UD 3

Administración de Apache

[1. Servidor Web Apache para Linux 1](#_Toc496599355)

[*1.1.* *Instalación de Apache y arranque del servicio* 1](#_Toc496599356)

[*1.2.* *Estructura de archivos de configuración* 1](#_Toc496599357)

[*1.3.* *Habilitar / Deshabilitar módulos* 1](#_Toc496599358)

[*1.4.* *Servidores Virtuales en Apache* 2](#_Toc496599359)

[*1.5.* *Restricción de acceso mediante usuario y contraseña* 7](#_Toc496599360)

[*1.6.* *Espacio web para cada usuario* 8](#_Toc496599361)

[*1.7.* *Página web principal* 9](#_Toc496599362)

[*1.8.* *Comprobar que funciona* 9](#_Toc496599363)

[2. Apache web server en Windows 10](#_Toc496599364)

[*2.1.* *Instalación* 10](#_Toc496599365)

[*2.2.* *Iniciar y detener el servidor* 10](#_Toc496599366)

[*2.3.* *Configuración* 11](#_Toc496599367)

[*2.4.* *Testeando el servidor* 15](#_Toc496599368)

# Servidor Web Apache para Linux

El servidor web apache es una de las aplicaciones estrella del mundo Linux. Es el servidor web más implantado entre los distintos servidores que ofertan servicios web en Internet.

Entre las características más significativas destacamos:

* Es modular
* Capacidad para crear servidores virtuales
* Capacidad para crear servidores seguros https
* Capacidad para crear sitios privados

En este curso haremos uso de éstas y otras características de apache.

## *Instalación de Apache y arranque del servicio*

Instalación de apache2

**apt-get install apache2**

Con lo cual se instalarán los archivos necesarios para que funcione nuestro servidor web. Se instalará apache v2.

Arranque y parada del servicio:

**service apache2 start | stop | status | restart | reload**

## *Estructura de archivos de configuración*

Todos los ficheros de configuración del servidor Apache están disponibles en ***/etc/apache2.***

***/etc/apache2/apache2.conf***: configuración de directorio raíz por defecto, logs, procesos, etc. Este archivo incluye referencias al resto de ficheros de configuración.

***/etc/apache2/mods-available/: directorio de módulos disponibles***

***/etc/apache2/mods-enabled/: directorio de módulos habilitados***

***/etc/apache2/ports.conf: configuraciones de puertos.***

***/etc/apache2/mods-enabled/dir.conf***: archivos a mostrar al entrar en un directorio.

***/etc/apache2/sites-available/: configuración de sitios***

***/etc/apache2/sites-enabled/***: configuración de los sitios que están habilitados.

## *Habilitar / Deshabilitar módulos*

La inclusión de los módulos necesarios para ampliar la funcionalidad del servidor, se efectúa mediante la directiva LoadModule, seguido del nombre del módulo y el archivo (o librería) que lo contiene. Se utilizarán tantas directivas LoadModule, como módulos distintos se vayan a utilizar.

En Debian en el directorio /etc/apache2/mods-enabled tenemos enlaces simbólicos con la extensión .load, que indican el módulo que se va a cargar. Se pueden activar así o mediante la instrucción a2enmod (apache 2 enable module) seguida del nombre del módulo a habilitar. Por ejemplo:

# Para habilitar un módulo en Apache 2 se usa: a2enmod

# Habilitamos por ejemplo el módulo userdir

**a2enmod userdir**

# Para deshabilitar un módulo en Apache 2 se usa: a2dismod

# Deshabilitamos por ejemplo el módulo userdir

**a2dismod userdir**

## *Servidores Virtuales en Apache*

Nuestro servidor web puede servir múltiples webs a la vez, empleando una IP o varias.

Para poder hacer esto, Apache necesita diferenciar el acceso a cada una de las webs, y para ello lo puede hacer de dos formas:

1. Configurando puertos distintos para cada web, y el acceso se hará a través de cada puerto.

2. A través de los diferentes nombres de dominio alojados en el servidor y que se conectan a un mismo puerto 80 (por ejemplo).

### Acceso a través de puertos distintos

Para configurar el acceso a diferentes webs, a través de puertos diferenciados, lo primero que tenemos que hacer es configurar los VirtualHost en nuestro servidor de Apache, e indicar los puertos dónde queremos que escuche nuestro servidor web.

Editaremos el fichero /etc/apache2/ports.conf:

**#Vamos a habilitar los puertos dónde va a escuchar nuestro servidor Apache:**

**nano /etc/apache2/ports.conf**

**Listen 80**

**Listen 9999**

**# Si queremos que Apache escuche en unas IP específicas tendremos que poner también la IP:puerto**

**Listen 192.168.2.250:80**

**# Mediante la directiva NameVirtualHost indicaremos que vamos a tener hosts virtuales. # El asterisco indica que vamos a recibir peticiones de VirtualHost en todas las IP en las que # está escuchando nuestro servidor web. Si nos interesa una IP específica la indicaremos. # A continuación pondremos : y el número de puerto.**

**NameVirtualHost \*:80**

**# Para cada servidor virtual, crearemos una línea como la anterior. Por ejemplo, añadimos una más:**

**NameVirtualHost \*:9999**

Si queremos que se muestren webs diferentes al acceder por un puerto u otro, editaremos las directivas correspondientes dentro del fichero /etc/apache2/sites-available/default, que es el sitio habilitado por defecto y configuraremos los parámetros de DocumentRoot, DirectoryIndex, ServerName, etc.

### Acceso a través de nombres de dominios diferentes

En este caso el puerto será común a todos los servidores virtuales, con lo que en principio el fichero **ports.conf** podemos tener algo como:

**NameVirtualHost \*:80**

**Listen 80**

Crearemos un fichero con la configuración que queramos dar al nuevo sitio virtual, de esta manera evitaremos que el fichero **default** crezca demasiado en tamaño y se haga complicada su edición. Ese nuevo fichero lo haremos en **/etc/apache2/sites-available/**.

