There are no translations available.
En el artículo presentamos las principales características y funcionalidades de este potente servidor web, en su versión 2.2 y en entorno Ubuntu.

# Apache 2.2: servidor web

## 1 Introducción

Apache 2.2 es un servidor web de software libre desarrollado por la Apache Software Foundation (http://www.apache.org/) cuyo objetivo es servir o suministrar páginas web (en general, hipertextos) a los clientes web o navegadores que las solicitan.

En el artículo presentamos las principales características y funcionalidades de este potente servidor web, en su versión 2.2 y en entorno Ubuntu.

## 2 Apache 2.2

La arquitectura utilizada es cliente/servidor, es decir, el equipo cliente hace una solicitud o petición al equipo servidor y éste la atiende.

En el equipo cliente se ejecuta una aplicación llamada 'navegador o cliente web' que:

-

Sirve de interfaz con el usuario: atiende sus peticiones, muestra los resultados de las consultas y proporciona al usuario un conjunto de herramientas que facilitan su comunicación con el servidor.

-

Se comunica con el servidor web: transmite las peticiones de los usuarios.

El protocolo utilizado para la transferencia de hipertexto es **HTTP** (HiperText Transfer Protocol) que está basado en el envío de mensajes y establece el conjunto de normas mediante las

cuales se envían las peticiones de acceso a una web y la respuesta de esa web.

HTTP es un protocolo sin estado, es decir, no recuerda nada relativo a conexiones anteriores a la actual. La conexión sólo tiene la duración correspondiente a la transmisión de la página solicitada si la encuentra, y si no la encuentra devuelve un código de error.

El servidor web Apache 2.2 proporciona contenidos al cliente web o navegador como:

1.

**Páginas estáticas**: es el uso más generalizado que se hace de un servidor web. De esta forma se transfieren archivos HTML, imágenes, etc y no se requiere un servidor muy potente en lo que al hardware se refiere.

2.

**Páginas dinámicas**: la información que muestran las páginas que sirve Apache cambia ya que se obtiene a partir de consultas a bases de datos u otras fuentes de datos. Son, por tanto, páginas con contenido dinámico, cambiante.

Su curioso nombre hace referencia a sus orígenes. Cuando el proyecto inicial (Centro Nacional de Actividades de Supercomputación, NCSA, Universidad de Illinois) fue abandonado por su principal desarrollador, Rob McCool, diferentes webmasters comenzaron a desarrollar 'parches' para el código fuente de este servidor inicial y mediante el correo electrónico sincronizaban sus aportaciones. De esta forma apareció el proyecto Apache, cuyo nombre se debe a: **A PAtCHy server .**

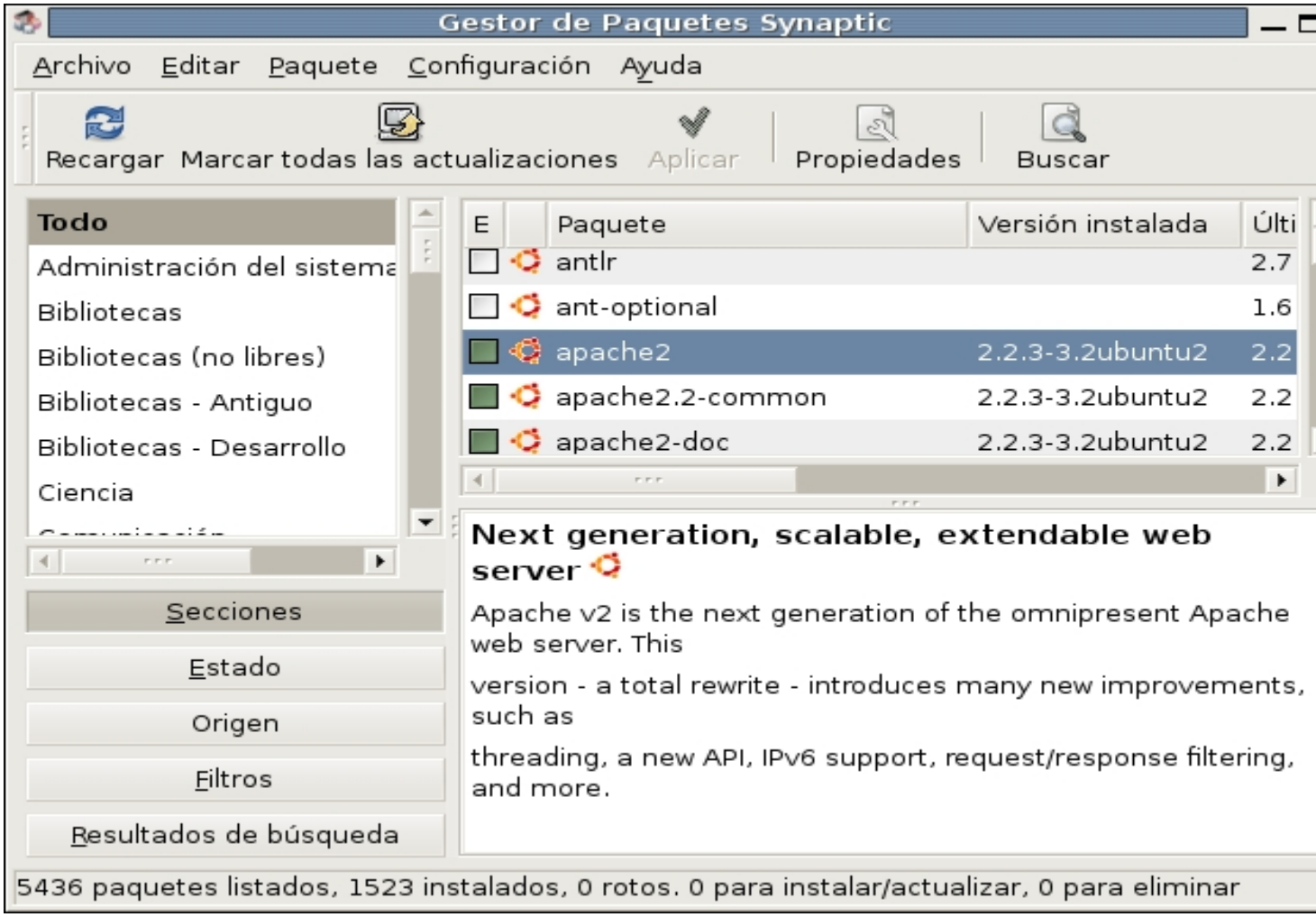
La primera versión de Apache es la 0.6 en 1995, y en la actualidad la versión disponible es la 2.2 ( http://httpd.apache.org/docs/2.2/en/ ).

