Despliegue de Aplicaciones Web

Tema 2: Servidores Web

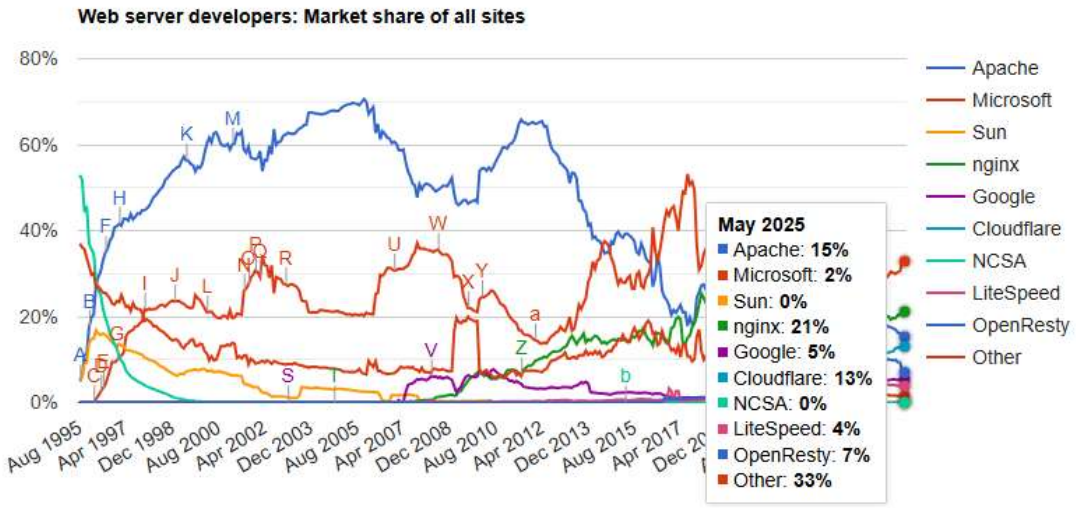
1. Introducción

Un servidor web es un programa que se ejecuta de forma continua en un ordenador (también llamada así la máquina que lo ejecuta), que se mantiene a la espera de peticiones por parte de clientes y contesta a éstas de forma adecuada, sirviendo una página web que será mostrada en el navegador o mostrando el mensaje correspondiente si se detectó algún error.

Un buen servidor web tiene que ser capaz de servir a muchos usuarios diferentes de la web a la vez, cada uno de los cuales solicitará diferentes páginas. Los servidores web procesan archivos escritos en diferentes lenguajes de programación como PHP, Python, Java y otros.

Los convierten en archivos HTML estáticos y le entregan estos archivos al navegador de los usuarios de la web.

Servidores web como Apache, Microsoft IIS o Nginx son los más utilizados hoy en día, siendo Apache el que se ha mantenido durante muchos años destacado en la primera posición, y que ha sido sobrepasado por nginx en los últimos tiempos.



1.1. Servicio de ficheros estáticos

Si al acceder a una página web no es necesaria la ejecución de código en el lado del servidor o en el del cliente entonces se entiende que la página es estática, de lo contrario será dinámica.

Para ofrecer páginas web estáticas, solamente es necesario que el servidor web disponga de soporte html/xhtml/css o incluso solamente html/xhtml:

- En cuanto a configuración y administración del servidor: se necesita un soporte mínimo de instalación del servidor Apache, esto quiere decir que, entre otros, no se necesita soporte para, por ejemplo, Python.
- En cuanto a rendimiento del servidor: no necesita de ejecución de código ni en el lado del servidor ni en el del cliente, por lo habrá un menor coste de memoria, una mayor rapidez en el acceso a la información de la página, etc.

Para poder ofrecer páginas estáticas mediante el servidor Apache simplemente se copiaría la página en la ruta correspondiente donde se quiera que se visione.

1.2. Contenido dinámico

Muchas veces, cuando se accede a una página, si tiempo después se accede a la misma, no es inusual que haya cambiado. Ya sea porque haya cambiado su contenido o que posea contenido dinámico, dependiente del código ejecutado en el servidor o en el cliente.

Cuando se accede a una página y dependiendo si se trata de un usuario u otro el contenido es distinto, o que presionando en una imagen de la página se produce un efecto, o que el contenido cambia dependiendo del navegador… supone que la página haya sido modificada por una interacción con el usuario y/o el navegador, y por lo tanto sea una página dinámica:

- Una página dinámica necesita más recursos del servidor web que una estática, ya que consume más tiempo de CPU y más memoria.
- En cuanto a la configuración y administración del servidor web: cuántos más módulos haya que soportar, más habrá que configurar y mantener(actualizar).
- En cuanto a seguridad del servidor web: cuántos más módulos más posibilidades de problemas de seguridad, así que si la página web dinámica necesita de ejecución en el servidor es imprescindible controlar que es lo que se ejecuta.

2. HTTP

La web funciona bajo el protocolo HTTP (o HTTPS en su caso). El navegador web utiliza este protocolo para comunicarse con el servidor Web donde está alojada la página que solicita.

Cada petición HTTP que el navegador realiza al servidor, se divide en 2 partes:

- Las cabeceras (Headers).
- El cuerpo de la respuesta (Response body).