**# Por ejemplo:**

**nano /etc/apache2/sites-available/servidor1.local**

Para cada servidor virtual se deberá crear un fichero como el mencionado anteriormente, y dentro se creará un contenedor donde se incluirán sus correspondientes directivas (ServerName, DocumentRoot, ServerAdmin, DirectoryIndex). Cada contenedor, generalmente, está delimitado por las etiquetas con formato <VirtualHost direccion-de-red:puerto> y </VirtualHost>.

Por ejemplo, éste sería el contenido mínimo para un servidor virtual servidor1.local:

**<VirtualHost \*:80>**

**ServerName** [**www.servidor1.local**](http://www.servidor1.local)

**DocumentRoot /var/www/servidor1.local**

**ServerAdmin** [**webmaster@servidor1.local**](mailto:webmaster@servidor1.local)

**DirectoryIndex index.html index.php**

**</VirtualHost>**

Por tanto se deberán crear tantos contenedores de este tipo como servidores virtuales se necesiten. Como valor dirección-de-red se indicará uno de los valores previamente definidos con las directivas NameVirtualHost (“\*” o una dirección IP), dependiendo por donde queremos que el servidor virtual atienda las solicitudes. </source>

### Habilitar / Deshabilitar sitios web

Para habilitar un sitio virtual que tenemos definido en la carpeta **/etc/apache2/sites-available/** lo haremos mediante el comando **a2ensite** nombredelsitio. Por ejemplo:

**# Suponiendo que en /etc/apache2/sites-available tenemos un sitio** [**www.pruebas.local**](http://www.pruebas.local) **# Lo habilitaremos con:**

**a2ensite** [**www.pruebas.local**](http://www.pruebas.local)

**# Para deshabilitarlo lo haremos con:**

**a2dissite** [**www.pruebas.local**](http://www.pruebas.local)

**# Reniciaremos el servicio a continuación:**

**service apache2 restart**

### Directivas utilizadas para la configuración de servidores virtuales y principal

* **ServerName**: indica el nombre que se le dará al servidor web.
* **DocumentRoot**: indica el directorio raíz del sitio para el servidor.
* **ServerAdmin**: indica la dirección de correo del webmaster del sitio.
* **DirectoryIndex**: indica los ficheros que podrán actuar como página índice del sitio.

La configuración de un sencillo servidor web, requerirá modificar la directiva de la sección 2 DocumentRoot, indicando el directorio que ejercerá como ráiz del sitio web. Esta directiva tendrá asociada una estructura contenedora cuyo nombre de directorio coincidirá con el indicado en DocumentRoot.

Es conveniente organizar el sitio web mediante directorios, aunque es posible que Apache pueda acceder a otros que no se encuentran en la jerarquía del sitio que se ha definido, como por ejemplo para la ejecución de un script de una página dinámica. Se suele definir una estructura denominada contenedor para cada uno de los directorios a los que accederá Apache, incluyéndose una para el directorio raíz del sitio del servidor principal.

Estas estructuras, se reconocen por una etiqueta de inicio con el formato <Directory nombre\_directorio> y otra de fin </Directory>. Dentro de estas estructuras se definen directivas que por lo general, establecen distintos permisos sobre el directorio que tiene asociado.

|  |  |  |
| --- | --- | --- |
| Directiva | Opciones | Función |
| Options | None | No establece ninguna opción. |
|  | All | Establece todas las opciones. |
|  | Indexes | Permite visualizar páginas índice existentes en el directorio. |
|  | FollowSymlinks | Permite seguir los enlaces simbólicos del directorio. |
|  | ExecCGI | Admite la ejecución de scripts CGI. |
| AllowOverride | None | No establece ninguna opción. El servidor no leerá los archivos .htaccess |
|  | All | Establece todas las opciones. Permite usar las directivas especificadas en .htaccess |
|  | FileInfo | Muestra la información de los archivos del directorio. |
| Order | allow,deny | Primero aplicará los permisos de allow y luego los de deny. |
|  | deny,allow | Primero aplicará los permisos de deny y luego los de allow. |
| Allow | from all | Admite cualquier acceso al directorio. |
|  | from IP | Admite cualquier acceso al directorio proveniente de la dirección IP indicada. |
|  | from dominio | Admite cualquier acceso al directorio desde el dominio especificado. |
| Deny | from all | Deniega cualquier acceso al directorio. |
|  | from IP | Deniega cualquier acceso al directorio proveniente de la dirección IP indicada. |
|  | from dominio | Deniega cualquier acceso al directorio desde el dominio especificado. |

### Permisos de los directorios

Se puede dar distintos permisos de acceso a los directorios que conforman nuestro espacio web.

La definición de los permisos se hace mediante etiquetas, de manera que en el fichero de configuración ***/etc/apache2/sites-available/000-default*** tiene la siguiente:

**#**

**<Directory /var/www/html>**

Options Indexes FollowSymLinks MultiViews

AllowOverride None

Order allow,deny

allow from all

# This directive allows us to have apache2’s default start page

# in /apache2-default/, but still have / go to the right place

RedirectMatch ^/$ /apache2-default/

**</Directory>**

**#**

En la imagen superior se ven los permisos que tienen la carpeta “/var/www/html” (en la que está guardada la página WEB) y las carpetas que están debajo de ella – salvo que se indique lo contrario. Se pueden crear tantas etiquetas como se desee, para señalar diferentes permisos.

