REDES: Capítulo 8

APACHE WEB SERVER

Capítulo 07 - APACHE WEBSERVER

ÍNDICE

Capítulo 07 - APACHE WEBSERVER

Servicio web

El **Servidor web** es un programa que se encarga de transferir archivos de texto de un equipo llamado “servidor” a otro llamado “cliente” usando un protocolo llamando http.

Los archivos de texto transferidos están escritos generalmente en el lenguaje de marcado HTML porque este lenguaje es interpretado por el navegador.

HTML no es un lenguaje de programación sino lo que se conoce como un **código de marcación**.

No utiliza sentencias lógicas sino “tags” que indican dónde irán ubicados los elementos en la página. Si queremos que nuestros sitios web contengan sentencias lógicas que puedan evaluar acciones del usuario o consultar bases de datos tendremos que trabajar con lenguajes de programación como PHP.

Al trabajar con este tipo de lenguajes el cliente de todas formas no recibe los archivos con el código fuente del lenguaje que hemos usado sino un archivo de texto plano que el cual es interpretado por el servidor, siendo este último el que escribe el texto en HTML luego de haber interpretado la sintaxis. Este resultado es devuelto al cliente en el lenguaje que el entiende.

El proceso que se lleva a cabo es el siguiente: el servidor web tiene los módulos adecuados para traducir de PHP a código a HTML de manera tal que el cliente pueda ver la página web solicitada.

¿Qué es un sitio web?

Antes de empezar a trabajar con sitios tenemos que tener una definición precisa de que es:

> Un sitio web es un directorio con archivos que son accedidos a través de una dirección comúnmente llamada URL, (www.sitio_ejemplo.com), la cual es enviada por red desde el servidor al navegador cliente.

Cuando el webserver interpreta la dirección que el cliente le solicita, inmediatamente el va al directorio asociado con la URL y transfiere un archivo que por defecto se llama index.html, cada vez que el cliente hace click en un link el webserver transfiere un archivo si esto fuera necesario o nos lleva a otra parte de la página según sea el caso.

> De aquí se desprende que podemos tener muchos directorios en un mismo servidor que son accedidos desde distintas URLs.

El trabajo de Apache, nuestro webserver, es identificar cual es la URL pedida por el cliente llegar hasta el directorio asociado a esta y transferir los archivos requeridos.

Instalación del webserver Apache

Para instalar apache vamos a usar el comando apt-get o aptitude de la siguiente manera:

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

o bien:

```
# aptitude install apache2
```

Cuando trabajamos con apache tenemos que tener en cuenta los directorios involucrados en la configuración. Podemos dividir las configuraciones en los siguientes directorios:

· /etc/apache2: directorio de configuración principal de Apache.
· /etc/apache2/sites-available: directorio de configuración de VirtualHost disponibles.
· /etc/apache2/sites-enable: directorio de VirtualHost activos.
· /etc/apache2/conf-available: directorio de configuraciones especiales de Apache.
· /etc/apache2/conf-enable: directorio de configuraciones activas.
· /etc/apache2/mods-available: módulos disponibles en Apache2.
· /etc/apache2/mods-enabled: módulos activos de Apache.
· /var/log/apa2che2: directorio donde se guardan los logs.

