

# Configuración básica de un servidor HTTP Apache

El protocolo HTTP (*HyperText Transfer Protocol*) está detrás del origen de la web. Se trata de un protocolo cliente/servidor, el cliente solicita al servidor HTTP que le transmita una "página", generalmente estructurada en formato HTML (*HyperText Markup Language*). Una página puede corresponder a un archivo o ser generada dinámicamente por un componente del servidor.

Apache es el programa servidor HTTP más conocido y uno de los más usados en el mundo.

Su nombre viene de su origen: en los años 1990, en los Estados Unidos, el NCSA (*National Center for Supercomputing Applications*) había creado uno de los primeros servidores HTTP, `httpd`. Un grupo de programadores propuso un conjunto de parches (*patches*) para este programa, antes de crear su propia versión, Apache (*a patchy server, un servidor parcheado*), en 1995. Apache HTTP Server está hoy día desarrollado bajo el control de la fundación *Apache Software Foundation*, que gestiona muchos proyectos de software libre (con una licencia específica, la licencia Apache).

El programa Apache HTTP Server, originario de Unix, ha sido implementado en Linux y en otros sistemas operativos (como Windows). Muchos servidores web se basan en una arquitectura llamada LAMP: Linux/Apache/MySQL/PHP.

La versión principal actual de Apache es la versión número 2, reescrita completamente en 2002. En 2020, la versión estable es la versión 2.4.43.

## 1. Archivo de configuración

El archivo de configuración del servidor Apache puede contener centenas de directivas, debido al gran número de funcionalidades y de módulos disponibles para este programa. En el marco de este tema de la certificación, estudiaremos los elementos esenciales sobre la configuración básica de un servidor HTTP.

## a. Formato del archivo de configuración

El nombre y la ubicación del archivo de configuración del servidor Apache varían según las versiones y según las distribuciones: `httpd.conf`, `apache.conf` o `apache2.conf`. Se trata de un archivo de texto, compuesto por directivas. Algunas directivas son globales, otras están declaradas dentro de una sección y solamente se aplican a esta sección.

El archivo está autodocumentado por muchas líneas de comentarios, que se encuentran después del carácter `#` y hasta el final de la línea.

Una directiva está compuesta por una palabra clave (sin distinción entre mayúsculas y minúsculas) y por uno o varios argumentos (con una distinción entre mayúsculas y minúsculas). La directiva se termina al final de la línea, excepto si va precedida por el carácter `\`, que provoca que se prolongue en la línea siguiente.

Una sección es un conjunto de líneas limitadas por una línea `<PalabraClave [Argumentos]>` y una línea `</PalabraClave>`.

### Ejemplo

Extractos de un archivo de configuración de un servidor HTTP Apache en una distribución CentOS 8.

Archivo `/etc/httpd/conf/httpd.conf`:

Comentarios:

```
#
# This is the main Apache HTTP server configuration file. It contains
# the configuration directives that give the server its instructions.
# See <URL:http://httpd.apache.org/docs/2.4/> for detailed information.
# In particular, see
# <URL:http://httpd.apache.org/docs/2.4/mod/directives.html>
# for a discussion of each configuration directive.
```

Directiva simple:

```
ServerRoot "/etc/httpd"
```

Sección:

```
<Directory "/var/www">
  AllowOverride None
  # Allow open access:
  Require all granted
</Directory>
```

## b. Directivas globales y directivas de sección

Una directiva que no figure en una sección es global, y se aplicará a todos los elementos, excepto a los que estén asociados a secciones donde esté definida de nuevo.

Una directiva que figura en una sección es local para la sección, solamente se aplicará a los elementos relativos a esta sección.

Si una directiva global figura en una sección, con un valor diferente, el valor de la directiva local en la sección se aplica a los elementos asociados a la sección y solamente a ellos (el valor local "cubre" al valor global).

### Ejemplo

Directiva en el interior de una sección `Directory`:

```
<Directory /var/www/html>
  Options FollowSymLinks
</Directory>
```

La directiva `Options FollowSymLinks` (seguir los enlaces simbólicos) se aplica al directorio asociado a la sección, incluso si el valor es diferente en otra parte del archivo de configuración.

## c. Directivas básicas

Estas directiva se encuentran en la mayoría de las configuraciones de servidores HTTP Apache.

<code>ServerRoot</code> Camino	Directorio raíz de los archivos de configuración y de los archivos de registro.
<code>User</code> NombreUsuario	Cuenta de usuario asociado a los procesos servidores de Apache.
<code>Group</code> NombreGrp	Grupo de usuario asociado a los procesos servidores de Apache.
<code>Listen</code> [Dirección:]Puerto	Dirección y puerto en el que el servidor HTTP se pone en escucha.
<code>DocumentRoot</code> Camino	Directorio raíz de los archivos de datos (paginas HTML).
<code>Include</code> <code>conf.modules.d/*.conf</code>	Inclusión de los archivos de configuración de los módulos.
<code>IncludeOptional</code> <code>conf.d/*.conf</code>	Inclusión de los archivos de configuración opcionales.
<code>TypesConfig</code> Camino	Camino de acceso al archivo <code>mime.types</code> .
<code>LogLevel</code> warn	Nivel de registro.
<code>ErrorLog</code> CaminoArchivo	Archivo de registro de los errores (si el camino es relativo, a partir de <code>ServerRoot</code> ).



Con las distribuciones de tipo Debian, la configuración del servidor HTTP Apache 2 usa variables de entorno, definidas en el script shell `/etc/apache2/envvars`.

### Ejemplo

Archivo de configuración mínimo de Apache 2, en un servidor CentOS 8:

```
vi /etc/httpd/conf/httpd.conf
Listen 80
User apache
Group apache
ServerRoot "/etc/httpd"
DocumentRoot "/var/www/html"
LogLevel warn
ErrorLog "logs/error_log"
Include conf.modules.d/*.conf
IncludeOptional conf.d/*.conf
TypesConfig /etc/mime.types
```

Archivo de configuración mínimo de Apache 2, en un servidor Debian 10:

```
vi /etc/apache2/apache2.conf
Listen 80
User www-data
Group www-data
ServerRoot "/etc/apache2"
DocumentRoot "/var/www/html"
LogLevel warn
ErrorLog /var/log/apache2/error.log
IncludeOptional mods-enabled/*.load
IncludeOptional mods-enabled/*.conf
```

## d. Validación de la sintaxis

La opción `-t` del ejecutable del servidor HTTP Apache permite comprobar la sintaxis del archivo de configuración, sin cargarlo ni iniciar el daemon.

### Ejemplo

Control de la sintaxis del archivo de configuración de Apache 2, en un servidor CentOS 8:

```
httpd -t  
Syntax OK
```

Introducimos un error de sintaxis en el archivo (`ServerBoot` en lugar de `ServerRoot`):

```
vi /etc/httpd/conf/httpd.conf  
ServerBoot "/etc/httpd"  
Listen 80  
Include conf.modules.d/*.conf  
User apache  
Group apache  
DocumentRoot "/var/www/html"  
ErrorLog "logs/error_log"  
LogLevel warn  
IncludeOptional conf.d/*.conf  
TypesConfig /etc/mime.types  
httpd -t  
AH00526: Syntax error on line 1 of /etc/httpd/conf/httpd.conf:  
Invalid command 'ServerBoot', perhaps misspelled or defined by a module  
not included in the server configuration
```

También podemos controlar el archivo de configuración usando el comando de control del servidor HTTP Apache, `apachectl` (o `apache2ctl`), con el subcomando `configtest`:

```
apachectl configtest
```

## 2. Inicio y paro del servidor HTTP Apache

El inicio y el paro del servidor HTTP Apache se hacen con un script de arranque `init System V` o, en las versiones recientes de la mayoría de las distribuciones Linux, usando `systemd`.