Por ejemplo, con el siguiente código, se permite el acceso a una carpeta llamada “contenidos públicos” situada dentro de la carpeta “/var/www/html”:

**<DIRECTORY “/var/www/html/contenidos publicos”>  
   Options Indexes FollowSymLinks  
   AllowOverride None  
   Order deny,allow  
   Allow from all  
</DIRECTORY>**

Con las siguientes líneas se prohíbe el acceso a una carpeta llamada contenidos personales situada dentro de la carpeta “/var/www/html”:

**<DIRECTORY “/var/www/html/contenidos personales”>  
   Options Indexes FollowSymLinks  
   AllowOverride None  
   Order allow,deny  
   Deny from all  
</DIRECTORY>**

Para crear un filtro por IP, se utilizan las opciones “Allow” y “Deny”. Cuando las utilizamos, tenemos que tener en cuenta que el orden (línea *ORDER*) es muy importante, ya que no es lo mismo denegar los permisos y luego darlos que darlos y luego denegarlos. Para asignar a una IP concreta el permiso para ver el contenido, escribiríamos lo siguiente:

**.......  
    Order deny,allow  
    Deny from all  
    Allow from 192.168.1.25   
</DIRECTORY>**

Si tenemos en marcha un servidor DNS y queremos dar a todos los ordenadores de un dominio concreto permiso para ver el contenido, escribiremos lo siguiente:

**.......  
   Order deny,allow  
   Deny from all  
   Allow from dw2.org**

Si, por el contrario, queremos impedirle la entrada a un ordenador concreto –por ejemplo-, deberemos cambiar el orden e incluir ese impedimento como una excepción:

**.......  
   Order allow,deny  
   Allow from all  
   Deny from irakasle.dw2.org  
</DIRECTORY>**

### Permisos de archivos

Se puede prohibir el acceso a un fichero (o tipo de fichero) almacenado en una carpeta determinada. Para ello, dentro de la etiqueta del directorio tendremos que crear otra etiqueta que haga referencia al archivo. Por ejemplo:

**<DIRECTORY “/var/www/html/contenidos publicos”>  
   Options Indexes FollowSymLinks  
   <FILES prohibido.htm>  
      Order allow,deny  
      Deny from all  
   </FILES>  
   AllowOverride FileInfo  
   Order deny,allow  
   Allow from all  
</DIRECTORY>**

En el ejemplo de arriba, se puede ver todo el contenido del directorio “/var/www/html/contenidos públicos”, excepto el fichero “prohibido.htm”.

También es posible hacer referencia a todos los archivos que hay en nuestro espacio web completo, colocando la etiqueta <FILE> fuera de las etiquetas del directorio. Por ejemplo:

**<FILES prohibido.htm>  
   Order allow,deny  
   Deny from all  
</FILES>**

Además, podemos permitir o denegar la entrada hacia los ficheros de un tipo determinado. Por ejemplo, hacia los ficheros html, de la siguiente forma:

**<FILES \*.html>  
   Order allow,deny  
   Deny from all  
</FILES>**

### Creación de alias

Un alias le ofrece al usuario la posibilidad de acceder a carpetas concretas que están dentro del servidor web, ahora bien, permaneciendo ocultas esas carpetas y su organización. De esta manera, el usuario que entra en el servidor web no sabrá cuál es la organización de los ficheros ni qué nombre tiene cada carpeta.

Para crear el alias se introduce, antes de la etiqueta < Directory ...... > de la carpeta a la que queremos dar un alias, la siguiente línea:

**Alias  /nombre\_simbolico  “camino\_de\_la\_carpeta\_en\_el\_ordenador”.**

Veamos un ejemplo:

**Alias /icons/ “/var/www/html/extra/pics/icons/”**

**<Directory “/var/www/html/extra/pics/icons/”>**

**Options Indexes MultiViews**

**AllowOverride None**

**Order allow,deny**

**Allow from all**

**</Directory>**

En este caso, si escribimos *“*[*http://Dirección\_IP\_del\_servidor\_WEB*](http://Dirección_IP_del_servidor_WEB)*/icons”* en la barra de direcciones del navegador web, abrirá el fichero principal (index.html) que está en el directorio “/var/www/html/extra/pics/icons/”.

## *Restricción de acceso mediante usuario y contraseña*

Para restringir el acceso a una página web mediante usuario y contraseña en primer lugar debemos modificar el fichero */etc/apache2/sites-available/default* y dentro del directorio donde queremos restringir el acceso cambiamos a *AllowOverride AuthConfig*

**<Directory /var/www/intranet>**

**Options Indexes FollowSymLinks Multiviews**

**AllowOverride AuthConfig**

**Order deny, allow**

**Allow from all**

**</Directory>**

A continuación crearemos dos ficheros más dentro del directorio en cuestión */var/www/intranet*. El primero se llamará *.htaccess* y su contenido será el siguiente:

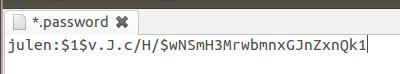
**AuthType Basic**

**AuthName “Se requiere autenticación para acceder a la intranet”**

**AuthUserFile /var/www/intranet/.password**

**Require valid-user**

El segundo se llamará .*password* e introduciremos la lista de usuarios con sus contraseñas que tendrán acceso al sitio web restringido. Su aspecto será:

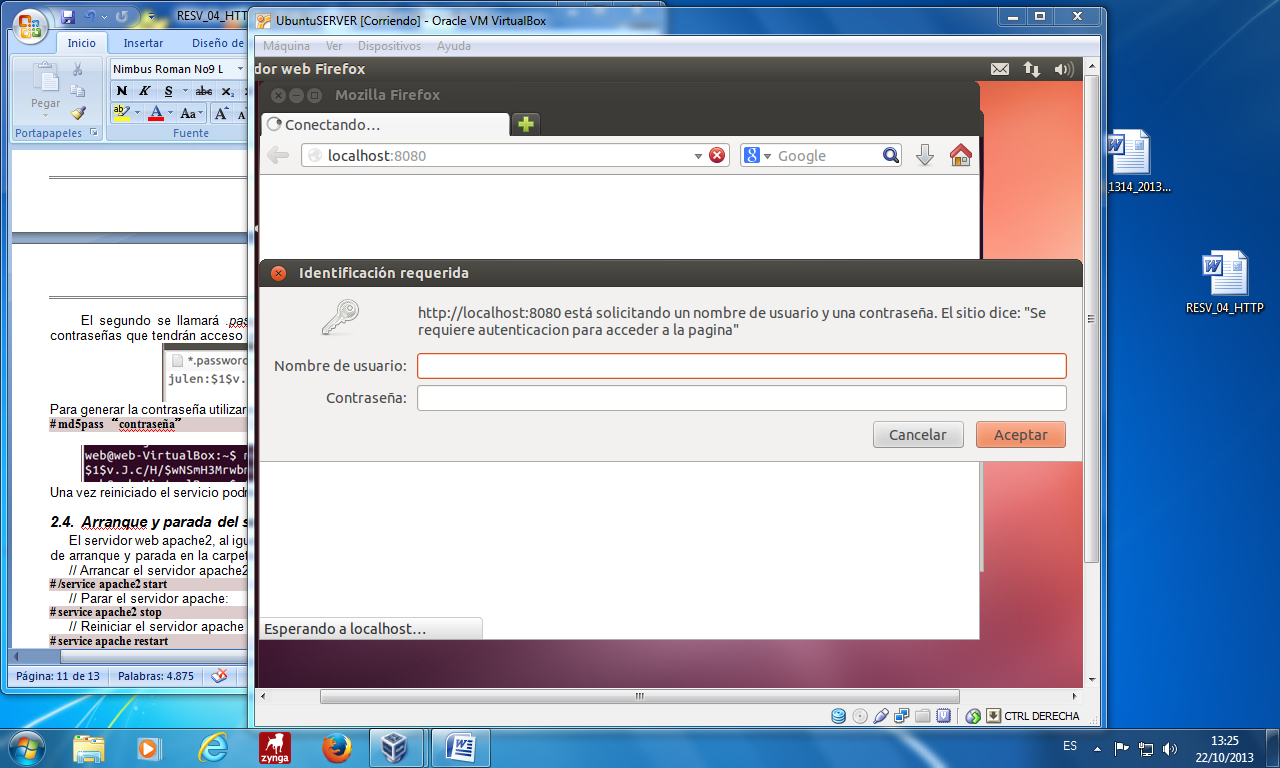


Para generar la contraseña utilizaremos el comando

**# md5pass “contraseña”**



Una vez reiniciado el servicio podremos comprobar que el acceso está restringido



## *Espacio web para cada usuario*

Para poder tener espacios web personales debemos activar el servicio “userdir”. De otra manera será imposible acceder desde fuera a una página personal. El comando para activar el servicio es el siguiente:

**a2enmod userdir**

Después de activarlo, cada usuario deberá crear un directorio llamado public\_html dentro de su directorio personal (/home/usuario) y alojar allí la página principal de su espacio web, que debería llamarse index.html. Al igual que hemos hecho con el directorio de la página web principal del servidor, también podemos cambiar el nombre del directorio que usaremos para páginas personales en el archivo **/etc/apache2/mods-available/userdir.conf**.

Cada usuario del sistema dispondrá de un espacio web que se almacena dentro de su carpeta home en una carpeta llamada ‘public\_html’. Si dicha carpeta no existe, el propio usuario puede crearla y copiar dentro de ella su página web. Los permisos recomendados son 644 para que el ‘grupo’ y el ‘resto’ de usuarios tengan acceso de lectura y así se puedan visualizar las páginas.

Para acceder vía web a la página de un usuario, desde un navegador debemos acceder directamente con la dirección IP a: <http://ip-del-servidor/~login-usuario/>

El caracter ‘~’comúnmente conocido como gusanillo y que se obtiene con Alt Gr + 4 sirve para indicar a apache que debe servir la página desde el home del usuario (en Linux el ‘gusanillo’ equivale a la carpeta home). Ejemplo, si hemos creado un usuario javier y éste ha creado la carpeta /home/javier/public\_html y ha copiado en ella su página web, desde cualquier PC de la red podremos acceder a dicha carpeta yendo a la dirección <http://ip-del-servidor/~javier/>. Para que la página aparezca automáticamente, es necesario crear un archivo llamado index.html.

**Carpetas web de los usuarios**

* **Carpeta web de javier:** /home/javier/public\_html
* **Acceso a la web de javier:** <http://ip-del-servidor/~javier/>

## *Página web principal*

Cuando desde un ordenador intentan acceder a las páginas de nuestro servidor, el servidor web sabe que tiene que ir a buscar esas páginas a donde le diga DocumentRoot. Pero de esa carpeta, ¿el servidor qué archivo muestra?

Ese parámetro se configura en ***/etc/apache2/mods-available/dir.conf*** donde aparece una lista nombres de páginas que el navegador intentará mostrar.

Por defecto existirá *index.html index.cgi index.pl index.php index.xhtml index.htm*. No obstante podemos se pueden poner tantas páginas como queramos separadas por un espacio en blanco, e irá buscando una tras otra hasta que encuentre alguna.

## *Instalación de certificados y acceso seguro con HTTPS*

Si queremos usar conexiones seguras mediante el protocolo HTPS para acceso a determinadas páginas, tendremos que configurar Apache para que cargue el módulo ssl , aunque en algunas versiones de Apache 2 suele venir integrado.

Comprobaremos si está o no instalado consultando el directorio ***/etc/apache2/mods-enabled***, y en el caso de que no esté lo habilitaremos con **a2enmod ssl**.

**a2enmod ssl**

Necesitaremos tener instalado además **OpenSSL** (***apt-get install openssl***), que se encarga de la implementación SSL para llevar a cabo transferencias seguras. Este módulo no se recomienda en configuraciones con más de un servidor virtual que utilicen el mismo puerto de escucha, ya que la conexión SSL del cliente se establece con una IP y puerto determinado del servidor. Apache no sabrá con qué servidor virtual debe utilizar SSL, escogiendo el primero que se encuentra, y no utilizará conexiones seguras para el resto de servidores virtuales.

Más información de por qué no se puede usar SSL con virtual hosts basados en nombre: *http:/ /httpd.apache.org/docs/2.0/ssl/ssl\_faq.html#vhosts*

Para corregir este problema se utiliza la extensión TLS denominada SNI (Server Name Indication o Indicación de Nombre de Servidor), necesitando tener instalado el paquete libapache2-mod-gnutls (para cargar el módulo gnutls en vez de ssl).

Más información sobre SNI en: http:/ / en. wikipedia. org/ wiki/ Server\_Name\_Indication

Se podría tener un servidor virtual principal para el sitio principal y que atiende en el puerto 80, y un servidor virtual para el sitio seguro que atiende en el **puerto 443** y que utiliza SSL.