# 3 Instalación. Arranque y parada.

La instalación la realizamos desde el entorno gráfico utilizando la herramienta Synaptic (**Sistema -> Administración -> Gestor de paquetes Synaptic**
).

Marcamos para instalar los paquetes siguientes: **apache2**, **apache2.2-common, apache2-mpm-worker** y **apache2-utils.**
Aplicamos los cambios y hacemos un seguimiento de la instalación desplegando la pestaña Detalles.

La versión que se instala es la 2.2.3.




Si todo ha ido bien al abrir el navegador web Firefox (Aplicaciones -> Internet -> ) e ir a http://l

ocalhost
deberá aparecer una página indicando que el servidor Apache2 está instalado y ejecutándose.

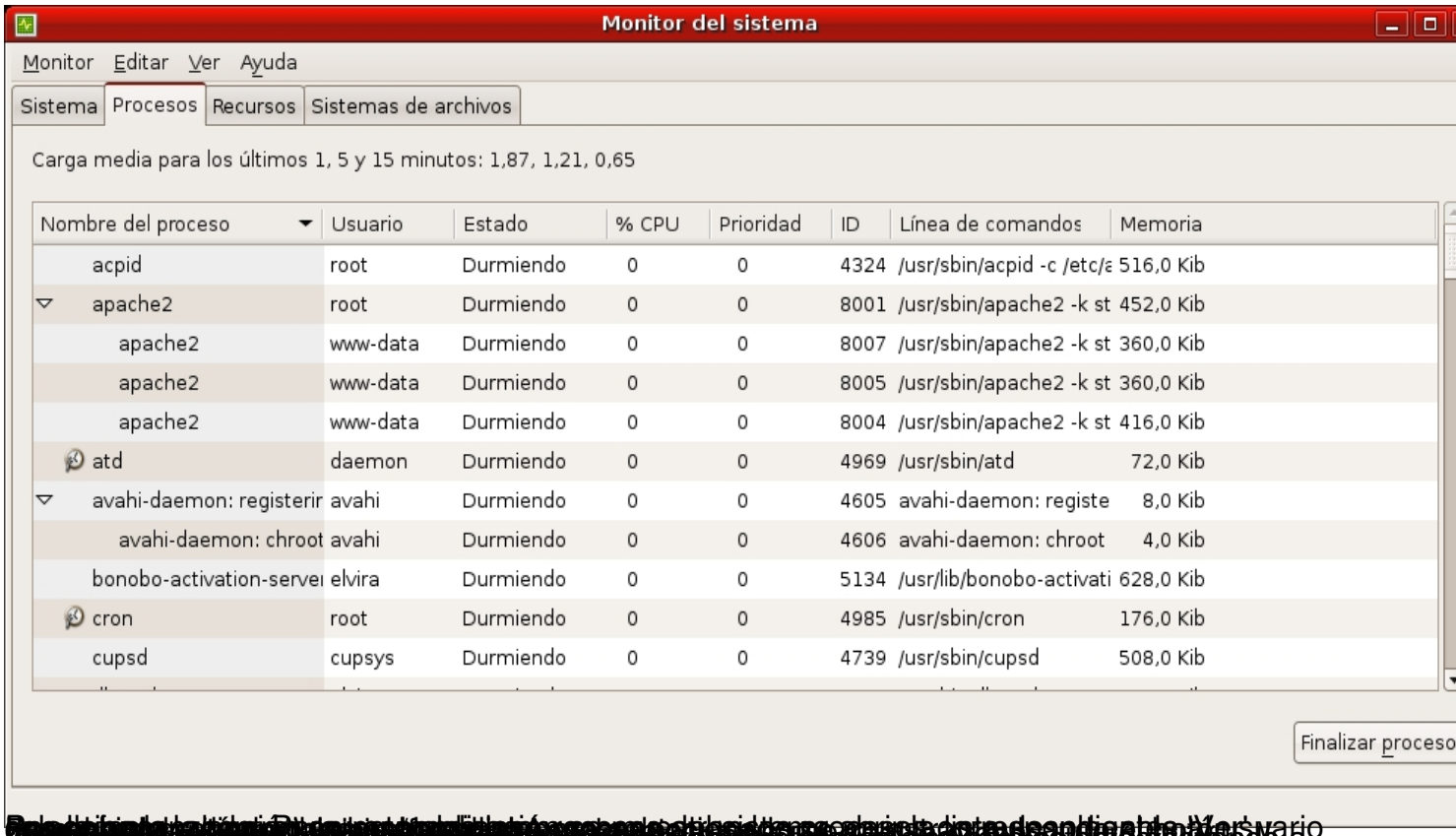
Si se dispone de un nombre de dominio cualificado que se puede resolver mediante **DNS** o localmente con el archivo
**/etc/hosts**
, podemos utilizarlo para invocar al servidor desde el cliente web.