### a. Paro/inicio por systemd

El nombre del daemon puede cambiar con respecto a las versiones del servidor HTTP Apache y según las distribuciones. En la mayoría de los casos, será `httpd` (Red Hat) o `apache2` (Debian).

#### Ejemplos

Control del servidor HTTP Apache, hecho por `systemd`, en un servidor CentOS 8:

Inicio del servidor HTTP Apache:

```
systemctl start httpd
ps -ef | grep httpd
root    445    1  0 14:09?   00:00:00 /usr/sbin/httpd -DFOREGROUND
apache  446   445  0 14:09?   00:00:00 /usr/sbin/httpd -DFOREGROUND
apache  447   445  0 14:09?   00:00:00 /usr/sbin/httpd -DFOREGROUND
apache  449   445  0 14:09?   00:00:00 /usr/sbin/httpd -DFOREGROUND
apache  450   445  0 14:09?   00:00:00 /usr/sbin/httpd -DFOREGROUND
```

El servidor se ha iniciado, con un proceso principal asociado a la cuenta de superusuario, y cuatro procesos servidores, asociados a la cuenta de usuario de servicio `apache`.

Estado del servidor:

```
systemctl status httpd
httpd.service - The Apache HTTP Server
Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled;
vendor preset: disabled)
Active: active (running) since Wed 2020-05-13 14:09:32 CEST; 1min 51s ago
Docs: man:httpd.service(8)
Main PID: 445 (httpd)
Status: "Running, listening on: port 80"
```

Tasks: 213 (limit: 23516)

Memory: 34.8M

CGroup: /system.slice/httpd.service

445 /usr/sbin/httpd -DFOREGROUND

446 /usr/sbin/httpd -DFOREGROUND

447 /usr/sbin/httpd -DFOREGROUND

449 /usr/sbin/httpd -DFOREGROUND

450 /usr/sbin/httpd -DFOREGROUND

mayo 13 14:09:32 centos8.midns.es systemd[1]: Starting The Apache HTTP Server...

mayo 13 14:09:32 centos8.midns.es systemd[1]: Started The Apache HTTP Server.

mayo 13 14:09:32 centos8.midns.es httpd[445]: Server configured, listening on: port 80

Paro del servidor:

**systemctl stop httpd**

**ps -ef | grep httpd**

Control del servidor HTTP Apache, mediante `systemd`, en un servidor Debian 10:

El programa ejecutable del servidor HTTP Apache se llama `apache2` en las distribuciones Debian recientes.

Inicio del servidor HTTP Apache:

**systemctl start apache2**

**ps -ef | grep apache2**

root 5588 1 0 14:24? 00:00:00 /usr/sbin/apache2 -k start

www-data 5589 5588 0 14:24? 00:00:00 /usr/sbin/apache2 -k start

www-data 5590 5588 0 14:24? 00:00:00 /usr/sbin/apache2 -k start

El servidor se ha iniciado, con un proceso principal asociado a la cuenta de superusuario, y dos procesos servidores, asociados a la cuenta de usuario de servicio `www-data`.

Estado del servidor:



**systemctl status apache2**

```

apache2.service - The Apache HTTP Server
Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor
preset: enabled)
Active: active (running) since Wed 2020-05-13 14:24:28 CEST; 1min 17s ago
Docs: https://httpd.apache.org/docs/2.4/
Process: 5584 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
Main PID: 5588 (apache2)
Tasks: 55 (limit: 4558)
Memory: 13.0M
CGroup: /system.slice/apache2.service
        5588 /usr/sbin/apache2 -k start
        5589 /usr/sbin/apache2 -k start
        5590 /usr/sbin/apache2 -k start

```

```

mayo 13 14:24:28 debian10 systemd[1]: Starting The Apache HTTP Server...
mayo 13 14:24:28 debian10 apachectl[5584]: AH00558: apache2: Could not reliably
determine the server's fully qualified domain name, using
mayo 13 14:24:28 debian10 systemd[1]: Started The Apache HTTP Server.

```

Paro del servidor:

```

systemctl stop apache2
ps -ef | grep apache2


```

## b. Comprobación del servidor HTTP Apache

Podemos comprobar la página de inicio por defecto del servidor HTTP Apache, usando un navegador de Internet y solicitando la URL `http://localhost` o `http://dirección_IP_servidor`.

### Ejemplo

Página de inicio por defecto de un servidor HTTP Apache2 de una distribución Debian 10:



## Apache2 Debian Default Page

**It works!**

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

### Configuration Overview

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented in `/usr/share/doc/apache2/README.Debian.gz`**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:

```

/etc/apache2/
|-- apache2.conf
|   |-- ports.conf
|-- mods-enabled
|   |-- *.load
|   |-- *.conf
|-- conf-enabled
|   |-- *.conf
|-- sites-enabled
|   |-- *.conf

```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.

Esta página de inicio da acceso a la documentación de Apache2, si el paquete de software `apache2-doc` está instalado en el servidor.

### c. Paro/arranque puntual

Se puede gestionar puntualmente el paro y el inicio del servidor HTTP Apache, usando la opción `-k stop|start` de su ejecutable:

```
httpd -k stop|start
```

0

`apache2 -k stop|start`

Este método está desaconsejado fuera de las fases de test, porque se corre el riesgo de desincronizar el servidor con la gestión efectuada por `systemd`.

### Ejemplo

Paro y reinicio del servidor HTTP Apache, en una distribución CentOS 8:

```

httpd -k start
ps -ef | grep httpd
root    870    1  0 14:29?   00:00:00 httpd -k start
apache  871    870  0 14:29?   00:00:00 httpd -k start
apache  872    870  0 14:29?   00:00:00 httpd -k start
apache  873    870  0 14:29?   00:00:00 httpd -k start
apache  874    870  0 14:29?   00:00:00 httpd -k start
httpd -k stop
ps -ef | grep httpd

```

También podemos, con las mismas restricciones, usar el comando de control del servidor HTTP Apache, `apachectl` (o `apache2ctl`), con la opción `-k`:

`apachectl -k stop|start`

### Ejemplo

Paro y reinicio del servidor HTTP Apache, en una distribución Debian 10:

```

apachectl -k start
ps -ef | grep apache
root    5861    1  0 14:42?    00:00:00 /usr/sbin/apache2 -k start
www-data 5862  5861    0 14:42?    00:00:00 /usr/sbin/apache2 -k start
www-data 5863  5861    0 14:42?    00:00:00 /usr/sbin/apache2 -k start
apachectl -k stop
ps -ef | grep apache

```

### 3. Archivos de registro

Los archivos del registro del servidor HTTP Apache son muy importantes para la seguridad. Permiten monitorizar la actividad del servidor, constatar los errores que puedan ocurrir y también conocer qué clientes HTTP han efectuado o han intentado efectuar peticiones de páginas en nuestro servidor.

Los dos archivos de registro por defecto son `access.log` y `error.log`, su camino de acceso depende de la configuración del servidor HTTP Apache. Con respecto al nivel de seguimiento especificado en la directiva `LogLevel`, pueden ser más o menos voluminosos. También podemos configurar usando directivas el formato de cada registro.

Por defecto, los archivos de registro son comprimidos y archivados automáticamente según los parámetros de rotación configurables.