Para ofrecer páginas de forma segura en un sitio web, el servidor necesita un certificado que apruebe su autentiticidad. Para generar un certificado autofirmado hay que ejecutar la instrucción **make-ssl-cert**, y se nos solicitará información para completar ese certificado. Tendremos que indicarle como parámetro el fichero con extensión “.pem”, que se generará al final de la ejecución y que contendrá las claves para efectuar el cifrado asimétrico por el servidor.

Es necesario habilitar el puerto de escucha mediante la directiva Listen 443. Además serán necesarias las siguientes directivas:

* **SSLRequireSSL**: se usa en el contenedor <Directory> y fuerza la utilización de SSL, evitando su desactivación.
* **SSLEngine**: habilita SSL (valor on) en un servidor virtual, si se deshabilitó en el servidor principal.
* **SSLCertificateFile**: indica la ruta del fichero (.pem) que contiene el certificado.

Para crear un certificado autofirmado que guardaremos en la carpeta ***/etc/apache2/certificados/certificado1.pem*** ejecutaremos el siguiente comando:

**# Creamos el directorio dónde almacenaremos los certificados:**

**mkdir -p /etc/apache2/certificados**

**# Generamos el fichero .pem que contendrá el certificado:**

**# Con esta instrucción o bien con:**

**make-ssl-cert /usr/share/ssl-cert/ssleay.cnf /etc/apache2/certificados/miweb.local.pem**

**openssl req $@ -new -x509 -days 365 -nodes –out /etc/apache2/certificados/miweb.local.pem –keyout /etc/apache2/certificados/miweb.local.pem**

**# Respondemos a las preguntas que se nos hacen en la generación del certificado.**

**# Damos permisos de lectura al certificado**

**chmod 600 /etc/apache2/certificados/miweb.local.pem**

**# Habilitamos el soporte SSL**

**a2enmod ssl**

**# Reiniciamos el servicio**

**service apache2 restart**

Tendremos que **editar el archivo /etc/apache2/ports.conf** para indicar que queremos que nuestro servidor web también escuche en el puerto 443. Lo más normal es crear un servidor virtual que escuche en el puerto 80 y el mismo pero escuchando en el puerto 443:

**# Editamos el fichero /etc/apache2/ports.conf:**

**nano /etc/apache2/ports.conf**

**# Añadimos los servidores virtuales:**

**Listen 80**

**Listen 443**

**NameVirtualHost \*:80**

**NameVirtualHost \*:443**

Ahora nos queda **revisar y configurar los contenedores de los sitios disponibles** para habilitar las opciones SSL. En el directorio /etc/apache2/sites-available revisaremos los contenedores:

**# Editamos el fichero /etc/apache2/sites-available/miweb.local**

**nano /etc/apache2/sites-available/miweb.local**

**# Comprobamos los contenedores y activamos las opciones correspondientes SSL:**

**# Comenzamos con el sitio SSL:**

**<VirtualHost \*:443>**

**ServerName miweb.local**

**DocumentRoot /var/www/miweb.local/htdocs**

**DirectoryIndex index.html**

**SSLEngine on**

**SSLCertificateFile /etc/apache2/certificados/miweb.local.pem**

**<Directory /var/www/miweb.local/htdocs>**

**Options Indexes FollowSymLinks MultiViews**

**AllowOverride none**

**Order allow,deny**

**Allow from all**

**SSLRequireSSL**

**</Directory>**

</VirtualHost>

<VirtualHost \*:80>

**ServerName miweb.local**

**DocumentRoot /var/www/miweb.local/htdocs**

**DirectoryIndex index.html**

**</VirtualHost>**

**# Si tenemos un sitio default-ssl y lo habilitamos, revisaremos en ese fichero**

**# las rutas del certificado y activamos estas opciones:**

**SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire**

**SSLCertificateFile /etc/ssl/certs/server.crt**

**SSLCertificateKeyFile /etc/ssl/private/server.key**

**# En el ejemplo anterior se permite el acceso a miweb.local en modo seguro y modo normal.**

**# Si quisiéramos crear dos webs diferentes para el acceso seguro o normal,**

**# simplemente modificaríamos los directorios DocumentRoot dónde se alojan los ficheros y ya está.**

Por último nos falta habilitar el nuevo sitio que hemos hecho:

**a2ensite miweb.local**

**# Reiniciamos el servidor apache para aplicar cambios:**

**service apache2 restart**

## *Comprobar que funciona*

Para comprobar que apache funciona perfectamente, desde el navegador de cualquier estación de trabajo de nuestro centro, debemos dirigirnos a ‘http://ip-del-servidor’. Si tenemos el DNS funcionando, podemos acceder al nombre que tiene el ordenador, visualizando la siguiente pantalla:



Si la prueba se hace desde el ordenador en el que está instalado el servidor web, podemos utilizar la dirección IP 127.0.0.1 o el nombre de equipo [*http://localhost*](http://localhost), ya que hacen referencia al mismo ordenador.

### Monitorización del servicio

Si queremos monitorizar el servicio, escribiremos el siguiente comando en la terminal:

**ps aux | grep apache**

### Utilización de puertos

También es posible ver qué puertos utiliza Apache, es decir, que puertos tiene abiertos el ordenador, utilizando el siguiente comando:

**cat /etc/services | grep http**

# Apache web server en Windows

El servidor web Apache puede ser instalado a su vez en sistemas operativos Windows. Aunque existen ciertas diferencias respecto a la versión de GNU/Linux el proceso es muy similar.