Podemos hacer una comprobación de que realmente está ejecutándose el servidor web Apache 2.2 ejecutando en una terminal la orden siguiente:

**#ps axl | grep apache2**

Deberán aparecer los procesos lanzados por Apache y que están escuchando en el puerto 80 las peticiones de los clientes.

Desde el entorno gráfico ir a **Sistema -> Administración -> Monitor del sistema**




# 4 Configuración básica

El archivo de configuración de Apache 2.2 en Ubuntu es **/etc/apache2/apache2.conf**. Es un archivo muy extenso en el que todas las directivas disponibles están explicadas, incluyendo ejemplos de utilización.

Contiene una directiva por línea y no se hace distinción entre mayúsculas y minúsculas. Las líneas que comienzan con el carácter '#' se consideran comentarios.

En http://httpd.apache.org/docs/2.2/en/mod/quickreference.html existe una guía completa de las directivas disponibles. Para el artículo sólo se van a comentar aquellas directivas que se utilizen en los ejemplos de configuración.

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

1.

Las que se evalúan para cada petición que se recibe y se aplican las directivas que se incluyen. Dentro de este grupo están **<Directory>, <Files>, <Location>, <VirtualHost>** entre otras. Las secciones **<Directory>** y **<Files>** están relacionadas con el sistema de archivos y **<Location>** está relacionada con el espacio web.

2.

Las que se evalúan sólo al inicio o reinicio del servidor, como **<IfDefine>** e **<IfModule>**. Si al iniciar el servidor las condiciones son las adecuadas, las directivas que incluyen estas secciones se aplicarán a todas las peticiones que se reciban.

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

**<Directory /var/www>**

**Order Allow,Deny**

**Allow from dominio.com**

**Deny from pc01.dominio.com**

**</Directory>**

Que indica que el directorio /**var**/**www**/ (origen por defecto de los contenidos web) está configurado para permitir el acceso a cualquier máquina del dominio
**dominio**
**.com**
excepto al
**host**
**pc01**
.

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

-

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

**Include /etc/apache2/conf.d**

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

-

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

-

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

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

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

-

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

-

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

-

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

-

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

# 5 Utilización de módulos

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

Existen dos tipos de módulos:

1.

Los que se compilan de forma "estática" cada vez que se compila Apache2.

2.

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

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

**#a2enmod userdir**

Module userdir installed; run /etc/init.d/apache2 force-reload to enable.

**#/etc/init.d/apache2 force-reload**

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

La directiva **UserDir** disponible para este módulo indica el nombre del subdirectorio, dentro del directorio home de cada usuario, donde estarán los archivos HTML que podrán ser servidos por Apache2. Por defecto el subdirectorio es **public_html**.

**UserDir public_html**

Esta directiva irá incluida en el archivo de configuración por defecto para este módulo /**etc**/**apache2**/**mods-available**/**userdir.conf**
:

**<IfModule mod_userdir.c>**

```
UserDir public_html

UserDir disabled root

<Directory /home/*/public_html>

AllowOverride FileInfo AuthConfig Limit

Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec

</Directory>

</IfModule>
```

La sección **<IfModule>** anterior:

-

Indica que el directorio donde estarán los archivos html de los usuarios es **public_html** (podemos cambiar el nombre).