En el cuerpo de la respuesta está el código HTML, CSS, Javascript, etc… necesario para que la web se vea y funcione. Pero, además, tanto el navegador como el servidor, en cada petición, envían unas cabeceras que sirven para configurar diversos comportamientos, o simplemente para dejar alguna marca con información.

Como ya se ha visto con anterioridad, el protocolo HTTPS permite que la información viaje de forma segura entre el cliente y el servidor, mientras que HTTP envía la información en texto plano, por lo que cualquiera que accediese a la información transferida entre el cliente y el servidor podría ver el contenido exacto y textual de la información.

Para asegurar la información, el protocolo HTTPS requiere de certificados y, siempre y cuando sean validados, la información será transferida cifrada. Pero cifrar la información requiere un tiempo de computación, por lo que saldrá perjudicado el rendimiento del servidor web. Por ello hay que plantearse si toda la información transferida entre el cliente y servidor tiene que ir cifrada.

Un servidor web, como Apache, puede emitir certificados, pero no siempre todos los navegadores lo aceptarán y puede incluso que lo interpreten como peligroso. Esto suele pasar porque los navegadores poseen en su configuración una lista de Entidades Certificadoras que verifican, autentican y dan validez a los certificados. Eso no quiere decir que no se puedan crear certificados en un servidor web, pero suelen utilizarse mayoritariamente para sitios internos o externos en los que solamente puede acceder personal autorizado.

La utilización del protocolo HTTPS no excluye ni impide el protocolo HTTP, los dos pueden convivir en un mismo dominio.

A continuación, se va a ver cómo funcionan estos protocolos.

En el protocolo HTTP cuando escribes una dirección URL en el navegador básicamente lo que ocurre es:

- Se obtiene la IP del sitio al que se solicita la información y se busca en ella si un servidor web aloja la página que ha sido solicitada en el puerto 80, que es el puerto asignado por defecto al protocolo HTTP.
- Si el servidor web aloja la página ésta será enviada al navegador.

Cuando lo que se escribe en el navegador es una URL con llamada al protocolo HTTPS, el procedimiento es algo más complejo que el anterior:

- Se obtiene la IP de la máquina que contenga la página solicitada, y se busca el servidor web que aloja la página, solicitada en el puerto 443, puerto TCP asignado por defecto al protocolo HTTPS.
- Antes de enviar la página al navegador se inicia una negociación SSL(TSL), en la que, entre otras cosas, el servidor envía su certificado -el navegador, aunque es poco habitual, también puede enviar el suyo-. Si el certificado es firmado por un Entidad Certificadora de confianza se acepta el certificado y se cifra la comunicación con él, transfiriendo así la página web de forma cifrada.

Es posible que un servidor web, para una determinada página, espere los protocolos HTTP y HTTPS en puertos TCP distintos del 80 y 443 respectivamente. Eso sí, es necesario que cuando se visite la página web, a mayores en la dirección URL, se especifique el puerto, por ejemplo:

http://www.tupagina.local:85

https://www.tupagina.local:444

2.1. Tipos MIME

El estándar Extensiones Multipropósito de Correo de Internet o MIME (Multipurpose Internet Mail Extensions), especifica como un programa debe transferir archivos de texto, imagen, audio, vídeo, etc. MIME está especificado en seis RFCs(Request for Comments) :RFC2045, RFC 2046, RFC 2047, RFC 4288, RFC4289, RFC2077.

Cada identificador de tipo MIME consta de dos partes. La primera indica la categoría general a la que pertenece el archivo como, por ejemplo, "text". La segunda parte detalla el tipo de archivo específico como, por ejemplo, "html". Un identificador de tipo MIME "text/html", por ejemplo, indica que el archivo es una página web estándar.

Cuando se transfiere una página web, y el navegador intenta abrir un archivo, el estándar MIME le permite saber con qué tipo de archivo está trabajando para que el programa asociado pueda abrirlo correctamente. Esta información formará parte de las cabeceras HTTP.

En esas cabeceras de la respuesta del servidor existe el campo Content-Type, donde el servidor avisa del tipo MIME de la página. Con esta información, el navegador sabe como debe presentar los datos que recibe.

El navegador del cliente también participa, además de estar capacitado para interpretar el concreto tipo MIME que el servidor le envía, también puede, en el diálogo previo al envío de datos, informar que tipos MIME puede aceptar la cabecera http_accept, así por ejemplo una cabecera http_accept tipo de un navegador sería:

text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

El valor */* significa que el navegador aceptará cualquier tipo MIME.

2.2. Contenido de una cabecera HTTP

Petición del navegador

```
GET /tutorials/other/top-20-mysql-best-practices/ HTTP/1.1
Host: net.tutsplus.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.5) Gecko/20091102 Firefox/
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: PHPSESSID=r2t5uvjq435r4q7ib3vtdjq120
Pragma: no-cache
Cache-Control: no-cache
```

La primera línea de la solicitud HTTP es llamada línea solicitud y consiste de 3 partes:

- El primer dato indica que tipo de solicitud es. Puede ser GET, POST y HEAD.
- La ruta, que normalmente es la parte de la url tras el dominio. Por ejemplo, cuando se solicita "https://net.tutsplus.com/tutorials/other/top-20-mysql-best-practices/" , la porción de la ruta es "/tutorials/other/top-20-mysql-best-practices/".
- La parte de protocolo, contiene "HTTP" y la versión(1.1 en navegadores modernos).

