

Tema 3. Implantación, configuración y administración de servidores web

Servidores web.....	3
Apache.....	3
Instalación.....	3
Comprobar el estado del servicio.....	4
Detener el servidor.....	5
Iniciar el servicio.....	5
Reiniciar el servidor.....	5
Más información.....	5
Configuración.....	6
Configuración principal.....	7
apache2.conf.....	7
ports.conf.....	7
sites-available/enabled.....	8
Cómo crear un hosting compartido (Creación de hosts virtuales).....	10
Instalación y configuración de módulos.....	14
Soporte para PHP.....	14
Soporte para Python (WSGI).....	15
Autenticación HTTP.....	16
Autenticación HTTP con .htaccess.....	19
Impedir que se liste un directorio.....	19
Limitar el ancho de banda.....	20
Reescritura de URLs.....	20
Personalizar las páginas de error.....	21
Ficheros log.....	21
Ficheros access.log.....	21
Ficheros error.log.....	22
Seguridad.....	23
Configurar SSL/TLS en Apache.....	23
Protección frente a atacantes.....	27
Despliegue de sitios web.....	27
Estadísticas web.....	27
Monitorización.....	29

Servidores web

Aunque son muchos los servidores web que podemos encontrar disponibles en Internet, a la hora de la verdad, entre casi la totalidad de todos los sitios web disponibles hoy en día Internet sólo se están utilizando unos 4 diferentes.

Existen un par de servidores web ([Apache](#) y [Microsoft IIS](#)) de uso general y muy extendido que se reparten casi la totalidad del mercado. Por otra parte está [nginx](#), relativamente reciente, y que ha cogido bastante popularidad en poco tiempo. También aparece alguna solución de Google debido a la popularidad de sus servicios. A partir de ahí, el resto de soluciones, quizás por ser más específicas para productos o tecnologías más concretas y menos extendidas, apenas significan nada en el reparto total. Quizás, junto con el resto de servidores, también puede destacar [lighttpd](#), puesto que actualmente se está hablando bastante de él por su bajo consumo de memoria.

En este tema nos centraremos exclusivamente en el servidor web *Apache* por ser el más utilizado actualmente (e históricamente) con bastante diferencia con respecto a los demás. Más adelante, centraremos algunas de las prácticas en la instalación y configuración de otros servidores web que hemos comentado anteriormente.

Apache



Instalación

Puesto que en nuestro caso trabajamos con *Debian Linux* no necesitamos ir al sitio web del [Proyecto Apache](#) para descargarnos el software. Simplemente, usando el gestor de paquetes, instalaremos el servidor web, su serie de utilidades y la documentación:

```
alumno@daw:~$ sudo apt-get install apache2 apache2-utils
```

Una vez instalado todo el software, podremos echar un vistazo a la carpeta `/etc/apache2`, que es donde se almacena toda la configuración del servidor:

```
alumno@daw:~$ ls /etc/apache2
apache2.conf  conf-enabled  magic          mods-enabled
sites-available
conf-available  envvars      mods-available  ports.conf
sites-enabled
```

La estructura de directorios queda de la siguiente manera:

```
/etc/apache2/
```

```
|-- apache2.conf
|   |-- ports.conf
|-- mods-enabled
|   |-- *.load
|   |-- *.conf
|-- conf-enabled
|   |-- *.conf
|-- sites-enabled
|   |-- *.conf
```

Y a continuación, echaremos un vistazo a cada uno de los ficheros/carpetas para ver cual es su cometido. Hay que tener en cuenta que la configuración de *Apache* está muy estructurada, por lo que es necesario saber en qué fichero/directorio corresponde configurar cada parte. Realmente es mucho más cómodo disponer de varios ficheros de configuración pequeños que de uno muy grande. Eso también hace que sea más modular a la hora de activar o desactivar según que características que iremos viendo más adelante.

- **apache2.conf**: El fichero de configuración principal de *Apache*, donde se pueden realizar cambios generales
- **envvars**: Contiene la configuración de las variables de entorno
- **ports.conf**: Contiene la configuración de los puertos donde *Apache* escucha
- **conf-available**: Contiene ficheros de configuración adicionales para diferentes aspectos de *Apache* o de aplicaciones web como *phpMyAdmin*
- **conf-enabled**: Contiene una serie de enlaces simbólicos a los ficheros de configuración adicionales para activarlos. Puede activarse o desactivarse con los comandos `a2enconf` o `a2disconf`
- **mods-available**: Contiene los módulos disponibles para usar con *Apache*
- **mods-enabled**: Contiene enlaces simbólicos a aquellos módulos de *Apache* que se encuentran activados en este momento
- **sites-available**: Contiene los ficheros de configuración de cada uno de los hosts virtuales configurados y disponibles (activos o no). Se crean utilizando los comandos `a2enmod` y `a2dismod` que más adelante explicaremos con más detalle
- **sites-enabled**: Contiene una serie de enlaces simbólicos a los ficheros de configuración cuyos hosts virtuales se encuentran activos en este momento. Se crean a través de los comandos `a2ensite` y `a2dissite` que más adelante explicaremos con más detalle

Apache es lo que se conoce como un demonio, gestionado en *Debian* a través del comando `service`, de forma que podemos gestionar el servidor en función de las opciones que pasemos.

Comprobar el estado del servicio

Si queremos saber cómo se encuentra nuestro servidor *Apache*, podemos ejecutar el siguiente comando y se mostrará un detalle de cuál es el estado del mismo. Resulta especialmente útil cuando, por alguna razón desconocida, éste está caído y no tenemos claro porqué. Con este comando podemos encontrar, normalmente, una descripción clara del problema.

```
alumno@daw:~$ sudo service apache2 status
● apache2.service - The Apache HTTP Server
```

```

Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor
preset: enabled)
Active: active (running) since Sun 2017-09-24 19:25:37 IST; 4h 54min
ago
Process: 17895 ExecStop=/usr/sbin/apachectl stop (code=exited,
status=0/SUCCESS)
Process: 17902 ExecStart=/usr/sbin/apachectl start (code=exited,
status=0/SUCCESS)
Main PID: 17906 (/usr/sbin/apach)
Tasks: 6 (limit: 4915)
Memory: 28.2M
CPU: 297ms
CGroup: /system.slice/apache2.service
├─17906 /usr/sbin/apache2 -k start
├─17907 /usr/sbin/apache2 -k start
├─17908 /usr/sbin/apache2 -k start
├─17909 /usr/sbin/apache2 -k start
├─17910 /usr/sbin/apache2 -k start
└─17911 /usr/sbin/apache2 -k start

```

```

Sep 24 19:25:37 zenbook systemd[1]: Starting The Apache HTTP Server...
Sep 24 19:25:37 zenbook apachectl[17902]: AH00558: apache2: Could not
reliably determine the server's fully
Sep 24 19:25:37 zenbook systemd[1]: Started The Apache HTTP Server.

```

Detener el servidor

Podemos detener el servidor, que permanecerá detenido hasta el siguiente reinicio o hasta que sea arrancado manualmente

```
alumno@daw:~$ sudo service apache2 stop
```

Iniciar el servicio

Aunque se inicia automáticamente en cada arranque del equipo, quizás lo hayamos parado y queramos iniciarlo de nuevo

```
alumno@daw:~$ sudo service apache2 start
```

Reiniciar el servidor

Se trata de parar y arrancar el servicio con un sólo comando. Muy útil si hemos realizado algún cambio en la configuración. Hay que tener en cuenta que *Apache* lee la configuración una vez en el arranque y no la vuelve a leer hasta la siguiente vez que inicia. Así, para hacer efectivo el más mínimo cambio tendremos que reiniciarlo con este comando

```
alumno@daw:~$ sudo service apache2 restart
```

Más información

También es posible algunos datos sobre su ejecución con el siguiente comando

```
alumno@daw:~$ apachectl status
```

```
Apache Server Status for localhost (via ::1)
```

```
Server Version: Apache/2.4.25 (Debian) mod_perl/2.0.10 Perl/v5.24.1
```

```
Server MPM: prefork
```

```
Server Built: 2017-07-18T18:37:33
```

```
-----
```

```
Current Time: Monday, 25-Sep-2017 00:24:09 IST
```

```
Restart Time: Monday, 25-Sep-2017 00:21:47 IST
```

```
Parent Server Config. Generation: 1
```

```
Parent Server MPM Generation: 0
```

```
Server uptime: 2 minutes 21 seconds
```

```
Server load: 2.08 3.37 2.45
```

```
Total accesses: 1 - Total Traffic: 1 kB
```

```
CPU Usage: u0 s0 cu0 cs0
```

```
.00709 requests/sec - 7 B/second - 1024 B/request
```

```
1 requests currently being processed, 6 idle workers
```

```
___W___.....  
.....  
.....
```

```
Scoreboard Key:
```

```
"_" Waiting for Connection, "S" Starting up, "R" Reading Request,  
"W" Sending Reply, "K" Keepalive (read), "D" DNS Lookup,  
"C" Closing connection, "L" Logging, "G" Gracefully finishing,  
"I" Idle cleanup of worker, "." Open slot with no current process
```

Y por último, si queremos conocer la versión del servidor que tenemos instalada, podemos hacerlo con el siguiente comando

```
alumno@daw:~$ apache2 -v
```

```
Server version: Apache/2.4.25 (Debian)
```

```
Server built: 2017-07-18T18:37:33
```

Configuración

Configuración principal

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

apache2.conf

`apache2.conf` es el fichero de configuración principal. Actualmente hay directivas generales de funcionamiento del servidor. Sólo se comentarán las más utilizadas puesto que algunas raras veces se modifican.