-

El usuario *root* está desactivado (por seguridad). Si no estuviese desactivado accediendo a **servidor.apache2.com/~root** se podría llegar al directorio root.

-

La sección **<Directory>** está indicando que los home de todos los usuarios podrán ser listados sus contenidos ( **Options Indexes)**. Es decir, cualquier usuario cuyo home contenga el directorio **public_html** podrá publicar su contenido en el servidor Apache2 correspondiente.

Si *usuario1*, *usuario2*,... son usuarios que tienen login en el sistema podemos poner el carácter **'~'** en los caminos de las URLs. Por ejemplo si escribimos:

**http://servidor.apache2.com/~usuario1**

se accede al home del usuario *usuario1* y no al directorio /**var**/**www**/ al que accederíamos si pusiésemos sólo **servidor.apache2.com**

.

Si se quiere acceder a la página web sin necesidad de utilizar el carácter '~' habrá que definir un alias en /**etc**/**apache2**/**conf.d**/**alias**.

**alias /usuario1/ /home/usuario1/public_html/**

## 6 Archivos .htaccess

Los archivos **.htaccess** permiten a los usuarios que no tienen permisos modificar la configuración y así poder ejercer algún control sobre el comportamiento de su parte del servidor Apache2.

Las directivas que modifican el comportamiento se colocan en un archivo **.htaccess** situado en el directorio al que tiene que afectar, junto a todos sus subdirectorios.
E
l archivo
**.htaccess**
se carga cada vez que se solicita un documento.

Las modificaciones introducidas no requieren reiniciar el servidor web.

Para que el servidor haga caso de los archivos **.htaccess** hay que incluir la directiva **AllowOverride**
(permite sobreescritura) dentro de la sección
**<Directory>**
que interese.

Si se quiere deshabilitar completamente la utilización de estos archivos hay que incluir la directiva:

```
AllowOverride None
```

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

En el archivo de configuración /**etc**/**apache2**/**apache2.conf** encontramos:

```
AccessFileName .htaccess
```

Los motivos fundamentales para no permitir (o no recomendar) la utilización de archivos **.htaccess**
son los siguientes:

1.

El primero es de funcionamiento.

Cuando se activa AllowOverride para permitir el uso de archivos **.htaccess**, Apache2 mirará en cada directorio buscando estos archivos. De esta forma se buscan tanto si existen como si no y entran en funcionamiento tanto si se utilizan como si no.
Además Apache2 buscará archivos
**.htaccess**
en todos los directorios de niveles superiores realizando un conjunto de accesos al sistema de archivos adicionales que relentizan su funcionamiento.

2.

El segundo es de seguridad.

Se está permitiendo que los usuarios modifiquen la configuración del servidor y pueden ser cambios de los que no se tiene control.

# 7 Hosts virtuales

Trabajar con Hosts Virtuales consiste en ejecutar más de un sitio web en el mismo servidor.

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

La utilización de hosts virtuales tiene como ventajas la versatilidad para crear diferentes sitios web configurables; el precio, ya que se necesita sólo una máquina para alojar varios servidores web; una configuración del sistema sirve para todos los servidores web; se actualiza sólo una vez y no requiere ningún software ni hardware adicional.

Apache soporta dos tipos de hosts virtuales:

1.

Hosts virtuales basados en nombres

Permiten alojar varios nombres de host (o dominios) en una misma máquina (IP). Todos los hosts virtuales que comparten la misma IP deben declararse mediante la directiva **NameVirtual Host** .

2.

Hosts virtuales basados en IP

Una máquina responde de diferente manera a diferentes direcciones IP. Es decir, tenemos múltiples IPs asignadas al sistema y queremos que cada una de ellas soporte un sitio web.

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

### 7.1 Host virtual basado en nombre

Para usar hosts virtuales basados en nombres se debe especificar en el servidor qué dirección IP se va a usar mediante la directiva **NameVirtualHost**.

Si escribimos:

**NameVirtualHost ***

el '*' indica que el servidor acepta todas las solicitudes entrantes.

¿Cómo sabe Apache2 si una petición va dirigida a uno u otro host virtual?

La respuesta está en las cabeceras del HTTP/1.1. Cuando un navegador envía una petición al servidor usando el protocolo HTTP/1.1 envía una cabecera del tipo **host: nombre_de_un_host** con la que Apache2 puede diferenciar las peticiones de los distintos hosts virtuales.

El siguiente paso es crear un bloque **<VirtualHost>** para cada host diferente que se quiera alojar en el servidor.

**<VirtualHost * >**

Con la directiva <**NameVirtualHost *>** se le está diciendo a Apache2 que se activan los hosts virtuales por nombre para la IP dada en **ServerName** (externo), y los siguientes grupos <VirtualHost *>...</VirtualHost> definen los hosts virtuales de nuestro servidor.

