

# **TEMA 5: Apache**

Módulo

Despliegue de aplicaciones web

para los ciclos

Desarrollo de aplicaciones web



Despliegue FP-GS; Tema5:Apache

© Gerardo Martín Esquivel, Noviembre de 2022

Algunos derechos reservados.

Este trabajo se distribuye bajo la Licencia "Reconocimiento-No comercial-Compartir igual 3.0 Unported" de Creative Commons disponible en <http://creativecommons.org/licenses/by-nc-sa/3.0/>

<b>5.1 El servicio web.....</b>	<b>4</b>
5.1.1 Apache.....	5
<b>5.2 Instalación de Apache.....</b>	<b>6</b>
5.2.1 Instalación en Windows Server.....	6
5.2.2 Instalación en Ubuntu.....	7
<b>5.3 Arranque y parada del servicio.....</b>	<b>8</b>
5.3.1 Arranque y parada del servidor Apache en Windows Server.....	8
5.3.2 Arranque y parada del servidor Apache en Ubuntu.....	8
Iniciar el servicio.....	8
Parar el servicio.....	8
Reiniciar el servicio.....	9
Comprobar el estado del servicio.....	9
<b>5.4 Comprobación de la instalación.....</b>	<b>9</b>
<b>5.5 Ficheros y carpetas importantes.....</b>	<b>10</b>
5.5.1 El DocumentRoot y el DirectoryIndex.....	10
<b>5.6 Configuración.....</b>	<b>11</b>
5.6.1 Directivas básicas.....	12
Include.....	12
DocumentRoot.....	12
Alias.....	13
Redirect.....	14
ServerRoot.....	14
DirectoryIndex.....	14
Listen.....	14
ErrorDocument.....	15
ErrorLog.....	15
PidFile.....	15
User.....	16
Group.....	16
TypesConfig.....	16
DefaultType.....	16
5.6.2 Secciones.....	16
5.6.3 Directivas para la configuración por secciones.....	17
<Directory>.....	17
<File>.....	17
<Location>.....	18
Opciones habituales.....	18
5.6.4 Otros archivos de configuración.....	19
<b>5.7 Utilización de módulos.....</b>	<b>20</b>
5.7.1 Habilitar y deshabilitar módulos en Linux.....	20
Para habilitar un módulo.....	20
Para deshabilitar un módulo.....	20
5.7.2 Habilitar y deshabilitar módulos en Windows.....	21
5.7.3 El módulo userdir.....	21

<b>5.8 Autenticación y control de acceso.....</b>	<b>22</b>
5.8.1 Autenticación básico.....	22
5.8.2 Autenticación HTTP Digest.....	24
<b>5.9 Archivos .htaccess.....</b>	<b>25</b>
AllowOverride.....	25
<b>5.10 Registro de errores (logging).....</b>	<b>26</b>
5.10.1 El fichero ErrorLog.....	26
5.10.2 Directivas para la Configuración de Logs.....	27
LogLevel.....	27
ErrorLog.....	27
TransferLog.....	27
LogFormat.....	27
CustomLog.....	27
5.10.3 El módulo mod-status.....	28
5.10.4 El módulo mod-info.....	28
5.10.5 El paquete webalizer.....	29
<b>5.11 Hosts virtuales.....</b>	<b>29</b>
5.11.1 Host virtual basado en nombre.....	30
5.11.2 Host virtual basado en IP.....	31
5.11.3 Host virtual basado en puertos.....	32
5.11.4 Hosts Virtuales dinámicos.....	33
5.11.5 Directivas para la Configuración de Hosts Virtuales.....	34
DocumentRoot.....	34
ServerName.....	34
VirtualHost.....	34
NameVirtualHost.....	34
VirtualDocumentRoot.....	34
Include.....	34
UseCanonicalName.....	34
TransferLog.....	34
<b>5.12 HTTPS y certificados digitales.....</b>	<b>35</b>
5.12.1 Cifrado asimétrico.....	35
5.12.2 Certificados.....	35
5.12.3 Generar un certificado.....	36
5.12.4 Configuración HTTPS en Apache.....	37
<b>5.13 Webdav.....</b>	<b>37</b>
Activación y configuración de Webdav.....	37
Acceso a Webdav desde Windows.....	37
Acceso a Webdav desde Linux.....	38
<b>5.14 La herramienta apache2ctl.....</b>	<b>39</b>

## 5.1 El servicio web

Un servidor web ofrece el servicio web a otros equipos (clientes web). Es decir, les permite consultar las páginas que el servidor publica. El protocolo que usan cliente y servidor web es el protocolo **HTTP** (o también el **HTTPS**).

El protocolo **HTTP** se usa habitualmente en su versión **1.1** (las versiones posteriores no suponen cambios significativos). Los detalles de este protocolo se especifican en la norma **RFC 2068/2616**. Con la versión **1.1** se consiguen conexiones persistentes (no se cierra la conexión tras el envío de cada parte de una página), peticiones simultáneas (se puede realizar más de una petición al webserver con una sola conexión **TCP**) y nuevos métodos (aparte del **GET**, **POST**, **HEAD** de **HTTP 1.0** se suman **PUT**, **COPY**, **DELETE**, **TRACE**, **OPTIONS** y otros más).

**Ejemplo: Solicitud de un fichero a un servidor web.**



```
administrador@Laptop:~$ telnet servidorlinux00.daw00.net 80
Trying 192.168.20.115...
Connected to servidorlinux00.daw00.net.
Escape character is '^]'.
GET /indice.html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<HTML>
  <HEAD>
    <TITLE>
      Servidor Linux Gerardo: indice.html
    </TITLE>
    <META http-equiv="content-type" content="text/html; charset=UTF-8">
  </HEAD>
  <BODY>
    <H1>indice.html</H1>
    <P>En servidor Linux de Gerardo</P>
  </BODY>
</HTML>
Connection closed by foreign host.
administrador@Laptop:~$
```

En el ejemplo aparecen marcadas con color amarillo las dos líneas que escribe el usuario. En la primera de ellas:

```
telnet servidorlinux00.daw00.net 80
```

Estamos iniciando una conexión **telnet** con un equipo (**servidorlinux00.daw00.net**) que sabemos que dispone de un servidor web activo. Además indicamos que queremos hacer la conexión a través del puerto **80** que es el puerto en el que tradicionalmente escuchan los servidores web.

**Nota:** Una conexión **telnet** es una conexión en modo texto con un equipo remoto a través de una red de ordenadores.

**Nota:** Un servidor web podría estar a la escucha por cualquier puerto, no necesariamente por el **80**. Pero eso obligará a que todos los usuarios de la web tengan que conocer ese puerto.

Una vez iniciada la conexión con el servidor web usamos un comando (**GET**) del protocolo **HTTP versión 1.1** que permite solicitar un fichero. En este caso solicitamos el fichero **indice.html**:

```
GET /indice.html
```

Lo que aparece después de este comando es la respuesta del servidor web: el contenido del fichero pedido, que, como puedes ver, es un documento **HTML**.

**Nota:** Si la petición del fichero se hace desde un **navegador web**, éste interpretará el código **HTML** y "dibuja" la página. Pero el proceso de conexión con el servidor web es el mismo.

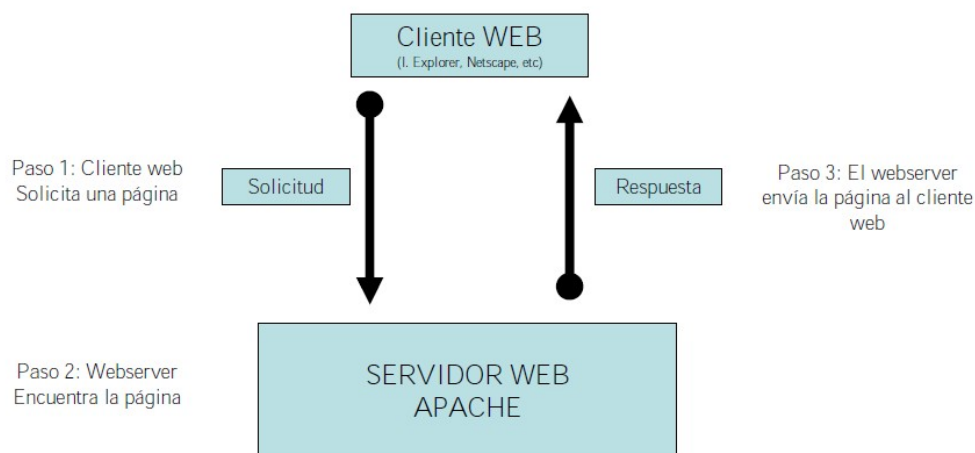
### 5.1.1 Apache

**Apache** es un servidor web de software libre creado en 1996. Es el más utilizado y en la actualidad está mantenido y actualizado por la Apache Software Foundation.

Es un software de código abierto, multiplataforma, modular y extensible. Es rápido y eficiente y tiene interfaces para bases de datos. Soporta la **versión 1.1 de HTTP**.

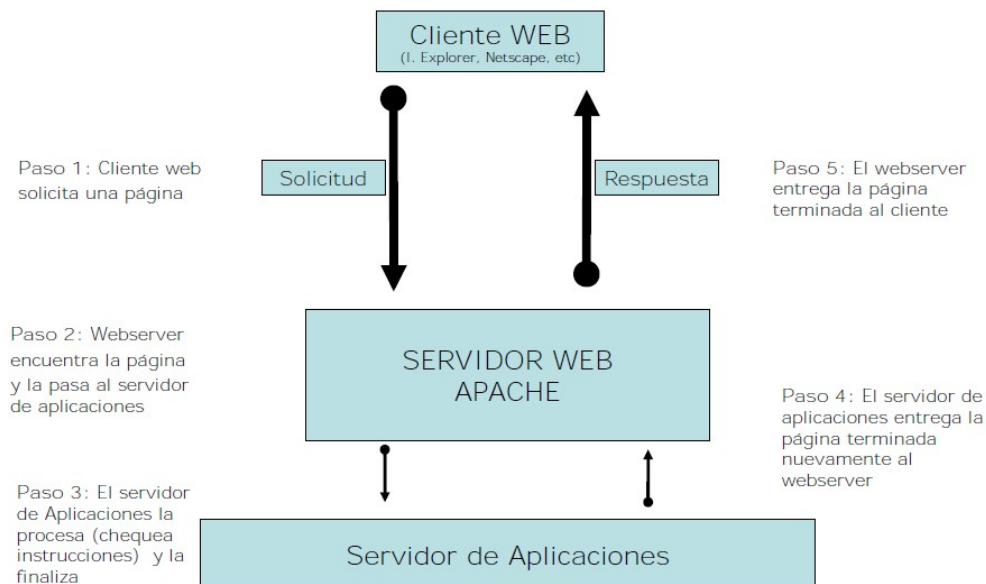
Una de las mejores características de **Apache** es que se trata de un sistema modular. Eso le permite cargar y descargar módulos sin modificar el núcleo. Los módulos los pueden crear terceras partes y hacen al sistema más ágil y con menos necesidad de recursos.

