán el efecto contrario, no obstante, las sentencias Require, si son varias o hay alguna negación se incluyen en directivas tipo: *<RequireAll>... </RequireAll>*, *<RequireAny> ... </RequireAny> o <RequireNone> ... </RequireNone>*. De forma que la primera se deben cumplir todas la condiciones para dar acceso, en la segunda que se cumpla alguna de las condiciones y en la última que no se cumpla ninguna de las directivas que contiene.

Directivas para la Autenticación

Veamos las directivas que hemos utilizado:

AuthType Establece el tipo de autenticación de usuario que se utilizará.

Los valores posibles son Basic y Digest.

AuthName Establece el nombre para el espacio protegido.

AuthUserFile Establece el fichero del servidor que guarda la información de usuarios y sus contraseñas encriptadas (para autenticación Básica).

AuthDigestFile Establece el fichero del servidor que guarda la información de usuarios y sus digests (para autenticación Digest).

AuthGroupFile Establece el fichero del servidor que guarda la información de grupos de usuarios (para autenticación Básica).

Require Establece qué usuarios podrán acceder al recurso protegido.

Los valores posibles son: user, group, valid-user o cualquier combinación de ellos. Al utilizar user y/o group, debe añadirse una lista separada por espacios con los nombres de los usuarios y/o los grupos autorizados.

Require [not] ip Establece mediante dirección de red o subred los sistema que pueden acceder al recurso.

Require [not] host Establece mediante el nombre de un host o subdominio los sistemas que podrán acceder al recurso.

AccessFileName Establece el nombre para los ficheros de configuración distribuidos.

AllowOverride Establece qué directivas están permitidas en los ficheros de configuración distribuidos. Los valores posibles son: AuthConfig, FileInfo, Indexes, Limit y Options.

AuthDigestProvider Indica el encargado de suministrar el Digest por defecto file. O bien, usando los módulos apropiados, usar bases de datos donde se almacenan los usuarios y contraseñas.

RequireAll Incluye un grupo de directivas de autorización de las cuales ninguna debe fallar y al menos una se debe cumplir para que conceda el acceso al recurso.

RequireAny Incluye un grupo de directivas de autorización de las cuales al menos una se debe cumplir para que conceda el acceso al recurso.

RequireNone Incluye un grupo de directivas de autorización de las cuales ninguna se debe cumplir para que conceda el acceso al recurso.

Configuración Distribuida .htaccess

Con los ficheros *.htaccess* se puede modificar las directivas de acceso al directorio en que se encuentra, tendría el mismo efecto que si estas directivas se establecen en la configuración del servidor virtual dentro de la directiva <*Directory*>. Apache aconseja no usar este tipo de configuraciones si podemos editar el fichero de configuración donde se usa el directorio. Los ficheros .htaccess relentizan la respuesta del servidor.

Las directivas incluidas en un fichero *.htaccess* se aplican en el directorio donde se encuentra y en sus subdirectorios, tenga en cuenta que se puede aplicar en cascada varios de estos ficheros reescribiéndose directivas establecidas en configuraciones previas.

Para poder usar los ficheros de configuración distribuida debe establecerse la directiva *AllowOverride* en la definición *Directory* en el fichero de configuración del servidor virtual. *AllowOverride* indica qué tendrá efecto en el directorio de los que incluya el fichero *.htaccess*.

Por ejemplo, si se quiere modificar el conjunto de caracteres por defecto en el contenido de un directorio mediante *.htaccess*, pondremos en él la directiva *AddDefaultCharset utf-8* pero sólo tendrá efecto si en la definición Directory de este servidor virtual aparece la directiva: *AllowOverride FileInfo* sino, no permitirá sobreescribir esta característica y cambiar el juego de caracteres al que hemos indicado en *.htaccess*. Para permitir todo tipo de cambios se puede establecer *AllowOverride All* . Otro uso típico de *.htaccess* es para establecer la autenticación de usuarios con acceso al contenido del directorio, para esto debe establecerse *AllowOverride AuthConfig* y en el *.htaccess* pondremos por ejemplo:

```
    AuthType Digest
    AuthName profes
    AuthDigestProvider file
    AuthUserFile /etc/apache2/passwd/digest_profes
    require valid-user
```

Índices

En ocasiones puede resultar imposible mantener actualizada la página de índice de un directorio. Por ejemplo, si se trata de un directorio de descargas en el que varios usuarios pueden subir información para que otros accedan a ella. Además, para qué vamos a trabajar haciendo índices en HTML si Apache puede hacerlos por nosotros.

El funcionamiento normal de Apache2 consiste que si cuando recibe una solicitud por parte de un cliente en el directorio de acceso no encuentra ningún fichero del tipo `DirectoryIndex` entonces mostrará la lista de contenidos del directorio, esta configuración por defecto la podemos cambiar si no deseamos que se muestre el contenido del directorio eliminamos `Indexes` en la directiva `Options` o poniendo:

`Options -Indexes` .