El resto de la solicitud contiene cabeceras HTTP como pares "Nombre: Valor". Contienen información diversa sobre la solicitud y el navegador. Por ejemplo, User-Agent provee información sobre la versión del navegador y el sistema operativo que se está usando, Accept-Encoding le dice al servidor si el navegador acepta salida comprimida como gzip, etc. Los datos de cookies también se transmiten dentro de una cabecera HTTP y, si hubiera una url referida, estaría en la cabecera también.

Los tipos de solicitud implican:

- GET: Recupera un Documento***.*** Principal método para recuperar HTML, imágenes, JavaScript, etc. La mayor parte de la información que se carga en ***el*** navegador ha sido solicitada usando este método. Una vez que el HTML carga, el navegador enviará nuevas solicitudes GET para imágenes.
- POST: Envía datos al Servidor. Aunque es posible enviar datos usando GET y la cadena de consulta, en muchos casos POST sería preferible. Enviar grandes cantidades de datos usando GET no es práctico y tiene sus limitaciones. Solicitudes de método POST pueden también hacerse vía AJAX, aplicaciones, cURL, etc. Y todos los formularios para subir archivos requieren que usen el método POST.
- HEAD: Recupera Información de Cabecera. Idéntico a GET, salvo que el servidor no devolverá el contenido en la respuesta. Con esto el navegador, por ejemplo, puede revisar si un documento ha sido modificado, para temas de almacenamiento en caché. O cuando una página contiene gran cantidad de enlaces, es posible periódicamente enviar solicitudes HEAD a todos ellos para revisar enlaces rotos.

Respuesta del servidor

La primera línea de la respuesta es la de estado, seguida por Cabeceras HTTP como pares "Nombre: Valor". Algunos de los datos más habituales en la cabecera de respuesta HTTP son:

- Date: Indica la fecha y hora exacta a la que el servidor ha realizado la respuesta para que conste (la hora del servidor).
- Content-type: El Tipo Mime de la respuesta (en caso de que sea una respuesta), o del contenido subido vía POST/PUT (en caso de que sea una Request)
- Content-Length: El tamaño de la respuesta en octetos (8 bits)
- Server: indica el tipo de servidor HTTP empleado.
- Age: tiempo que ha estado el objeto servidor almacenado en un proxy cache(seg)
- Cache-control: lo usa el servidor para decirle al navegador que objetos cachear, durante cuanto tiempo, etc..
- Content-Encoding: se indica el tipo de codificación empleado en la respuesta
- Expires: indica una fecha y hora a partir del cual la respuesta HTTP se considera obsoleta. Usado para gestionar caché.
- Location: usado para especificar una nueva ubicación en casos de redirecciones.
- P3P: se usa para especificar el tipo de política de privacidad empleado en la web. No está muy extendido.
- Set-Cookie: sirve para crear cookies. Las famosos cookies viajan entre el servidor y el navegador a través de estas cabeceras HTTP.

El código de respuesta HTTP puede ser de alguno de los tipos siguientes:

- 1xx: Mensajes.
- 2xx: Operación exitosa.
- 3xx: Redirección.
- 4xx: Errores del cliente.
- 5xx: Errores del servidor.

En el Aula Virtual de la asignatura tienes disponibles un par de enlaces con más información sobre los datos de las cabeceras y los distintos códigos de estados de respuesta.

EJERCICIO 1

- Descarga la aplicación WireShark y ejecútala.
- Selecciona Captura->Iniciar.
- Abre una pestaña del navegador Chrome y accede a www.apache.org.
- Ve a WireShark y detén la captura.
- Busca alguna trama HTTP en donde la petición sea GET/HTTP/1.1. Botón derecho sobre ella->Seguir->Flujo TCP.
- Plasma en un documento:

- Algunas de las cabeceras vistas en esta unidad. Explica que significa cada una de ellas y su contenido.
- Localiza 4 cabeceras que no conozcas, busca su significado y el de su contenido.

- Busca otra trama en la que se envíe una imagen, para ello busca en la cabecera de la trama un tipo MIME de algún tipo de imagen.

3. Apache HTTP Server

EJERCICIO 2

- Vamos a preparar el entorno de trabajo. Para ello debes instalar la última versión LTS de Ubuntu Server y configurar la tarjeta de red para asignar una IP de forma estática. Esta IP debe pertenecer a la red del colegio.

3.1. ¿Qué es Apache?

Apache HTTP Server (httpd) es un software de servidor web de código abierto y gratuito, que garantiza el rendimiento, estabilidad y seguridad de un servidor web, mantenido y desarrollado por la Apache Software Foundation y disponible para Windows y GNU/Linux, entre otros.

La licencia de software, bajo la cual el software de la fundación Apache es distribuido, es una parte distintiva de la historia de Apache HTTP Server y de la comunidad de código abierto. La Licencia Apache permite la distribución de derivados de código abierto y cerrado a partir de su código fuente original.

httpd es el programa servidor del protocolo Apache HTTP. Está diseñado para ser ejecutado como un proceso autónomo, no interactivo y en segundo plano(daemon).

3.2. Algunas características de Apache

Es un servidor web con soporte para HTTP/1.1 y HTTP/2.0.