## *Instalación*

Los pasos para hacer la instalación son los siguientes:

1. Instala ***vcreditsx64*** (o ***vcreditsx86***, dependiendo del procesador de tu equipo). Se debería instalar la version del 2012 para evitar problemas.
2. Extrae los ficheros contenidos en el fichero ***httpd-2.4.10-win64-VC11.zip*** . Mueve o copia la carpeta ***Apache24*** a ***C:/***. ***C:/Apache24*** sera la carpeta de instalación en vez de *C:\Program Files\Apache Software Foundation\Apache2.2*.
3. Abre el terminal (**cmd**), entra en la carpeta **C:/Apache24/bin** y ejecuta el siguiente commando para instalar Apache como servicio:

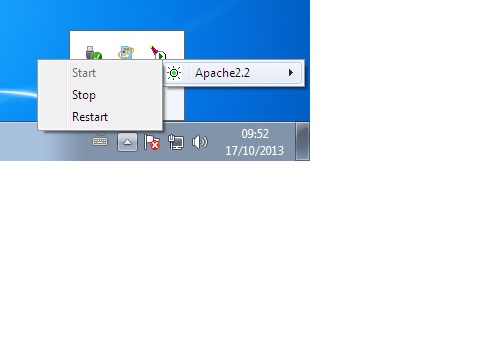
**C:/Apache24/bin> httpd.exe –k install**

1. Ejecuta ***ApacheMonitor.exe*** (también está en ***C:/Apache24/bin***). Aparecerá en la barra de tareas. Personaliza la barra de tareas para que Apache Monitor se muestre siempre.
2. Probablemente tengas que añadir una excepción en el Firewall de Winidows para permitir el acceso a los puertos de Apache. Ve al Firewall y selecciona: “***Permitir un programa o una característica a través de Firewall de Windows***”, busca ***httpd.exe*** (dentro de ***C:/Apache24/bin***) y añádelo como una excepción.

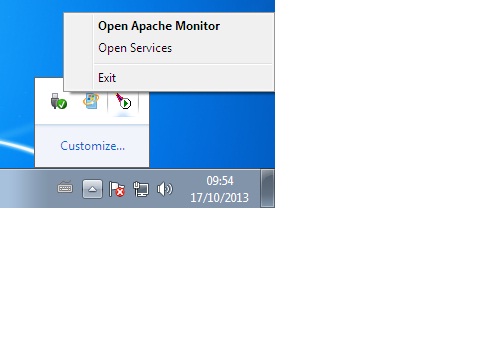
## *Iniciar y detener el servidor*

Por defecto el servidor web Apache está configurado para iniciar el servicio automáticamente cuando el equipo se enciende. Si el icono de la barra de tareas está en verde y “play”, el servicio está iniciado.

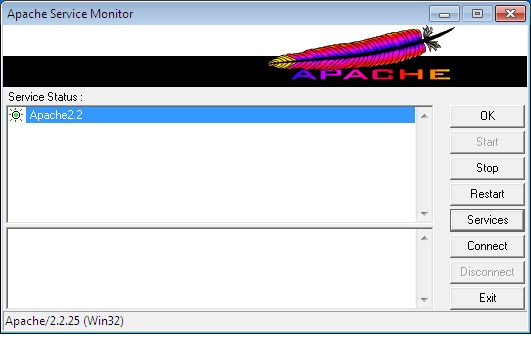
Podemos detenerlo, reiniciar o iniciar el servicio clicando en el icono de Apache y escogiendo la opción deseada.



Podemos abrir el Monitor de Apache clicando con el botón derecho en el icono de Apache y seleccionando la opción “Open Apache Monitor”



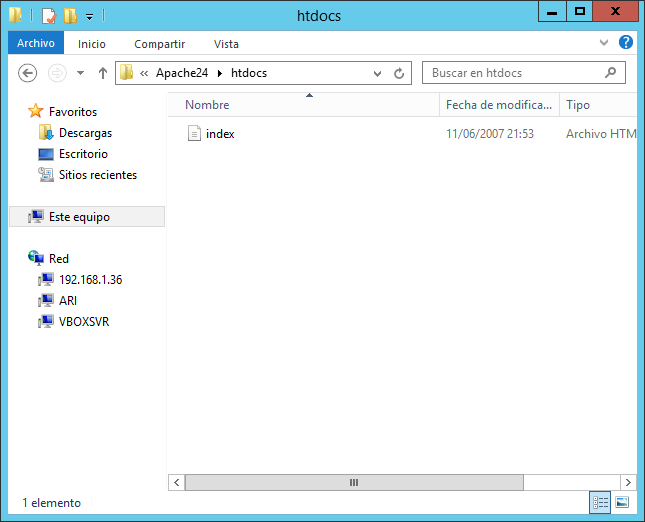
Una vez abierto el monitor de Apache podemos ejecutar la opción deseada.



## *Configuración*

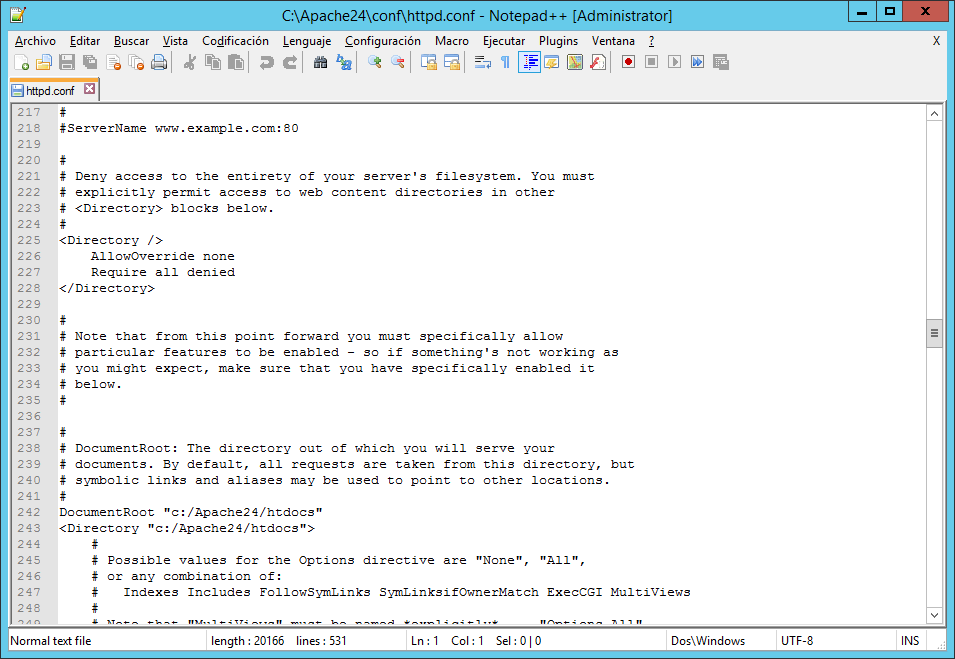
### Directorio donde se almacena el sitio web

El directorio por defecto donde se almacena nuestro sitio web es ***C:/Apache24/htdocs***, y el nombre de la página de inicio es index.html.