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

Como ejemplo de utilización añadimos un host virtual a un servidor web ya existente. Suponemos que el servidor web existente (host virtual por defecto) dispone de su configuración como host virtual en el archivo **/etc/apache2/sites-available/default**.

Supongamos que ya se está sirviendo el dominio servidor.dominio.com y se quiere añadir el host virtual virtual.dominio.com , que apunta a la misma dirección IP.

Para preparar el nuevo sitio virtual habrá que:

1.

Editar el archivo /**etc**/**hosts** y añadir la línea siguiente (si no se está utilizando DNS):

**192.168.1.1 virtual.dominio.com virtual**

Si el aula dispone de servicio DNS se configura para que ambos dominios servidor.dominio.com  virtual.dominio.com ) apunten a la misma dirección IP. Esto se hace mediante el tipo de registro **CNAME** (nombre canónico).

1.

Crear el directorio /**var**/**www**/**virtual**

2.

Crear un archivo llamado **index.html** con el contenido:  *Servidor virtual virtual.dominio.com*

3.

Crear el archivo de configuración para el nuevo sitio virtual que llamaremos '**virtual**' con el contenido que se indica:

```
        NameVirtualHost *

<VirtualHost *>
    ServerName virtual.dominio.com


DocumentRoot /var/www/virtual


<Directory /var/www/virtual>


Options FollowSymLinks


AllowOverride None


</Directory>


</VirtualHost>
```

4. Activar el sitio ejecutando la orden:

```
#a2ensite virtual
```

5. Reiniciar el servidor para que lea los cambios realizados en el archivo de configuración.

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

1.

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

2. Ejecutar la siguiente orden para comprobar los hosts virtuales configurados[1]:

```
#apache2 -S
```

### 7.2 Host virtual basado en IP

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

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

Si no se dispone de un equipo con varias interfaces de red se puede probar creando alias de la tarjeta disponible.

Para ello hay que editar el archivo **/etc/network/interfaces** y añadir la porción de código siguiente adecuando los datos al aula:

```
auto eth0:0
```

```
iface eth0:0 inet static
```

```
address 192.168.1.2
```

```
netmask 255.255.255.0
```

```
network 192.168.1.0
```

```
gateway 192.168.1.100
```

Se está creando un alias de la tarjeta ethernet eth0 y se le asigna la IP 192.168.1.2. Se supone que la tj eth0 tiene IP 192.168.1.1.

A partir de este momento el sistema ya dispone de 2 interfaces de red, una real y un alias que permite definir un host virtual alojado en el mismo servidor web, además del dominio inicial serv idor.dominio.com. La configuración para el host virtual
virtual.dominio.com
es la siguiente y podría ser incluida tanto en el archivo
**default**
como en un archivo diferenciado dentro de
**sites-available**
:

```
<VirtualHost 192.168.1.2>
```

```
ServerName virtual.dominio.com
```

```
DocumentRoot /var/www/virtual
```

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

CustomLog /var/log/apache2/virtual-access.log combined

</VirtualHost>
```

También es posible configurar hosts virtuales mixtos, es decir basados en nombres y en IP.

### 7.3 Host virtual basado en puertos

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

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

```
NameVirtualHost 192.168.1.1:80

<VirtualHost 192.168.1.1:80>

ServerName servidor.dominio.com

DocumentRoot /var/www/servidor80
```

```
</VirtualHost>

<VirtualHost 192.168.1.1:443>

ServerName servidor.dominio.com

DocumentRoot /var/www/servidor443

</VirtualHost>
```

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

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

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

# 8 Autenticación y control de acceso

Respecto al proceso de Autenticación de usuarios en Apache 2 existen dos métodos:

- 

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

- 

Digest: el usuario en el navegador web introduce su login y contraseña y se envían al servidor **cifrados**
.

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

### 8.1 Autenticación básica

El módulo que controla este método de autenticación es **mod_auth_basic** y tiene la ventaja de que está soportado por todos los navegadores web. Por el contrario, tiene el inconveniente de que el login y la contraseña no van cifradas del navegador web al servidor.

En el archivo /**etc/apache2/sites-available/default**, o en el archivo relativo al host virtual correspondiente, habrá que añadir un bloque **<Directory>...</Directory>** por cada directorio que se quiera proteger:

```
<Directory "/var/www/privado">

AuthType Basic

AuthName "Directorio privado"
```

**AuthUserFile /etc/apache2/passwd/.htpasswd**

**Require valid-user**

**</Directory>**

Donde:

-

**AuthName**: nombre del **dominio de autenticación**. Define el conjunto de recursos que estarán sujetos a los mismos requisitos de autenticación. También es el texto que aparecerá en la ventana que pide el usuario y la clave.

-

**AuthType**: tipo de autenticación.

**Basic**: la contraseña se negocia sin encriptar

**Digest**: la contraseña se negocia encriptada

- **AuthUserFile**: ubicación del archivo de texto que contendrá los nombres de usuario y contraseñas usadas en la autenticación HTTP básica. Se suele llamar **.htpasswd.**
Previamente hay que crear el directorio **/etc/apache2/passwd/.**

- **Require**: usuarios que tienen acceso a los recursos especificados. Opciones disponibles:

-
-

**valid-user**: cualquier usuario incluido en el archivo de contraseñas **.htpasswd**.

-

**user <lista de usuarios>**: lista de usuarios de **.htpasswd**, separados por espacios, que pueden acceder.

-

**Satisfy**: al utilizar esta directiva determina si se deben cumplir todos los requisitos (**All**) o cualquiera (
**Any**
).

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

**#htpasswd -c /etc/apache2/passwd/.htpasswd nombre_usuario**

La opción **-c** permite crear el archivo **.htpasswd** con el primer usuario dado de alta, que además no tiene porque ser un usuario existente en el sistema.

Los permisos del archivo **.htpasswd** deben ser **644**, es decir lectura y escritura para el dueño,

que es root y lectura para el grupo y los otros.

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

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

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




## 8.2 Autenticación HTTP Digest

El módulo que controla este método de autenticación es **mod_auth_digest**. Tiene la ventaja de que el login y la contraseña van cifradas del navegador web al servidor. Por el contrario, tiene el inconveniente de que no está soportado por todos los navegadores web.

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

```
#a2enmod auth_digest
```

**#/etc/init.d/apache2 force-reload**

Utiliza **MD5** (Message Digest Authentication) para generar un **hash** que es el que se transmite o envía al servidor.

En el archivo /**etc**/**apache2**/**sites-available**/**default** habrá que añadir un bloque **<Directory>... </Directory>**
por cada directorio que queramos proteger:

**<Directory "/var/www/privado">**

**AuthName "Directorio privado"**

**AuthType Digest**

**AuthDigestDomain http://servidor.dominio.com/privado/**

Donde:

**AuthName**: indica el nombre del dominio de autenticación (realm).

**AuthType**: indica que el método a usar es 'Digest'.>

**AuthDigestProvider**: indica el soporte utilizado para la autenticación. Por defecto es file (archivo).

**AuthDigestDomain**: dominio protegido con autenticación digest.

**AuthUserFile**: indica donde se encuentra el archivo de contraseñas que ahora llamamos **.htdigest** .

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

**#htdigest /etc/apache2/passwd/.htdigest zona_privada nom_usuario**

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

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

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

**#htdigest -c /etc/apache2/passwd/.htdigest "Directorio privado" usuario1**

En el archivo de configuración **/etc/apache2/sites-available/default** hay que añadir un bloque **<Directory>...</Directory>**
para el directorio que queremos proteger:

**Alias /privado /var/www/privado**

**<Directory "/var/www/privado">**

**AuthType digest**

**AuthName "Directorio privado"**

**AuthUserFile /etc/apache2/passwd/.htdigest**

**Require user usuario1**

**</Directory>**

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




# 9 Analizador de logs awstats

Desde Edubuntu vamos a utilizar el analizador de accesos de Apache **Awstats** (Advanced Web Statistics) cuya página oficial es
http://www.awstats.org

**Awstats** soporta cualquier sistema operativo, ya que al estar escrito en Perl es suficiente que el servidor que lo va a interpretar tenga el módulo correspondiente instalado. Puede generar estadísticas en 33 idiomas, entre los cuales se encuentran el *castellano.*

Su funcionamiento se basa en:

- la lectura de los archivos **access.log** de Apache2.
- a partir de estos archivos **Awstats** genera sus propios archivos con las estadísticas. Esta generación se hace de forma periódica con un **cron**.
- para mostrar las estadísticas, **Awstats** lee los archivos que él ha generado y muestra los resultados como HTML estático en un navegador.

## 9.1 Instalación de awstats

La instalación en Edubuntu es:

```
# apt-get install awstats
```

Los archivos y directorios que instala/crea son los siguientes:

| Archivos/Directorios | Descripción |
| --- | --- |
| **/usr/share/doc/awstats/** | Directorio donde se guardan archivos de ejemplo y ayuda. |
| **/usr/share/awstats/lang/awstats-es.txt** | Archivo de idioma. Es un archivo de texto. |
| **/usr/share/awstats/plugins/** | Directorio en el que se instalan los plugins que vienen por defecto con el paquete. |
| **/usr/share/awstats/icon/** | Directorio donde se instalan los iconos que se van a utilizar para mostrar las estadísticas. |

**/usr/lib/cgi-bin/awstats.pl**

Archivo que va a generar las estadísticas a partir de los logs del sistema. Escrito en Perl.

**/etc/cron.d/awstats**

Entrada de cron para actualizar las estadísticas periódicamente.

**/etc/awstats/awstats.conf**

Archivo de configuración de **Awstats**

### 9.2 Configuración de awstats y funcionamiento

El archivo de configuración de **awstats** es /**etc/awstats/awstats.conf.**>

Es importante comprobar que se tiene activado el módulo **cgid** ya que es necesario para el funcionamiento de **Awstats**.

### A. Tipo de logs

Hay que asegurarse de que el servidor Apache2 'logea' los accesos de manera combinada utilizando la siguiente directiva:

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

Por otro lado, también hay que asegurarse de que Apache2 hace los logs de la forma que necesitamos. Para ello hemos de comprobar que los logs tienen el siguiente formato:

**servidor.dominio.com 192.168.0.1 - - [13/May/2007:16:45:52 +0200] "GET /webalizer HTTP/1.1" 20**

que se corresponde con una entrada del archivo /**var/log/apache2/access.log**.

Indicamos a **Awstats** el formato en que están los logs mediante la directiva **LogFormat:**

| Valor | Descripción |
| --- | --- |
| **1** | Apache2 'logea' los accesos de manera combinada (NCSA combined/XLF/ELF log format) |

| |
|---|
| **2** |

| |
|---|
| Fomato de log de IIS (W3C log format) |

| |
|---|
| **3** |

| |
|---|
| Formato de log nativo Webstar |

| |
|---|
| **4** |

| |
|---|
| Formato de log nativo de Apache o Squid (NCSA common log format). Con LogFormat=4 algunos nave |

| |
|---|
| **5** |

| |
|---|
| Formato de log nativo de ISA server. |

| |
|---|
| **6** |

| |
|---|
| Formato de log combinado de Lotus Notes |

Comprobar las siguientes líneas en /**etc**/**awstats**/**awstats.conf:**

```
LogFormat=1
```

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

**B. Idioma**

**Awstats** viene preconfigurado para mostrar los logs y estadísticas en inglés. Hay que cambiarlo a castellano. Los archivos de idioma han sido instalados en el directorio /**usr**/**share**/**awstats**/**lang**/
.

Comprobamos que en /**etc**/**awstats**/**awstats.conf** está disponible esta ruta que es donde debe buscar los archivos de idioma.

```
DirLang="/usr/share/awstats/lang"
```

Para que el idioma por defecto sea el castellano, hemos que sustituir el valor de la directiva **Lang**
'en' por 'es'.

```
Lang="es"
```

La página que se genera muestra en la parte superior izquierda unas banderas que permiten elegir el idioma que queramos que muestre la página.

**C. Imágenes**

Para que las imágenes que contiene la página que se va a generar se vean correctamente, hay que indicarle a **Awstats** donde están ubicadas. La directiva **DirIcons** en **/etc/awstats/awstats.conf** contiene la ruta de las imágenes relativa a la ubicación de la página web.

Utilizamos los *alias* del servidor Apache2. Tenemos que añadir la siguiente línea al archivo de configuración de apache **/etc/apache2/apache2.conf** o crear un archivo **alias** en el directorio **conf.d/**.

```
Alias /awstats-icon/ /usr/share/awstats/icon/
```

y, entonces en la directiva **DirIcons** escribimos el nombre del alias creado:

```
DirIcons="/awstats-icon"
```

**D. Intérprete de Perl**

El archivo **awstats.pl** necesita del intérprete de Perl para generar las estadísticas de los logs. Hay que asegurarse de que está bien configurado indicando al script en Perl donde se encuentra el intérprete. La primera línea del archivo **/usr/lib/cgi-bin/awst**

**ats.pl** es:

```
#!/usr/bin/perl
```

**E. Dominio del sitio**

Como en un mismo host pueden existir distintos dominios (hosts virtuales), la directiva **SiteDomain** indica a cuál de esos dominios nos estamos refiriendo.

En nuestro caso:

```
SiteDomain="servidor.apache2.com"
```

**F. Formato del informe**

Indicamos con la directiva **LogType** el tipo de archivo que contendrá el informe generado:

```
LogType=W
```

W indica informe web

M para archivos mail

F para archivos de log FTP

Con esto quedaría finalizada la configuración básica de **Awstats**.

Para visualizar la página web generada ir a http://servidor.apache2.com/cgi-bin/awstats.pl.

Como puede observarse las estadísticas están vacías. Eso es así porque **awstats.pl** no lee los datos de los logs directamente sino que genera un archivo de texto y muestra los datos a partir de dicho texto. Los archivos de texto generados se encuentran en:

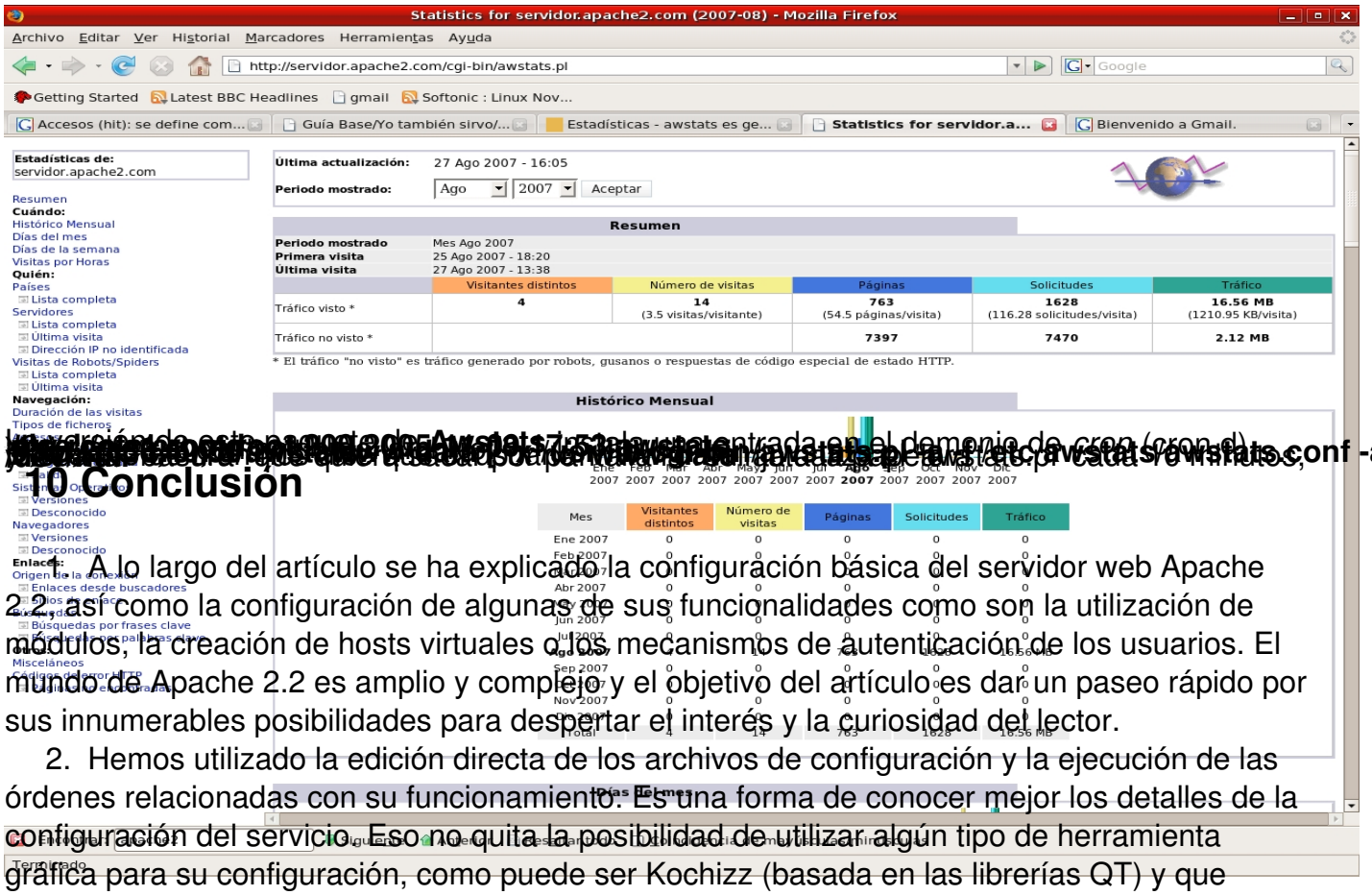
```
# ls -l /var/lib/awstats/
```

```
-rw-r--r-- 1 root root 6445 2007-08-13 20:08 awstats022008.txt
```

Para generar estos archivos por primera vez o actualizarlos manualmente, ejecutamos el archivo **awstats.pl** con los siguientes parámetros:

```
#/usr/lib/cgi-bin/awstats.pl -config=servidor -update
```

Volver a ir al navegador y comprobar que cambian las estadísticas y aparecen ya datos.




## 10 Conclusión

1. A lo largo del artículo se ha explicado la configuración básica del servidor web Apache 2.2, así como la configuración de algunas de sus funcionalidades como son la utilización de módulos, la creación de hosts virtuales o los mecanismos de autenticación de los usuarios. El mundo de Apache 2.2 es amplio y complejo y el objetivo del artículo es dar un paseo rápido por sus innumerables posibilidades para despertar el interés y la curiosidad del lector.

2. Hemos utilizado la edición directa de los archivos de configuración y la ejecución de las órdenes relacionadas con su funcionamiento. Es una forma de conocer mejor los detalles de la configuración del servicio. Eso no quita la posibilidad de utilizar algún tipo de herramienta gráfica para su configuración, como puede ser Kochizz (basada en las librerías QT) y que puede ser tema de un artículo específico para esta herramienta.

## NOTAS

1)Si en el archivo /etc/apache2/sites-available/default existen varias entradas <VirtualHost> por defecto las páginas que se mostrarán son las del primer host virtual de la lista. Es importante entonces que, si se dispone deun servidor 'oficial' con varios hosts virtuales esteservidor 'oficial' también aparezca como <VirtualHost> y esté en primer lugar.