En cuanto a su arquitectura se puede destacar lo siguiente:

- Está estructurado en módulos.
- Cada módulo contiene un conjunto de funciones relativas a un aspecto concreto del servidor.
- La funcionalidad de estos módulos puede ser activada o desactivada al arrancar el servidor.
- Los módulos de Apache se pueden clasificar en tres categorías:
    - Módulos base: Se encargan de las funciones básicas.
    - Módulos multiproceso: Encargados de la unión de los puertos de la máquina, aceptando las peticiones y atendiéndolas.
    - Módulos adicionales: para añadir funcionalidad al servidor.

Esta estructura basada en módulos permite habilitar o deshabilitar distintas funcionalidades tales como:

- Módulos de seguridad como mod_security.
- Módulos de caché como Varnish.
- Módulos de personalización de cabeceras como mod_headers.
- Etc.

Este software incluye, entre otros:

- envío mensajes de error personalizados.
- esquemas de autenticación.
- módulo de host virtual que permite ejecutar múltiples sitios simultáneamente.

- servicio de nombres de dominio DNS.
- SMTP (Simple Mail Transfer Protocol).
- FTP (File Transfer Protocol).

4. Instalación de Apache

4.1. Instalación

Para instalar Apache se puede descargar el software o, si se va a instalar en una máquina Linux, descargarlo usando el gestor de paquetes. Se va a instalar el servidor web, sus utilidades y la documentación (revisar que ese es el paquete actual de apache):

```
sudo apt install apache2 apache2-utils
```

También es posible realizar la instalación mediante la descarga del código fuente de la aplicación desde la web del proyecto Apache. Este código debería descomprimirse, compilarse y ya después instalarse.

Para comprobar si se ha instalado correctamente el servidor, se puede intentar acceder a él desde un navegador escribiendo en la barra de direcciones "localhost" o la ip "127.0.0.1". También se puede comprobar desde otra máquina de la red a través de la IP de la máquina donde esté instalado. Si efectivamente está bien instalado, debería aparecer la página de bienvenida que por defecto Apache muestra en estos casos.

4.2. Manejar el estado del servicio

Los comandos que se ven a continuación sirven para ver el estado del servidor y para arrancar o parar el servicio.

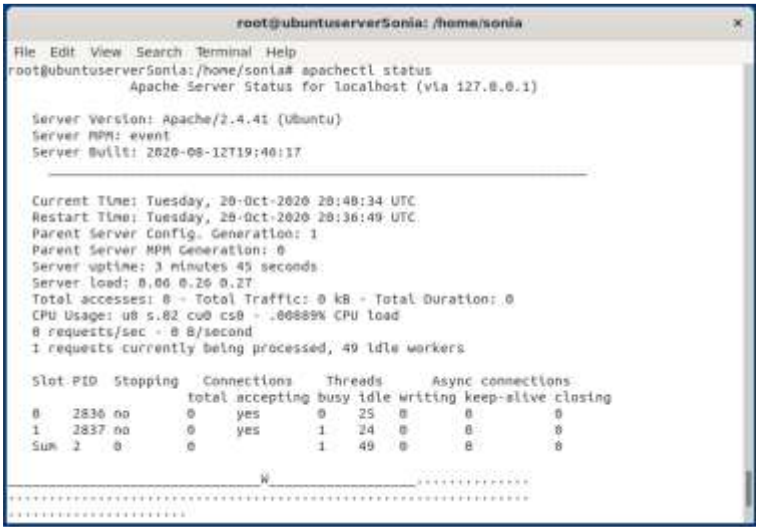
Estado de Apache

Para saber el estado del servidor se utiliza el comando que se ve a continuación. Este comando es bastante útil cuando el servidor está caído y no se sabe la razón. Al ejecutar este comando puede dar una explicación clara del problema:

```
sudo systemctl status apache2
```

Para conocer más datos del estado del servidor, se puede utilizar el siguiente comando (si no funcionara, habría que instalar antes un navegador para línea de comandos como links, links2 o lynx):

```
apachectl status
```



Detener el servidor

Para detener el servidor hasta el siguiente reinicio o hasta que sea arrancado de nuevo manualmente, existe el siguiente comando:

```
sudo systemctl stop apache2
```

Iniciar el servicio

Apache se iniciará automáticamente cada vez que se arranque el equipo en el que se aloja, pero si se ha parado y se quiere volver a iniciar, se hará mediante el comando:

```
sudo systemctl start apache2
```

Reiniciar el servidor

En este caso, se procedería a parar y arrancar el servidor con un solo comando. Esto es algo que se suele hacer cuando se realiza algún cambio en la configuración ya que Apache lee su configuración cuando arranca y no vuelve a acceder a ella, por eso es necesario reiniciarlo en cada cambio que se realizase y se quiera hacer efectivo.

```
sudo systemctl restart apache2
```

Recargar la configuración

Este comando permitirá recargar la configuración sin parar las conexiones (lo que si haría un reinicio)

```
sudo systemctl reload apache2
```

Deshabilitar/habilitar inicio automático

Dado que Apache está configurado por defecto para activarse cuando arranque el sistema operativo, existe un comando que permite modificar este hecho.

```
sudo systemctl disable apache2
```

Y para habilitarlo de nuevo

```
sudo systemctl enable apache2
```

Versión instalada

Es posible conocer la versión de Apache instalada con el siguiente comando:

```
apache2 –v
```