Configuración básica de Apache

El archivo de configuración principal de Apache es /etc/apache2/apache2.conf, este archivo contiene las directivas de configuración básicas del webserver:

```
# This is the main Apache server configuration file.  It contains the
# configuration directives that give the server its instructions.
# See http://httpd.apache.org/docs/2.4/ for detailed information about
# the directives and /usr/share/doc/apache2/README.Debian about De-
bian specific
# hints.
#
#
# Summary of how the Apache 2 configuration works in Debian:
# The Apache 2 web server configuration in Debian is quite different to
# upstream's suggested way to configure the web server. This is because
Debian's
# default Apache2 installation attempts to make adding and removing
modules,
# virtual hosts, and extra configuration directives as flexible as pos-
sible, in
```

```
# order to make automating the changes and administering the server
as easy as
# possible.

# It is split into several files forming the configuration hierarchy
outlined
# below, all located in the /etc/apache2/ directory:
#
#    /etc/apache2/
#    |-- apache2.conf
#    |    `--  ports.conf
#    |-- mods-enabled
#    |    |-- *.load
#    |    `-- *.conf
#    |-- conf-enabled
#    |    `-- *.conf
#    `-- sites-enabled
#         `-- *.conf
#
#
# * apache2.conf is the main configuration file (this file). It puts the
pieces
#   together by including all remaining configuration files when star-
ting up the
#   web server.
#
# * ports.conf is always included from the main configuration file. It is
#   supposed to determine listening ports for incoming connections
which can be
```

```
#   customized anytime.
#
# * Configuration files in the mods-enabled/, conf-enabled/ and si-
tes-enabled/
#   directories contain particular configuration snippets which manage
modules,
#   global configuration fragments, or virtual host configurations,
#   respectively.
#
# They are activated by symlinking available configuration files from their
#   respective *-available/ counterparts. These should be managed by
using our
#   helpers a2enmod/a2dismod, a2ensite/a2dissite and a2enconf/a2dis-
conf. See
#   their respective man pages for detailed information.
#
# * The binary is called apache2. Due to the use of environment va-
riables, in
#   the default configuration, apache2 needs to be started/stopped with
#   /etc/init.d/apache2 or apache2ctl. Calling /usr/bin/apache2 di-
rectly will not
#   work with the default configuration.

# Global configuration
#


#
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
#
# NOTE!  If you intend to place this on an NFS (or otherwise network)
# mounted filesystem then please read the Mutex documentation (available
# at <URL:http://httpd.apache.org/docs/2.4/mod/core.html#mutex>);
# you will save yourself a lot of trouble.
```

```
#
# Do NOT add a slash at the end of the directory path.
#
#ServerRoot “/etc/apache2”

#
# The accept serialization lock file MUST BE STORED ON A LOCAL DISK.
#
Mutex file:${APACHE_LOCK_DIR} default

#
# PidFile: The file in which the server should record its process
# identification number when it starts.
# This needs to be set in /etc/apache2/envvars
#
PidFile ${APACHE_PID_FILE}

#
# Timeout: The number of seconds before receives and sends time out.
#
Timeout 300

#
# KeepAlive: Whether or not to allow persistent connections (more than
# one request per connection). Set to “Off” to deactivate.
#
KeepAlive On

#
# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an unlimited amount.
# We recommend you leave this number high, for maximum performance.
#
```

```
MaxKeepAliveRequests 100

#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection.
#
KeepAliveTimeout 5

# These need to be set in /etc/apache2/envvars
User ${APACHE_RUN_USER}
Group ${APACHE_RUN_GROUP}

#
# HostnameLookups: Log the names of clients or just their IP addresses
# e.g., www.apache.org (on) or 204.62.129.132 (off).
# The default is off because it'd be overall better for the net if people
# had to knowingly turn this feature on, since enabling it means that
# each client request will result in AT LEAST one lookup request to the
# nameserver.
#
HostnameLookups Off

# ErrorLog: The location of the error log file.
# If you do not specify an ErrorLog directive within a <VirtualHost>
# container, error messages relating to that virtual host will be
# logged here.  If you *do* define an error logfile for a <VirtualHost>
# container, that host's errors will be logged there and not here.
#
ErrorLog ${APACHE_LOG_DIR}/error.log
```

```
#
# LogLevel: Control the severity of messages logged to the error_log.
# Available values: trace8, ..., trace1, debug, info, notice, warn,
# error, crit, alert, emerg.
#It is also possible to configure the log level for particular modules,e.g.
# “LogLevel info ssl:warn”
#
LogLevel warn

# Include module configuration:
IncludeOptional mods-enabled/*.load
IncludeOptional mods-enabled/*.conf

# Include list of ports to listen on
Include ports.conf

# Sets the default security model of the Apache2 HTTPD server. It does
# not allow access to the root filesystem outside of /usr/share and /var/www.
# The former is used by web applications packaged in Debian,
# the latter may be used for local directories served by the web server. If
# your system is serving content from a sub-directory in /srv you must allow
# access here, or in any related virtual host.
<Directory />
     Options FollowSymLinks
     AllowOverride None
     Require all denied
</Directory>
```

```
<Directory /usr/share>
     AllowOverride None
     Require all granted
</Directory>

<Directory /var/www/>
     Options Indexes FollowSymLinks
     AllowOverride None
     Require all granted
</Directory>

#<Directory /srv/>
#    Options Indexes FollowSymLinks
#    AllowOverride None
#    Require all granted
#</Directory>

# AccessFileName: The name of the file to look for in each directory
# for additional configuration directives.See also the AllowOverride
# directive.
#
AccessFileName .htaccess

#
# The following lines prevent.htaccess and.htpasswd files from being
# viewed by Web clients.
#
<FilesMatch “^\.ht”>
     Require all denied
</FilesMatch>
```

```
#
# The following directives define some format nicknames for use with
# a CustomLog directive.
#
# These deviate from the Common Log Format definitions in that theyuse %O
# (the actual bytes sent including headers) instead of %b(the size of the
# requested file), because the latter makes it impossible to detectpartial
# requests.
#
# Note that the use of %{X-Forwarded-For}i instead of %h is not re-
commended.
# Use mod_remoteip instead.
#
LogFormat “%v:%p %h %l %u %t \”%r\” %>s %O \”%{Referer}i\” \”%{User-
Agent}i\”” vhost_combined
LogFormat “%h %l %u %t \”%r\” %>s %O \”%{Referer}i\” \”%{User-Agent}
i\”” combined
LogFormat “%h %l %u %t \”%r\” %>s %O” common
LogFormat “%{Referer}i -> %U” referer
LogFormat “%{User-agent}i” agent

# Include of directories ignores editors’ and dpkg’s backup files,
# see README.Debian for details.

# Include generic snippets of statements
IncludeOptional conf-enabled/*.conf

# Include the virtual host configurations:
IncludeOptional sites-enabled/*.conf

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Las variables del archivo las vamos a comentar cuando configuremos nuestros propios VirtualHost.

Configuración inicial de Apache

Apache viene configurado para servir el sitio cuya URL localhost está definida en el archivo /etc/hosts el cual está asociado con la dirección de red 127.0.0.1.

Para esta dirección de red no necesitamos tener configurado DNS, ya que Linux para resolver esta dirección utiliza el archivo /etc/hosts el cual tiene la siguiente sintaxis:

```
127.0.0.1  localhost
127.0.1.1  equipo1
```

Esto indica que la dirección 127.0.0.1 (nuestra dirección de loopback) apunta a localhost.

Los desarrolladores de Apache aprovecharon esta dirección y crearon el archivo **/etc/apache2/sites-available/000-default.conf** y para que este VirtualHost funcione crearon el link de activacion en **/etc/apache2/sites-enable/000-default.conf**

Probando la configuración

Una vez instalado vamos a levantar el servicio para ver si funciona:

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

Una vez que lo hemos hecho levantamos el navegador y tipiamos la dirección localhost. Si nos nuestra una página de test el servicio funciona correctamente y podremos seguir trabajando en nuestra configuración.

Archivo de configuración apache2.conf

El archivo apache2.conf está dividido en tres partes:
· Global: incluye las variables principales de Apache.
· Main: en la cual se configura el VirtualHost por defecto.
· VirtualHost: la parte destinada a los VirtualHost.

La primera variable indica dónde se encuentra la configuración principal de Apache2:

```
ServerRoot “/etc/apache2”
```

La segunda variable nos dice dónde está el archivo de lockeo de Apache:

```
LockFile /var/lock/apache2/accept.lock
```

La variable PidFile nos muestra dónde se almacena toda la información sobre el proceso principal de apache2:

```
PidFile /var/run/apache2.pid
```

La variable Timeout nos dice cuántos segundos espera Apache antes de enviar un la señal de conexión cerrada:

```
Timeout 300
```

KeepAlive es la variable que determina si se permite o no más de una conexión por IP (cuando varios equipos navegan a través de un proxy).

A estas conexiones se las conoce como persistentes. Si no las permitimos, los clientes que naveguen a través de un Servidor Proxy no podrán visitar los sitios que deseen. Si queremos desactivarlo podemos setear Off.

```
KeepAlive On
```

La **variable MaxKeepAliveRequests** define el máximo número de conexiones persistentes que están permitidas.

Si queremos que sean ilimitadas el valor a setear tendrá que ser 0. Se recomienda dejarlo en un valor alto para una mejor performance:

```
MaxKeepAliveRequests 100
```

KeepAliveTimeout es la variable que define el número de segundos de espera para la siguiente conexión desde el mismo cliente:

```
KeepAliveTimeout 15
```

Server apache2 está configurado para trabajar con programación multihilos. En esta sección podemos ver cómo es el manejo de hilos por parte de Apache2:

```
## Server-Pool Size Regulation (MPM specific)
# prefork MPM
# StartServers .... Número de procesos que se levantan por defecto
# MinSpareServers . Minimo número de procesos que se esperan.
# MaxSpareServers . Maximo número de procesos esperados
# MaxClients ..... Maximo núnero de procesos para cada server levantado
# MaxRequestsPerChild . Maximo número de procesos hijos para cada servidor
<IfModule prefork.c>
     StartServers        5
     MinSpareServers     5
     MaxSpareServers     10
     MaxClients          20
     MaxRequestsPerChild 0
</IfModule>
```

```
Configuración del segundo hilo.
     <IfModule worker.c>
          StartServers        2
          MaxClients          150
          MinSpareThreads     25
          MaxSpareThreads     75
          ThreadsPerChild     25
          MaxRequestsPerChild   0
     </IfModule>

Configuración del tercer hilo.
     <IfModule perchild.c>
          NumServers          5
          StartThreads        5
          MinSpareThreads     5
          MaxSpareThreads     10
          MaxThreadsPerChild  20
          MaxRequestsPerChild   0
          AcceptMutex fcntl
     </IfModule>
```

Las variables que veremos a continuación son las que determinan a nombre de quién corre el proceso de Apache2:

```
User www-data
Group www-data
```

Veamos a continuación cómo se indica a dónde y siguiendo qué método se deben guardar los logs dentro del archivo /var/log/access_log:

```
LogFormat “%h %l %u %t \”%r\” %>s %b \”%{Referer}i\” \”%{User-
Agent}i\”” combined
LogFormat “%h %l %u %t \”%r\” %>s %b” common
LogFormat “%{Referer}i -> %U” referer
LogFormat “%{User-agent}i” agent
```

Los errores se guardarán en el archivo que se indica en la variable ErrorLog:

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

Veamos a continuación cómo podemos incluir nuevos módulos a nuestro Servidor Apache:

```
Include /etc/apache2/mods-enabled/*.load
Include /etc/apache2/mods-enabled/*.conf
```

El archivo en donde podremos colocar nuestros módulos es:

```
Include /etc/apache2/httpd.conf
```

La lista de puertos habilitados también debe ser incluida de la siguiente forma:

```
Include /etc/apache2/ports.conf
```

Para la configuración del VirtualHost usaremos la siguiente línea de inclusión:

```
Include /etc/apache2/conf.d/[^.#]*
```

Si queremos agregar íconos tendremos que ubicarlos en el siguiente directorio:

```
Alias /icons/ “/usr/share/apache2/icons/”
<Directory “/usr/share/apache2/icons”>
    Options Indexes MultiViews
    AllowOverride None Order
    allow,deny Allow from all
</Directory>
```

Los archivos de error que se muestran en pantalla cuando no puede alcanzarse una página ó hay algún problema en el servidor están definidos en el siguiente lugar:

```
#ErrorDocument 500 “The server made a boo boo.”
#ErrorDocument 404 /missing.html
#ErrorDocument 404 “/cgi-bin/missing_handler.pl”
#ErrorDocument 402 http://www.example.com/subscription_info.html
```

Apache trabaja en el directorio *errors* pero nosotros podemos incluir nuestros propios errores agregando un alias. Nos advertirá que los coloquemos todos juntos y que usemos un estándar para no generar confusiones.

```
# # Putting this all together, we can Internationalize error responses.
# # We use Alias to redirect any /error/HTTP_<error>.html.var response to
# our collection of by-error message multi-language collections. We use
# includes to substitute the appropriate text.
# # You can modify the messages' appearance without changing any of the
# default HTTP_<error>.html.var files by adding the line;
#
# Alias /error/include/ "/your/include/path/"
#
# which allows you to create your own set of files by starting with the
# /usr/local/apache2/error/include/ files and
# copying them to /your/include/path/, even on a per-VirtualHost basis.
#
<IfModule mod_negotiation.c>
     <IfModule mod_include.c>
          Alias /error/ "/usr/share/apache2/error/"
     <Directory "/usr/share/apache2/error">
          AllowOverride None
          Options IncludesNoExec
          AddOutputFilter Includes html
          AddHandler type-map var
          Order allow,deny
          Allow from all
          LanguagePriority en es de fr
          ForceLanguagePriority Prefer Fallback
     </Directory>
     ErrorDocument 400 /error/HTTP_BAD_REQUEST.html.var
     ErrorDocument  401 /error/HTTP_UNAUTHORIZED.html.var
     ErrorDocument  403 /error/HTTP_FORBIDDEN.html.var
     ErrorDocument  404 /error/HTTP_NOT_FOUND.html.var
```