En versiones anteriores, este fichero, aparecían opciones interesantes como puertos donde escucha el servidor, configuración del sitio principal, de los módulos habilitados (o no) y los hosts virtuales. Actualmente todas esas configuraciones han sido trasladadas a ficheros de configuración específicos, por lo que en el fichero general quedan pocas opciones interesantes que habitualmente se modifiquen.

```
. . .  
<Directory /var/www/>  
    Options Indexes FollowSymlinks  
    AllowOverride None  
    Require all granted  
</Directory>  
. . .
```

```
. . .  
AccessFileName .htaccess  
. . .
```

```
. . .  
Include ports.conf  
. . .
```

KeeAlive On/Off

```
. . .  
KeepAlive On  
. . .
```

ports.conf

En este fichero de configuración se establecen los puertos en los que escucha *Apache*. Puesto que es algo muy estándar no es habitual modificar el fichero, quizás para configurar el soporte para *HTTPS* como haremos más adelante.

```
# If you just change the port or add more ports here, you will likely
also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf
```

```
Listen 80
```

```
<IfModule ssl_module>
    Listen 443
</IfModule>
```

```
<IfModule mod_gnutls.c>
    Listen 443
</IfModule>
```

sites-available/enabled

En estas dos carpetas se almacenan los ficheros de configuración de los hosts virtuales. En *sites-available* se crean los ficheros de configuración (estén o no activos) y aquellos que queremos que estén activos se vinculan con la carpeta *sites-enabled* para que se tenga en cuenta su configuración, ya que *Apache* sólo carga los que se encuentran en esta última carpeta.

```
alumno@daw:~$ ls /etc/apache2/sites-available
000-default.conf default-ssl.conf
```

En este caso sólo nos encontramos con la configuración del sitio principal y tal como *Apache* lo prepara con la instalación por defecto. Más adelante, veremos cómo configurar el sitio principal y el resto de hosts virtuales cuando lo necesitemos.

000-default.conf

```
<VirtualHost *:80>
    # The ServerName directive sets the request scheme, hostname and
    port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header
    to
    # match this virtual host. For the default virtual host (this file)
    this
    # value is not decisive as it is used as a last resort host
    regardless.
    # However, you must set it for any further virtual host explicitly.
    #ServerName www.example.com
```



```

ServerAdmin webmaster@localhost
DocumentRoot /var/www/html

# Available loglevels: trace8, ..., trace1, debug, info, notice,
warn,
# error, crit, alert, emerg.
# It is also possible to configure the loglevel for particular
# modules, e.g.
#LogLevel info ssl:warn

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

# For most configuration files from conf-available/, which are
# enabled or disabled at a global level, it is possible to
# include a line for only one particular virtual host. For example
the
# following line enables the CGI configuration for this host only
# after it has been globally disabled with "a2disconf".
#Include conf-available/serve-cgi-bin.conf
</VirtualHost>

```

En el anterior fichero de configuración del sitio principal se pueden ver una serie de directivas muy importantes:

- **ServerName:** Aunque se encuentra comentada, se utiliza para definir el dominio de Internet para el que se configura este sitio. Puesto que ahora estamos realizando pruebas en nuestro propio equipo no tendría sentido habilitarlo, puesto que no estamos dando servicio a ningún dominio real. Más adelante veremos cómo *engañar* al equipo y configuraremos al completo un sitio de *Apache* para ver cómo funciona totalmente esta directiva
- **ServerAdmin:** Sirve para definir la dirección de correo del administrador del servidor web. En caso de error se mostrará al usuario y será a quién deba dirigirse para notificar cualquier incidencia
- **DocumentRoot:** Es muy importante puesto que define la ruta del equipo donde se encuentran todos los ficheros que se están sirviendo en cada momento a través del sitio web que define este fichero. Todo lo que haya en este directorio y por debajo de él está, en principio, siendo ofrecido a través de la web.
- **ErrorLog:** Define en qué fichero (como texto) se van a ir registrando todos los errores que se produzcan
- **CustomLog:** Define un fichero donde se registra toda la información posible (dependiendo de cómo se configure) sobre las visitas que llegan al sitio web y sus usuarios. Es la información que, más adelante, nos podrá permitir sacar las estadísticas de visitas de nuestra página web

Y aquí podemos ver como la configuración del sitio principal tiene un vínculo en `sites-enabled` para que *Apache* cargue la configuración y el sitio esté operativo.

```
alumno@daw:~$ ls -la /etc/apache2/sites-enabled
total 8
drwxr-xr-x 2 root root 4096 Nov 19 2015 .
drwxr-xr-x 8 root root 4096 Sep 22 10:26 ..
lrwxrwxrwx 1 root root 35 Nov 19 2015 000-default.conf ->
../sites-available/000-default.conf
```

Cómo crear un hosting compartido (Creación de hosts virtuales)

El hosting compartido consiste en el mantenimiento de diferentes sitios web (independientes entre ellos) en el mismo equipo, compartiendo recursos. Es la forma más económica puesto que, al poder compartir los recursos, es posible crear y mantener un gran número de éstos en el mismo equipo.

```
alumno@daw:~$ sudo vim /etc/apache2/sites-available/misitio.com.conf
```

misitio.com.conf

```
<VirtualHost *:80>
    ServerAdmin webmaster@misitio.com
    DocumentRoot "/var/www/html/misitio.com"
    ServerName misitio.com
    ServerAlias www.misitio.com
    ErrorLog "misitio.com-error.log"
    CustomLog "misitio.com-access.log" combined
</VirtualHost>
```

Cuando hayamos terminado de configurar el nuevo host virtual, podemos activarlo utilizando el comando `a2ensite` (**apache2 enable site**). Automáticamente se creará un enlace simbólico con la configuración del sitio de `sites-available` en `sites-enabled`:

```
alumno@daw:~$ sudo a2ensite misitio.com
Enabling site misitio.com.
To activate the new configuration, you need to run:
    systemctl reload apache2
```

Básicamente podríamos haber hecho lo mismo con el siguiente comando:

```
alumno@daw:~$ sudo ln -s ../sites-available/misitio.com.conf
/etc/apache2/sites-enabled/misitio.com.conf
```

En cualquier caso tendremos que reiniciar el servicio de *Apache* para que se active la nueva configuración

```
alumno@daw:~$ service apache2 restart
```

También podemos desactivar un host virtual con el comando `a2dissite` (**apache2 disable site**). El sitio permanecerá desactivado hasta que lo volvamos a activar con `a2ensite`

```
alumno@daw:~$ a2dissite misitio.com
Site misitio.com disabled.
To activate the new configuration, you need to run:
systemctl reload apache2
```