#### **Ejemplo: Modelo HTTP 1.1 con páginas estáticas.**

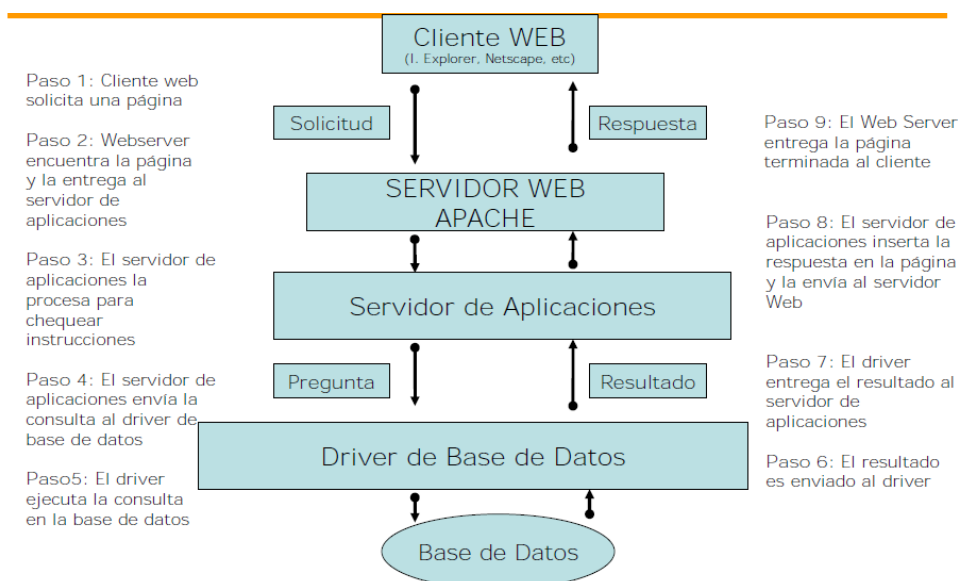


Las páginas estáticas ya están creadas en su totalidad. Sólo necesitan de un servidor web

#### **Ejemplo: Modelo HTTP 1.1 con páginas dinámicas.**



Las páginas dinámicas se generan en función de ciertos datos que aporta el usuario (por ejemplo: el resultado de una búsqueda o la cesta de la compra). Requieren de un servidor web y servidor de aplicaciones.

**Ejemplo: Modelo HTTP 1.1 con páginas dinámicas + base de datos.**

En la mayoría de las ocasiones, las aplicaciones web hacen uso de bases de datos.

## 5.2 Instalación de Apache

### 5.2.1 Instalación en Windows Server

Desde la web <https://archive.apache.org/dist/httpd/binaries/win32/> descargamos el instalador que tendrá un nombre parecido a:

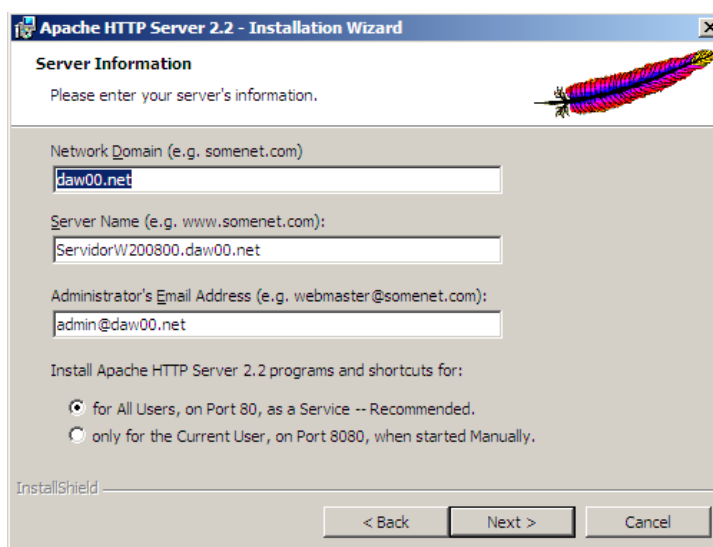
**`httpd-2.2.25-win32-x86-openssl-0.9.8y.msi`**

**Nota:** Junto al enlace de descarga tienes también la huella digital del fichero en diversos formatos (**md5**, **sha1**, etc.). Si quieres asegurarte que el fichero no ha sufrido alteraciones durante la descarga puedes calcular su huella digital y compararla con estas.

Tras la descarga, lo ejecutamos y seguimos las instrucciones. La única pantalla donde debes escribir algo es la que ves a la derecha.

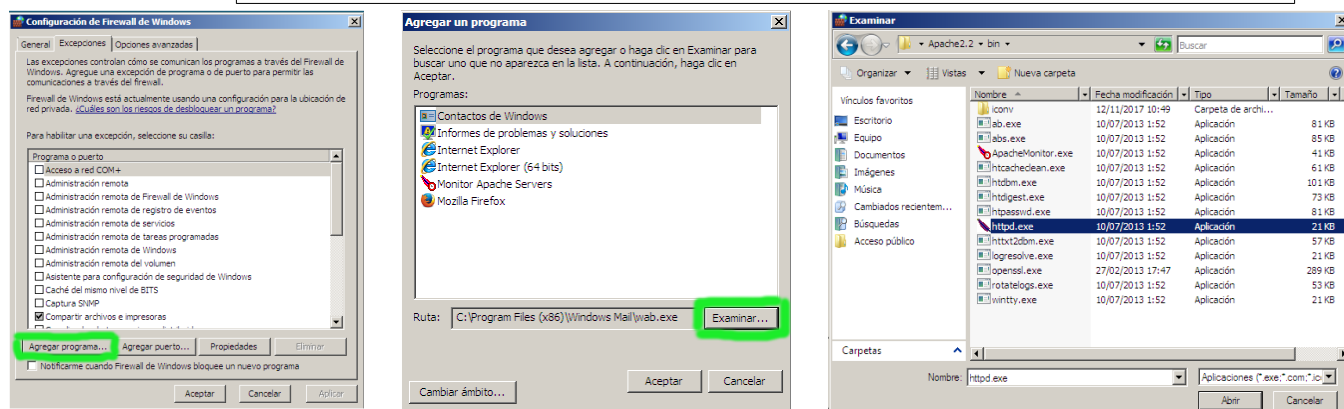
Observa que debes indicar el dominio, el nombre del equipo y la dirección del administrador del sistema.

Con un par de pantallas más, que no requieren especial atención, tendrás instalado el servidor.



Para que el servidor pueda estar operativo es muy importante configurar el **firewall de Windows** para que permita las conexiones externas al servidor. Para ello nos vamos a:

**INICIO/PANEL DE CONTROL/SEGURIDAD/DEJAR PASAR UN PROGRAMA A TRAVÉS DEL FIREWALL DE WINDOWS/AGREGAR PROGRAMA/EXAMINAR**



Y buscamos el programa:

**C:\Archivos de programa (x86)\Apache Software Foundation\Apache2.2\bin\httpd.exe**

Para que la instalación esté disponible desde cualquier localización en la línea de comandos, debemos añadir la ruta del ejecutable de **Apache** a la variable de entorno **PATH**. Desde:

**INICIO/PANEL DE CONTROL/SISTEMA Y MANTENIMIENTO/SISTEMA/CONFIGURACIÓN AVANZADA DEL SISTEMA/VARIABLES DE ENTORNO/VARIABLES DEL SISTEMA**

añadimos (al final, sin borrar nada) nuestra ruta, precedida de un punto y coma (;):

**lo que había previamente;C:\Archivos de programa (x86)\Apache Software Foundation\Apache2.2\bin**

### 5.2.2 Instalación en Ubuntu

La instalación de **Apache** en **Ubuntu** sólo requiere de la instalación del paquete **apache2**:


**sudo apt-get install apache2**


### 5.3 Arranque y parada del servicio

**Nota:** Al iniciar o reiniciar el servicio debemos estar atentos al mensaje que devuelve porque puede ocurrir que haya problemas que impidan ese inicio. Si se da ese caso, debes buscar los mensajes generados en el fichero de error (ver el apartado de **Ficheros y carpetas importantes**).

#### 5.3.1 Arranque y parada del servidor Apache en Windows Server

Un icono de **Apache** en la esquina inferior derecha nos indicará:

➤ Si el servidor está apagado: 

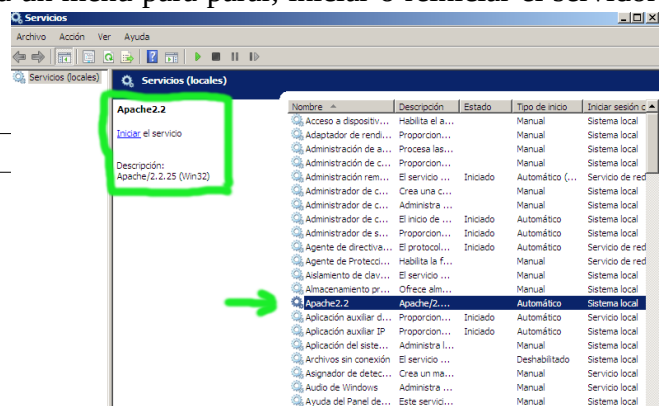
➤ Si el servidor está activo: 

Desde ese mismo icono podemos acceder a un menú para parar, iniciar o reiniciar el servidor web. Si, por algún motivo, no aparece el servidor web, también puedes acudir al listado de servicios de **Windows**:

#### INICIO/HERRAMIENTAS ADMINISTRATIVAS/SERVICIOS

Desde esta pantalla, donde aparecen todos los servicios de **Windows**, podemos seleccionar el servicio web (**Apache 2.2**).

Una vez seleccionado, en la zona de la izquierda aparecen enlaces para iniciar, detener o reiniciar el servicio.



#### 5.3.2 Arranque y parada del servidor Apache en Ubuntu

##### INICIAR EL SERVICIO

```
sudo /etc/init.d/apache2 start
```

o bien

```
sudo service apache2 start
```

o bien

```
systemctl start apache2
```

##### PARAR EL SERVICIO

```
sudo /etc/init.d/apache2 stop
```

o bien

```
sudo service apache2 stop
```

o bien

```
systemctl stop apache2
```



**REINICIAR EL SERVICIO**

```
sudo /etc/init.d/apache2 restart
```

o bien

```
sudo service apache2 restart
```

o bien

```
systemctl restart apache2
```

**COMPROBAR EL ESTADO DEL SERVICIO**

```
sudo /etc/init.d/apache2 status
```

o bien

```
sudo service apache2 status
```

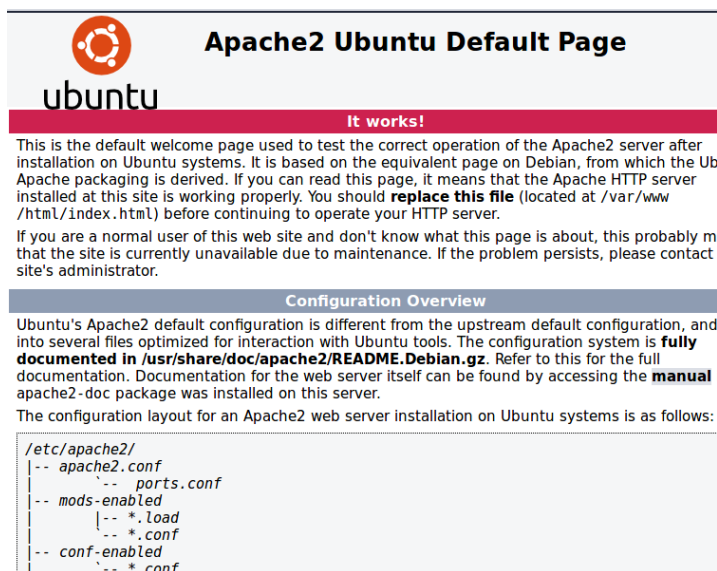
o bien

```
systemctl status apache2
```

## 5.4 Comprobación de la instalación

Para comprobar que el servidor se ha instalado y está funcionando basta con abrir un navegador web en cualquier equipo de la red y escribir la **dirección IP** del equipo donde tienes el servidor web. Debería aparecer una página de prueba que viene con la configuración inicial.