#### Ejemplo

*Contenido del directorio de los registros Apache 2 en una distribución Debian:*

```

ls /var/log/apache2
access.log    access.log.3.gz error.log.10.gz error.log.2.gz
error.log.5.gz error.log.8.gz
access.log.1  error.log    error.log.11.gz error.log.3.gz
error.log.6.gz error.log.9.gz
access.log.2.gz error.log.1  error.log.12.gz error.log.4.gz
error.log.7.gz other_vhosts_access.log

```

Extractos de los archivos de registros actuales:

**vi /var/log/apache2/access.log**

```
192.168.1.24 - - [15/May/2020:09:11:48 +0200] "GET /hola.php HTTP/1.1"
200 363 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleW
ebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.138 Safari/537.36"
[...]
```

**vi /var/log/apache2/error.log**

```
[Fri May 15 00:00:25.527997 2020] [mpm_event:notice] [pid 8527:tid
140445564359808] AH00489: Apache/2.4.38 (Debian) configured -- resuming
normal operations
[Fri May 15 00:00:25.528090 2020] [core:notice] [pid 8527:tid
140445564359808] AH00094: Command line: '/usr/sbin/apache2'
[Fri May 15 08:29:33.022653 2020] [mpm_event:notice] [pid 8527:tid
140445564359808] AH00491: caught SIGTERM, shutting down
[Fri May 15 08:29:33.115813 2020] [mpm_prefork:notice] [pid 16008] AH00163:
Apache/2.4.38 (Debian) configured -- resuming normal operations
[...]
```

## 4. Los módulos Apache

Apache es un programa muy modular. Está compuesto por el ejecutable principal y un gran número de módulos independientes, bajo la forma de módulos de objetos cargables dinámicamente y compartidos (archivos `*.so`, *shared object*). Esta arquitectura facilita el mantenimiento y la capacidad de evolución del programa, la creación o la modificación de un módulo no tendrá ningún impacto en el programa principal o en los otros módulos. También permite limitar la memoria usada, porque solamente se cargarán los módulos realmente utilizados.

Los módulos que se tienen que cargar tienen que estar declarados en el archivo de configuración del servidor HTTP Apache, y necesitan muy a menudo sus propias directivas de configuración.



Ciertos módulos esenciales están directamente contenidos en el ejecutable del servidor HTTP Apache, durante su compilación. Son módulos estáticos, no compartidos.

### a. Directiva de carga de un módulo

La directiva `LoadModule` especifica la carga de un módulo según la sintaxis siguiente:

```
LoadModule idModule CaminoArchivoMódulo
```

Donde:

`idModule`: es el identificador del módulo.

`CaminoArchivoMódulo`: camino de acceso del archivo de módulo objeto correspondiente. Si el camino es relativo, parte del directorio raíz del servidor definido en la directiva `ServerRoot`.

#### Ejemplo

*Directiva de carga de un módulo de gestión de scripts CGI.*

*Apache2 en una distribución CentOS 8:*

*Las directivas de carga de los módulos se encuentran en los archivos de inclusión del directorio `/etc/httpd/conf.modules.d/`. Para los módulos CGI, el archivo es `01-cgi.conf`, que contiene en particular esta línea:*

```
LoadModule cgid_module modules/mod_cgid.so
```

`cgid_module` es el identificador del módulo. El módulo en sí se encuentra en el archivo: `/etc/httpd/modules/mod_cgid.so`.

*Apache2 en una distribución Debian 10:*

Las directivas de carga de los módulos se encuentran en los archivos de inclusión del directorio `/etc/apache2/mods-available` (módulos cargables). Para los módulos CGI, el archivo es `cgid.load`, que contiene en particular esta línea:

```
LoadModule cgid_module /usr/lib/apache2/modules/mod_cgid.so
```

`cgid_module` es el identificador del módulo. El módulo en sí se encuentra en el archivo `/usr/lib/apache2/modules/mod_cgid.so`.

## b. Lista de los módulos incluidos y cargados

Dos opciones en el ejecutable del servidor HTTP Apache permiten obtener la lista de los módulos estáticos integrados en el ejecutable (opción `-l`) y la del conjunto de los módulos, estáticos o dinámicos (opción `-M`).

### Ejemplo

Lista de los módulos Apache2 con una distribución CentOS 8:

#### **httpd -l**

Compiled in modules:

```
core.c
mod_so.c
http_core.c
```

#### **httpd -M**

Loaded Modules:

```
core_module (static)
so_module (static)
http_module (static)
access_compat_module (shared)
[...]
proxy_ajp_module (shared)
proxy_balancer_module (shared)
proxy_connect_module (shared)
proxy_express_module (shared)
proxy_fcgi_module (shared)
proxy_fdpass_module (shared)
```

```
proxy_ftp_module (shared)
proxy_http_module (shared)
[...]
```

Si el servidor Apache no es usado como servidor proxy, podríamos desactivar la carga de los numerosos módulos `proxy*`.

Lista de los módulos Apache2 con una distribución Debian 10:

#### **apache2 -l**

Compiled `in` modules:

```
core.c
mod_so.c
mod_watchdog.c
http_core.c
mod_log_config.c
mod_logio.c
mod_version.c
mod_unixd.c
```

#### **apache2 -M**

Loaded Modules:

```
core_module (static)
so_module (static)
watchdog_module (static)
http_module (static)
log_config_module (static)
logio_module (static)
version_module (static)
unixd_module (static)
access_compat_module (shared)
alias_module (shared)
auth_basic_module (shared)
authn_core_module (shared)
authn_file_module (shared)
authz_core_module (shared)
authz_host_module (shared)
authz_user_module (shared)
autoindex_module (shared)
deflate_module (shared)
```



```

dir_module (shared)
env_module (shared)
filter_module (shared)
mime_module (shared)
mpm_event_module (shared)
negotiation_module (shared)
reqtimeout_module (shared)
setenvif_module (shared)
status_module (shared)

```

### c. Configuración de los módulos

La elección de los módulos dinámicos que se tienen que cargar depende de las funciones que se pueden implementar en el servidor HTTP. Existen más de cien módulos, algunos obligatorios (*core modules*), otros implementan tratamientos específicos (gestión de LDAP, proxy FTP, SSL...), a menudo con dependencias entre módulos.

A modo de ejemplo, la certificación LPIC2 propone detallar la implementación de módulos que permiten el soporte de dos lenguajes de script en el lado del servidor: Perl y PHP.



La configuración de los módulos cambia con respecto a las versiones de Apache y las implementaciones. Las que se van a describir a continuación corresponden a Apache 2.4x, para una distribución CentOS 8 y una distribución Debian 10.

### d. Configuración del módulo Perl

Perl es un lenguaje (semi) interpretado que permite ejecutar scripts. Fue uno de los más usados al principio de la web para administrar páginas HTML dinámicas, y su base instalada sigue siendo importante.

El módulo Perl de Apache se ejecuta automáticamente cuando una URL correspondiente a un script Perl es solicitada por un cliente HTTP. El módulo es un interpretador Perl que leerá directamente y ejecutará el script para generar dinámicamente una página HTML que

se le devolverá al cliente.

El módulo Perl tiene que ser instalado por un paquete de software. El paquete instala el archivo módulo objeto, un archivo de configuración para cargar el módulo y un ejemplo de archivo de configuración del módulo y de declaración de un directorio destinado a almacenar los scripts Perl ejecutables a través del servidor HTTP Apache.

Una vez que el módulo haya sido instalado y configurado, hay que crear el directorio de almacenamiento de los scripts Perl, bajo la raíz de los documentos del servidor, y solicitar al servidor HTTP Apache que recargue su configuración.