En estos casos, donde empezamos a tener numerosos ficheros de configuración, es donde resulta útil el comando `service apache2 status` ya que, en caso de que tengamos algún error que impida arrancar el servicio, nos dará la información necesaria para encontrarlo. En el siguiente ejemplo he escrito mal a idea un fichero de configuración de un sitio virtual. Al arrancarlo con `a2ensite` me ha dado un error y con `service apache2 status` visualizo toda la información y puedo ver cuál es el motivo del error.

```
alumno@daw:~$ service apache2 status
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor
   preset: enabled)
   Active: active (running) (Result: exit-code) since Mon 2017-09-25
19:43:40 IST; 38s ago
     Process: 31465 ExecStop=/usr/sbin/apachectl stop (code=exited,
status=0/SUCCESS)
     Process: 31607 ExecReload=/usr/sbin/apachectl graceful (code=exited,
status=1/FAILURE)
     Process: 31472 ExecStart=/usr/sbin/apachectl start (code=exited,
status=0/SUCCESS)
   Main PID: 31476 (/usr/sbin/apach)
      Tasks: 6 (limit: 4915)
     Memory: 28.0M
        CPU: 354ms
    CGroup: /system.slice/apache2.service
            └─31476 /usr/sbin/apache2 -k start
            └─31477 /usr/sbin/apache2 -k start
            └─31478 /usr/sbin/apache2 -k start
            └─31479 /usr/sbin/apache2 -k start
            └─31480 /usr/sbin/apache2 -k start
            └─31481 /usr/sbin/apache2 -k start
```

```
Sep 25 19:43:40 zenbook systemd[1]: Starting The Apache HTTP Server...
Sep 25 19:43:40 zenbook apachectl[31472]: AH00558: apache2: Could not
reliably determine the server's fully
Sep 25 19:43:40 zenbook systemd[1]: Started The Apache HTTP Server.
Sep 25 19:44:12 zenbook systemd[1]: Reloading The Apache HTTP Server.
Sep 25 19:44:12 zenbook apachectl[31607]: AH00526: Syntax error on line
1 of /etc/apache2/sites-enabled/misi
Sep 25 19:44:12 zenbook apachectl[31607]: <VirtualHost> directive
missing closing '>'
Sep 25 19:44:12 zenbook apachectl[31607]: Action 'graceful' failed.
Sep 25 19:44:12 zenbook apachectl[31607]: The Apache error log may have
more information.
Sep 25 19:44:12 zenbook systemd[1]: apache2.service: Control process
exited, code=exited status=1
```

```
Sep 25 19:44:12 zenbook systemd[1]: Reload failed for The Apache HTTP Server.
```

Si finalmente comprobamos que la configuración es correcta, nuestro host virtual estará funcionando correctamente. Si suponemos que nuestro servidor está corriendo en Internet y que además ya somos dueños del dominio (y éste apunta a nuestro equipo), los usuarios ya podrán empezar a visitar nuestra nueva web **misitio.com**.

En nuestro caso, en clase, vamos a decirle a nuestro ordenador que el dominio *misitio.com* apunte a nuestro propio equipo, así podremos comprobar que todo funciona. Para ello editaremos el fichero `/etc/hosts` que funciona como DNS local

```
alumno@daw:~$ sudo vim /etc/hosts
```

Y añadimos la IP que queremos asignar al dominio *misitio.com* (en nuestro caso nuestro propio equipo, 127.0.0.1)

```
127.0.0.1    localhost
127.0.1.1    daw
127.0.0.1    misitio.com

# The following lines are desirable for IPv6 capable hosts
::1          localhost ip6-localhost ip6-loopback
ff02::1      ip6-allnodes
ff02::2      ip6-allrouters
```

Simplemente haciendo ping contra el dominio podemos ver que responde nuestro propio equipo, por lo que todo funciona correctamente

```
alumno@daw:~$ ping misitio.com
PING misitio.com (127.0.0.1) 56(84) bytes of data:
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.026 ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0.029 ms
64 bytes from localhost (127.0.0.1): icmp_seq=3 ttl=64 time=0.030 ms
```

Preparamos ahora una web de bienvenida para el nuevo sitio web:

index.html

```
<!DOCTYPE>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Bienvenido a misitio.com</title>
</head>
<body>
  <p>Estás en misitio.com</p>
</body>
</html>
```

Y así ahora que lo visitamos podremos ver que la página que visitamos es diferente en función de si vamos al host virtual `misitio.com`...

```
alumno@daw:~$ links http://misitio.com
```

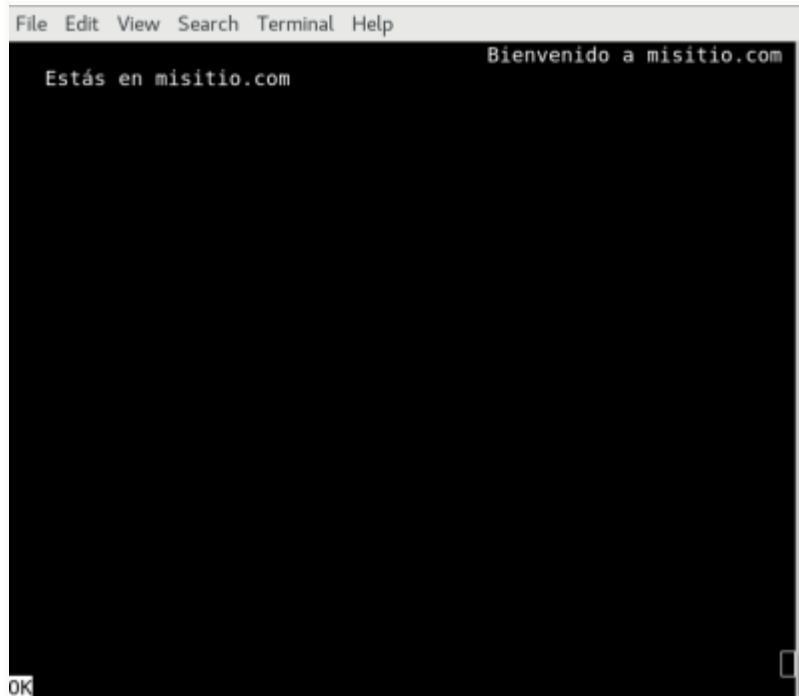


Figure 2: misitio.com

... o al sitio principal (con `http://localhost`)

```
alumno@daw:~$ links http://localhost
```

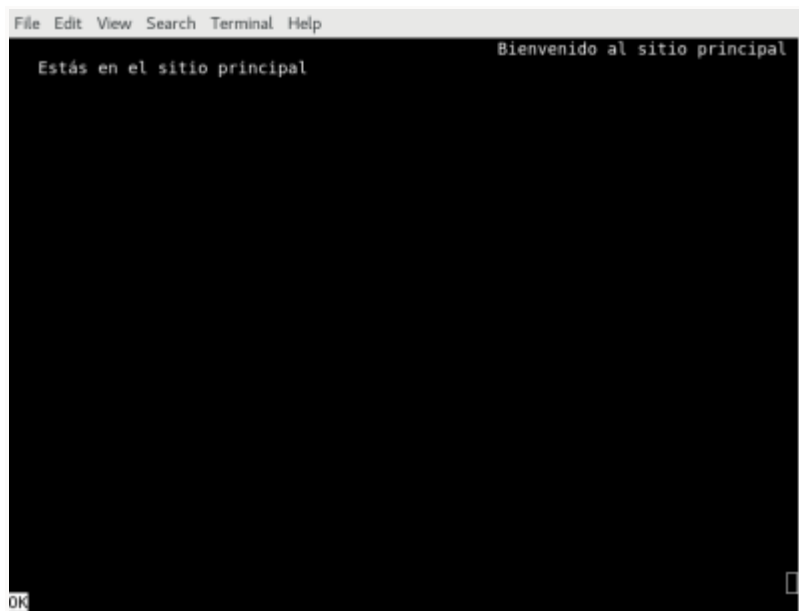


Figure 3: Sitio principal

Y como curiosidad, podemos ver como las visitas realizadas al host virtual se van registrado en su correspondiente fichero `log`, tal y como habíamos configurado antes:

```
alumno@daw:~$ tail /var/log/apache2/misitio.com-access.log
```

```
127.0.0.1 - - [22/Sep/2017:11:17:36 +0100] "GET / HTTP/1.1" 200 449 "-"
"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/60.0.3112.90 Safari/537.36 OPR/47.0.2631.80"
127.0.0.1 - - [22/Sep/2017:11:17:51 +0100] "GET / HTTP/1.1" 200 464 "-"
"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/60.0.3112.90 Safari/537.36 OPR/47.0.2631.80"
127.0.0.1 - - [22/Sep/2017:11:18:39 +0100] "GET / HTTP/1.1" 200 485 "-"
"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/60.0.3112.90 Safari/537.36 OPR/47.0.2631.80"
127.0.0.1 - - [22/Sep/2017:11:21:54 +0100] "GET / HTTP/1.1" 200 485 "-"
"Links (2.14; Linux 4.9.0-3-amd64 x86_64; GNU C 6.3; text)"
. . .
```

Instalación y configuración de módulos

En esta parte veremos cómo extender la funcionalidad del servidor web *Apache* añadiendo y configurando algunos módulos que, aunque se deben de instalar manualmente a parte, empiezan a ser parte indispensable de este servidor:

- Soporte para lenguaje PHP
- Soporte para Python a través de WSGI
- Autenticación HTTP
- Control del ancho de banda
- Reescritura de URLs
- Y también, aunque en el apartado de seguridad, como configurar *Apache* para conexiones seguras HTTPS