Si el servidor está en un equipo **Ubuntu** la página de bienvenida será algo similar a la imagen de la derecha.



Si el servidor está instalado en un equipo **Windows**, la página mostrará simplemente el mensaje **"It works!"** (funciona):

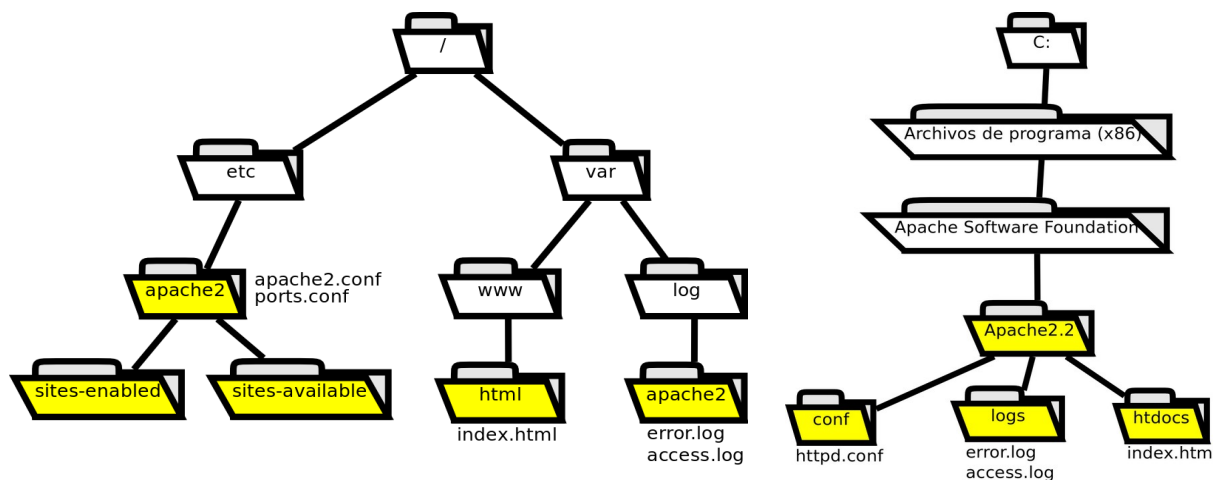
# It works!

## 5.5 Ficheros y carpetas importantes

Aquí podemos ver los principales ficheros y carpetas en la configuración de **Apache**. Como puedes ver, la ubicación es bastante distinta en sistemas **Windows** o sistemas **Linux**. Esta ubicación es también variable entre versiones, de manera que si has instalado **Apache** mediante algún instalador como los **XAMPP** es posible que sea distinta a lo que mostramos aquí.

	Windows	Ubuntu
<b>ServerRoot:</b> carpeta principal de la instalación	C:\Archivos de programa (x86)\Apache Foundation Software\Apache2.2	/etc/apache2
<b>DocumentRoot:</b> carpeta donde cuelgan los documentos que sirve el servidor <b>Apache</b>	...Apache2.2\htdocs	/var/www/html
Carpeta con los ficheros de configuración	...Apache2.2\conf	/etc/apache2
Fichero principal de configuración	httpd.conf	apache2.conf
Fichero con el registro de errores	...Apache2.2\logs\error.log	/var/log/apache2/error.log
Fichero con el registro de accesos	...Apache2.2\logs\error.log	/var/log/apache2/access.log

Esta es la ubicación de carpetas y ficheros en **Linux** (izquierda) y en **Windows** (derecha).



### 5.5.1 El DocumentRoot y el DirectoryIndex

Se conoce como **DocumentRoot** a la raíz del subárbol que se expone en el servidor web, la carpeta desde la que cuelgan todos los ficheros y carpetas que son visibles a través del servidor web. En la imagen anterior el **DocumentRoot** es:

- **/var/www/html** (en la instalación de **Linux**)
- **C:\Archivos de programa (x86)\Apache Software Foundation\Apache2.2\htdocs** (en la instalación de **Windows**)

No obstante, el **DocumentRoot** puede cambiar en cada instalación de **Apache** que te encuentres. Si quieres usar una carpeta distinta como **DocumentRoot** puedes hacerlo usando la directiva adecuada (en los siguientes apartados están las directivas de **Apache**).

**Nota:** Es posible visualizar páginas alojadas fuera de este subárbol usando **localizaciones virtuales** (ver la directiva **Alias**).

El **DirectoryIndex** es el fichero que se sirve cuando no se pide ninguno explícitamente.

Cuando un usuario pide ver un fichero del servidor web, se le sirve sin más. Cuando un usuario solicita una carpeta del servidor web, se busca un fichero con el nombre por defecto (el que se haya establecido como **DirectoryIndex**). Si no se encuentra ese fichero en esa carpeta, entonces se le mostrará un listado de los ficheros y carpetas que allí se encuentran.

El **DirectoryIndex** lo podemos establecer con la directiva de igual nombre y en realidad no se indica un nombre de fichero, sino una lista de nombres de ficheros en orden. Por ejemplo, podemos establecer que en primer lugar se busque el fichero **index.html**, si no lo encuentra que busque **index.htm**, después **index.php**, etc.

La directiva **DirectoryIndex** puede consultarse en los apartados siguientes.

## 5.6 Configuración

La configuración de un servidor **Apache** se establece en varios ficheros de configuración. Son ficheros de texto con directivas para el servidor y comentarios para los administradores (las líneas que comienzan con el carácter almohadilla #- son comentarios). La configuración se lleva a cabo mediante las llamadas **directivas de Apache**, palabras reservadas con una función específica.

Son muchas las directivas que admite **Apache** y puedes encontrar listados muy completos en Internet, por ejemplo en: <http://httpd.apache.org/docs/2.2/en/mod/quickreference.html>, que es la página oficial de **Apache**. **Apache** es muy estricto en la interpretación de estas directivas, de modo que tendrás que hacerlo exactamente como se indica en cuanto a mayúsculas/minúsculas, símbolos de puntuación, espacios, etc.

En cada instalación de **Apache** suele haber un fichero principal de configuración. En los esquemas del apartado anterior son:

- **/etc/apache2/apache2.conf** (en la instalación de **Linux**)
- **C:\Archivos de programa (x86)\Apache Software Foundation\Apache2.2\conf\httpd.conf** (en la instalación de **Windows**)

**Nota:** Cada instalación de **Apache** puede localizar y nombrar el fichero principal de configuración de forma distinta según la versión, el sistema operativo, el instalador, etc.

**Nota Importante:** Además del fichero de configuración principal hay otros ficheros de configuración que influyen en el comportamiento de **Apache**. Busca en el fichero principal las líneas que tienen la directiva **Include** porque esta directiva hace llamadas a otros ficheros para incorporar su contenido. A veces es posible que el contenido de esos otros ficheros esté en contradicción con el fichero principal, creando confusión.

### 5.6.1 Directivas básicas

#### **INCLUDE**

La directiva **Include** se escribe seguida de un nombre de fichero(s) (con su ruta). Indica que debe tenerse en cuenta la configuración allí escrita como si estuviese en este punto del fichero. La ruta puede ser absoluta (si comienza por barra -/-) o relativa al **ServerRoot**.

**Ejemplo:**

```
Include ports.conf
```

Incluye el contenido del fichero **ports.conf** (que, puesto que no tiene ruta, debe encontrarse en el **ServerRoot**) en el punto donde está esta directiva.

**Ejemplo:**

```
Include sites-enabled/*.conf
```

Incluye el contenido de todos los ficheros (en orden alfabético) con extensión **.conf** que se encuentren en la carpeta **sites-enabled** que cuelga del **ServerRoot**.

**Nota:** Es importante tener en cuenta todos los ficheros de configuración que se incluyen mediante la directiva **Include**, porque es posible que el contenido de alguno de ellos esté en contradicción con otras directivas.

**Nota:** El **ServerRoot** puede modificarse con la directiva de igual nombre.

#### **DOCUMENTROOT**

Establece el directorio que será raíz del subárbol que se expone en el servidor web, es decir, la carpeta de la que cuelgan los ficheros y carpetas visibles a través del servidor. El **DocumentRoot** establecido puede cambiar en cada instalación de **Apache** que te encuentres y, con esta directiva, lo puedes cambiar tú. Si se escribe una ruta relativa se entiende que arranca en el **ServerRoot**.

**Ejemplo:**

```
DocumentRoot /var/www/html
```

Indica que el subárbol visible empieza en la carpeta **/var/www/html**. Esto significa que las **localizaciones** de carpetas y ficheros del servidor se señalan teniendo en cuenta esta carpeta como raíz. Por ejemplo la localización **/ciclos** hace referencia a la carpeta **/var/www/html/ciclos**.

**Nota aclaratoria (localizaciones Vs rutas):** Cuando hablamos de **rutas** nos referimos a lugares del **sistema de archivos**. Cuando hablamos de **localizaciones** nos referimos a lugares del **servidor web** (visibles desde el exterior) y su raíz es el **DocumentRoot**. Observa la imagen en las páginas siguientes que compara **localizaciones** y **rutas**.

Las **localizaciones** siempre toman como raíz el **DocumentRoot**. Las **rutas** toman como raíz la raíz del sistema de archivos (**Windows** o **Linux**). Cuando en directivas de **Apache** aparezca una ruta relativa hará referencia al **ServerRoot**.

Algunas directivas trabajan con localizaciones porque indican contenidos visibles, por ejemplo la directiva **ErrorDocument**. No obstante, otras directivas trabajan con rutas porque hacen referencia a ficheros no visibles desde el exterior, como **Include**.

Cuando escribimos una **URL** en un navegador, no escribimos **rutas** sino **localizaciones**.

**Nota:** Es posible visualizar páginas alojadas fuera del subárbol del **DocumentRoot** usando localizaciones virtuales (ver la directiva **Alias**).