```
    ErrorDocument  405 /error/HTTP_METHOD_NOT_ALLOWED.html.var
    ErrorDocument  408 /error/HTTP_REQUEST_TIME_OUT.html.var
    ErrorDocument  410 /error/HTTP_GONE.html.var
    ErrorDocument  411 /error/HTTP_LENGTH_REQUIRED.html.var
    ErrorDocument  412 /error/HTTP_PRECONDITION_FAILED.html.var
    ErrorDocument  413 /error/HTTP_REQUEST_ENTITY_TOO_LARGE.html.var
    ErrorDocument  414 /error/HTTP_REQUEST_URI_TOO_LARGE.html.var
    ErrorDocument  415 /error/HTTP_SERVICE_UNAVAILABLE.html.var
    ErrorDocument  500 /error/HTTP_INTERNAL_SERVER_ERROR.html.var
    ErrorDocument  501 /error/HTTP_NOT_IMPLEMENTED.html.var
    ErrorDocument  502 /error/HTTP_BAD_GATEWAY.html.var
    ErrorDocument  50310 /error/HTTP_SERVICE_UNAVAILABLE.html.var
    ErrorDocument  506 /error/HTTP_VARIANT_ALSO_VARIES.html.var
    </IfModule>
</IfModule>
```

La variable que sigue es muy importante porque nos indica cómo debe llamarse la página que el servidor tiene que transferir por defecto:

```
    DirectoryIndex index.html index.cgi index.pl index.php index.xhtml
```

Con la variable UserDir podremos permitir que cada usuario que trabaje en el servidor tenga su sitio personal en su home directory:

```
    # UserDir is now a module
    #UserDir public_html
    #UserDir disabled root
    #<Directory /home/*/public_html>
    # AllowOverride FileInfo AuthConfig Limit
    # Options Indexes SymLinksIfOwnerMatch IncludesNoExec
    #</Directory>
```

La **variable AccessFileName** es muy interesante ya que nos va a permitir trabajar con un archivo llamado .htaccess.

Este archivo se procesa antes de transferir el DirectoryIndex:

```
AccessFileName .htaccess
```

Esta variable impide que todo lo que comienza con .ht se transfiera:

```
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
</Files>
```

Veamos el directorio dónde vamos a configurar nuestros VirtualHost:

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

Creación de un VirtualHost Local

Para **crear un sitio web local** que sólo podamos ver desde nuestro equipo, primero tenemos que **modificar el archivo /etc/hosts**.

A este archivo le agregaremos el nombre y el número IP de nuestro equipo. Suponga-mos que el sitio que queremos crear se llama "www.sitio.com.ar".

Entonces el archivo hosts debería verse de la siguiente forma:

```
127.0.0.1 localhost.localdomain localhost
10.10.3.1 equipo1.red.carreralinux.com.ar equipo1
10.10.3.1 www.sitio.com.ar
```

La tercera línea del archivo apunta la dirección 10.10.3.1 al sitio www.sitio.com.ar. Esto es importante ya que nuestro equipo antes de ir a buscar un sitio en Internet lo buscará localmente.

El orden de búsqueda está definido en el archivo /etc/host.conf derivado desde el archivo /etc/nsswitch.conf.

Crear el directorio del sitio

Es hora de ir al directorio que aloja los sitios en general, el directorio suele ser /var/www en algunas distribuciones /var/www/html. En Apache este directorio está siendo apuntado por la variable DocumentRoot.

En este directorio vamos a crear un archivo llamado index.html. Este archivo está definido en el archivo de configuración en la variable DirectoryIndex.

Veamos paso a paso todo el proceso. Vamos al directorio /var/www:

```
cd /var/www/html
```

Creamos un directorio llamado sitio_local:

```
mkdir sitio_local
```

Cambiamos al directorio recientemente creado:

```
cd sitio_local
```

Luego creamos un archivo llamado index.html y le ponemos el siguiente contenido:

```
<html>
    <head>
        <title> Este es nuestro Sitio local</title>
    </head>
    <body>
        <h1 align=”center”> Bienvenidos al sitio local </h1>
    </body>
</html>
```

El próximo paso es declarar el directorio como un VirtualHost.

Declarar un directorio como VitrualHost

Tenemos que asociar www.sitio.com.ar con la dirección de red 10.10.3.1. Para lograrlo vamos a ir al directorio /etc/apache2/sites-available. Allí tenemos que colocar las definiciones de sitios:

```
cd /etc/apache2/sites-available
```

Creamos un archivo llamado sitios.conf que contendrá la información de nuestro sitio local que estamos elaborando:

```
<VirtualHost *:80>
    ServerAdmin webmaster@sitio.com.ar
    ServerName www.sitio.com.ar

    DocumentRoot /var/ww/sitio_local

    ErrorLog /var/log/apache2/errores_sitio_local
    CustomLog /var/log/apache2/accessos_sitio_local combined
</VirtualHost>
```

Ahora tenemos que activar el sitio para que Apache pueda controlarlo. Apache solo publica los sitios que son configurados en /etc/apache2/sites-enabled/.

Entonces para activarlo ejecutamos el siguiente comando:

```
a2ensite /etc/apache2/sites-available/sitios.conf
```

El último paso que nos queda es recargar el servicio de Apache y verificar nuestro sitio:

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

Levantamos el navegador y tratamos de acceder al sitio que acabamos de alojar en nuestro servidor.

Veamos a continuación un ejemplo de un VirtualHost que incluye la configuración de cgi:

```
<VirtualHost *:80>
    ServerName www.sitio.com.ar
    ServerAlias sitio.com.ar
    ServerAdmin webmaster@sitio.com.ar

    DocumentRoot /var/www/otro_sitio

    ScriptAlias /cgi-bin/ “/var/www/sitio_local/cgi-bin/”
    AddHandler cgi-script .cgi

    <Directory “/var/www/sitio_local/cgi-bin”>
        AllowOverride None
        Options None
        Order allow,deny
        Allow from all
    </Directory>

    ErrorLog /var/log/apache2/errores_sitio_local
    CustomLog /var/log/apache2/accesos_sitio_local combined
</VirtualHost>
```

Crear un directorio para acceso restringido

Es hora de crear un directorio al cual se pueda acceder mediante el uso de un usuario y un password.

Vamos al directorio de nuestro sitio y creamos un subdirectorio llamado restringido:

```
mkdir restringido
```

Nos cambiamos al directorio...

```
cd restringido/
```

Copiamos la página index.html del directorio anterior:

```
cp ../index.html
```

La modificamos para darnos cuenta que estamos en ese directorio:

```
<html>
    <head>
        <title> Este es nuestro Sitio local </title>
    </head>
<body>
      <h1 align=”center”> Directorio Restringido </h1>
</body>
</html>
```

Una vez que ya tenemos nuestra página index vamos a crear un archivo que contendrá los usuarios que podrán acceder al sitio. Para lograrlo usaremos el comando htpasswd.

Es importante aclarar que el archivo no debe estar dentro del sitio.

```
cd /etc/apache2
```

Al ejecutar el comando estamos creando un archivo llamado pass y dentro del archivo un usuario llamado prueba. Veamos como lo hacemos:

```
htpasswd -c pass prueba
New password:
Re-type new password:

Adding password for user prueba
```

Una vez hecho esto cambiamos el dueño del archivo de tal manera que Apache2 pueda leerlo. Recordemos que Apache2 corre con usuario www-data y lo mismo ocurre con el grupo:

```
chown www-data:www-data pass
```

Preparamos nuestro VirtualHost para que sepa que tiene que trabajar con el archivo .htaccess escribiendo la configuración de la siguiente manera:

```
<VirtualHost *:80>
    ServerAdmin webmaster@sitio.com.ar
    ServerName www.sitio.com.ar

    DocumentRoot /var/www/sitio_local

    AccessFileName .htaccess
    ErrorLog /var/log/apache2/errores_sitio_local
    CustomLog /var/log/apache2/accessos_sitio_local combined
</VirtualHost>
```

Como podemos ver hemos agregado al archivo la variable AccessFileName que apunta al archivo .htaccess que no puede ser transferido.

Por último crearemos el archivo .htaccess dentro del directorio restringido:

```
cd /var/www/sitio_local/restringido
```

Creamos el archivo .htaccess y le agregamos las siguientes lineas:

```
AuthName www.sitio.com.ar
AuthType Basic
AuthUserFile /etc/apache2/pass require valid-user
```

Recargamos el servicio de Apache

```
/etc/init.d/apache2 force-reload
```

Y probamos que todo funcione accediendo a nuestro sitio desde el navegador web que utilicemos. Antes de transferir la página index.html nos preguntará por el usuario y el password.

Este servicio es sumamente importante. Todos los sitios que navegamos tienen configuraciones similares a las usadas en los ejemplos.