Soporte para PHP

Para instalar el soporte para PHP en *Apache*, lo primero que tenemos que hacer es instalar los paquetes para darle soporte. Podemos, por ejemplo, instalar los de PHP5 y los de la recién liberada versión 7

```
alumno@daw:~$ sudo apt-get install php5 php7.0
```

Hay que tener en cuenta que la instalación del paquete de PHP 5 (php5) tiene como dependencia el paquete `libapache-mod-php5` que instala el módulo de apache para dar soporte a esta tecnología. El paquete para la versión 7 de PHP (php7.0) trae también como dependencia el módulo correspondiente para esta versión (`libapache-mod-php7.0`)

Si además queremos añadir soporte para Bases de Datos, tendremos que instalar a parte el SGBD que nos interese. En el caso de PHP el más utilizado es MySQL, por lo que instalaremos éste:

```
alumno@daw:~$ sudo apt-get install mysql-server
```

Y también el soporte del lenguaje PHP para conectarse con *MySQL*:

```
alumno@daw:~$ sudo apt-get install php5-mysql php7.0-mysql
```

Una vez instalado el soporte para PHP, la mejor forma de comprobar que todo funciona correctamente es invocar a la función `phpinfo()`, que nos proporciona información muy detallada sobre su instalación. Así, podemos preparar un brevísimo script PHP con las siguientes líneas y copiarlo en el directorio del sitio principal (`/var/www/html`)

```
<?php
phpinfo();
```

Y al cargar dicho script obtendremos una web con varias tablas donde se detalla en profundidad toda la configuración de PHP para nuestro servidor *Apache*:


PHP Version 5.6.30-0+deb8u1 	
System	Linux zenbook 4.9.0-3-amd64 #1 SMP Debian 4.9.30-2+deb9u3 (2017-08-06) x86_64
Build Date	Feb 8 2017 08:50:48
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/05-opcache.ini, /etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-json.ini, /etc/php5/apache2/conf.d/20-mcrypt.ini, /etc/php5/apache2/conf.d/20-mysql.ini, /etc/php5/apache2/conf.d/20-mysqli.ini, /etc/php5/apache2/conf.d/20-pdo_mysql.ini, /etc/php5/apache2/conf.d/20-readline.ini
PHP API	20131106
PHP Extension	20131226
Zend Extension	220131226
Zend Extension Build	API20131226,NTS
PHP Extension Build	API20131226,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	enabled
Registered PHP Streams	https, ftps, compress.zlib, compress.bzip2, php, file, glob, data, http, ftp, phar, zip
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, bzip2.*, convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, mcrypt.*, mdecrypt.*

Figure 4: Información sobre la instalación de PHP

Resulta especialmente útil en el caso de que queramos extender extensiones al lenguaje, puesto que es la manera más efectiva de saber si dichas librerías se encuentran ya instaladas y activas. Algunas de las librerías más utilizadas son las de gráficos (*gd* y *imagick*) y una librería para cachear y mejorar así el rendimiento (*apcu*):

```
alumno@daw:~$ sudo apt-get install php-gd php-imagick php-apcu
```

Además, en el caso de que hayamos optado por instalar PHP junto con MySQL, puede resultar de mucha utilidad instalar un gestor para el SGBD, en este caso como aplicación web. Se trata de *phpMyAdmin*, conocida aplicación web para la gestión de Bases de Datos *MySQL*.

```
alumno@daw:~$ sudo apt-get install phpmyadmin
```

Soporte para Python (WSGI)

```
alumno@daw:~$ sudo apt-get install libapache2-mod-wsgi-py  
libapache2-mod-wsgi-py3
```

```
WSGIScriptAlias / /path/to/mysite.com/mysite/wsgi.py  
WSGIProxyHome /path/to/venv  
WSGIProxyPath /path/to/mysite.com
```

```
<Directory /path/to/mysite.com/mysite>  
<Files wsgi.py>  
Require all granted  
</Files>  
</Directory>
```

Autenticación HTTP

La autenticación HTTP es un método por el cual podemos proteger carpetas dentro de nuestro servidor web (en este caso con *Apache*).

Lo primero que tendremos que tener instalado es el paquete `apache2-utils`

```
alumno@daw:~$ sudo apt-get install apache2-utils
```

Que como podemos observar en la descripción detallada del paquete, contiene una serie de utilidades para, entre otras cosas, manipular archivos de autenticación (con el comando `htpasswd`) que serán los ficheros a través de los cuales configuraremos el acceso (mediante usuario/contraseña) a esas carpetas protegidas dentro de nuestro servidor web.

```
alumno@daw:~$ apt-cache show apache2-utils  
Package: apache2-utils  
Source: apache2  
Version: 2.4.25-3+deb9u2  
Installed-Size: 377  
Maintainer: Debian Apache Maintainers <debian-apache@lists.debian.org>  
Architecture: amd64  
Depends: libapr1 (>= 1.4.8-2~), libaprutil1 (>= 1.5.0), libc6 (>= 2.14), libssl1.0.2 (>= 1.0.2d)  
Description-en: Apache HTTP Server (utility programs for web servers)  
Provides some add-on programs useful for any web server. These  
include:  
- ab (Apache benchmark tool)  
- fcgidstarter (Start a FastCGI program)  
- logresolve (Resolve IP addresses to hostnames in logfiles)  
- htpasswd (Manipulate basic authentication files)  
- htdigest (Manipulate digest authentication files)  
- htdbm (Manipulate basic authentication files in DBM format, using  
APR)  
- htcacheclean (Clean up the disk cache)
```



```
- rotatelog (Periodically stop writing to a logfile and open a new one)
- split-logfile (Split a single log including multiple vhosts)
- checkgid (Checks whether the caller can setgid to the specified group)
- check_forensic (Extract mod_log_forensic output from Apache log files)
- httxt2dbm (Generate dbm files for use with RewriteMap)
Description-md5: f1e2440381fa90571f125990da6a52fc
Homepage: http://httpd.apache.org/
Multi-Arch: foreign
Section: httpd
Priority: optional
Filename:
pool/updates/main/a/apache2/apache2-utils_2.4.25-3+deb9u2_amd64.deb
Size: 216528
MD5sum: b509377ca37975d449a1595bcd416327
SHA1: 4eeb5e76e8ac1ed12c6fdcb9d2f0c0468a54b50e
SHA256:
9562aacafb42cb6b08acbcc479487e9556fadcb180e7b6743696f1d9b0a6d5b4
```

Una vez instalado `apache2-utils`, estaremos en disposición de ejecutar el comando `htpasswd` que se utiliza tanto para crear el fichero por primera vez y añadir un usuario a la zona protegida que vamos a crear, como para añadir otros usuarios más adelante.

Creamos primero un usuario `santi` con su contraseña y almacenamos el fichero en la ruta de configuración de nuestro servidor web. Como el fichero todavía no existe utilizamos la opción `-c`

```
alumno@daw:~$ sudo htpasswd -c /etc/apache2/.htpasswd santi
New password:
Re-type new password:
Adding password for user santi
```

Y añadimos un segundo usuario. En este caso el fichero ya existe y no es necesario especificar ninguna opción adicional al ejecutarlo.

```
alumno@daw:~$ sudo htpasswd /etc/apache2/.htpasswd otro_usuario
New password:
Re-type new password:
Adding password for user otro_usuario
```

Si echamos un vistazo al fichero, veremos como quedan almacenados usuario / contraseña, y ésta última cifrada.

```
alumno@daw:~$ cat /etc/apache2/.htpasswd
santi:$apr1$mDMINEK5$34dXLMYERhHz7QEmhtK.x/
otro_usuario:$apr1$tYxVwhYM$8guRCfIs5aIJovO8YazTL/
```

A continuación, asignamos la propiedad del fichero y el grupo al usuario y grupo sobre el que se ejecuta *Apache* en nuestro Linux, `www-data`

```
alumno@daw:~$ chown www-data:www-data /etc/apache2/.htpasswd
```

Puesto que teníamos ya creados algunos hosts virtuales, vamos a seleccionar uno de ellos para proteger dentro del mismo una de las carpetas. En este caso teníamos el sitio `misitio.com` ya creado como aparece aquí debajo:

```
<VirtualHost *:80>
    ServerAdmin webmaster@misitio.com
    DocumentRoot "/var/www/html/misitio.com"
    ServerName misitio.com
    ServerAlias www.misitio.com
    ErrorLog "misitio.com-error.log"
    CustomLog "misitio.com-access.log" combined
</VirtualHost>
```

Vamos a añadir la directiva `Directory` como aparece a continuación para proteger un directorio al que llamaremos `usuarios` donde suponemos que hay cierta información a la que no todo el mundo debe acceder. Configuraremos dicha zona para indicar que es una zona restringida y que sólo los usuarios que están especificados en ese fichero pueden entrar

```
<VirtualHost *:80>
    ServerAdmin webmaster@misitio.com
    DocumentRoot "/var/www/html/misitio.com"
    ServerName misitio.com
    ServerAlias www.misitio.com
    ErrorLog "logs/misitio.com-error.log"
    CustomLog "logs/misitio.com-access.log" combined

    <Directory "/var/www/html/misitio.com/usuarios">
        AuthType Basic
        AuthName "Acceso Restringido a Usuarios"
        AuthUserFile /etc/apache2/.htpasswd
        Require valid-user
    </Directory>
</VirtualHost>
```

Donde:

- **AuthType Basic:** Define el tipo de autenticación
- **AuthName:** Define el mensaje que se mostrará al usuario cuando se le solicite el usuario y contraseña para acceder
- **AuthUserFile:** Se indica la ruta al fichero que hemos creado (tal y como se explica más arriba) con los usuarios/contraseña que tienen permitido el acceso
- **Require:** Indica que sólo se podrá acceder con un usuario válido. Si sólo queremos que se pueda acceder con un único usuario podemos indicar `Require user santi` si sólo queremos que el usuario `santi` pueda acceder a esa carpeta.

