Instalación de Servidor Web Apache en Debian

Sumario

Instalacion de Servidor Web Apache en Debian

```
# Instalación de Apache2
apt-get install apache2

# Arranque y parada del servicio:
service apache2 start | stop | status | restart | reload

# Rutas de ficheros de configuración:
/etc/apache2

/etc/apache2/apache2.conf: configuración de directorio raiz por defecto, logs, procesos, etc.
/etc/apache2/httpd.conf: configuraciones de usuario.
/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/sites-available/: configuración de sitios
/etc/apache2/sites-enabled/: configuración de los sitios que están habilitados.
```

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.

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.

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.

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 usertrack
a2enmod usertrack

# Para deshabilitar un módulo en Apache 2 se usa: a2dismod
# Deshabilitamos por ejemplo el módulo usertrack
a2dismod usertrack
```

Proteger directorios

Hay dos formas de proteger directorios en Apache:

Antes de ejecutar cualquier de los dos métodos, hay que asegurarse que el módulo auth_basic está habilitado en: **/etc/apache2/mods-enabled**.

1.- Creando fichero .htaccess en el directorio a proteger:

```
# Para proteger un directorio en Apache2.
# Dentro del directorio que queremos proteger creamos un fichero .htaccess con este contenido:
 AuthName "Acceso Restringido a este recurso"
AuthType Basic
AuthUserFile /usr/local/apache2/etc/.htpasswd
Require valid-user

# Si queremos permitir el acceso sólo a un usuario específico, pondremos:
Require user nombreusuario

# Crearemos el fichero de contraseñas, y lo pondremos
# en un directorio fuera de la raíz web, para más seguridad:
mkdir -p /usr/local/apache2/etc

# Creamos el fichero de contraseñas .htpasswd (lo ponemos oculto para dar más seguridad),
# con un usuario al que llamamos admin (nos pedirá la contraseña al pulsar ENTER).
# Creación del fichero de contraseñas:
htpasswd -c /usr/local/apache2/etc/.htpasswd admin

# Para añadir más usuarios al fichero de contraseñas (por ejemplo antonio):
htpasswd /usr/local/apache2/etc/.htpasswd antonio

# Editar el directorio virtual en el fichero de configuración del Apache2 a proteger en:
```

```
nano /etc/apache2/sites-available/default

# Activar la opción AllowOverride All:
    <Directory /var/www/>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride All

    Order allow,deny
    allow from all
    </Directory>

# Por último reiniciar el servicio Apache:
service apache2 restart
```

2.- Editando la configuración del sitio web: En lugar de crear un fichero .htaccess con la configuración de autenticación, editaremos la configuración del sitio web.

```
# Crearemos el fichero de contraseñas, y lo pondremos
# en un directorio fuera de la raíz web, para más seguridad:
mkdir -p /usr/local/apache2/etc

# Creamos el fichero de contraseñas .htpasswd (lo ponemos oculto para dar más seguridad),
# con un usuario al que llamamos admin (nos pedirá la contraseña al pulsar ENTER).
# Creación del fichero de contraseñas:
htpasswd -c /usr/local/apache2/etc/.htpasswd admin

# Para añadir más usuarios al fichero de contraseñas (por ejemplo antonio):
htpasswd /usr/local/apache2/etc/.htpasswd antonio

# Editamos el sitio web que queremos proteger y el directorio en particular:
nano /etc/apache2/sites-available/default

#Añadimos las directivas necesarias en el directorio a proteger:
AuthType Basic
AuthName "Acceso Restringido a este recurso"
AuthUserFile /usr/local/apache2/etc/.htpasswd
Require valid-user

# Si queremos permitir el acceso sólo a un usuario específico, pondremos:
Require user pepe antonio maria
```

Información adicional sobre fichero .htaccess

```
# Directivas para redireccionar todas las peticiones HTTP
# a peticiones HTTPS