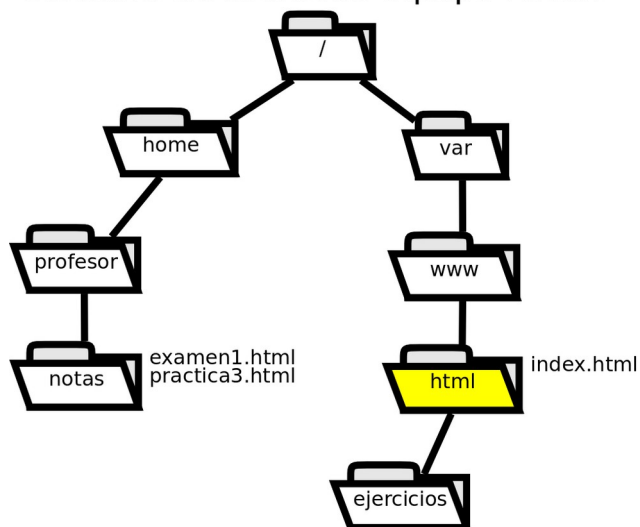
**Ejemplo: Localizaciones Vs Rutas.**

Observa las siguientes imágenes: en la izquierda vemos las carpetas del sistema de archivos de un sistema **Linux**. En ese sistema se ha instalado un servidor web **Apache** cuyo **DocumentRoot** es **/var/www/html** (en amarillo). Suponemos que en la configuración de ese servidor web se incluyen las dos siguientes líneas:

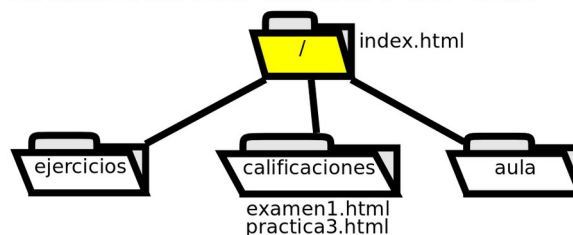
```
Alias /calificaciones /home/profesor/notas
Redirect /aula http://iesmurgi.org
```

Entonces las **localizaciones del servidor web** serían las que aparecen en la imagen de la derecha, donde la carpeta que actúa como **DocumentRoot** (en amarillo) es la raíz, por eso la representamos con la barra (/).

Sistema de archivos equipo Linux



Localizaciones del servidor web



Localización	Apunta a:
<b><i>http://midominio.dom</i></b>	<b><i>/var/www/html</i></b>
<b><i>http://midominio.dom/index.html</i></b>	<b><i>/var/www/html/index.html</i></b>
<b><i>http://midominio.dom/ejercicios</i></b>	<b><i>/var/www/html/ejercicios</i></b>
<b><i>http://midominio.dom/calificaciones</i></b>	<b><i>/home/profesor/notas</i></b>
<b><i>http://midominio.dom/calificaciones/practica3.html</i></b>	<b><i>/home/profesor/notas/practica3.html</i></b>
<b><i>http://midominio.dom/aula</i></b>	<b><i>http://iesmurgi.org</i></b>

**ALIAS**

Establece un alias o apodo para referenciar una carpeta desde una localización. Es útil para poder servir documentos que se encuentran fuera del **DocumentRoot**.

**Ejemplo:**

```
Alias /calificaciones /home/profesor/notas
```

Con este alias indicamos que se puede acceder a la carpeta **/home/notas/profesor** del sistema de archivos, como si fuese la localización **/calificaciones** del servidor web.

**REDIRECT**

Establece un alias para poder referenciar un dominio externo desde una localización.

**Ejemplo:**

```
Redirect /aula http://iesmurgi.org
```

Con este redireccionamiento indicamos que se puede acceder al dominio externo **http://iesmurgi.org** como si fuese la localización **/aula** de nuestro servidor web.

**SERVERROOT**

Establece la carpeta base del servidor. Todas las rutas relativas que aparezcan en los ficheros de configuración partirán de esta carpeta.

**Ejemplo:**

```
ServerRoot /etc/apache2
```

Establece la carpeta **/etc/apache2** como base del servidor **Apache**.

**DIRECTORYINDEX**

Establece el nombre del fichero que se sirve por defecto.

Cuando un usuario pide ver un fichero del servidor web, se le sirve sin más. Cuando un usuario solicita una carpeta del servidor web, se busca un fichero con el nombre por defecto (el que se haya establecido como **DirectoryIndex**).

**Ejemplo:**

```
DirectoryIndex index.html index.htm index.php
```

Con este ejemplo, si el usuario escribe el nombre de una carpeta, sin indicar el fichero que quiere ver, se busca el fichero **index.html**, si no existe se busca **index.htm**, después **index.php**. Si no se encuentra ninguno de ellos, se le mostrará un listado de los ficheros y carpetas que allí se encuentran (siempre que esté activa la opción **Options Indexes** para esa localización).

**LISTEN**

Establece la dirección **IP** y/o el puerto en el que escuchará **Apache**. Por defecto, **Apache** escuchará en todas las direcciones **IP** habilitadas en la máquina.

**Ejemplo:**

```
Listen 192.168.20.100:80
```

El servidor web atenderá las solicitudes que lleguen al puerto **80** por la tarjeta de red que tiene asignada la dirección **192.168.20.100**.

**Ejemplo:**

```
Listen 80
```

El servidor web atenderá las solicitudes que lleguen al puerto **80** por cualquier tarjeta de red del equipo.

**Nota:** Lo habitual es que el servicio web esté a la escucha del puerto **80**. De este modo todos los usuarios que quieran usarlo sabrán donde está.

**Nota importante:** Por defecto no se escucha en ningún puerto, así que si tu configuración no incluye esta directiva el servidor web no funcionará.

## **ERRORDOCUMENT**

Cada petición **HTTP** que se realiza a nuestro servidor web devuelve un código, tanto si la operación ha sido exitosa como si ha generado un error. Los códigos son números de 3 cifras con el siguiente significado:

- Si empiezan por 1 (**1xx**) son solamente informativos.
- Si empiezan por 2 (**2xx**) la petición se ha resuelto con éxito.
- Si empiezan por 3 (**3xx**) se requiere una redirección.
- Si empiezan por 4 (**4xx**) se ha producido un error en el cliente web.
- Si empiezan por 5 (**5xx**) se ha producido un error en el servidor web.

En [https://es.wikipedia.org/wiki/Anexo:Códigos\\_de\\_estado\\_HTTP](https://es.wikipedia.org/wiki/Anexo:Códigos_de_estado_HTTP) tienes un informe detallado de estos códigos de respuesta.

Los códigos que corresponden a un error ya tienen asignado un mensaje genérico. Pero con la directiva **ErrorDocument** podemos personalizar la respuesta a ese error con uno de los siguientes:

- Un simple mensaje.

```
ErrorDocument 404 "No se encuentra esa página en este servidor."
```

- Un documento HTML situado en el propio servidor.

```
ErrorDocument 404 /errores/noEncontrada.html
```

La ubicación de ese archivo se marca desde el **DocumentRoot** (es una localización y no una ruta).

- Una redirección a otro dominio externo.

```
ErrorDocument 404 http://otrodominio.dom
```

- Mantener el mensaje por defecto.

```
ErrorDocument 404 default
```

Este último tiene utilidad si, por ejemplo, hemos establecido una respuesta genérica a un error en todo el servidor y queremos desactivarla en una subcarpeta.

**Nota:** El código **404** corresponde a la situación en la que se solicita una página que no existe. En cada caso tendremos que usar el código cuya respuesta queremos personalizar.

## **ERRORLOG**

Establece el fichero de registro de errores de **Apache**.

En este fichero se registrarán los fallos de acceso, intentos de acceso a recursos sin autorización, páginas no encontradas, etc. Será de gran utilidad cuando se produzcan errores.

## **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**). 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).

**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 como presentarlo (mostrando una página web, ejecutando un plugin, guardándolo en disco, etc.)

**DefaultType**

Establece el tipo **Mime** por defecto para aquellos ficheros cuya extensión no figure en la lista de tipos **Mime**.

**Nota:** Recuerda que en <http://httpd.apache.org/docs/2.2/en/mod/quickreference.html> existe una guía completa de las directivas disponibles.

### 5.6.2 Secciones

En el archivo de configuración existen secciones de configuración que agrupan directivas y que pueden ser de dos tipos:

- Las que **se evalúan para cada petición que se recibe** y se aplican las directivas que se incluyen. Dentro de este grupo están **<Directory>**, **<Files>**, **<Location>**, **<VirtualHost>** entre otras. Las secciones **<Directory>** y **<Files>** están relacionadas con el sistema de archivos (rutas) y **<Location>** está relacionada con el espacio web (localizaciones).
- Las que **se evalúan sólo al inicio o reinicio del servidor**, como **<IfDefine>** e **<IfModule>**. Si al iniciar el servidor las condiciones son las adecuadas, las directivas que incluyen estas secciones se aplicarán a todas las peticiones que se reciban.

Como ejemplo se incluye esta sección **Directory** tomada del archivo **/etc/apache2/apache2.conf**

```
<Directory /var/www>
    Order Allow,Deny
    Allow from dominio.com
    Deny from pc01.dominio.com
</Directory>
```



Que indica que el directorio `/var/www/` está configurado para permitir el acceso a cualquier máquina del dominio **dominio.com** excepto al host **pc01**.

Dentro de los bloques podemos cambiar algunas directivas de las que hemos visto antes como **DirectoryIndex**, de manera que establecemos un comportamiento distinto dentro de la zona afectada por este bloque (el directorio, la localización o el fichero). Las más habituales en los bloques son las directivas que definen el acceso a esa zona.

```
#Fichero de Configuración POR BLOQUES

#Establecemos la configuración de cada directorio
<Directory /var/www>
    #No permitimos 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 acceder a los ficheros .htaccess del directorio
    <Files .htaccess>
        order allow,deny
        deny from all
    </Files>
</Directory>
```

### 5.6.3 Directivas para la configuración por secciones

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.

#### <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 una ruta completamente distinta a la que él especificó.

También hay una directiva <LocationMatch> que permite utilizar expresiones regulares. Así, un mismo bloque de configuración puede aplicarse a varias localizaciones.

### OPCIONES HABITUALES

**Options Indexes:** Si se está accediendo a una carpeta, sin especificar el fichero y no existe el fichero señalado en **DirectoryIndex**, mostrará un listado (índice) del contenido de la carpeta. Si esta opción no está activada, esa situación provocará una respuesta **403 (forbidden)**.

**Options FollowSymLinks:** Permite al usuario seguir los enlaces simbólicos que aparecen en una carpeta.

**Options MultiViews:** Permite ampliar la búsqueda del usuario a todos los ficheros que comienzan por los caracteres escritos (y cualquier extensión). También actúa sobre **DirectoryIndex**, de modo que **DirectoryIndex index**, combinado con **Multiviews** permite buscar **index.html**, **index.php**, etc.

**Require ip N\_ip, Require user usuario:** Restringe el acceso a esta ruta o localización a los equipos cuya **IP** coincide con **N\_ip** o a los usuarios **usuario**.

**Order:** Indica el orden en que se aplican los permisos y denegaciones. **Order allow,deny** significa que primero se aplican los permisos y después las denegaciones. **Order deny,allow** significa que primero se aplican las denegaciones y después los permisos.

**Ejemplo: Acceso exclusivamente a los equipos del dominio apache.org**

```
Order Deny,Allow
Deny from all
Allow from apache.org
```

**Ejemplo: Denegado al dominio foo.apache.org y permitido al resto del dominio apache.org**

```
Order Allow,Deny
Allow from apache.org
Deny from foo.apache.org
```

#### 5.6.4 Otros archivos de configuración

Dentro del directorio `/etc/apache2/` existen otros archivos y directorios de configuración:

**conf.d/**: contiene archivos de configuración asociados a módulos específicos. Los archivos de este directorio son incluidos en `/etc/apache2/apache2.conf`:

```
Include /etc/apache2/conf.d
```

Por ejemplo, se puede crear un archivo dentro de este directorio llamado `alias` que contenga todos los alias creados por el administrador. Se utilizan los Alias para asociar direcciones **URL** con directorios que no pertenecen al directorio origen por defecto (`/var/www/`).