A continuación, se puede crear un script Perl de test y solicitar su ejecución desde un cliente HTTP, por ejemplo, un navegador de Internet en la máquina local.

### Ejemplo

*Este ejemplo muestra la instalación, la configuración y la prueba del módulo Perl, en un servidor HTTP Apache2 de una distribución CentOS 8.*

*Comprobamos si el módulo Perl para Apache ha sido instalado:*

```
yum list mod_perl
```

*Si no estuviera instalado, lo instalamos:*

```
yum install mod_perl
```

```
[...]
```

```
Tamaño total de la descarga: 5.4 M
```

```
Tamaño instalado: 13 M
```

```
¿Está de acuerdo [s/N]?:
```

```
[...]
```

```
Instalado:
```

```
mod_perl-2.0.11-1.el8.x86_64
```

```
perl-CPAN-Meta-2.150010-396.el8.noarch
```

```
perl-Encode-Locale-1.05-9.el8.noarch
```

```
perl-Time-HiRes-1.9758-1.el8.x86_64
```

```
annobin-8.78-1.el8.x86_64
```

```
dwz-0.12-9.el8.x86_64
```

```

efi-srpm-macros-3-2.el8.noarch
ghc-srpm-macros-1.4.2-7.el8.noarch
go-srpm-macros-2-16.el8.noarch
ocaml-srpm-macros-5-4.el8.noarch
openblas-srpm-macros-2-2.el8.noarch
perl-CPAN-Meta-Requirements-2.140-396.el8.noarch
perl-CPAN-Meta-YAML-0.018-397.el8.noarch
perl-ExtUtils-Command-1:7.34-1.el8.noarch
perl-ExtUtils-Install-2.14-4.el8.noarch
perl-ExtUtils-MakeMaker-1:7.34-1.el8.noarch
perl-ExtUtils-Manifest-1.70-395.el8.noarch
perl-ExtUtils-ParseXS-1:3.35-2.el8.noarch
perl-JSON-PP-1:2.97.001-3.el8.noarch
perl-Test-Harness-1:3.42-1.el8.noarch
perl-devel-4:5.26.3-416.el8.x86_64
perl-srpm-macros-1-25.el8.noarch
perl-version-6:0.99.24-1.el8.x86_64
python-srpm-macros-3-37.el8.noarch
python3-rpm-macros-3-37.el8.noarch
qt5-srpm-macros-5.11.1-2.el8.noarch
redhat-rpm-config-120-1.el8.noarch
rust-srpm-macros-5-2.el8.noarch
systemtap-sdt-devel-4.1-6.el8.x86_64
perl-BSD-Resource-1.291.100-11.el8.x86_64
perl-Linux-Pid-0.04-40.el8.x86_64

```

!Listo!

31 paquetes han sido instalados.

Comprobamos la instalación del archivo módulo objeto del módulo:

```

ls /etc/httpd/modules/*perl*
/etc/httpd/modules/mod_perl.so

```

Comprobamos el archivo de configuración de la carga del módulo:

```
ls /etc/httpd/conf.modules.d/*perl*
/etc/httpd/conf.modules.d/02-perl.conf
vi /etc/httpd/conf.modules.d/02-perl.conf
LoadModule perl_module modules/mod_perl.so
```

Modificamos el archivo de configuración del módulo para declarar el directorio de almacenamiento de los scripts Perl. Este directorio será accesible para todo el mundo, y sus páginas serán accesibles usando un alias `/perl`:

```
vi /etc/httpd/conf.d/perl.conf:
Alias /perl /var/www/html/perl
<Directory /var/www/html/perl>
    SetHandler perl-script
    PerlResponseHandler ModPerl::Registry
    PerlOptions +ParseHeaders
    Options +ExecCGI
    Order allow,deny
    Allow from all
</Directory>
```

Creamos el directorio de los scripts Perl, bajo el directorio raíz de los datos del servidor y determinamos los derechos de acceso:

```
mkdir /var/www/html/perl
chmod 755 /var/www/html/perl
```

Recargamos el servidor HTTP Apache:

```
systemctl reload httpd
```

Comprobamos que el módulo Perl esté cargado correctamente:

```
httpd -M | grep -i perl
perl_module (shared)
```

Creamos un script Perl de prueba. Para el nombre del archivo podemos utilizar la extensión `.html`, porque todo archivo del directorio será considerado como un script Perl y enviado al módulo Perl para analizarlo y ejecutarlo:

```
vi /var/www/html/perl/index.html
# Encabezado que muestra el tipo de contenido generado:
print "Content-type: text/plain\n";
# CUIDADO: la línea blanca siguiente es OBLIGATORIA
print "\n";
print "=====\n";
print "Bienvenido a mi sitio Perl\n";
print "=====\n";
```

Comprobamos el script desde un navegador local, utilizando el alias del directorio Perl:

`http://localhost/perl/index.html` o `http://localhost/perl`, porque el servidor está configurado para que la página por defecto sea `index.html`.

El navegador muestra la página de texto generada por la ejecución del script:

```
=====
Bienvenido a mi sitio Perl
=====
```

El módulo Perl está operativo.

## e. Configuración del módulo PHP

PHP es un lenguaje interpretado que permite ejecutar scripts, escritos completamente en PHP o integrados en el interior de páginas HTML.

Se trata de un lenguaje muy usado para generar páginas HTML dinámicas y su base instalada es muy importante. El lenguaje ha evolucionado mucho a lo largo de los años, en particular para resolver múltiples problemas de seguridad. La versión principal en 2020 es la 7.x.



La configuración del servidor PHP, que interpreta y ejecuta los scripts PHP, es compleja y debe ser estudiada con cuidado para prevenir posibles fallos de seguridad.

El módulo PHP de Apache se ejecuta automáticamente cuando una URL que corresponde a un script PHP es solicitada por el cliente HTTP. El módulo es un servidor PHP que leerá y ejecutará directamente el script para generar dinámicamente una página HTML que será mostrada al cliente.

El módulo PHP tiene que ser instalado usando un paquete de software. El paquete instala el archivo módulo objeto, un archivo de configuración para cargar el módulo y un ejemplo de archivo de configuración del módulo y de declaración de un directorio destinado a contener los scripts PHP ejecutables por el servidor HTTP Apache.

La configuración del módulo consiste, particularmente, en asociarle los archivos de extensiones (`.php`, `.php7` ...). Cuando el servidor HTTP Apache recibe una petición relativa a estos archivos, la trata usando el módulo PHP que interactúa con el servidor PHP. La página HTML generada después de este tratamiento se envía al servidor HTTP Apache, que la transmite al cliente HTTP.

Una vez que el módulo esté instalado y configurado, hay que crear el directorio de almacenamiento de los scripts PHP, bajo la raíz de los documentos del servidor, y solicitar al servidor HTTP Apache que recargue su configuración.

A continuación podemos crear un script PHP de prueba y solicitar su ejecución desde un cliente HTTP, por ejemplo un navegador de Internet en la máquina local.