Para terminar, reiniciamos el servicio de Apache:

```
alumno@daw:~$ sudo service apache2 restart
```

Ahora podemos probar a acceder a la URL <http://misitio.com/usuarios> y veremos como aparece una ventana al estilo *popup* solicitando que nos identifiquemos mediante un usuario y una contraseña.

Autenticación HTTP con .htaccess

También es posible, y quizás más cómodo, activar y configurar la autenticación *HTTP* creando un fichero *.htaccess* de forma que podamos almacenarlo dentro de la carpeta del sitio que queremos proteger.

Antes de nada tenemos que modificar la configuración general de Apache para permitir que las propiedades sobre el directorio raíz puedan ser modificadas. Cambiaremos el valor `AllowOverride All` para permitir dichas modificaciones. Así, la política sobre dicho directorio podrá ser diferente a la establecida en la carpeta si así se define con algún fichero *.htaccess*

[/etc/apache2/apache2.conf](#)

```
. . .
<Directory /var/www/>
    Options Indexes FollowSymLinks
    AllowOverride All
    Require all granted
</Directory>
. . .
```

A continuación, sobre la carpeta que queremos proteger con contraseña, creamos un fichero *.htaccess* con el siguiente contenido:

[/var/www/html/misitio.com/usuarios/.htaccess](#)

```
AuthType Basic
AuthName "Acceso Restringido a Usuarios"
AuthUserFile /etc/apache2/.htpasswd
Require valid-user
```

Y, para terminar, tendremos que reiniciar el servicio de Apache:

```
alumno@daw:~$ sudo service apache2 restart
```

Impedir que se liste un directorio

Una vez que ya hemos modificado la configuración de Apache para que permita que las propiedades sobre el directorio raíz y sus subdirectorios puedan ser modificadas (ver punto anterior), podemos también impedir, por ejemplo, que los ficheros de un directorio puedan ser

listados por cualquier visitante. Basta con crear un fichero `.htaccess` y añadir la siguiente línea que indica que no puede listar el directorio.

`Options -Indexes`

Limitar el ancho de banda

Otro de los módulos de *Apache*, (`bw`, `bandwidth`), permite controlar el ancho de banda de los usuarios que se conectan a nuestro sitio web en función de numerosos parámetros. En este apartado veremos como instalarlo, activarlo y configurar un par de ejemplos:

Para instalarlo, como siempre, con la herramienta `apt-get`:

```
alumno@daw:$ sudo apt-get install libapache2-mod-bw
```

Y ahora, tomando como ejemplo el sitio virtual creado en anteriores apartados, vamos a configurarlo para controlar el ancho de banda en todo el sitio, el máximo, el mínimo y el número de conexiones simultáneas por conexión (por usuario)

```
<VirtualHost *:80>
    ServerAdmin webmaster@misitio.com
    DocumentRoot /var/www/html/misitio.com
    ServerName misitio.com
    ServerAlias www.misitio.com
    ErrorLog ${APACHE_LOG_DIR}/misitio.com-error.log
    CustomLog ${APACHE_LOG_DIR}/misitio.com-access.log combined

    BandwidthModule On
    ForceBandWidthModule On
    Bandwidth all 64
    MinBandwidth all -1
    MaxConnection all 2
</VirtualHost>
```

También es posible, por ejemplo, limitar la velocidad en un determinado directorio para determinados ficheros. Por ejemplo, si almacenamos en un directorio específico los ficheros más grandes (videos por ejemplo) y queremos limitar la velocidad sólo en el caso de que la gente se baje esos ficheros

```
. . .
<Directory "/var/www/example1.com/public_html/media">
    LargeFileLimit * 10240 102400
</Directory>
. . .
```

En [la documentación del módulo](#) se pueden encontrar otros tantos ejemplos para limitar el ancho de banda siguiendo diferentes criterios y parámetros.

Reescritura de URLs

La reescritura de URLs es un recurso muy útil y potente tanto para la administración de un site como para potenciar el posicionamiento de este. `mod_rewrite` se usa más comúnmente para transformar URL feas y crípticas en lo que se conoce como "URL amigables" o "URL limpias". Para gestionar correctamente estas reescrituras es muy útil saber usar las expresiones regulares, que caen fuera del objetivo del Módulo de Despliegue de Aplicaciones Web, sin embargo me parece acertado incluir un manual de cómo se puede gestionar este apartado https://code.tutsplus.com/es/una-guia-detallada-de-mod_rewrite-para-apache--net-6708t

```
alumno@daw:~$ sudo a2enmode rewrite
```

```
<VirtualHost *:80>
    . . .
    RewriteEngine On
    Options FollowSymLinks
</VirtualHost>

. . .
    RewriteRule "^/antiguo_index\.php$" "index.html" [R]
. . .
```

Personalizar las páginas de error

```
<VirtualHost *:80>
    ServerAdmin webmaster@misitio.com
    DocumentRoot "/var/www/html/misitio.com"
    ServerName misitio.com
    ServerAlias www.misitio.com

    ErrorLog "misitio.com-error.log"
    CustomLog "misitio.com-access.log" combined

    ErrorDocument 404 /error_404.html
    ErrorDocument 500 /error_500.html
</VirtualHost>
```

Ficheros log

Ficheros access.log

```
192.168.1.2 - - [01/Nov/2017:22:29:23 +0100] "GET /formulario2.html
HTTP/1.1" 200 721 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100
Safari/537.36 OPR/48.0.2685.35"
192.168.1.2 - - [01/Nov/2017:22:29:43 +0100] "GET /formulario2.html
HTTP/1.1" 200 753 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5)
```

```

AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100
Safari/537.36 OPR/48.0.2685.35"
192.168.1.2 - - [01/Nov/2017:22:29:43 +0100] "GET /formulario2.html
HTTP/1.1" 200 752 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100
Safari/537.36 OPR/48.0.2685.35"
192.168.1.2 - - [01/Nov/2017:22:29:56 +0100] "GET /formulario2.html
HTTP/1.1" 200 753 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100
Safari/537.36 OPR/48.0.2685.35"
192.168.1.2 - - [01/Nov/2017:22:30:20 +0100] "GET /formulario2.html
HTTP/1.1" 200 762 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100
Safari/537.36 OPR/48.0.2685.35"
192.168.1.2 - - [01/Nov/2017:23:24:34 +0100] "GET /test HTTP/1.1" 301
570 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100
Safari/537.36 OPR/48.0.2685.35"
192.168.1.2 - - [01/Nov/2017:23:24:34 +0100] "GET /test/ HTTP/1.1" 200
652 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100
Safari/537.36 OPR/48.0.2685.35"
192.168.1.2 - - [01/Nov/2017:23:24:34 +0100] "GET /icons/blank.gif
HTTP/1.1" 200 431 "http://192.168.1.6/test/" "Mozilla/5.0 (Macintosh;
Intel Mac OS X 10_9_5) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/61.0.3163.100 Safari/537.36 OPR/48.0.2685.35"
192.168.1.2 - - [01/Nov/2017:23:24:34 +0100] "GET /icons/back.gif
HTTP/1.1" 200 499 "http://192.168.1.6/test/" "Mozilla/5.0 (Macintosh;
Intel Mac OS X 10_9_5) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/61.0.3163.100 Safari/537.36 OPR/48.0.2685.35"
192.168.1.2 - - [01/Nov/2017:23:27:12 +0100] "GET /test/ HTTP/1.1" 200
653 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100
Safari/537.36 OPR/48.0.2685.35"