RewriteEngine on
DirectoryIndex index.php
RewriteCond %{SERVER_PORT} ^80$
RewriteRule .* https://%{SERVER_NAME}%{REQUEST_URI} [R,L]

# Directiva para redireccionar los .html a sus correspondientes .php
# pasando el parámetro definido como inc (Query)
RewriteRule ^([^/]+).html$ index.php?inc=$1 [QSA,L]
```

Para más información sobre este tipo de directivas, visitar: http://www.askapache.com/htaccess/ssl-example-usage-in-htaccess.html

Servidores Virtuales en Apache

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

Para poder hacer ésto, 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
DocumentRoot /var/www/servidor1.local
ServerAdmin 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
```

```
# Lo habilitaremos con:
a2ensite www.pruebas.local

# Para deshabilitarlo lo haremos con:
a2dissite www.pruebas.local

# Reniciaremos el servicio a continuación:
service apache2 restart
```

Estructura web recomendada para múltiples sitios web

En muchas organizaciones mantienen una estructura determinada a la hora de colocar aplicaciones en sus servidores web. Aquí va una guía de cómo organizar y estructurar el servicio web:

1.- Modificar la raíz del sitio web, como si fuera una aplicación web: en nuestro caso al instalar Debian la raíz del sitio web es /**var**/**www**. Lo que haremos es crear una carpeta llamada **htdocs** por ejemplo y modificaremos la directiva **DocumentRoot** en el fichero /**etc**/**apache2**/**sites-available**/**default** al directorio /**var**/**www**/**htdocs**.

```
nano /etc/apache2/sites-available/default

DocumentRoot /var/www/htdocs

 <Directory /var/www/htdocs/>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    allow from all
</Directory>
```

2.- Si queremos instalar una nueva aplicación web en nuestro servidor web haremos:

- Crear el directorio principal de la aplicación en /var/www/, siguiendo esta estructura:

```
mkdir -p /var/www/aplicacion/htdocs
```

- Dentro de la carpeta htdocs de la aplicación, copiaremos todos los archivos de la misma.

3.- Una vez copiados todos los archivos tendremos que crear el fichero de configuración de Apache para esa aplicación. Ese fichero lo crearemos en el directorio /etc/apache2/sites-available/.

Un ejemplo de configuración para una web de programación de aplicaciones podría ser:

```
# Esta recomendación es demasiado extensa. Se cita a modo de ejemplo, pero no se
# utilizará en ningún ejercicio.
nano /etc/apache2/sites-available/aplicacion

<Directory /var/www/htdocs/aplicacion>
        Order allow,deny
        Allow from all
            Options +FollowSymLinks
        Allowoverride all
        RewriteEngine on
        php_admin_value open_basedir /var/www/htdocs/aplicacion:/var/www/properties/:
/var/www/aplicacion:/var/www/logs/:/tmp/
</Directory>
<Location /aplicacion>
        SetEnv PROPDIR /var/www/aplicacion/properties
        SetEnv RUTAAPP /var/www/aplicacion/htdocs
        SetEnv FILEDIR /var/www/aplicacion/files
</Location>
```

4.- Por último quedaría realizar un enlace simbólico desde /var/www/htdocs/ al directorio htdocs de nuestra aplicación:

```
# Creamos el enlace simbólico desde la carpeta raíz de nuestro servidor web a nuestra aplicación
cd /var/www/htdocs
ln -s /var/www/aplicacion/htdocs/ aplicacion
```

```
# De esta forma al poner http://ipservidor/aplicacion nos conectará
# con la carpeta /var/www/aplicacion/htdocs

# Reiniciamos el servicio web
service apache2 restart
```

De esta forma tendremos físicamente nuestra aplicación web, fuera del directorio raíz del servidor.

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
```

Ampliación de información sobre VirtualHost en Apache

http://httpd.apache.org/docs/2.0/vhosts/examples.html

--Veiga 19:05 13 abr 2012 (CEST)