### Ejemplo

*El ejemplo siguiente muestra la instalación, la configuración y la prueba del módulo PHP, en un servidor HTTP Apache2 de una distribución Debian 10.*

*Comprobamos si PHP está instalado en el sistema:*

```
apt list php7
```

Si no está instalado, lo instalamos:

```

apt-cache search php7|more
libapache2-mod-php7.3 - server-side, HTML-embedded scripting language (Apache 2
module)
php7.3 - server-side, HTML-embedded scripting language (metapackage)
apt-get install php7.3
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
libapache2-mod-php7.3 php-common php7.3-cli php7.3-common php7.3-json
php7.3-opcache php7.3-readline
Paquetes sugeridos:
php-pear
Se instalarán los siguientes paquetes NUEVOS:
libapache2-mod-php7.3 php-common php7.3 php7.3-cli php7.3-common php7.3-json
php7.3-opcache php7.3-readline
0 actualizados, 8 nuevos se instalarán, 0 para eliminar y 8 no actualizados.
Se necesita descargar 4.014 kB de archivos.
Se utilizarán 17,5 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n]
[...]
Configurando libapache2-mod-php7.3 (7.3.31-1~deb10u1) ...

Creating config file /etc/php/7.3/apache2/php.ini with new version
Module mpm_event disabled.
Enabling module mpm_prefork.
apache2_switch_mpm Switch to prefork
apache2_invoke: Enable module php7.3
Configurando php7.3 (7.3.31-1~deb10u1) ...
Procesando disparadores para man-db (2.8.5-2) ...

```

Se ha instalado el servidor PHP, así como el módulo para Apache2.

Comprobamos que el servidor se ha instalado correctamente:

```
php -v
```

PHP 7.3.31-1~deb10u1 (cli) (built: Oct 24 2021 15:18:08) ( NTS )  
 Copyright (c) 1997-2018 The PHP Group  
 Zend Engine v3.3.31, Copyright (c) 1998-2018 Zend Technologies  
 with Zend OPcache v7.3.31-1~deb10u1, Copyright (c) 1999-2018, by Zend Technologies

Comprobamos la instalación del archivo módulo objeto del módulo:

```
find /lib/apache2 -name '*php*'
/lib/apache2/modules/libphp7.3.so
```

Comprobamos los archivos de configuración de la carga del módulo:

```
find /etc/apache2 -name '*php*'
/etc/apache2/mods-enabled/php7.3.load
/etc/apache2/mods-enabled/php7.3.conf
/etc/apache2/mods-available/php7.3.load
/etc/apache2/mods-available/php7.3.conf
```

En una distribución Debian, la configuración de los módulos Apache2 se hace en diferentes directorios: `/etc/apache2/mods-available` para los módulos cargables, `/etc/apache2/mods-enabled` para los módulos que se cargarán efectivamente.

Comprobamos el archivo de configuración del módulo, para especificar las extensiones de los archivos que serán asociados al módulo PHP:

```
vi /etc/apache2/mods-enabled/php7.3.conf
<FilesMatch ".+\.ph(ar|p|tml)$">
    SetHandler application/x-httpd-php
</FilesMatch>
<FilesMatch ".+\.phps$">
    SetHandler application/x-httpd-php-source
    # Deny access to raw php sources by default
    # To re-enable it's recommended to enable access to the files
    # only in specific virtual host or directory
```



```

Require all denied
</FilesMatch>
# Deny access to files without filename (e.g. '.php')
<FilesMatch "\.ph(ar|p|ps|tml)$">
    Require all denied
</FilesMatch>

```

Recargamos el servidor HTTP Apache:

```
systemctl reload apache2
```

Comprobamos que el módulo PHP esté bien cargado:

```

apache2 -M | grep php
php7_module (shared)

```

Creamos un archivo página PHP de prueba. Para el nombre del archivo, usamos el sufijo `.php`, porque todos los archivos que tengan esa extensión serán considerados como scripts PHP y enviados al módulo PHP para que su análisis y ejecución:

```

vi /var/www/html/buenosdias.php
<html>
<head>
<title>Test página PHP</title>
</head>
<body>
<?php
// Script PHP integrado en la página HTML
echo '<p>Bienvenido a mi sitio PHP</p>';
$date = date("d-m-Y H:i");
Print("<p>Fecha y hora:$date</p>");
?>
</body>
</html>

```

Comprobamos la página PHP desde un navegador local: `http://localhost/buenosdias.php`.

El navegador muestra la página, con el texto generado por la ejecución del script PHP integrado entre las etiquetas `<?php?>`:

```
Bienvenido a mi sitio PHP
Fecha y hora:15-05-2021 09:25
```

El módulo PHP está operativo.

## 5. Control de los recursos asignados al servidor

Diferentes directivas permiten especificar los recursos mínimos y máximos asignables al servidor HTTP Apache. Estos parámetros son importantes para gestionar la carga del servidor y también, desde el punto de vista de la seguridad, para evitar que ataques de tipo denegación de servicio (*denial of service*) provoquen una saturación de los recursos de memoria y procesador del sistema que alojan al servidor HTTP.

En el arranque del servidor, se lanza un daemon, asociado a la cuenta de superusuario. Este proceso inicializa el servidor, pero no gestiona directamente las peticiones de los clientes HTTP. Crea procesos hijos asociados a una cuenta de servicio que no es de tipo superusuario, y son estos los que atienden las peticiones de los clientes.

Si el número de clientes aumenta, el proceso padre puede crear otros procesos hijos para tratar sus peticiones.

El número máximo y mínimo de procesos hijos, durante el arranque, pueden ser especificados por directivas del archivo de configuración, con valores por defecto variables si las directivas no están presentes en el archivo.



Si el servidor HTTP funciona en modo sin thread (módulo `mpm_prefork`, el modo de las primeras versiones del servidor Apache), un proceso servidor hijo solamente puede tratar a un único cliente. Si funciona en modo thread o hybride (módulo `mpm_worker_module` o `mpm_event_module`), un proceso servidor hijo puede tratar a distintos clientes.

Las principales directivas de control de los recursos son las siguientes:

<code>StartServers</code>	Número de procesos servidores durante el arranque del servidor HTTP Apache.
<code>MinSpareServers</code>	Número mínimo de procesos servidores inactivos (modo <code>prefork</code> solamente).
<code>MaxSpareServers</code>	Número máximo de procesos servidores inactivos (modo <code>prefork</code> solamente).
<code>MaxClients</code>	Número máximo de clientes conectados simultáneamente al servidor HTTP Apache.
<code>MaxRequestWorkers</code>	Número máximo de peticiones que pueden ser tratadas simultáneamente (antiguo nombre: <code>MaxClients</code> ).
<code>MaxRequestsPerChild</code>	Número máximo de peticiones que pueden ser tratadas por un proceso servidor HTTP Apache.

### Ejemplo

Configuramos las directivas de recursos, para un servidor HTTP Apache en una distribución Debian 10:

```
vi /etc/apache2/apache2.conf
```

```
StartServers 10
MinSpareServers 5
MaxSpareServers 15
MaxRequestWorkers 100
MaxRequestsPerChild 1000
```

Comprobamos el archivo de configuración:

```
apache2 -t
Syntax OK
```

Reiniciamos el servidor HTTP Apache:

```
systemctl start apache2
ps -ef | grep apache
root  17432  1 0 13:34?    00:00:00 /usr/sbin/apache2 -k start
www-data 17433 17432 0 13:34?    00:00:00 /usr/sbin/apache2 -k start
www-data 17434 17432 0 13:34?    00:00:00 /usr/sbin/apache2 -k start
www-data 17435 17432 0 13:34?    00:00:00 /usr/sbin/apache2 -k start
www-data 17436 17432 0 13:34?    00:00:00 /usr/sbin/apache2 -k start
www-data 17437 17432 0 13:34?    00:00:00 /usr/sbin/apache2 -k start
www-data 17438 17432 0 13:34?    00:00:00 /usr/sbin/apache2 -k start
www-data 17439 17432 0 13:34?    00:00:00 /usr/sbin/apache2 -k start
www-data 17440 17432 0 13:34?    00:00:00 /usr/sbin/apache2 -k start
www-data 17441 17432 0 13:34?    00:00:00 /usr/sbin/apache2 -k start
www-data 17442 17432 0 13:34?    00:00:00 /usr/sbin/apache2 -k start
```

Hay un proceso principal, asociado a la cuenta de usuario `root`, y diez procesos servidores, asociados a la cuenta de usuario `www-data`.