A continuación, veremos cómo podemos manejar la forma en que se muestran estos contenidos.

Apache dispone de directivas no sólo para realizar índices muy potentes, sino también para personalizarlos añadiendo o quitando opciones. Veamos cómo configurarlos:

```
<VirtualHost www.depinfo.iesjc>
    ServerName "www.depinfo.iesjc"
    DocumentRoot "/var/www/depinfo"
    <Directory /var/www/curso/depinfo/formularios>
        #Establecemos índices bonitos para este directorio
        IndexOptions FancyIndexing
        #Añadimos una descripción para algunos recursos
        AddDescription "Formulario de Contacto" contacto.html
        #Hacemos que algunos elementos no aparezcan en el índice
        IndexIgnore *.jpg
```

```
        IndexIgnore ..
    </Directory>
</VirtualHost>
```

Hay muchas directivas que nos permitirán configurar los índices, las puedes consultar en https://httpd.apache.org/docs/2.4/mod/mod_autoindex.html . Aquí hay algunas:

IndexOptions Establece las opciones para los índices. Pueden ser, entre otras: DescriptionWidth, FancyIndexing, FoldersFirst, IconsAreLinks, IconHeight, IconWidth, IgnoreCase, ScanHTMLTitles ...

IndexIgnore Establece qué ficheros no aparecerán en el índice.

AddDescription Añade una descripción para uno o varios ficheros del índice.

AddIcon Asocia un icono a un fichero o tipo de ficheros.

IndexOptions FancyIndexing Permite ordenar el listado mediante clics en las cabeceras de las columnas.

SuppressColumnSorting Elimina los enlaces de las cabeceras que permiten la ordenación del contenido

Redirecciones

En ocasiones es necesario mover las cosas de sitio. Hay que mover directorios de un sitio a otro, trasladar ficheros ... y queremos que nuestros clientes sigan accediendo como lo hacían antes. El módulo *mod_rewrite* sirve para reescribir las URL de las solicitudes que recibe nuestro servidor de forma que pueda servir las páginas incluso si se ha trasladado el contenido solicitado, tanto dentro del servidor actual o a otro servidor diferente.

Para conseguirlo tenemos tres mecanismos:

Redirecciones con Alias

Alias nos permite dar acceso a recursos que no están en la dirección indicada por la directiva DocumentRoot.

Por ejemplo,

```
<VirtualHost www.depinfo.iesjc>
    ServerName www.depinfo.iesjc
```

```
    DocumentRoot /var/www/depinfo
    #Damos acceso a un directorio que NO está en DocumentRoot
    Alias /calendario_examenes /var/www/calendario
    #También podemos crear accesos a directorios de DocumentRoot
    Alias /exam1 /var/www/depinfo/desarrollo/exam1/test
</VirtualHost>
```

Redirecciones con Redirect

Redirect asocia una URL antigua con una nueva. La nueva URL es notificada al cliente para que realice la conexión con ella.

Por ejemplo:

```
<VirtualHost www.depinfo.iesjc>
    ServerName www.depinfo.iesjc
    DocumentRoot "/var/www/depinfo"
    #Redireccionamos a los clientes a la nueva web
    Redirect /info http://www.iesjoancoromines.org
</VirtualHost>
```

Rewrite

Todo lo anterior, y mucho más, podemos hacerlo con Rewrite. Rewrite es un módulo muy extenso y completo que, básicamente, aplica un patrón a una URL y realiza sustituciones en ella.

Por ejemplo:

Cargamos el módulo que necesitamos: `#a2enmod rewrite`

```
NameVirtualHost 192.168.0.51
<VirtualHost www.depinfo.iesjc>
    ServerName www.depinfo.iesjc
    DocumentRoot "/var/www/depinfo"
    #Activamos el motor de rewrite
    RewriteEngine on
    #Establecemos el fichero de log
    RewriteLog /tmp/log_rewrite
```

```
    #Establecemos el nivel de log
    RewriteLogLevel 9
    #La dirección www.depinfo.iesjc/info/ la redirigimos a iesjoancoromies.org
    RewriteRule ^/info/$ http://www.iesjoancoromines.org
</VirtualHost>
```

Pero hay reglas de rewrite mucho más complicadas y útiles. Por ejemplo, imaginemos que tenemos una página web cuyo contenido depende de un parámetro (típico de páginas PHP, Perl, etc.). El problema es que ni a los seres humano ni tampoco a los buscadores les gusta recordar una URL del tipo *http://www.depinfo.iesjc/prog.php?opcion=1*. Preferimos las URLS directas.

¿Qué podemos hacer?