4.3. Algunos directorios importantes

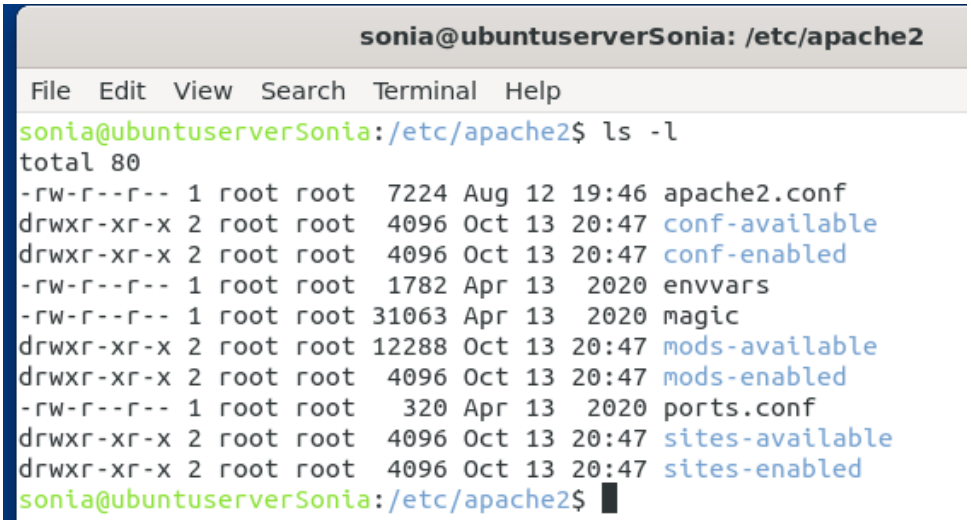
Contenido

Apache, por defecto, utiliza como carpeta raíz /var/www/html/, y ahí es donde se puede encontrar la página de bienvenida anteriormente mencionada (index.html).

Esto puede modificarse haciendo los cambios pertinentes en los ficheros de configuración.

Configuración del servidor

La configuración de Apache se almacenará en el directorio /etc/apache2:



En ese directorio, por ejemplo, está el fichero principal de configuración de Apache: /etc/apache2/apache2.conf

La configuración de Apache está muy estructurada, por lo que es conveniente saber que fichero/directorio hay que tocar según qué parte se quiera modificar. Aunque más adelante se verán más en profundidad algunos ficheros o directorios según se vaya avanzando con la configuración y uso de Apache, a continuación, se incluye una pequeña descripción de que hace cada uno. El hecho de que la configuración esté repartida en diferentes ficheros y no toda en uno de mayores dimensiones hace más fácil su mantenimiento, además de resultar más modular a la hora de activar o desactivar según qué características:

- 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 pertinentes, que se verán más adelante.
- sites-enabled: Contiene enlaces simbólicos a los ficheros de configuración cuyos hosts virtuales se encuentran activos en este momento.

Logs del servidor

En el directorio /var/log/apache2/ se pueden encontrar ficheros de log:

- access.log: por defecto, almacenará información de cada petición que llegue al servidor.
- error.log: por defecto, en este documento se irá almacenando información de los errores que se produzcan.

EJERCICIO 3

- Instala Apache en tu servidor Ubuntu.
- Comprueba desde el navegador de Ubuntu que se ha instalado correctamente.
- Haz la misma comprobación anterior desde tu equipo Windows y dale la ip de tu servidor a un compañero para que haga lo mismo desde su ordenador.
- Accede al directorio donde se aloja la página html que muestra el aviso de instalación correcta cuando se accede a través del navegador.
- Para el servicio e intenta acceder desde el navegador.
- Reinicia Ubuntu y comprueba que Apache se ha reiniciado con él, a pesar de haberlo detenido antes del reinicio.

4.4. Servicio de páginas estáticas

Una vez instalado Apache, sin ningún tipo de configuración adicional, ya sería posible servir páginas estáticas.

Como se ha comentado anteriormente, en la instalación de Apache se crea la página web en /var/www/html/index.html referenciada a través del archivo default (/etc/apache/sites-available/default). Ese archivo default contiene la configuración por defecto, generada en la instalación de Apache, para esa página. Si solamente se pretende servir una página web la forma más fácil y rápida de hacerlo sería sustituyendo la página index.html, referenciada en default, por la página que se quiera servir, por ejemplo paginaPrincipal.html.

EJERCICIO 4

En este ejercicio se va a comprobar a través de la práctica el funcionamiento más simple de Apache a la hora de servir una sola página web:

- Accede a través de un navegador a la página por defecto de Apache.
- Ve a la carpeta donde Apache aloja dicho documento index.html y cámbiale el nombre para no perder su contenido.
- Crea un nuevo index.html en la misma ubicación que el original con el contenido de algún documento HTML que tengas.
- Pulsas F5 en el navegador para actualizar la página y se debería ver tu página.

Ahora vas a realizar los cambios necesarios para poder acceder tanto a index.html (con su contenido original) como a otra página que incluyas tu:

- Ve de nuevo al directorio anterior y cambia el nombre a index.html por otraPagina.html.