## 6. Hosts virtuales de un servidor Apache

Por defecto, el servidor HTTP Apache escucha en el puerto bien conocido 80, asociado al

protocolo HTTP, en todas las direcciones IP de la máquina. Si recibe una petición de página, va a buscar el archivo correspondiente a partir de la raíz de los documentos, definida por la directiva `DocumentRoot`.

No obstante, un servidor HTTP Apache puede gestionar varios sitios web diferentes. Cada sitio web virtual es independiente, dispone de su propia configuración y almacena sus páginas en un directorio específico.

Para distinguir los diferentes servidores web virtuales, llamados hosts virtuales (*virtual host*), podemos usar diferentes técnicas:

- ✓ Por dirección IP: el sistema debe gestionar distintas direcciones IP. El servidor HTTP Apache asocia a cada host virtual con una dirección IP diferente.
- ✓ Por número de puerto: el servidor HTTP Apache asocia a cada host virtual un número de puerto diferente.
- ✓ Por nombre de host: el sistema debe gestionar distintos nombres de host. El servidor HTTP Apache asocia un nombre de host diferente a cada host virtual.

#### a. Organización de los hosts virtuales

Cada host virtual está configurado por un conjunto de directivas que se encuentran en una sección `VirtualHost`.

No hay sitio web por defecto. Cada petición de URL corresponde a un sitio virtual, identificado por su dirección IP o por su nombre.

Sintaxis habitual de una sección `VirtualHost`:

```
<VirtualHost DirecciónIP[:NúmeroPort]>
ServerName NombreHost
DocumentRoot CaminoRaíz
ServerAdmin CorreoElectrónicoAdministrador
ErrorLog CaminoLog
TransferLog CaminoAccesoLog
</VirtualHost>
```

Donde:

<code>DirecciónIP[:NœmPort]</code>	Si la dirección está en IPv6, hay que colocarla entre corchetes. El puerto por defecto es el 80.
<code>ServerName</code>	Nombre de host del host virtual (obligatorio).
<code>DocumentRoot</code>	Directorio raíz del host virtual (obligatorio).
<code>ServerAdmin</code>	Dirección de correo electrónico del administrador del host virtual (opcional).
<code>ErrorLog</code> <code>CaminoLog</code>	Camino de acceso al archivo de registro de errores (opcional).
<code>TransferLog</code>	Camino de acceso al archivo de registro (opcional).

## b. Hosts virtuales por dirección IP

El sistema debe tener distintas direcciones IP. El servidor HTTP Apache recibe una petición en una dirección IP (y un número de puerto). Busca en su configuración el host virtual asociado a esta dirección y a este puerto, y trata la petición según la configuración de este host virtual.

La dirección IP del host virtual debe estar especificada en una directiva global `Listen`.

En el cliente, se puede especificar el host virtual solicitado definiendo su dirección IP en la URL, o su nombre de host si las diferentes direcciones están configuradas con diferentes nombres en el sistema de resolución de nombres del cliente (DNS, archivo `hosts`, etc.).

### Ejemplo

*Host virtual por dirección IP en un servidor HTTP Apache de una distribución CentOS 8.*

*Primero hay que atribuir una segunda dirección IP al sistema:*

```
ip -4 -br a
lo          UNKNOWN    127.0.0.1/8::1/128
enp38s0     UP          192.168.1.60/24
```

El sistema posee una dirección IPv4 `192.168.1.60/24`. Declaramos una segunda dirección, en la misma interfaz:

```
ip -4 -br a
lo          UNKNOWN    127.0.0.1/8
enp38s0     UP          192.168.1.60/24 192.168.1.61/24
```

Añadimos la nueva dirección en la zona DNS, asociada al nombre de host `www61` del dominio DNS local:

```
vi /var/named/db.midns.es
2020101601; serial
[...]
www61      IN  A 192.168.1.61
[...]
rndc reload
host www61
www61.midns.es has address 192.168.1.61
```

Configuramos un host virtual asociado a la nueva dirección IP, a través del nombre de host `www61.midns.es`:

```
vi /etc/httpd/conf/httpd.conf
[...]
<VirtualHost 192.168.1.61>
  ServerName www61.midns.es
  DocumentRoot /var/www/html/www61
  ErrorLog /var/log/httpd/www61err.log
  TransferLog /var/log/httpd/www61acc.log
</VirtualHost>
```

Creamos el directorio raíz del host virtual:

```
mkdir /var/www/html/www61
```

Comprobamos la sintaxis del archivo de configuración:

```
httpd -t  
Syntax OK
```

Recargamos la configuración del servidor HTTP Apache:

```
systemctl reload httpd
```

Después de haber creado una página de inicio en el directorio del host virtual, la comprobamos con el comando `wget`:

```
wget -O - www61.midns.es  
--2020-05-16 18:36:13-- http://www61.midns.es/  
Resolviendo www61.midns.es (www61.midns.es)... 192.168.1.61  
Conectando con www61.midns.es (www61.midns.es)[192.168.1.61]:80... conectado.  
Petición HTTP enviada, esperando respuesta... 200 OK  
Longitud:309 [text/html]  
Grabando a: « STDOUT »  
  
-          0%[          ] 0 --.  
-KB/s      <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
           "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">  
<html>  
<head>  
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>  
    <title>Servidor Web www61</title>  
</head>  
<body>  
<h2>Bienvenido al sitio www61.midns.es</h2>  
</body>  
</html>
```



- 100%[=====] 309 --.-KB/s en 0s

2020-05-16 18:36:13 (34,0 MB/s) — guardado [309/309]

La página ha sido recibida correctamente.

Desde el navegador de otra máquina, cliente DNS del servidor DNS local, solicitamos la URL:  
http://www61.midns.es:



También se puede usar la dirección IP como URL:



### c. Hosts virtuales por número de puerto

Se trata de una variante de la configuración anterior. El servidor HTTP Apache recibe una petición en una dirección y un número de puerto. Busca en su configuración el host virtual

asociado a esta dirección y a este puerto, y trata la petición según la configuración de este host virtual.

La ventaja de este método es que podemos usar una sola dirección IP. El inconveniente es que el cliente tiene que especificar el número del puerto del host virtual que se desea visitar, en la URL solicitada.

#### d. Hosts virtuales por nombre de host

El servidor HTTP Apache recibe una petición para un nombre de host en una dirección IP (y un número de puerto). Busca en su configuración el host virtual asociado a esta dirección, este puerto y este nombre de hosts y trata la petición según la configuración de este host virtual.

Desde el cliente, hay que especificar el nombre de host en la URL. Este nombre tiene que estar configurado en el sistema de resolución de nombres usado por el cliente (DNS, archivo `hosts`, etc.).