Podemos reescribir las peticiones que nos lleguen a una dirección “fácil”. Por ejemplo, si la opción 1 nos da información del centro y la opción 2 nos da información del calendario escolar, podemos crear dos direcciones “ficticias” que sean *http://www.depinfo.iesjc/centro/* y *http://www.depinfo.iesjc/calendario/* y poner en el fichero de configuración la siguientes directivas:

RewriteRule ^/centro/$ prog.php?opcion=1 [L]

RewriteRule ^/calendario/$ prog.php?opcion=2 [L]

Fíjate en la opción [L] al final. Indica al proceso que es una regla final. Si se aplica, no se seguirán aplicando más reglas.

Las reglas de rewrite se ejecutan en cascada. La URL de entrada para la segunda regla es la salida de la primera. Esto nos permite utilizar reglas encadenadas.

Las opciones son casi ilimitadas: podemos pasar partes de la URL como parámetros, podemos establecer condiciones, cadenas de reglas... Cualquier problema de redirección, por complicado que sea, puede resolverse con este módulo.

Directivas para la Configuración de Redirecciones

Hemos utilizado las siguientes directivas:

Alias Asocia URLs con ficheros del sistema**.**

Redirect Envía una redirección externa indicándole al cliente que cargue otra URL.

RewriteEngine Activa/Desactiva el motor de rewrite.

RewriteLog Establece el fichero de log para el motor de rewrite.

RewriteLogLevel Establece el nivel de depuración para el motor de rewrite.

RewriteRule Establece una regla para el motor de rewrite.

Logging - Registro de actividad

Los ficheros de log son un instrumento crítico para la seguridad de nuestro servidor. En este capítulo veremos cómo configurarlos para que Apache guarde la información que necesitamos, tanto para mejorar su rendimiento como para resolver problemas a los usuarios, conocer a qué secciones acceden más, intentos de acceso no autorizados, etc.

Pero no basta con configurar correctamente los logs para que guarden la información que queremos. También tenemos que consultarla periódicamente. Sino, cualquier información que pudiéramos registrar será inútil.

Existen muchos programas que te pueden ayudar a inspeccionar y gestionar los logs de Apache. Puedes encontrarlos en los paquetes apache2-utils, scanerrlog, visitors, vlogger, webdruid...

Ficheros de Log

Aunque ya los hemos utilizado, vamos a mencionarlos explícitamente.

Apache utiliza dos ficheros de log:

ErrorLog – Para registrar los errores que se producen al acceder a los recursos de Apache.

TransferLog – Para registrar los accesos que no producen error.

CustomLog Establece el fichero y el formato para un log personalizado.

Estas directiva se pueden establecer en el fichero de configuración de Apache2, pero lo más habitual es definirlas en cada servidor virtual para registrar su actividad.

ErrorLog

El contenido de ErrorLog es algo parecido a esto:

[Tue Nov 21 11:20:14 2006] [notice] caught SIGTERM, shutting down

[Tue Nov 21 11:20:15 2006] [notice] Apache/2.0.55 (Debian) configured

-- resuming normal operations

[Tue Nov 21 11:20:18 2006] [error] [client 127.0.0.1] unknown

directive "fsze="ssi.shtml"" in parsed doc /var/www/ssi.shtml

[Tue Nov 21 12:12:36 2006] [error] [client 192.168.0.51] File does

not exist: /var/www/curso/aym/no_existo

En él podemos ver los mensajes de cierre de Apache, de reinicio, errores de directivas SSI, páginas no encontradas, etc.

TransferLog

El contenido de TransferLog será parecido a esto:

192.168.0.51 - - [21/Nov/2006:10:22:32 +0100] "GET /cgi/aym.cgi
HTTP/1.1" 200 51
127.0.0.1 - - [21/Nov/2006:11:07:30 +0100] "GET /ssi.shtml HTTP/1.1"
200 130
192.168.0.51 - - [21/Nov/2006:11:43:20 +0100] "GET /index.shtml
HTTP/1.1" 200 348
192.168.0.51 - - [21/Nov/2006:12:12:36 +0100] "GET /no_existo
HTTP/1.1" 404 207

En él podemos ver desde qué dirección se ha accedido a nuestro servidor, en qué momento, qué petición se recibió, resultado, etc.

Directivas para la Configuración de Logs

Podemos utilizar las siguientes directivas para la configuración de los ficheros de log:

ErrorLog Establece el fichero que para el log de errores. Opcionalmente puede ser un programa que, por ejemplo, registre los errores en una base de datos.

LogLevel Establece el nivel de detalle para el log de errores. Los valores posibles son: emerg, alert, crit, error, warn, notice, info y debug.

TransferLog Establece el fichero que para el log de accesos. Opcionalmente puede ser un programa que, por ejemplo, registre los accesos en una base de datos.

LogFormat Define un formato de log para el log de accesos o para un log personalizado.

CustomLog Establece el fichero y el formato para un log personalizado.