- Vuelve a nombrar al index.html original (aquel que cambiaste de nombre) como index.html.
- Desde el servidor puedes acceder a otraPagina.html con 127.0.0.1/otraPagina.html

Por último, vas a comprobar cómo se pueden organizar diferentes páginas en directorios:

- Crea una carpeta dentro de /var/www/html/ llamada subcarpeta.
- Mueve el documento otraPagina.html a dicha carpeta.
- Accede a él desde el navegador teniendo en cuanta la carpeta en la que se ubica.

5. Configuración de Apache

Ya se ha visto que la carpeta raíz, de inicio, del sitio web creado por Apache es /var/www/html y que el nombre del archivo índice o por defecto es index.html. Si se quiere modificar este comportamiento y otros del servidor Apache habrá que modificar los archivos de configuración.

Para modificar la configuración del servidor Apache, es posible editar el archivo principal de configuración /etc/apache2/apache2.conf o editar alguno de los archivos que incluye éste. Hay un gran número de directivas para usar en la configuración del servidor, y en los siguientes apartados se van a ir viendo algunas.

5.1. Configuración principal

El directorio de Apache tendrá un aspecto similar al siguiente:

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

apache2.conf

Como ya se ha indicado, se trata del fichero de configuración principal. En versiones anteriores de apache2.conf aparecían opciones interesantes como puertos por los que escucha el servidor, configuración del sitio principal, de los módulos habilitados (o no) y los hosts virtuales, etc. Actualmente todas esas configuraciones han sido trasladadas a ficheros de configuración específicos, por lo que el fichero general rara vez es modificado.

- Para establecer el modelo de seguridad que se aplicará por defecto:

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

- AccessFileName .htaccess: El valor será el nombre del fichero que hay que buscar en cada directorio para directivas adicionales de configuración.
- Include ports.conf: Incluye la lista de puertos a través de los que escuchar. Referencia al fichero ports.conf.

Otras directivas interesantes que modifican esta configuración (pudiéndose encontrar en cualquiera de los archivos de configuración) son:

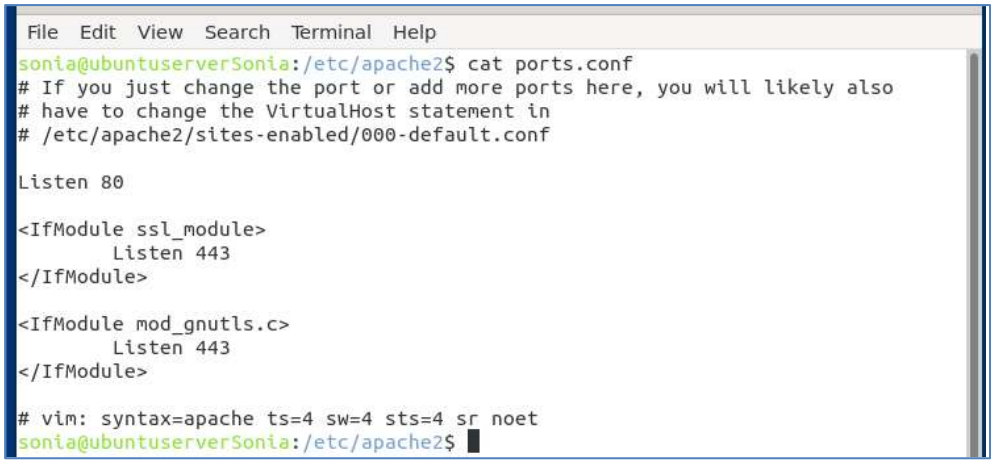
Directiva	Descripción	Ejemplo de uso
ServerName	Nombre de equipo y puerto que usa el servidor para identificarse ante los clientes. Es decir, indica el nombre del sitio web.	ServerName www.aulalinux.org:80
ServerRoot	Indica la carpeta en la que se ha instalado el servidor. No es la carpeta de archivos de configuración.	ServerRoot /usr/local/apache
ServerAdmin	Dirección de correo del administrador del servidor web. En caso de error se mostrará al usuario para que reporte cualquier incidencia.	ServerAdmin admin@serviciosHost.com
Listen	Especifica los puertos del servidor en los que escucha a los clientes y, opcionalmente, las IPs de los adaptadores de red que utiliza.	Listen 80 Listen 192.168.3.5:8080
Tiemout	Máximo tiempo en segundos que el servidor espera un nuevo mensaje de un cliente para no cerrar la conexión.	Timeout 300
KeepAlive	Para indicar que se mantenga (on) o no (off) una conexión con un cliente para varias transacciones.	KeepAlive on KeepAlive off
DocumentRoot	Define la ruta del equipo donde están todos los ficheros que se sirven en cada momento a través del sitio web. Todo lo que haya en este directorio está, en principio, siendo ofrecido a través de la web.	DocumentRoot var/web/
DirectoryIndex	Indica los nombres de las páginas web índice que puede entregar el servidor por defecto cuando en la petición HTTP la URL contiene sólo un nombre de carpeta.	DirectoryIndex index.html home.html index.php
NameVirtualHost	Especifica las direcciones IP y puerto en los que el servidor recibirá peticiones para sitios web virtuales basados en nombre. Si se usa esta directiva, se deben definir los sitios virtuales dentro de una directiva <VirtualHost> ... </VirtualHost>.	NameVirtualHost *:8
ErrorLog	Fichero donde se registrarán los errores que se produzcan	/var/log/ error_log