##### Ejemplo

*Host virtual por nombre de host en un servidor HTTP Apache de una distribución CentOS 8:*

*Añadimos el nuevo nombre en la zona DNS, como alias del nombre de host `centos8` del dominio DNS local:*

```
vi /var/named/db.midns.es
2020101602; serial [...]
www1      IN    CNAME  centos8
[...]
rndc reload
host www1
www1.midns.es is an alias for centos8.midns.es.
centos8.midns.es has address 192.168.1.60
```

*Configuramos un host virtual asociado al nombre de host `www1.midns.es`:*

```
vi /etc/httpd/conf/httpd.conf
```

```
[...] <VirtualHost www1.midns.es>
    ServerName www1.midns.es
    DocumentRoot /var/www/html/www1
    ErrorLog /var/log/httpd/ww1err.log
    TransferLog /var/log/httpd/ww1acc.log
</VirtualHost>
[...]
```

Creamos el directorio raíz del host virtual:

```
mkdir /var/www/html/www1
```

Comprobamos la sintaxis del archivo de configuración:

```
httpd -t
Syntax OK
```

Recargamos la configuración del servidor HTTP Apache:

```
systemctl reload httpd
```

Después de haber creado una página de inicio en el directorio del host virtual, comprobamos con el comando `wget`:

```
wget -O - www1.midns.es
--2020-05-16 18:46:23-- http://www1.midns.es/
Resolviendo www1.midns.es (www1.midns.es)... 192.168.1.60
Conectando con www1.midns.es (www1.midns.es)[192.168.1.60]:80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud:307 [text/html]
Guardao a:« STDOUT »

-      0%[          ] 0 --.
-KB/s      <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
```

```

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
  <title>Servidor Web www1</title>
</head>
<body>
<h2>Bienvenido al sitio www1.midns.es</h2>
</body>
</html>
-          100%[=====>]   307 --.-KB/s   ds 0s

2020-05-16 18:46:23 (36,3 MB/s) — escritos a stdout [307/307]

```

La página ha sido recibida correctamente.

Desde el navegador de otra máquina, cliente DNS del servidor DNS local, solicitamos la URL:  
<http://www1.midns.es>:



## 7. Control de los derechos de acceso de los usuarios

El servidor HTTP Apache permite controlar los derechos de acceso de los usuarios identificándolos y dándoles o no acceso a los diferentes directorios. Diferentes módulos permiten configurar múltiples métodos de autenticación para los usuarios que solicitan un acceso a un archivo gestionado por el servidor HTTP Apache.

### a. Sección de la declaración de un directorio: Directory

El control de accesos a los directorios se aplica a toda la arborescencia dentro del directorio, excepto si hay otros derechos posicionados en un subdirectorio.

Si el directorio está protegido en acceso, solo los usuarios autorizados podrán acceder a él.

La declaración de un directorio se hace usando la sección `Directory`, según la sintaxis siguiente:

```
<Directory CaminoDir>
[Directivas]
</Directory>
```

El camino de acceso del directorio puede estar especificado en camino absoluto o relativo a la raíz de los documentos configurada para el servidor o el host virtual que administra este directorio.

Pueden haber diferentes directivas especificadas en la sección, pero solo se aplicarán a ese directorio (y a sus subdirectorios, excepto si estos están asociados a una sección `Directory` específica).

### b. Directiva de control de acceso: Require

Dentro de una sección de directorio, esta directiva permite especificar qué usuarios tendrán acceso autorizado o rechazado al contenido del directorio.

Puede haber distintas directivas `Require` en la misma sección del directorio.

#### Sintaxis

```
Require [not] valid-user[user|group|ip|host [ Ident1 ... identN]
```

Donde:

<code>not</code>	Prohíbe el acceso.
<code>valid-user</code>	Cualquier usuario autenticado.
<code>user Ident ...</code>	Lista de los usuarios autorizados.
<code>group Ident ...</code>	Lista de los grupos de usuarios autorizados.
<code>ip Dir ...</code>	Lista de las direcciones IP autorizadas.
<code>host NombreDNS</code>	Lista de los nombres de hosts o de dominios DNS autorizados.

### Ejemplo

Este directorio es accesible para los usuarios del grupo `rrhh`, pero no para el usuario `becario`, incluso aunque esté dentro del grupo `rrhh`:

```
<Directory /var/www/html/rh
Require group rrhh
Require not user becario
</Directory>
```

### c. Directivas de autenticación

Existen muchos métodos de autenticación para los usuarios que solicitan acceso a un directorio. Están gestionados por módulos específicos.

Las siguientes directivas definen la autenticación deseada:

- `AuthName Título` : título de la ventana de diálogo de autenticación que será mostrada en el navegador cliente HTTP.

`AuthType TipoAutenticación` : método de autenticación que se utilizará. El tipo está relacionado con un módulo que tiene que estar cargado.

Después hay que usar directivas específicas para el método de autenticación deseado.

Algunos métodos de los más usados están descritos a continuación.

#### d. Método de autenticación local: `AuthType Basic`

Este método usa un archivo en el que están definidas las cuentas de los usuarios específicos del servidor HTTP Apache.

Para crear un usuario en este archivo, utilizamos el comando `htpasswd` :

```
htpasswd [-c|D] CaminoArchivo NombreUsuario
```

Donde:

<code>-c</code>	Crea el archivo.
<code>-D</code>	Suprime la cuenta de usuario del archivo.
<code>CaminoArchivo</code>	Camino de acceso del archivo de las cuentas de los usuarios.
<code>NombreUsuario</code>	Cuenta de usuario que se creará en el archivo.

El comando solicita que se teclee dos veces la contraseña que se asociará a la cuenta de usuario.

Si la cuenta del usuario existe, su contraseña se actualizará.

#### Ejemplo

Creación de un archivo de cuentas de usuarios con dos cuentas:

```

htpasswd -c passwd.httpd phi
New password: XXXX
Re-type new password: XXXX
Adding password for user phi
htpasswd passwd.httpd mar
New password: YYYY
Re-type new password: YYYY
Adding password for user mar
cat passwd.httpd
phi:$apr1$Fnw3KTni$eMMK.B/palaBZclOBTzLh/
mar:$apr1$I1M8AGBX$cdMbAwopgjTBdiCNpBU0C1

```

Para que el método de autenticación local sea válido, es necesario que tres módulos sean configurados durante la carga:

```

auth_basic_module
authn_file_module
authz_user_module

```



Los módulos necesarios pueden ser diferentes dependiendo de la versión del servidor HTTP Apache implementado.

El control de acceso por dirección IP o por nombres DNS (`Require ip` y `Require host`) está gestionado por el módulo `mod_authz_host`.

Este módulo hace obsoleto al antiguo módulo `mod_access_compat` que usaban las directivas siguientes para gestionar las direcciones o nombres de hosts autorizados o denegados:

```

Allow from all|DirIp|host
Deny from all|DirIp|host

```



En la sección del directorio que se protegerá, habrá que especificar dos directivas, además de las que ya se han visto anteriormente:

✓ `AuthType Basic`

Esta directiva especifica el tipo de autenticación utilizada.

✓ `AuthUserFile CaminoArchivo`

`CaminoArchivo` es el camino de acceso del archivo de las cuentas de usuarios, creado con el comando `htpasswd`.

O

`AuthGroupFile CaminoArchivo`

`CaminoArchivo` es el camino de acceso al archivo de los grupos de usuarios, si usamos el control de acceso por grupos de usuarios. El archivo especificado es un archivo de texto donde se declaran los grupos y sus miembros, usando el formato de línea siguiente:

`NombreGrupo:Usuario1 Usuario2 ... UsuarioN`

### Ejemplo

Configuración en un servidor HTTP Apache CentOS 8 de un directorio con el acceso reservado a los usuarios autenticados localmente.

Usamos el archivo de usuarios creado en el ejemplo anterior.