```

Ficheros error.log

```

[Wed Nov 01 20:17:21.357016 2017] [:error] [pid 1346] [client
192.168.1.2:60418] PHP Notice: Undefined index: email in
/var/www/html/alta_usuario.php on line 3
[Wed Nov 01 23:27:19.185428 2017] [mpm_prefork:notice] [pid 421]
AH00169: caught SIGTERM, shutting down
[Wed Nov 01 23:27:19.269594 2017] [mpm_prefork:notice] [pid 2396]
AH00163: Apache/2.4.25 (Debian) configured -- resuming normal
operations
[Wed Nov 01 23:27:19.269661 2017] [core:notice] [pid 2396] AH00094:
Command line: '/usr/sbin/apache2'
[Wed Nov 01 23:30:18.774503 2017] [mpm_prefork:notice] [pid 2396]
AH00169: caught SIGTERM, shutting down

```

```
[Wed Nov 01 23:30:18.861176 2017] [mpm_prefork:notice] [pid 2468]
AH00163: Apache/2.4.25 (Debian) configured -- resuming normal
operations
[Wed Nov 01 23:30:18.861250 2017] [core:notice] [pid 2468] AH00094:
Command line: '/usr/sbin/apache2'
[Wed Nov 01 23:30:20.733505 2017] [autoindex:error] [pid 2488] [client
192.168.1.2:50755] AH01276: Cannot serve directory /var/www/html/test/:
No matching DirectoryIndex
(index.html,index.cgi,index.pl,index.php,index.xhtml,index.htm) found,
and server-generated directory index forbidden by Options directive
[Wed Nov 01 23:30:21.884441 2017] [autoindex:error] [pid 2488] [client
192.168.1.2:50755] AH01276: Cannot serve directory /var/www/html/test/:
No matching DirectoryIndex
(index.html,index.cgi,index.pl,index.php,index.xhtml,index.htm) found,
and server-generated directory index forbidden by Options directive
[Wed Nov 01 23:30:22.444693 2017] [autoindex:error] [pid 2488] [client
192.168.1.2:50755] AH01276: Cannot serve directory /var/www/html/test/:
No matching DirectoryIndex
(index.html,index.cgi,index.pl,index.php,index.xhtml,index.htm) found,
and server-generated directory index forbidden by Options directive
```

Seguridad

Configurar SSL/TLS en Apache

SSL (Secure Socket Layer) es un protocolo de seguridad que nació con el objetivo de cifrar las comunicaciones entre los servidores web y los navegadores de forma que, si se interceptaba la conexión, nunca se pudiera desvelar el contenido de la misma. Con el paso del tiempo se han ido encontrando diversas vulnerabilidades críticas que han hecho que la recomendación sea usar un nuevo protocolo llamado TLS (Transport Secure Layer).

El primer paso para configurar SSL en *Apache* será crear el certificado y la clave, que se quedarán almacenados en `/etc/apache2/certs`. Para eso primero creamos la carpeta y luego el certificado y su correspondiente clave.

Hay que tener en cuenta que estamos creando un certificado autofirmado. Este tipo de certificados sólo se deben usar con el propósito de enseñar o hacer una demostración puesto que en la práctica no son válidos. El navegador no confiará en él porque somos nosotros quienes lo hemos firmado. Los certificados, para que sean válidos, deben ser validados por una entidad certificadora. Más adelante veremos como el navegador avisa al usuario de que el certificado no es fiable (aunque siempre le mostrará la opción de continuar a pesar de ello)

```
alumno@daw:~$ sudo mkdir /etc/apache2/certs
alumno@daw:~$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048
                    -keyout /etc/apache2/certs/apache2.key
                    -out /etc/apache2/certs/apache2.crt
Generating a 2048 bit RSA private key
.....+++
.....
writing new private key to '/etc/apache2/certs/apache2.key'
```

```
-----
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a
DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:Malaga
Locality Name (eg, city) []:Malaga
Organization Name (eg, company) [Internet Widgits Pty Ltd]:daw
Organizational Unit Name (eg, section) []:web
Common Name (e.g. server FQDN or YOUR name) []:daw.com
Email Address []:juan@daw.com
```

Ahora ya tenemos el certificado y su clave. Activamos entonces el módulo SSL de Apache:

```
alumno@daw:~$ sudo a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and
create self-signed certificates.
To activate the new configuration, you need to run:
    systemctl restart apache2
```

Y ahora configuramos un host virtual para que soporte conexión segura. En este caso he seleccionado el mismo sitio que antes hemos configurado como ejemplo de host virtual y he realizado los cambios necesarios para que ahora soporte conexión segura (HTTPS). Básicamente es cambiar el puerto donde escucha (ahora es 443, donde escuchan las conexiones seguras), activar el soporte para SSL e indicar donde están el certificado y la clave.

```
<VirtualHost *:443>
    ServerAdmin webmaster@misitio.com
    DocumentRoot /var/www/html/misitio.com
    ServerName misitio.com
    ServerAlias www.misitio.com
    ErrorLog ${APACHE_LOG_DIR}/misitio.com-error.log
    CustomLog ${APACHE_LOG_DIR}/misitio.com-access.log combined

    SSLEngine On
    SSLCertificateFile /etc/apache2/certs/apache2.crt
    SSLCertificateKeyFile /etc/apache2/certs/apache2.key
```



```
SSLProtocol All -SSLv3
</VirtualHost>
```

Para probarlo, abrimos cualquier navegador e introducimos la dirección <https://misitio.com>. Automáticamente, al usar HTTPS, se conectará al puerto 443 y el navegador intentará comprobar la validez del certificado. Si fuera válido se establecería la conexión segura y veríamos como el símbolo del candado en el navegador nos indica que la web es segura y podremos navegar sin problemas.

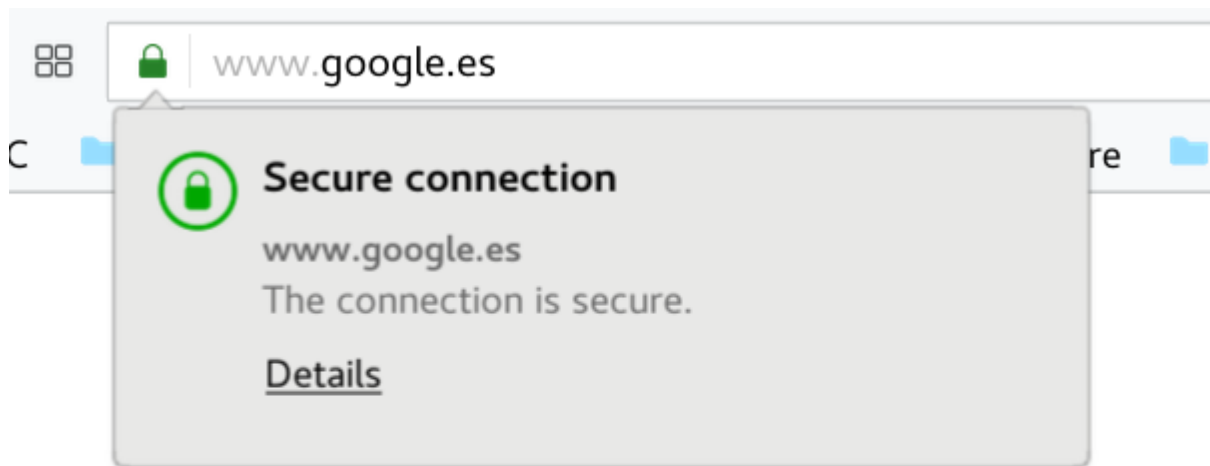


Figure 5: Conexión segura con Google

En nuestro caso hemos hecho todos los pasos correctamente pero, como hemos comentado anteriormente, nuestro certificado es autofirmado. El navegador mostrará el correspondiente error diciendo que quién firma el certificado es desconocido y avisará al usuario, quién podrá decidir no visitar la web si no fía o bien añadir a dicho sitio como excepción en el caso de que esté completamente seguro de que, a pesar del certificado, el sitio es totalmente fiable.