**httpd.conf**: archivo vacío. No se usa desde **Edubuntu** pero se mantiene por compatibilidad con otras versiones de **Apache2** para otros sistemas.

**Mods-available/**: este directorio contiene una serie de archivos **.load** y **.conf**.

- El archivo **.load** contiene directivas de configuración de **Apache** necesarias para la carga del módulo en cuestión. Ejemplo:

```
userdir.load
```

- El archivo **.conf** contiene directivas de configuración necesarias para la utilización del módulo en cuestión. Ejemplo:

```
userdir.conf
```

**mods-enabled/**: para activar un módulo para **Apache2** es necesario crear un enlace simbólico en este directorio a los archivos **.load** asociados con el módulo en **mods-available/**. También para **.conf** si existe. Por defecto la instalación de **Apache2** deja 'activados' un grupo de módulos.

**ports.conf**: directivas de configuración que indican puertos y direcciones **IP** donde **Apache2** escucha peticiones.

**Sites-available/**: similar a **mods-available/** excepto que contiene archivos de configuración para diversos hosts virtuales que podrían ser utilizados en **Apache2**. 'default' es el host por defecto.

**Sites-enabled/**: similar a **mods-enabled/** y contiene enlaces simbólicos a sitios de **sites-available/** que el administrador ha activado.

## 5.7 Utilización de módulos

Uno de los principales motivos por los que se utilizan los módulos en **Apache2** es que no todas las instalaciones de servidores web necesitan las mismas funcionalidades. Al modularizar cada servidor web incluye sólo aquello que necesita consiguiendo que el servicio sea mas ligero.

Existen dos tipos de módulos:

- Los que se compilan de forma "estática" cada vez que se compila **Apache2**. Se pueden consultar con el comando:

```
sudo apache2ctl -l      (si estás en Linux)
httpd -l                (si estás en Windows)
```

- Los que se cargan dinámicamente. Esto permite que **Apache2** cambie dinámicamente los módulos cada vez que se inicia, sin necesidad de recompilar todo el programa de nuevo. Esta opción se denomina **DSO** (Dynamic Shared Object, Objeto Compartido Dinámico) y los archivos correspondientes a estos módulos tienen extensión **.so**.

### 5.7.1 Habilitar y deshabilitar módulos en Linux

En la carpeta **/etc/apache2/mods-available** (módulos disponibles) hay dos ficheros por cada módulo (**.load** con la directiva (**LoadModule**) para cargar el módulo y **.conf** con las directivas que configuran el módulo).

En la carpeta **/etc/apache2/mods-enabled** (módulos habilitados) hay, por cada módulo que está realmente cargado, dos enlaces simbólicos que apuntan a los dos ficheros descritos anteriormente.

Puesto que en nuestro fichero de configuración principal hay un **Include** para todos los ficheros **.load** y **.conf** de esta carpeta, al crearse estos enlaces estamos añadiendo líneas a los ficheros de configuración.

Los cambios de configuración, si hubiera que hacerlos, se harán en los ficheros originales, que ya se reflejarán en los enlaces.

#### PARA HABILITAR UN MÓDULO

NO se hacen los enlaces a mano, sino que se usa el siguiente comando:

```
sudo a2enmod nombreModulo
```

**a2enmod** es un juego de palabras con **available to (2) enabled** (de disponible a habilitado).

Tras habilitar un módulo será necesario reiniciar el servicio.

#### PARA DESHABILITAR UN MÓDULO

```
sudo a2dismod nombreModulo
```

**a2dismod** es un juego de palabras con **available to (2) disabled** (disponible a deshabilitado).

Tras deshabilitar un módulo será necesario reiniciar el servicio.

### 5.7.2 Habilitar y deshabilitar módulos en Windows

En el caso de **Windows**, en lugar de un fichero **.load**, la directiva **LoadModule** se encuentra (comentada o no) directamente en el fichero principal de configuración (**.../Apache2.2/conf/httpd.conf**) y el fichero **.conf** se encuentra en la carpeta (**.../Apache2.2/conf/extra**) y es referenciado desde una línea **Include** (comentada o no) también del fichero principal de configuración.

En este caso, para habilitar un módulo, lo único que hay que hacer es decomentar esas dos líneas.

### 5.7.3 El módulo userdir

Este módulo permite añadir como localizaciones del servidor web una carpeta muy concreta de entre las carpetas privadas de cada usuario (**/home/nombreUsuario/html\_public** en **Linux** o **...\\Users\\nombreUsuario\\Documents\\Website** en **Windows**).

Para acceder a esas carpetas desde la red, la localización será el nombre del usuario precedido del símbolo virgulilla (~).

Localización	Apunta (en Linux) a:	Apunta (en Windows) a:
<b>http://midominio.dom/~pepe</b>	<b>/home/pepe/html_public</b>	<b>C:\\Users\\pepe\\Documents\\Website</b>

**Nota:** Naturalmente, en la configuración se puede cambiar el nombre de la subcarpeta que se quiere compartir (directiva **userdir** que se usa en el archivo **userdir.conf**), es decir, no tiene que ser **html\_public** o **Website**.

**Nota:** El módulo **userdir** solamente asigna una localización del servidor web a esas carpetas, para que sean accesibles. Si se quiere restringir el acceso a esas carpetas a través del servidor habrá que hacerlo explícitamente tal y como se detalla más adelante en este mismo tema.

El mecanismo para activar un módulo de **Apache2** disponible en el directorio **/etc/apache2/mods-available/** consiste en ejecutar como **root** (o un usuario sudo) la orden **a2enmod** (available 2 enable module) sobre él.

```
a2enmod userdir

/etc/init.d/apache2 restart
```

En concreto uno de los módulos mas utilizados es el que acabamos de activar, **userdir**, cuya misión es permitir que cualquier usuario del servidor pueda crear su espacio web en un directorio o carpeta dentro de su cuenta. Es decir, permite asociar sitios con los usuarios del sistema.

## 5.8 Autenticación y control de acceso

En muchas ocasiones queremos 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.

No obstante, 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.

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 más 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.

**Nota:** Teniendo en cuenta esto último, cuando estés haciendo las pruebas de autenticación tendrás que cerrar el navegador entre prueba y prueba, ya que después de un acceso válido te permitirá continuar accediendo sin autenticar.

Respecto al proceso de Autenticación de usuarios en **Apache2** existen dos métodos:

- **Básico o Simple:** el usuario en el navegador web introduce su login o nombre de usuario y contraseña y se envían al servidor sin cifrar.
- **Digest:** el usuario en el navegador web introduce su login y contraseña y se envían al servidor cifrados.

Estos dos métodos sólo autentican al usuario cuando intenta acceder a un recurso. Pero en ninguno de los dos métodos los datos que a continuación se envían del navegador web al servidor o viceversa van cifrados. Son métodos que controlan el acceso a los recursos, pero no protegen la información intercambiada en la comunicación cliente-servidor una vez se ha comprobado que el acceso es válido.

### 5.8.1 Autenticación básica

El módulo que controla este método de autenticación es **mod\_auth\_basic** (estará habilitado por defecto) y tiene la ventaja de que está soportado por todos los navegadores web. Por el contrario, tiene el inconveniente de que el login y la contraseña no van cifradas del navegador web al servidor.

En el archivo de configuración habrá que añadir un bloque **<Directory>...</Directory>** por cada directorio que se quiera proteger:

```
<Directory "/var/www/privado">
    AuthType Basic
    AuthName "Directorio privado"
    AuthUserFile /etc/apache2/passwd/.htpasswd
    Require valid-user
</Directory>
```

- **AuthType:** tipo de autenticación.

- ◆ **Basic:** la contraseña se negocia sin encriptar.
- ◆ **Digest:** la contraseña se negocia encriptada.
- **AuthName:** nombre del dominio de autenticación. Define el conjunto de recursos que estarán sujetos a los mismos requisitos de autenticación. También es el texto que aparecerá en la ventana que pide el usuario y la clave.
- **Require:** usuarios que tienen acceso a los recursos especificados. Opciones disponibles:
  - ◆ **valid-user:** cualquier usuario incluido en el archivo de contraseñas **.htpasswd**.
  - ◆ **user <lista de usuarios>:** lista de usuarios de **.htpasswd**, separados por espacios, que pueden acceder.
  - ◆ **Satisfy:** al utilizar esta directiva determina si se deben cumplir todos los requisitos (**All**) o cualquiera (**Any**).
- **AuthUserFile:** ubicación del archivo de texto que contendrá los nombres de usuario y contraseñas usadas en la autenticación **HTTP** básica. Se suele llamar **.htpasswd**. Previamente hay que crear el directorio **/etc/apache2/passwd/**.

Este fichero podrá ubicarse en cualquier otro sitio y tener cualquier otro nombre. Se trata de un fichero de texto que contiene un usuario por línea, con dos campos por usuario: el nombre y la contraseña encriptada. El comando **htpasswd** nos permite crear el fichero y añadirle usuarios. (Aquí hablamos de usuarios del servicio web, que no deberían confundirse con los usuarios del sistema).

Para crear usuarios para el método de autenticación **Básic** se utiliza la orden **htpasswd**.

```
htpasswd [-c] ruta/fichero nombre_usuario
```

La opción **-c** permite crear el archivo **.htpasswd**. Se usa exclusivamente al crear el primer usuario. (Si se usa la opción **-c** al crear los siguientes usuarios reseteamos el fichero cada vez).

Para seguir dando de alta usuarios no hay que poner el argumento **-c** de lo contrario creará siempre de nuevo el archivo con sólo el último usuario incorporado.

Los usuarios creados para **Apache2** no tienen porque estar dados de alta en el sistema, y si existen no tienen porque tener la misma contraseña.

Al ir a la **URL** **http://servidor.dominio.com/privado/** aparece la ventana:

(El aspecto de la ventana cambiará según el navegador, sistema operativo, etc.).

También podemos permitir el acceso a un recurso a un grupo completo. Para ello usamos la directiva **AuthGroupFile** indicando el nombre y



ubicación del fichero que contiene la información de grupos. Será un fichero de texto plano que engloba los usuarios en grupos con la siguiente sintaxis:

```
pac: usu1 usu2
desarrollo: prog1 prog2
```

Al incluir el nombre del grupo en la directiva **require** se está permitiendo el acceso a todos los miembros del grupo.

**Nota:** La autenticación **Basic** funciona exactamente igual en **Windows**: está habilitada por defecto, las directivas son las mismas, el programa de la línea de comandos funciona igual (**htpasswd.exe**), etc.

### 5.8.2 Autenticación HTTP Digest

Podemos mejorar algo la seguridad utilizando **Digest**. En lugar de enviar la contraseña en claro 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. 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.

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.

El módulo que controla este método de autenticación es **mod\_auth\_digest** (por defecto está deshabilitado, así que habrá que habilitarlo con **a2enmod**). Tiene la ventaja de que el login y la contraseña van cifradas del navegador web al servidor. Por el contrario, tiene el inconveniente de que no está soportado por todos los navegadores web.