Comprobamos si los módulos necesarios han sido cargados:

**httpd -M | grep auth\_basic\_module**

auth\_basic\_module (shared)

**httpd -M | grep authn\_file\_module**

authn\_file\_module (shared)

**httpd -M | grep authz\_user\_module**

authz\_user\_module (shared)

Los tres módulos están activos.

Declaramos un directorio con acceso protegido, en el archivo de configuración

`httpd.conf`:

```
vi /etc/httpd/conf/httpd.conf
[...]
<Directory /var/www/html/rh>
  AuthType basic
  AuthUserFile /etc/httpd/passwd.httpd
  AuthName "Identificación obligatoria"
  Require valid-user
</Directory>
```

Creamos el directorio, con una página de inicio:

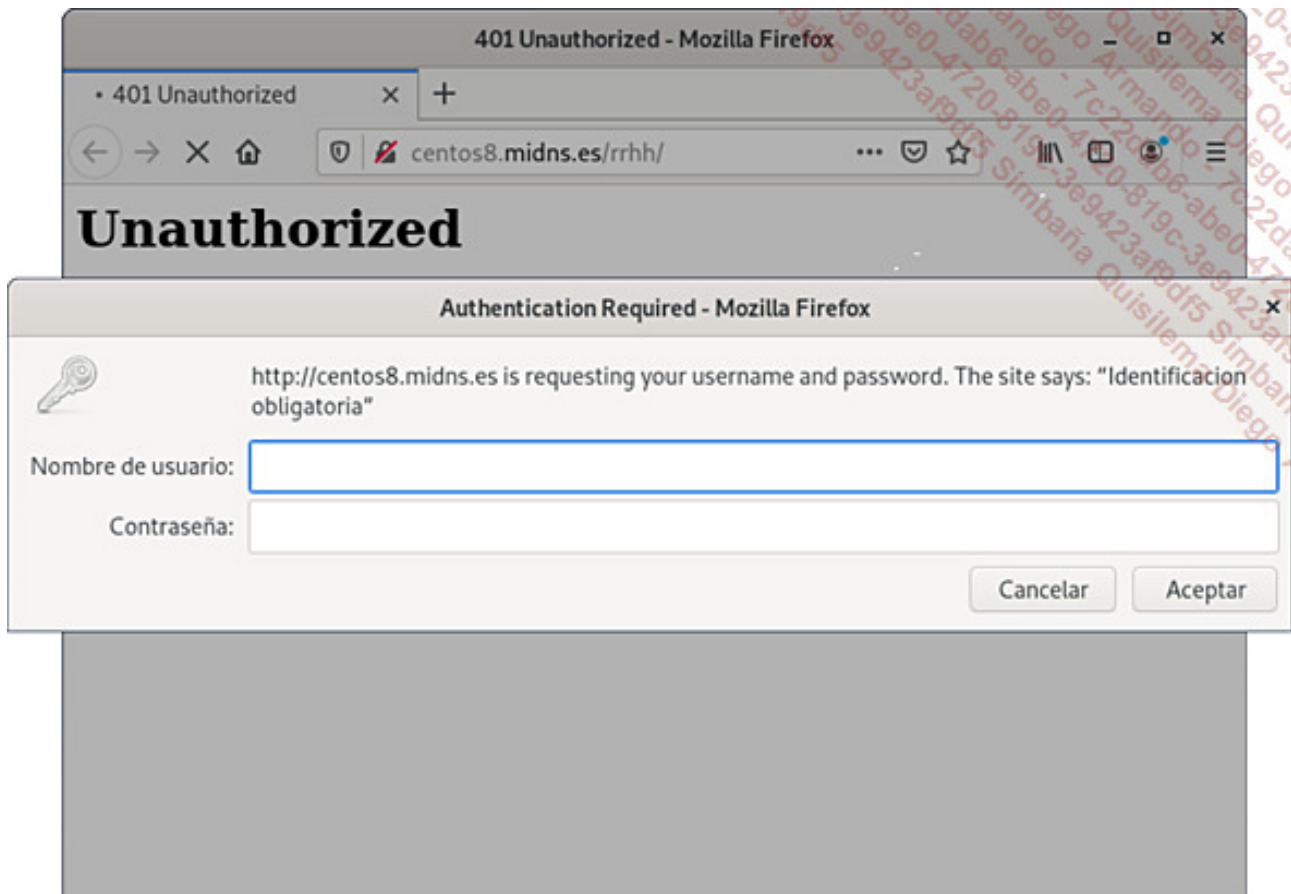
```
mkdir /var/www/html/rh
vi /var/www/html/rh/index.html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
  <title>Intranet RRHH</title>
</head>
<body>
<h2>Bienvenido a la intranet de los recursos humanos </h2>
</body>
</html>
```

Comprobamos la configuración y la recargamos:

```
httpd -t
Syntax OK
systemctl reload httpd
```

Intentamos acceder a la página de inicio desde un navegador:

<http://centos8.midns.es/rrhh>



El servidor HTTP Apache envía al cliente una ventana de identificación.

Después de haber tecleado el nombre de un usuario válido y su contraseña, se autorizará el acceso al directorio:



### e. Autenticación por anuario LDAP

El inconveniente de la autenticación local es que las cuentas de los usuarios son específicas del servidor HTTPD Apache y los usuarios no pueden cambiar la contraseña que se les ha atribuido.

Una solución más flexible es la de usar el anuario LDAP de la organización. En ese caso, el acceso a los directorios protegidos se hará pidiendo al servidor de anuario LDAP la autenticación de la información tecleada por el usuario en la ventana de autenticación.

Para ello habrá que cargar, además de los módulos anteriores, el módulo LDAP: `mod_authnz_ldap`.

También hay que configurar el control de acceso en la sección del directorio, con una directiva que especifique el servidor LDAP que se tendrá que interrogar:

```
AuthLDAPUrl IdAnuarioLDAP
```

La directiva `Require` permite opcionalmente definir los usuarios, los grupos LDAP o los DN LDAP (*Distinguished Name*) autorizados o no a acceder al directorio:

```
Require ldap-user Nombre1 ... NombreN
Require ldap-group Nombre1 ... NombreN
Require ldap-dn DistName
```

## f. Control de acceso por archivo .htaccess

Es posible definir los parámetros de control de acceso en un archivo situado en el directorio que se quiera proteger. Este archivo se llama, por defecto, `.htaccess`. Las directivas necesarias para el control están especificadas en este archivo en lugar de estar en la sección `Directory` asociada al directorio en el archivo de configuración del servidor HTTP Apache.

Esta técnica permite definir el control de acceso al directorio sin tener que modificar la configuración del servidor HTTP Apache. Sin embargo, no está aconsejada desde el punto de vista de la seguridad.

La directiva `AllowOverride` permite configurar el uso de los archivos `.htaccess`, globalmente o en un directorio:

AllowOverride Parám

He aquí los valores posibles de `Parám`:

<code>All</code>	Archivo <code>.htaccess</code> autorizado, para todas las directivas en el directorio (valor por defecto).
<code>None</code>	Archivo <code>.htaccess</code> ignorado.
<code>AuthConfig</code>	Solo se autorizan las directivas de autenticación en un archivo <code>.htaccess</code> .



Se aconseja especificar la directiva global `AllowOverride None` en el archivo de configuración del servidor HTTP Apache. De esta manera, Apache no tendrá que comprobar la presencia de un archivo `.htaccess` en el directorio de cada página solicitada. Se podrá, si se considera necesario, activar el uso del archivo en un directorio específico, usando la directiva `AllowOverride AuthConfig` en una sección `Directory` asociada al directorio.