EJERCICIO 5

- Accede al fichero apache2.conf y localiza las directivas que se han detallado en este apartado y que indican con el valor que tienen asignado.
- Busca todas las veces que aparece <Directory/> y que significa su contenido.

ports.conf

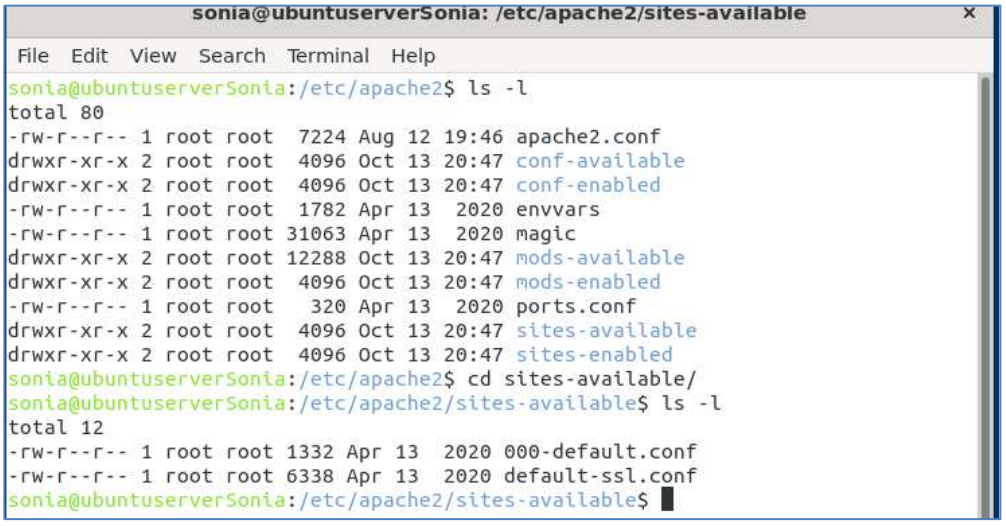
Archivo de configuración en el que se establecen los puertos en los que escucha Apache. Es un fichero que no se suele modificar porque los puertos de escucha suelen ser estándar, pero uno de los casos en los que si se cambia es para configurar el soporte para HTTPS. De inicio su aspecto será similar al de la siguiente imagen, donde se ve que apache escucha en el puerto 80, que si está activado el módulo ssl escuchará en el 443 y lo mismo para el módulo GnuTLS(extensión para proveer HTTPS):



sites-available/sites-enabled

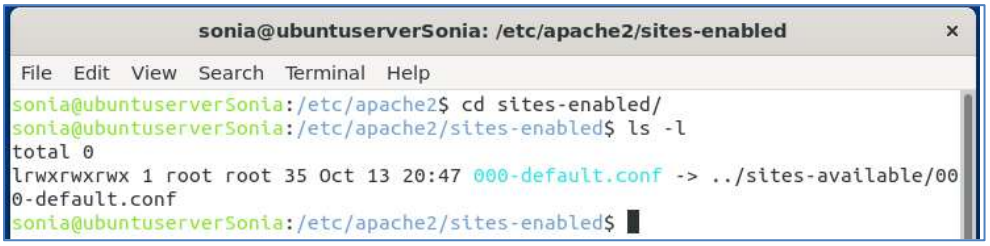
Se trata de dos carpetas que almacenan los ficheros de configuración de los host virtuales. En la primera de ellas se crean los ficheros de configuración (estén activos o no). De esos ficheros, se vinculan con la segunda aquellos que se quieran activos, para que se tenga en cuenta su configuración, ya que Apache sólo carga los que estén en sites-enabled.

En sites-available se localiza el fichero 000-default.conf que contendrá la configuración del servidor virtual por defecto. Un enlace a este fichero se encontrará en sites-enabled.



Tras la carga de configuración, el contenido inicial es la configuración del sitio principal establecido por defecto.

En la siguiente imagen se puede ver el vínculo que hay en sites-enabled para que Apache cargue la configuración y el sitio esté operativo.



EJERCICIO 6

- Accede, desde el navegador en la máquina Windows, a la página otraPagina.html que creaste en el ejercicio 4. Comprueba si la página se abre correctamente. Si no se abriera, realiza los cambios que consideres necesarios.
- Ya desde la terminal de Ubuntu, accede al contenido del fichero ports.conf y comprueba los valores configurados por defecto.
- Comprueba que en sites-available está creado el archivo default. Busca información de las directivas que contiene este fichero.
- En apache2.conf, busca la directiva que incluye los ficheros localizados en el directorio sites-enabled.
- Consultando el listado de directivas de la documentación de Apache, responde a las siguientes preguntas:
  - ¿Se permiten conexiones persistentes? ¿Qué directiva define este comportamiento?
  - ¿Qué fichero que por defecto recoge los errores? ¿Qué directiva lo define?