Lo primero que hay que hacer es activar dicho módulo. Para ello:

```
a2enmod auth_digest
/etc/init.d/apache2 restart
```

En el archivo de configuración habrá que añadir un bloque **<Directory>...</Directory>** por cada directorio que queramos proteger:

```
<Directory "/var/www/privado">
    AuthType Digest
    AuthName "Dominio"
    AuthDigestProvider file
    AuthUserFile "/etc/apache2/digest"
    Require valid-user
</Directory>
```

- **AuthType:** indica que el método a usar es **Digest**.
- **AuthName:** indica el nombre del dominio de autenticación (Atención porque tendremos que escribirlo exactamente igual al generar cada uno de los usuarios).



- **AuthDigestProvider:** indica el soporte usado para la autenticación. Por defecto **file**.
- **AuthUserFile:** indica donde se encuentra el archivo de contraseñas que ahora llamamos **.htdigest**.

La creación de usuarios en el método de autenticación **Digest** requiere la orden **htdigest**.

```
htdigest [-c] ruta/archivo dominio nom_usuario
```

El parámetro **dominio** debe coincidir exactamente con el nombre del dominio de autenticación dado en la directiva **AuthName** ya que, cuando se crea un usuario, se hace incluyéndolo a un dominio de autenticación concreto.

Si la directiva **Require** indica **'valid-user'**, se consideran usuarios válidos sólo los que pertenecen al dominio de autenticación dado en **AuthName** y las contraseñas sólo pueden utilizarse en este dominio.

En el ejemplo añadimos el usuario **usuario1** al archivo de contraseñas **/etc/apache2/passwd/.htdigest**. Si se utiliza el archivo **.htdigest** por primera vez y no existe, hay que incluir la opción **-c**:

```
htdigest -c /etc/apache2/passwd/.htdigest dominio usuario1
```

Ir a la URL **http://servidor.dominio.com/privado/** y aparece la ventana siguiente:

Como se puede observar la ventana de identificación es igual a la anterior con la salvedad de que los datos enviados a través de ella están encriptados.



**Nota:** La autenticación **Digest** funciona exactamente igual en **Windows**: está deshabilitada por defecto, las directivas son las mismas, el programa de la línea de comandos funciona igual (**htdigest.exe**).

## 5.9 Archivos **.htaccess**

Los ficheros de configuración no siempre están centralizados. Solamente el administrador de Apache puede acceder a los ficheros de configuración que conocemos. Si se quiere delegar parte de la configuración a otros usuarios se usan los ficheros de configuración distribuida.

Estos ficheros, situados en cualquier directorio sobrescriben las directivas que afectan a ese directorio. Hay que tener en cuenta que la decisión última siempre es del administrador que puede permitir o denegar la configuración distribuida en cualquier directorio con la directiva **AllowOverride**.

### **ALLOWOVERRIDE**

Esta directiva establece qué directivas están permitidas en los ficheros de configuración distribuidos. Los valores posibles son: **AuthConfig**, **FileInfo**, **Indexes**, **Limit** y **Options**. **AllowOverride all** permite todas las directivas y **AllowOverride None** no permite ninguna.

Para sobrescribir las directivas usamos un fichero con el nombre **.htaccess** (observa que comienza con un punto, lo que en **Linux** significa que será un archivo oculto), situado en el directorio al que afectará (junto con sus subdirectorios). El archivo **.htaccess** se carga cada vez que se solicita un documento por lo que las modificaciones introducidas no requieren reiniciar el servidor web.

Incluir un fichero **.htaccess** con directivas de configuración para **Apache** en un directorio tiene el mismo efecto que escribir esas directivas en una sección **<Directory>** para ese mismo directorio en el fichero de configuración general **/etc/apache2/apache2.conf**. La única diferencia está en que el fichero **.htaccess** lo puede modificar el usuario aunque no tenga permisos para modificar la configuración de **Apache**.

La directiva **AccessFileName** permite modificar el nombre de este archivo, cuyo valor por defecto es **.htaccess**.

### 5.10 Registro de errores (logging)

Los ficheros de registro (**log**) son un instrumento crítico para la seguridad de nuestro servidor. 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**.

#### 5.10.1 El fichero ErrorLog

Registra los errores que se producen al acceder a los recursos de **Apache**. 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 "fsz="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.

### 5.10.2 Directivas para la Configuración de Logs

#### LogLevel

Establece el nivel de detalle para el log de errores. Los valores posibles son, de mayor a menor importancia, son:

<i>Nivel</i>	<i>Descripción</i>	<i>Ejemplo</i>
<b>emerg</b>	Emergencias. Sistema parado	"Child cannot open lock file. Exiting"
<b>alert</b>	Acción que se necesita inmediatamente	"getpwuid: couldn't determine user name from uid"
<b>crit</b>	Condiciones críticas	"socket: Failed to get a socket, exiting child"
<b>error</b>	Condiciones de error	"Premature end of script headers"
<b>warn</b>	Advertencias	"child process 1234 did not exit, sending another SIGHUP"
<b>notice</b>	Normal, pero de interés	"httpd: caught SIGBUS, attempting to dump core in ..."
<b>info</b>	Información	"Server seems busy, (you may need to increase StartServers, or Min/MaxSpareServers)..."
<b>debug</b>	Mensajes de depuración	"Opening config file ..."

Cuando se especifica un nivel se muestran todos los mensajes de importancia mayor a ese nivel. Por ejemplo, cuando se indica **LogLevel warn**, los mensajes de nivel **emerg**, **alert**, **crit** y **error** también se registrarán.

Se recomienda usar al menos el nivel **crit**.

#### ErrorLog

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

#### TransferLog

Establece el fichero que registra los 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 registro de accesos o para un registro personalizado.

#### CustomLog

Establece el fichero y el formato para un registro personalizado.

```
CustomLog /var/log/apache2/access.log combined
```

### 5.10.3 El módulo *mod-status*

Este módulo nos permite monitorizar el rendimiento de *Apache*, generando un documento en formato *HTML*.

Tendremos que habilitar el módulo:

```
a2enmod status
```

y configurarlo en el fichero */etc/apache2/mods-enabled/status.conf*:

```
<Location /server-status>
    SetHandler server-status
    Order deny,allow
    Deny from all
    allow from IP_desde_la_que_queremos_acceder
</Location>
```

Una vez habilitado y configurado, podremos acceder a la información desde el navegador, indicando en la barra de direcciones:

```
IP/server-status
```

La directiva *SetHandler* establece el gestor que tratará la petición. Obsérvese que en la sección *Location* no definimos un *documentRoot*, el directorio donde está la información con la que respondemos a la petición *HTTP*. En cambio, en esta sección establecemos que cuando se haga una petición a la dirección *.../server-status*, la respuesta la proporcionará el gestor *server-status*

```
SetHandler server-status
```

### 5.10.4 El módulo *mod-info*

Este módulo proporciona una vista resumida de la configuración de *Apache*.

Tendremos que habilitar el módulo:

```
a2enmod info
```

y configurarlo en el fichero */etc/apache2/mods-enabled/info.conf*:

```
<Location /server-info>
    SetHandler server-info
    Order deny,allow
    Deny from all
    allow from IP_desde_la_que_queremos_acceder
</Location>
```

Una vez habilitado y configurado, podremos acceder a la información desde el navegador, indicando en la barra de direcciones:

```
IP/server-info
```

### 5.10.5 El paquete webalizer

**Webalizer** es un paquete disponible para distribuciones **Linux**, que usa el fichero de registro de accesos de **Apache** (**access.log**) y genera un documento **HTML**.

Necesitamos la carpeta **/var/www/html/webalizer** que será el lugar donde se genere el **HTML** con el nombre **index.html**. Y será necesario configurarlo a través del fichero **/etc/webalizer/webalizer.conf** donde prestaremos atención especialmente a dos directivas:

```
LogFile /var/log/apache2/access.log
```

para indicar cual es el fichero de registro desde el que **webalizer** obtiene la información a representar.

```
OutputDir /var/www/html/webalizer
```

para indicar la carpeta donde se generará el documento **HTML**.

**Nota:** **Webalizer** crea la carpeta **/var/www/webalizer** y la señala con la directiva **OutputDir**. Observa que nuestro servidor puede tener el **DocumentRoot** en otro sitio, por ejemplo, en **/var/www/html**. En ese caso tendremos que crear manualmente la carpeta **/var/www/html/webalizer** e indicarla en la directiva **OutputDir**.

Desde el terminal de **Linux** generamos el **HTML** con el comando:

```
sudo webalizer
```

y posteriormente accedemos desde un navegador con la dirección:

```
IP/webalizer/index.html
```

**Nota:** Al ejecutar **webalizer** buscará el fichero **access.log**. Si no se ha producido ningún acceso o intento de acceso, este fichero aún no se habrá generado y mostrará un mensaje de error. El documento **HTML** no se habrá generado.

## 5.11 Hosts virtuales

Si hacemos un **nslookup** a la dirección **www.bbva.es** obtenemos la **IP 89.107.176.83**. Si hacemos un **nslookup** a la dirección **IP 89.107.176.83** obtenemos hasta 6 dominios distintos. Lo que está pasando es que todos esos dominios están alojados en el mismo equipo y por eso responden a la misma dirección **IP**.

Mediante los hosts virtuales, **Apache2** permite la posibilidad de alojar varios dominios en una sola máquina. De esa forma, cuando una petición entra en el servidor **Apache2** desde un navegador web a través de una **IP** dada, **Apache2** comprueba el nombre de dominio que se está solicitando y muestra el contenido asociado a dicho nombre de dominio.

Trabajar con Hosts Virtuales consiste en ejecutar más de un sitio web en el mismo servidor y tiene como ventajas la versatilidad para crear diferentes sitios web configurables; el precio, ya que se necesita sólo una máquina para alojar varios servidores web; una configuración del sistema sirve para todos los servidores web; se actualiza sólo una vez y no requiere ningún software ni hardware adicional.

Apache soporta dos tipos de hosts virtuales:

- Hosts virtuales **basados en nombres**: Permiten alojar varios nombres de host (o dominios) en una misma máquina (**IP**). Todos los hosts virtuales que comparten la misma **IP** deben declararse mediante la directiva **NameVirtualHost**.
- Hosts virtuales **basados en IP**: Una máquina responde de diferente manera a diferentes direcciones **IP**. Es decir, tenemos múltiples **IPs** asignadas al sistema y queremos que cada una de ellas soporte un sitio web.

Para la definición de hosts virtuales se utiliza la sección **<VirtualHost>** y en ella se incluyen las directivas que se aplican a un determinado host virtual. Como mínimo debe incluir la directiva **DocumentRoot** y **ServerName**.

De la misma forma que configuramos los módulos, los sitios o dominios que atiende nuestro servidor se ayudan de dos carpetas: **sites-availables** y **sites-enabled**. Nosotros creamos un fichero de configuración para cada sitio en la carpeta **sites-availables**. Cada vez que queramos habilitar uno de estos sitios, creamos un enlace (en **sites-enabled**) a su fichero de configuración. El fichero de configuración general tiene un **include** a todos los ficheros que haya en el directorio **sites-enabled**.

Esto significa que, de entre todos los ficheros que haya en el directorio **sites-available**, aquellos que tengan un enlace simbólico en el directorio **sites-enabled**, serán incluidos en la configuración.