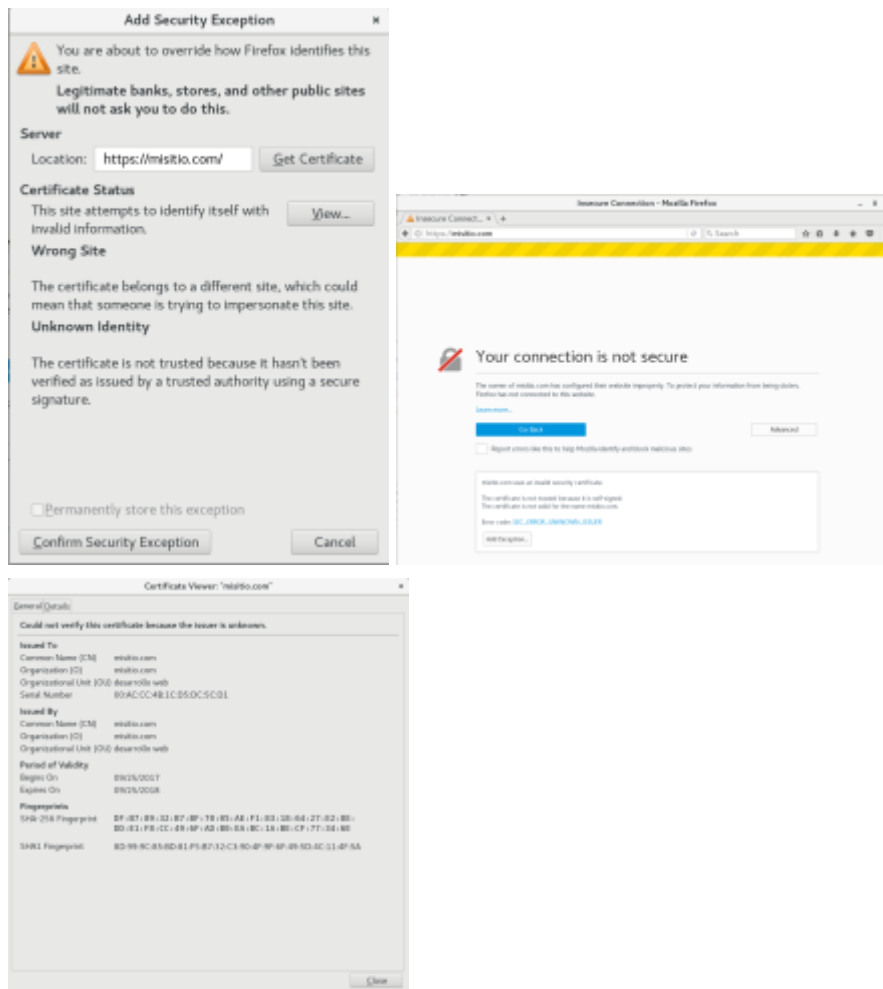
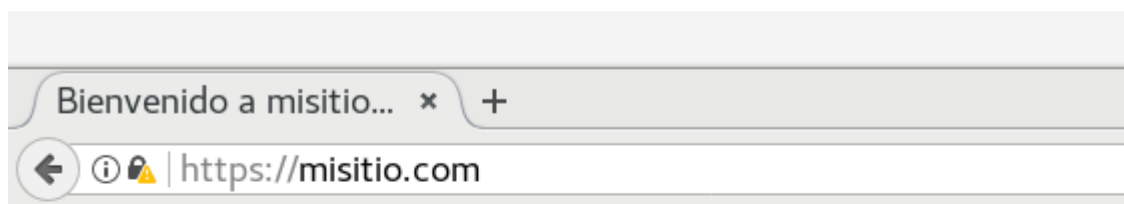


Figure 6: Certificado no seguro

Si finalmente aceptamos el certificado no seguro como excepción para el navegador (temporalmente o para siempre), podremos visitar la web utilizando una conexión segura.



Estás en misitio.com

Figure 7: Conexión segura utilizando HTTPS

Llegados a este punto nos puede interesar que todo el tráfico de la web se vea forzado a utilizar el protocolo seguro HTTPS. Incluso aunque el usuario introduzca la URL directamente y decide navegar utilizando HTTP (<http://...>) nosotros podemos redirigirle hacia la opción de utilizar la opción segura.

En ese caso podemos incluso configurar el host virtual no seguro con las opciones mínimas para redirigirlo al seguro. No haría falta ni incluir la opción `DocumentRoot`.

```
<VirtualHost *:80>
    ServerName www.misitio.com
    Redirect / https://www.misitio.com/
</VirtualHost>

<VirtualHost *:443>
    ServerName misitio.com
    DocumentRoot . . .
    . . .
</VirtualHost>
```

Finalmente podemos optar por una redirección permanente (de esta forma así se notificará a los buscadores) modificando la orden `Redirect` por la siguiente:

```
. . .
    Redirect permanent / https://www.misitio.com/
. . .
```

Protección frente a atacantes

```
. . .
ServerSignature Off
ServerTokens Prod
. . .
```

Despliegue de sitios web

Estadísticas web



Webalizer es una aplicación que podemos instalar para procesar el fichero `log` de Apache y generar un documento HTML con las estadísticas de nuestro sitio web.

Podemos instalarlo utilizando la herramienta `apt`.

```
alumno@daw:~$ sudo apt-get install webalizer
```

Y a continuación utilizar el comando `webalizer` para procesar el fichero `log` que queramos (en este caso el del sitio raíz) y éste generará una carpeta en `/var/www/webalizer` con un sitio web donde podremos visitar diferentes estadísticas sobre el uso del servidor.

```
alumno@daw:~$ sudo webalizer /var/log/apache2/access.log
```

Puesto que ahora por defecto en Debian la carpeta raíz para *Apache* es `/var/www/html`, tendremos que hacer un enlace simbólico dentro de dicha carpeta que apunte a la que *Webalizer* genera para poder visualizar el informe. También podría ser interesante proteger esa carpeta de alguna manera para que no fuera visible para cualquier visitante.

```
alumno@daw:/var/www/html/$ sudo ln -s ../webalizer webalizer
```

Ahora podemos visitar la página web generada donde podremos observar las estadísticas ya preparadas para nuestro sitio web (en este caso en <http://192.168.1.6/webalizer>, donde la IP será la de la máquina donde tenemos desplegado el sitio, o bien el dominio si lo tenemos)