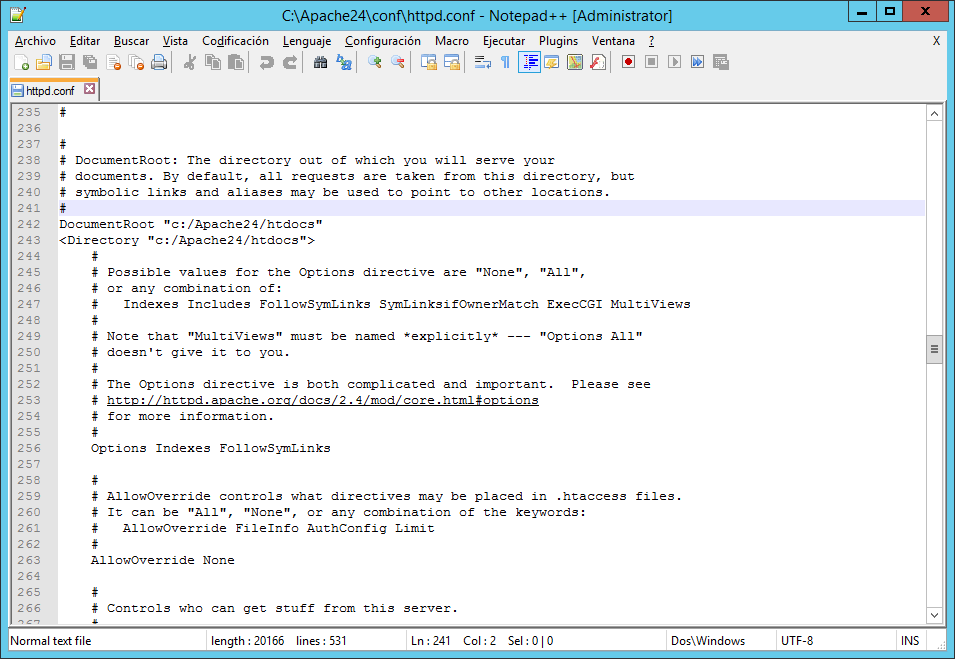


Evidentemente podemos cambiar de directorio. En Windows solamente trabajaremos sobre el fichero de configuración que se encuentra en ***C:/Apache24/conf/httpd.conf***

Para cambiar el directorio principal vamos a la línea ***DocumentRoot*** y la cambiamos:



Si cambiamos el directorio raíz deberíamos cambiar la ruta de la etiqueta ***Directory*** también:



### Permisos y aliases

La configuración de permisos cambia en la versión 2.4 de Apache con respecto a las versiones anteriores. Este tipo de líneas están depreciadas en las nuevas versiones:

**Order deny,allow**

**Allow from si2.org**

**Deny from all**

Ahora se utiliza la directiva “Require”. Veamos algunos ejemplos:

* Permitir acceso desde cualquier sitio:

**Require all granted**

* Denegar acceso desde cualquier sitio:

**Require all denied**

* Permitir acceso solo desde una IP:

**Require ip 192.168.2.25**

* Permitir acceso desde una red:

**Require ip 192.168.2.0/24**

o

**Require ip 192.168.2**

Presta atención al hecho de que los elementos no comunes de las direcciones pueden ser omitidos. Por ejemplo, para permitir el acceso solo desde las redes privadas se podría escribir algo así:

**Require ip 192.168 172.20 10**

* Permitir el acceso desde cualquier sitio menos desde una red:

**Require not ip 192.168.2.0/24**

Ten en cuenta que solo añadiendo “not” después de “Require” conseguimos el resultado opuesto.

* Permitir acceso solo desde un dominio determinado:

**Require host si2.org**

Ten en cuenta que hemos cambiado el término “ip” por “host” porque estamos especificando un nombre de dominio/equipo en vez de una dirección IP.

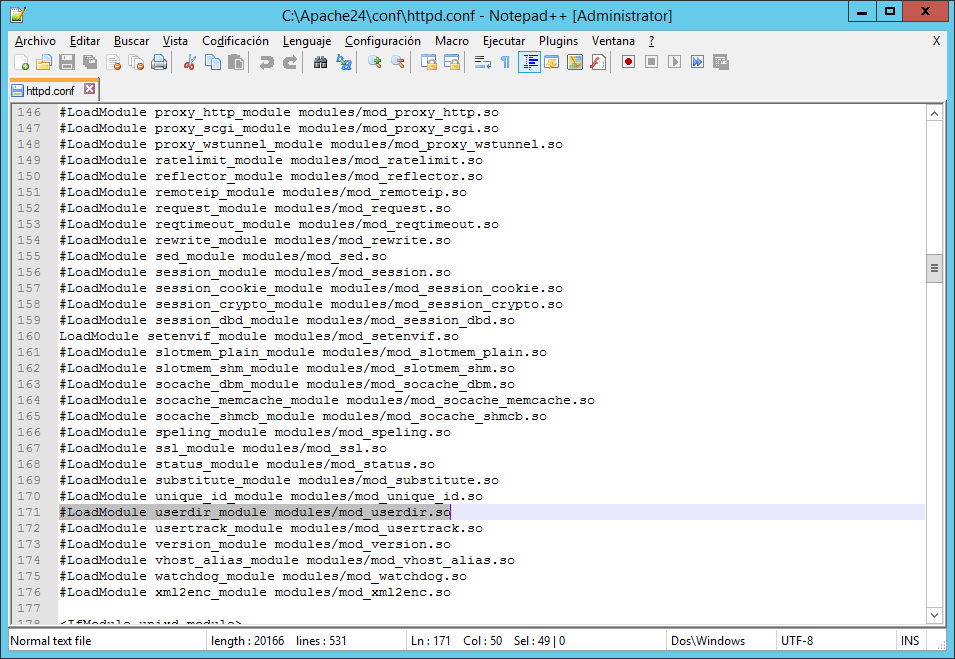
* Combinación de diferentes características:

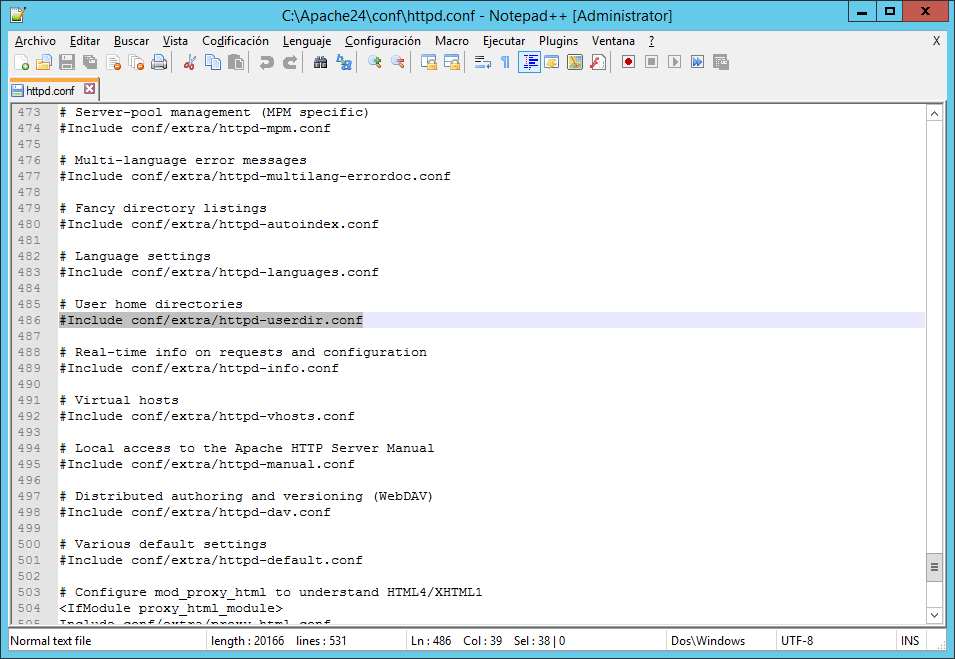
**Require not ip 172.20.14**

**Require not host si2.org si2.com**

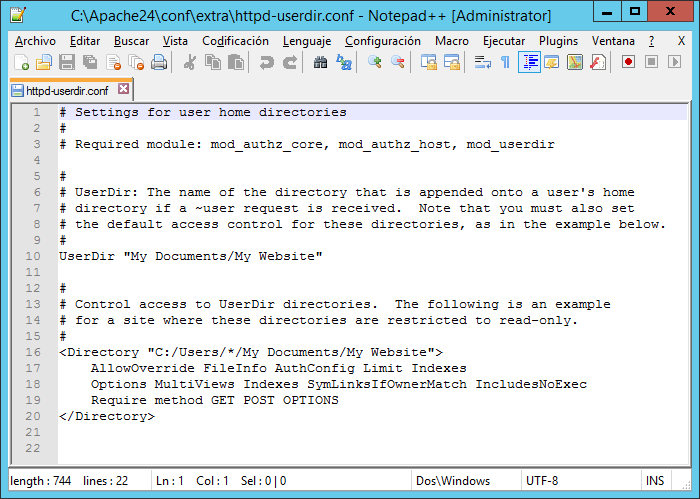
### Páginas web de usuarios

Si queremos configurar el servidor para alojar sitios web de usuarios debemos descomentar las siguientes líneas del fichero ***C:/Apache24/conf/httpd.conf*** :





Si queremos configurar permisos de los sitios web de usuarios lo haremos en el fichero ***C:/Apache24/conf/extra/httpd-userdir.conf***:



### Creación de un sitio web en un puerto diferente

Abre el fichero ***C:/Apache24/conf/httpd.conf*** y añade la siguiente línea

**Listen 8080**

Y descomenta la siguiente línea :

**Include conf/extra/httpd-vhosts.conf**

A continuación abre el fichero ***C:/Apache24/conf/extra/httpd-vhosts.conf*** y cambia/añade las siguientes líneas.

**NameVirtualHost \*:8080**

y

**<VirtualHost \*:8080>**

**DocumentRoot “C:/path”**

**</VirtualHost>**

### Petición de autenticación al acceder a un sitio web.

*Abre el fichero C:/Apache24/conf/extra/httpd-vhosts.conf, añade el directorio y la opción AllowOverride AuthConfig:*

**<Directory “C:/path”>**

**Options Indexes FollowSymLinks**

**AllowOverride AuthConfig**

**Order deny,allow**

**Allow from all**

**</Directory>**

Crea el fichero ***.htaccess*** dentro del path con el siguiente contenido

**AuthType Basic**

**AuthName “Autenticación requerida para acceder a la intranet”**

**AuthUserFile “C:/path/.htpassword”**

**Require valid-user**

Crea un fichero ***.htpassword*** el el path correspondiente y déjalo vacío.

Abre un terminal (***cmd***), y muévete a la carpeta ***C:/Apache24/bin***. Una vez ahí, ejecuta la siguiente orden para cada usuario:

**htpasswd “C:/path/.htpassword” julen**

**New password: (Enter password for julen)**

Re-type new password: (Confirm password for julen)

## *Testeando el servidor*

Para testear el funcionamiento del servidor podemos seguir las instrucciones del capítulo de GNU/Linux e intentar acceder al servidor web desde la propia máquina o desde otro equipo escribiendo la dirección IP o su nombre completo de dominio.

### Errores

Por defecto los errores se almacenarán en el fichero ***C:/Apache24/logs/error.log***.