Un fichero de configuración de sitio se activa con la orden:

```
a2ensite fichero (apache 2 enable site)
```

Un fichero de configuración de sitio se desactiva con la orden:

```
a2dissite fichero (apache 2 disable site)
```

Es conveniente utilizar un solo fichero de configuración de sitio a la vez.

### 5.11.1 Host virtual basado en nombre

Para usar hosts virtuales basados en nombres se debe especificar en el servidor qué dirección **IP** se va a usar mediante la directiva **NameVirtualHost**. Un asterisco **'\*'** indica que se aceptan todas las solicitudes entrantes y que se activan los hosts virtuales.

```
NameVirtualHost *
```

**Nota:** **Apache2** puede saber si una petición va dirigida a uno u otro host virtual gracias a la información contenida en las cabeceras del **HTTP/1.1**. Cuando un navegador envía una petición al servidor usando el protocolo **HTTP/1.1** envía una cabecera del tipo **host: nombre\_de\_un\_host** con la que **Apache2** puede diferenciar las peticiones de los distintos hosts virtuales.

Hay que crear un bloque **<VirtualHost>** para cada host virtual diferente.

```
<VirtualHost *>
```

Dentro de cada bloque **<VirtualHost>** se necesitará como mínimo una directiva **ServerName** para indicar a qué host se sirve y una directiva **DocumentRoot** para indicar dónde están los contenidos a servir dentro del sistema de archivos.

**Ejemplo: Crear un host virtual por nombre en el mismo servidor**

En el siguiente ejemplo añadimos un host virtual (**virtual.dominio.com**) a un servidor web ya existente (**servidor.dominio.com**). Suponemos que el servidor web existente (host virtual por defecto) dispone de su configuración en el archivo **/etc/apache2/sites-available/default** como host virtual. Para preparar el nuevo sitio virtual habrá que:

- Configurar el servicio **DNS** para que ambos dominios (**servidor.dominio.com** y **virtual.dominio.com**) apunten a la misma dirección **IP**.
- Crear el directorio **/var/www/virtual** y un archivo **index.html** con el contenido.
- Crear el archivo de configuración (**virtual.conf**) para el sitio con este contenido:

```
NameVirtualHost *  
  
<VirtualHost *>  
    ServerName virtual.dominio.com  
    DocumentRoot /var/www/virtual  
    <Directory /var/www/virtual>  
        Options FollowSymLinks  
        AllowOverride None  
    </Directory>  
</VirtualHost>
```

- Activar el sitio ejecutando la orden:

```
a2ensite virtual
```

- Reiniciar el servidor para que lea los cambios de configuración:

```
/etc/init.d/apache2 restart
```

- Ir al navegador y probar la **URL virtual.dominio.com**. Comprobar que se visualiza el contenido de **index.html** para ese sitio virtual.

**5.11.2 Host virtual basado en IP**

Si se tiene un sistema que dispone de varias direcciones **IP** y se quiere que cada una de ellas soporte un sitio web se deberá crear una sección virtual para cada dirección **IP**.

En este caso los servidores virtuales definidos reciben las solicitudes en función de la **IP** requerida, no del nombre del servidor.

**Ejemplo: Crear un host virtual por IP en el mismo servidor**

Supongamos un sistema que dispone de 2 interfaces de red (**192.168.1.1** y **192.168.1.2**) que permite definir un host virtual alojado en el mismo servidor web, además del dominio inicial **servidor.dominio.com**. La configuración para el host virtual **virtual.dominio.com** es la siguiente y podría ser incluida tanto en el archivo **default** como en un archivo diferenciado dentro de **sites-available**:

```
<VirtualHost 192.168.1.2>  
    ServerName virtual.dominio.com
```

```
DocumentRoot /var/www/virtual
ErrorLog /var/log/apache2/virtual-error.log
CustomLog /var/log/apache2/virtual-access.log combined
</VirtualHost>
```

### 5.11.3 Host virtual basado en puertos

Vamos a suponer que, para una misma dirección **IP**, el servidor web quiere mostrar diferente contenido según el puerto en el que se realiza la conexión **HTTP**.

Para ello habrá que indicar en la sección **<VirtualHost>** y en **NameVirtualHost** el puerto asociado y todos los puertos utilizados para atender peticiones deben estar **Listen**.

```
NameVirtualHost 192.168.1.1:80

<VirtualHost 192.168.1.1:80>
    ServerName servidor.dominio.com
    DocumentRoot /var/www/servidor80
</VirtualHost>

<VirtualHost 192.168.1.1:443>
    ServerName servidor.dominio.com
    DocumentRoot /var/www/servidor443
</VirtualHost>
```

En este caso el acceso al servidor web, si se llama igual, debe incluir el puerto por el cual se realiza la petición **HTTP**. En la **URL** habrá que escribir:

```
http://servidor.dominio.com:443/
```

Si se utiliza un puerto diferente al **80** (por defecto) hay que revisar el archivo **/etc/apache2/ports.conf** e incluir las líneas **Listen** correspondientes a los nuevos puertos de escucha de **Apache2**.

**Ejemplo: Hosts virtuales basados en nombre, IP y puerto**

```
#Fichero de Configuración para Hosts Virtuales
ServerRoot /etc/apache2
DocumentRoot /var/www
PidFile /var/run/apache2.pid
User www-data
Group www-data
ErrorLog /var/log/apache2/error.log
Listen 80
TypesConfig /etc/mime.types
DefaultType text/plain
DirectoryIndex index.html index.htm

NameVirtualHost 192.168.2.166
<VirtualHost aym.juntaex.es>
```



```
ServerName aym.juntaex.es
DocumentRoot /var/www/aym
ErrorLog /tmp/aym_ERROR.log
TransferLog /tmp/aym_ACCESS.log
</VirtualHost>

<VirtualHost bs.juntaex.es>
    ServerName bs.juntaex.es
    DocumentRoot /var/www/bs
    ErrorLog /tmp/bs_ERROR.log
    TransferLog /tmp/bs_ACCESS.log
</VirtualHost>

<VirtualHost 192.168.2.200:80>
    ServerName bs.juntaex.es
    DocumentRoot /var/www/bs
    ErrorLog /tmp/bs_ERROR.log
    TransferLog /tmp/bs_ACCESS.log
</VirtualHost>

<VirtualHost 192.168.2.200:8080>
    ServerName intranet.bs.juntaex.es
    DocumentRoot /var/www/bs/intranet
    ErrorLog /tmp/bs_intranet_ERROR.log
    TransferLog /tmp/bs_intranet_ACCESS.log
</VirtualHost>
```

#### 5.11.4 Hosts Virtuales dinámicos

Si tenemos que configurar decenas, cientos o incluso miles de sitios web tal y como ocurre en un **ISP**, tenemos la opción de Hosts Virtuales Dinámicos. Veamos un ejemplo:

```
#Fichero de Configuración para Hosts Virtuales Dinámicos
ServerName sabio
ServerRoot /etc/apache2
DocumentRoot /var/www
PidFile /var/run/apache2.pid
User www-data
Group www-data
ErrorLog /var/log/apache2/error.log
Listen 80
TypesConfig /etc/mime.types
DefaultType text/plain
ErrorLog /tmp/juntaex_ERROR.log
DirectoryIndex index.html index.htm

#Cargamos módulo para configuración dinámica de hosts virtuales
Include /etc/apache2/mods-available/vhost_alias.load
```

```
#Indicamos que se use como 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 y que sus administradores tengan permiso para gestionarlo. 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/[^.#]*
```

Incluye en el fichero de configuración todos los ficheros que hay en el directorio.

### 5.11.5 Directivas para la Configuración de Hosts Virtuales

#### DOCUMENTROOT

Establece el directorio que será raíz del subárbol que se expone en el servidor web, es decir, la carpeta de la que cuelgan los ficheros y carpetas visibles a través del servidor.

#### 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.

#### 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.

#### INCLUDE

Incluye uno o varios ficheros en el fichero de configuración.

#### USECANONICALNAME

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

#### TRANSFERLOG

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

## 5.12 HTTPS y certificados digitales

Cuando queramos seguridad en la comunicación con nuestro servidor **Apache** tendremos que usar el protocolo **HTTP** bajo **SSL**, conocido como **HTTPS**. Con este protocolo, la comunicación entre el servidor y el cliente se cifra de forma asimétrica, esto es, usando un sistema de dos claves: una pública y una privada.

### 5.12.1 Cifrado asimétrico

El **cifrado asimétrico** se llama así en contraposición con el cifrado simétrico en el que emisor y receptor usan la misma clave: el emisor para cifrar el mensaje, el receptor para descifrarlo. El principal problema que esto genera es que, puesto que ambos tienen que conocer la clave, esta tiene que viajar y podría ser interceptada.

En un cifrado asimétrico cada persona o entidad puede generar una pareja de claves: pública y privada.

- la **clave privada** nunca viaja puesto que se genera en el propio equipo y se usa para cifrar los mensajes a enviar. Si suponemos que el interesado la custodia correctamente, cualquier mensaje cifrado con esa clave pertenece a él indiscutiblemente.
- la **clave pública** si viaja. Se genera junto con la clave privada con la que forma pareja y se envía a todos aquellos que queramos puedan descifrar nuestro mensaje. El receptor necesita la clave pública porque ésta será la única forma de leer el contenido de un mensaje cifrado con su correspondiente clave privada.

Es cierto que un intruso podría conseguir nuestra clave pública y descifrar los mensajes que enviamos, pero de momento no nos preocupa eso, el objetivo que buscábamos y hemos conseguido es demostrar que nosotros somos el remitente del mensaje, puesto que nuestra clave pública ha servido para descifrar nuestro mensaje.

Si lo que necesitamos es que el mensaje sólo pueda ser leído por su destinatario tendremos que hacer uso de su clave pública. Al cifrar un mensaje con la clave pública de otro, sólo él podrá descifrarlo.

Y si necesitamos ambas cosas a la vez (que quede clara nuestra identidad y que sólo el destinatario correcto pueda leer el mensaje) tendremos que cifrar el mensaje usando nuestra clave privada y la clave pública del destinatario. Del otro lado, el destinatario tendrá que usar su clave privada y la pública nuestra para poder descifrar.

### 5.12.2 Certificados

Un **certificado** es un documento que recoge ciertos datos de una organización y su clave pública y que está firmado por una autoridad de certificación (**AC**). De este modo, cuando una persona, una entidad o una página web tiene que acreditar que es quien dice ser presenta su certificado. Que tú te creas lo que ese certificado dice dependerá de tu confianza en la **AC** que lo firma.

Una **autoridad de certificación (AC ó CA)** es una entidad de confianza responsable de emitir y revocar certificados digitales, empleando criptografía de clave pública. No existe un método para discernir si debes o no confiar en una **AC**. Puesto que se trata de una cuestión de confianza, dependerá de tu experiencia con esa organización, de que otras personas de tu confianza confíen en ella, de que se trate de una organización pública como la **FMNT**, etc.

Las autoridades de certificación se organizan en una estructura jerárquica en la que cada organización certifica la autenticidad de las organizaciones del nivel inferior. La organización que se encuentra en la cúspide de esta jerarquía es la denominada **AC raíz** y, puesto que no hay organizaciones de nivel superior, tiene que certificarse a sí misma.