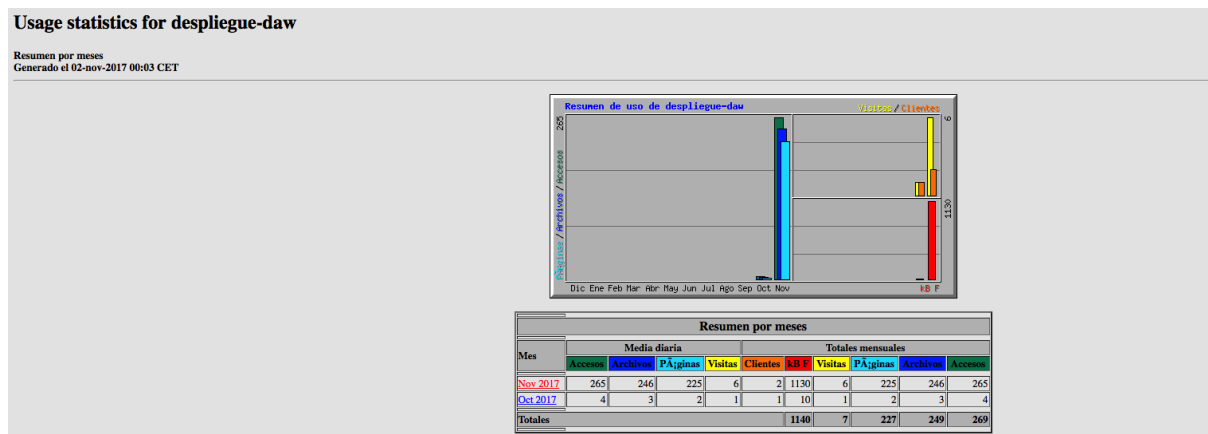


Figure 8: Estadísticas web con webalizer

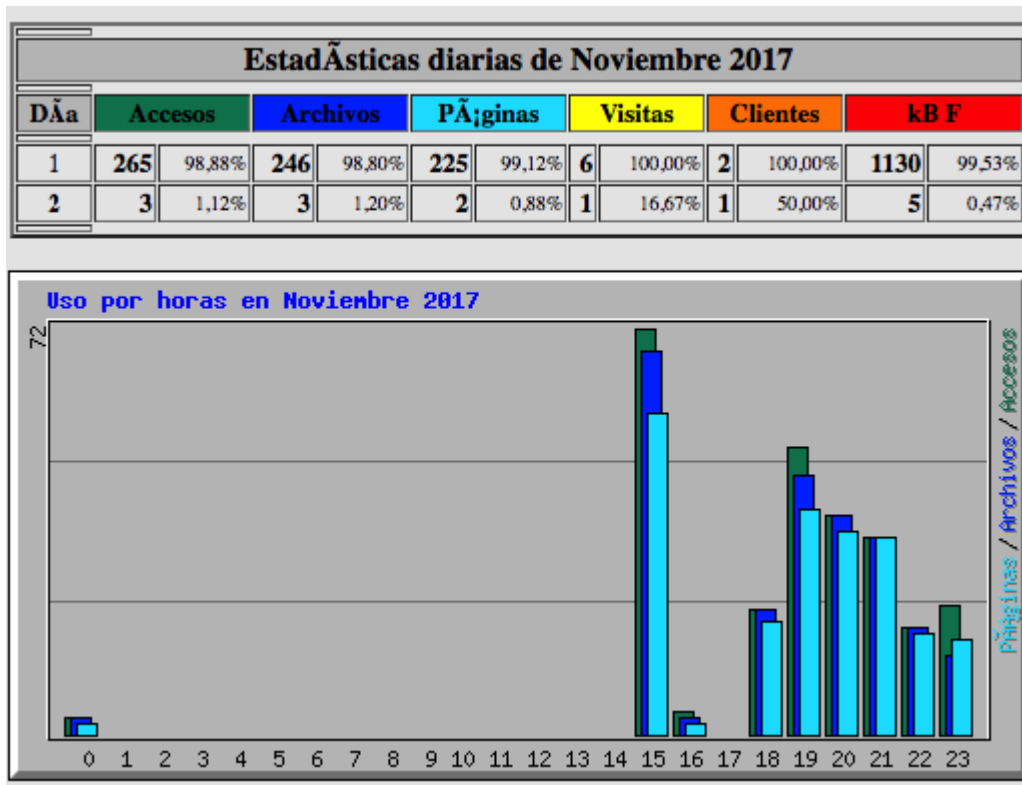


Figure 9: Estadísticas web con webalizer

Monitorización



[GoAccess](#) es un analizador visual de logs de Apache en tiempo real. De esa manera permite monitorizar el acceso y uso al servidor por parte de los usuarios en cada momento con numerosas métricas.

Lo primero de todo es instalarlo en nuestro servidor y, puesto que la versión que Debian trae por defecto puede no ser la más reciente, añadiremos al fichero `sources.list` de `apt` el repositorio oficial de los creadores de la herramienta. Luego lo instalaremos con `apt`

```
alumno@daw:~$ sudo echo "deb http://deb.goaccess.io/ stretch main" >>
/etc/apt/sources.list
alumno@daw:~$ sudo apt-get update
alumno@daw:~$ sudo apt-get install goaccess
```

Podemos usarla de dos maneras, visualizando los resultados por consola o en formato HTML como una web más.

Para visualizarlos desde la consola, basta localizar el fichero log de Apache que queremos monitorizar (en este caso el fichero general, pero podríamos pasar los ficheros `log` de los diferentes hosts virtuales que hayamos configurado) y ejecutamos la aplicación tal y como se muestra en el siguiente ejemplo:

```
alumno@daw:~$ sudo goaccess /var/log/apache2/access.log -c
```

Y tras elegir el formato de fichero `log` que usamos, podemos visualizar algo como la captura siguiente:

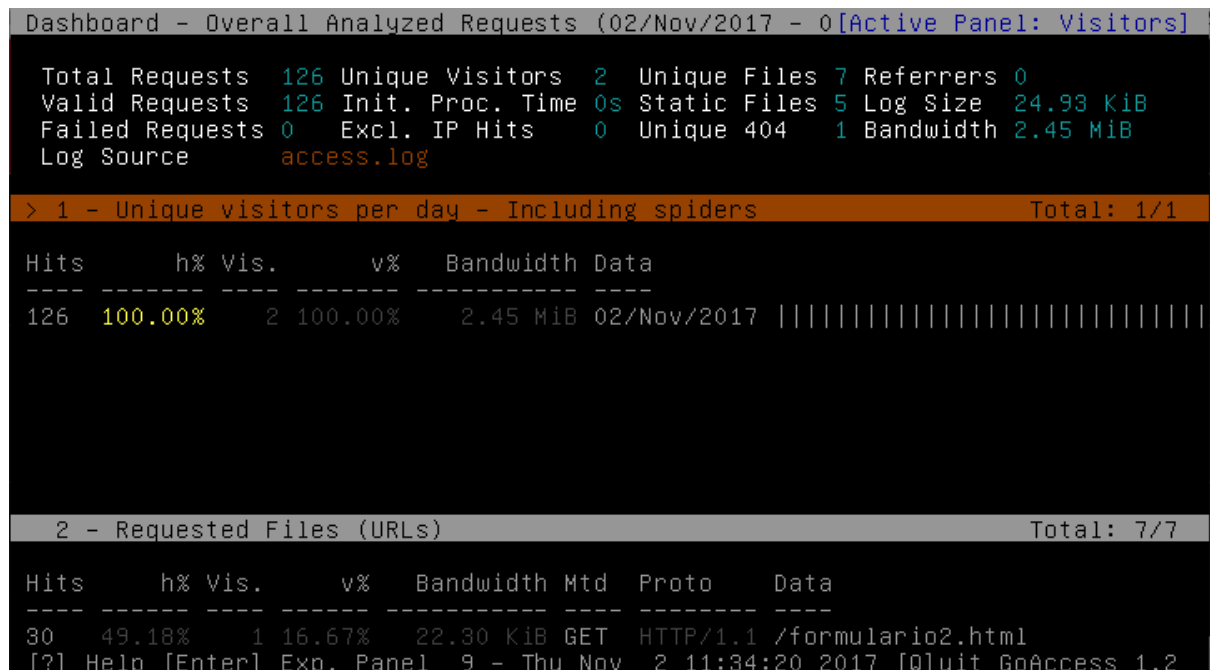


Figure 10: Monitorización de Apache en consola

También podemos pedirle a GoAccess que prepare un documento HTML en tiempo real donde podremos ver las estadísticas en tiempo real desde el navegador.

```
alumno@daw:~$ sudo goaccess /var/log/apache2/access.log -o
/var/www/html/report.html --log-format=COMBINED --real-time-html
WebSocket server ready to accept new client connections
```

Y este será el aspecto que tendrá, donde además podremos ir monitorizando el uso del servidor web puesto que se irá actualizando constantemente sin necesidad de recargar la página.

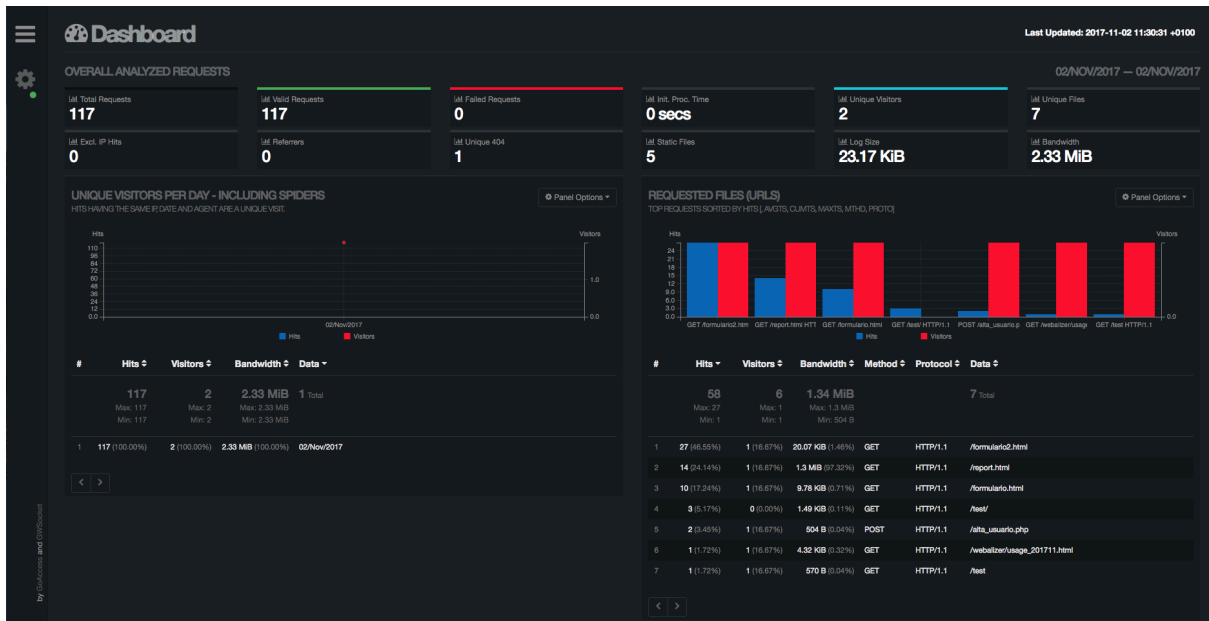


Figure 11: Monitorización de Apache en HTML en tiempo real