### 5.12.3 Generar un certificado

Cuando necesitemos un certificado para nuestra web tendremos que solicitarlo a una **AC**.

- Utilizando una herramienta como **openssl** generamos la clave privada, que se almacena en un fichero con la extensión **.key**.
- Con la clave privada y aportando los datos necesarios (**URL**, nombre de la entidad, ...) generamos una solicitud de certificado (en un fichero con extensión **.csr**) que tendremos que enviar a la **AC**.
- La **AC** nos envía el certificado firmado con el que podemos demostrar que somos la entidad que decimos ser. Naturalmente la **AC** tendrá sus medios para asegurarse de ello. Solo restará instalar el certificado en nuestro servidor **Apache**.

Un usuario puede decidir las autoridades que reconoce instalando sus certificados raíz en el navegador. Cuando se accede a una página web con **HTTPS** y ésta se identifica mediante su certificado, el navegador sabrá si debe confiar o no en ese certificado en función de que esté firmado por una autoridad de su confianza, una autoridad cuyo certificado raíz esté instalado. De no ser así, presentará al usuario el conocido mensaje de alerta de seguridad para que el usuario decida bajo su propia responsabilidad si admite o no el certificado.

Cuando estemos usando **Apache** fuera de un entorno de producción nos interesará usar certificados sin contratar y pagar los servicios de una **AC**. Para este propósito podemos generar nuestro propio certificado autofirmado.

- para generar la clave privada (en el fichero **clave.key**):

```
openssl genrsa -out clave.key 2048
```

- para generar la solicitud de certificado (en el fichero **solicitud.csr**). Para ello nos pedirá algunos datos:

```
openssl req -new -key clave.key -out solicitud.csr
```

- para generar el certificado (en el fichero **certificado.crt**):

```
openssl x509 -req -days 365 -in solicitud.csr -signkey clave.key -out certificado.crt
```

### 5.12.4 Configuración HTTPS en Apache

Una vez que tenemos un certificado, configuramos **Apache** con las siguientes directivas:

```
Listen 443
```

para que el servidor atienda las peticiones al puerto **443** que es el del protocolo **HTTPS**

```
SSLEngine on
```

para activar el protocolo **SSL**

```
SSLRequireSSL
```

en la sección (**Directory, Location,...**), donde queramos que el acceso sea **HTTPS**.

```
SSLCertificateFile ruta/certificado.crt
```

para indicar la ruta y nombre del fichero que contiene el certificado (**.crt**).

```
SSLCertificateKeyFile ruta/clave.key
```

para indicar la ruta y nombre del fichero que contiene la clave privada (**.key**).

### 5.13 Webdav

El servicio **Webdav** permite acceder a una carpeta del servidor desde el navegador de archivos del equipo cliente. Los permisos que sobre los ficheros y carpetas tendrá ese acceso pueden ser determinados en la configuración del propio módulo **Webdav**.

#### ACTIVACIÓN Y CONFIGURACIÓN DE WEBDAV

Hay que activar dos módulos:

```
sudo a2enmod dav  
sudo a2enmod dav_fs
```

Y en la carpeta que se quiere compartir establecer la directiva:

```
Dav on
```

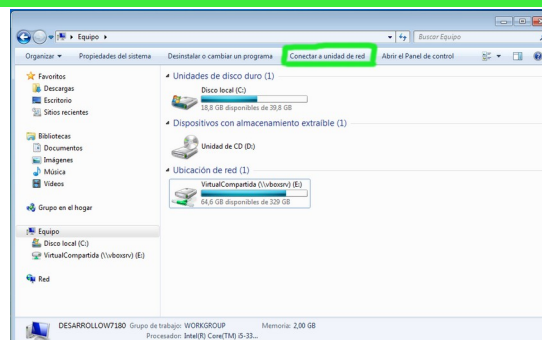
y, naturalmente, establecer autenticación con, por ejemplo, **Digest**.

Si queremos que los usuarios de la web puedan crear ficheros y/o carpetas en la carpeta compartida será necesario cambiar el grupo propietario de la carpeta al grupo de **Apache (www-data)** y otorgar permiso de escritura al grupo:

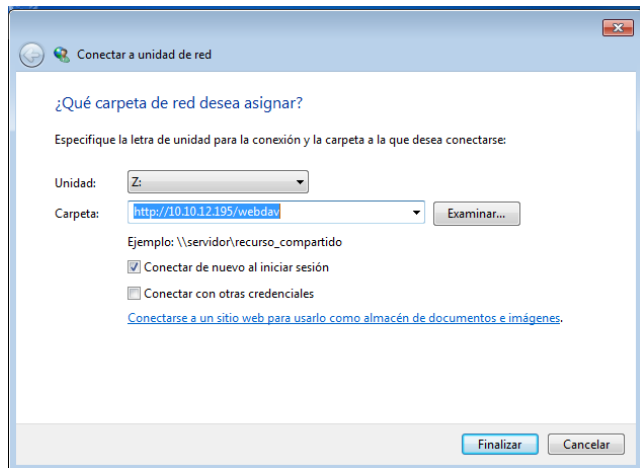
```
sudo chown root:www-data /rutaWebdav  
sudo chmod 775 /rutaWebdav
```

#### ACCESO A WEBDAV DESDE WINDOWS

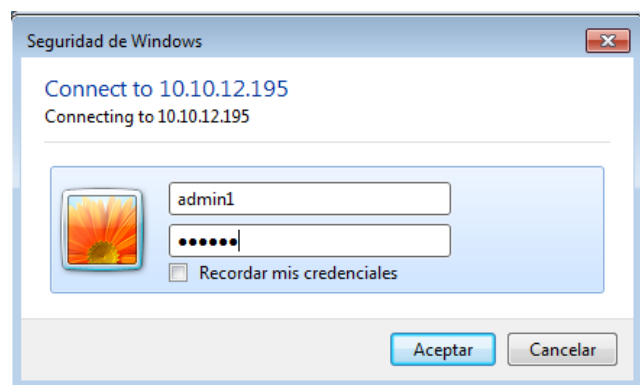
En una ventana Windows seleccionamos "**Conectar a unidad de red**"



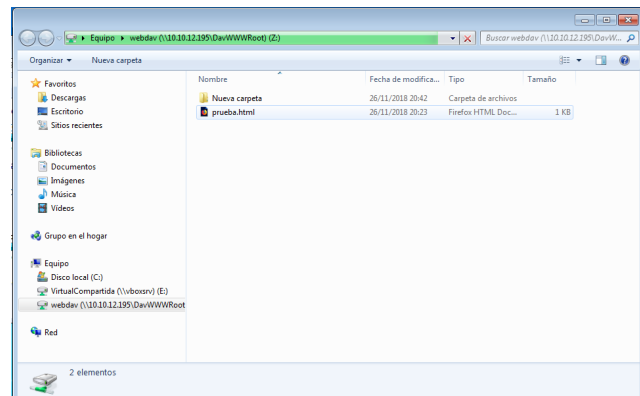
Y señalamos la ubicación de la carpeta **Webdav**:



Si hemos configurado la autenticación, pedirá que nos identifiquemos:

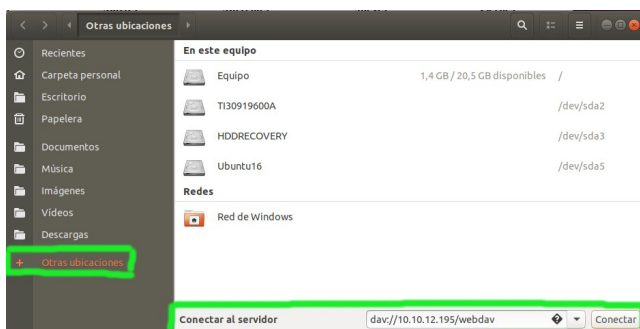


Y finalmente accedemos a la carpeta **Webdav**.

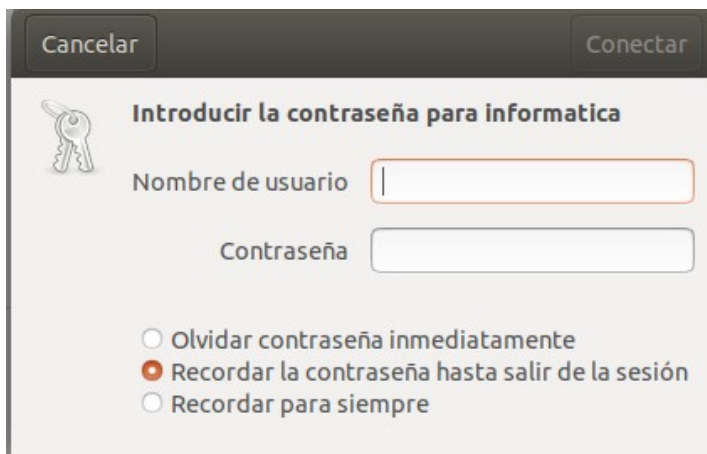


### ACCESO A WEBDAV DESDE LINUX

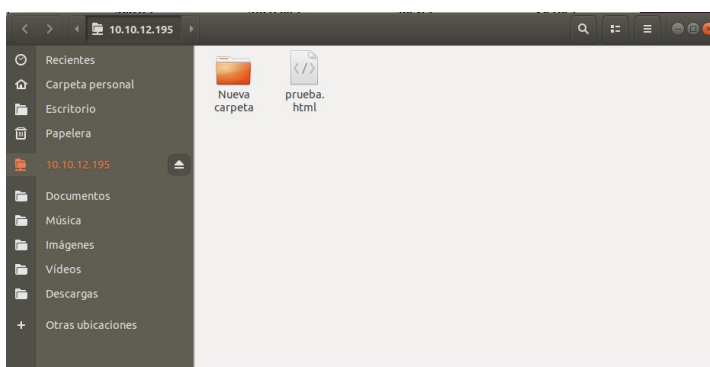
En una ventana **Linux**, seleccionamos "**Conectar con el servidor**" y señalamos la ubicación de la carpeta **Webdav**.



Si hemos configurado la autenticación, pedirá que nos identifiquemos:



Y finalmente accedemos a la carpeta **Webdav**.



### 5.14 La herramienta *apache2ctl*

Aunque el daemon de **Apache** es **apache2**, también disponemos del comando **apache2ctl**, que permite ejecutar herramientas de administración del demonio. De hecho, internamente, cuando ejecutamos las opciones básicas de **apache2**, éstas se traducen internamente en comandos **apache2ctl**.

Las opciones básicas de que dispone esta herramienta son:

- **start**: Inicia el servicio de **apache**.
- **stop**: Detiene el servicio de **apache**.
- **restart**: Reinicia el servicio de **apache**.
- **fullstatus**: Muestra un reporte del estado completo de **apache**.
- **status**: Muestra un reporte del estado breve de **apache**.
- **graceful**: Reinicia “delicadamente” el servicio **apache**. La diferencia con **restart** consiste en que a diferencia de éste, **graceful** no cancela las conexiones que tenga previamente abiertas... Se puede considerar como un “**restart** en caliente”.
- **configtest**: Ejecuta una prueba de los archivos de configuración de **apache**. En caso de que la configuración de **apache** esté bien, devuelve **Syntax OK**. En caso contrario, indica cuál es el error.

Se puede encontrar más información usando el comando **man apache2ctl**