6. Creación de host virtuales (hosting compartido)

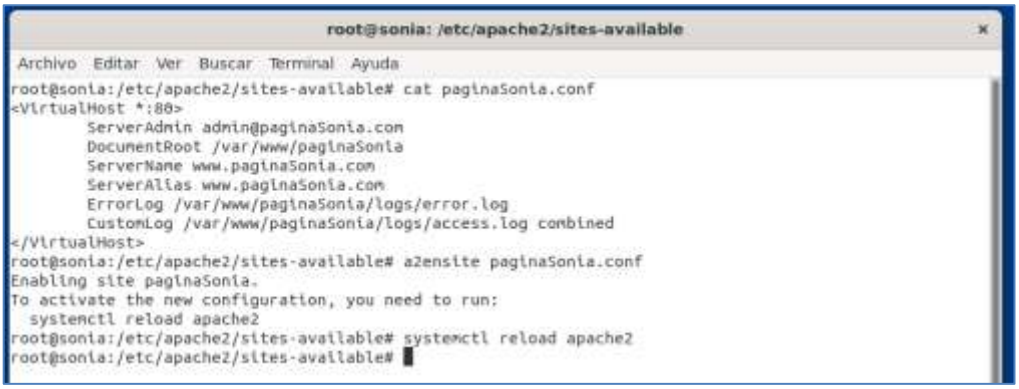
En puntos anteriores se ha visto que es posible alojar múltiples páginas web en el servidor web, pero pertenecientes todas al mismo dominio. Para alojar páginas de distintos dominios en un mismo servidor web, es necesaria la creación de host virtuales.

El hosting compartido consiste en el mantenimiento de diferentes sitios web (independientes entre ellos) en un mismo equipo, compartiendo recursos. Cada sitio web tendrá su virtualhost único e independiente de las demás.

Aunque se cree un host virtual para cada sitio, toda configuración que no se establezca para dicho sitio se heredará de la principal, de apache2.conf. Por ello, toda directiva que vaya a ser igual para todos los sitios alojados, solo necesitará ser definida en apache2.conf.

El archivo de configuración de cada sitio debe encontrarse en el directorio /sites-available:

Cuando se haya terminado de configurar el nuevo host virtual, se puede activar utilizando el comando a2ensite (**a**pache**2** **en**able **site**). Automáticamente se creará un enlace simbólico con la configuración del sitio de sites-available en sites-enabled, y reiniciar apache después:

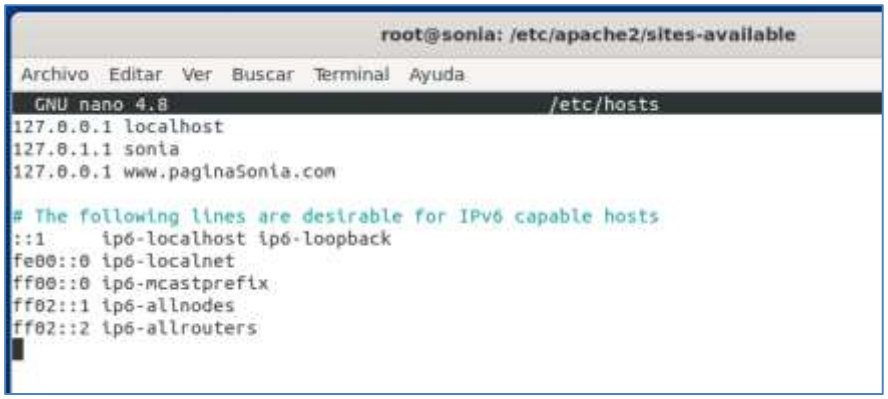


También podemos desactivar un host virtual con el comando a2dissite (**a**pache**2** **dis**able **site**). Para que surta efecto habrá que reiniciar el sistema apache de nuevo. El sitio permanecerá desactivado hasta que lo volvamos a activar con a2ensite.

En estos casos, cuando se empiezan a tener numerosos ficheros de configuración, es donde resulta útil el comando "service apache2 status" ya que, en caso de que exista algún error que impida arrancar el servicio, dará información para encontrarlo.

Si finalmente la configuración es correcta, el host virtual estará funcionando. Si el servidor estuviera conectado a Internet, y se fuese dueño del dominio (y éste apunta a nuestro equipo), los usuarios ya podrían empezar a visitar la nueva web.

En este caso, en clase, vamos a decirle al equipo que el dominio creado apunte al propio equipo, así es posible comprobar que todo funciona. Para ello se editará el fichero /etc/hosts que funciona como DNS local, y se añade la IP que se quiere asignar al dominio (en este caso, el propio equipo, 127.0.0.1):



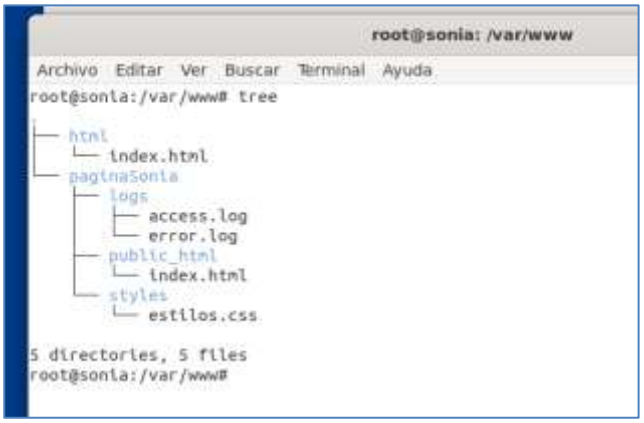
Ahora, al hacer ping contra el dominio, es posible ver que responde el equipo propio, por lo que todo funciona correctamente.

EJERCICIO 7

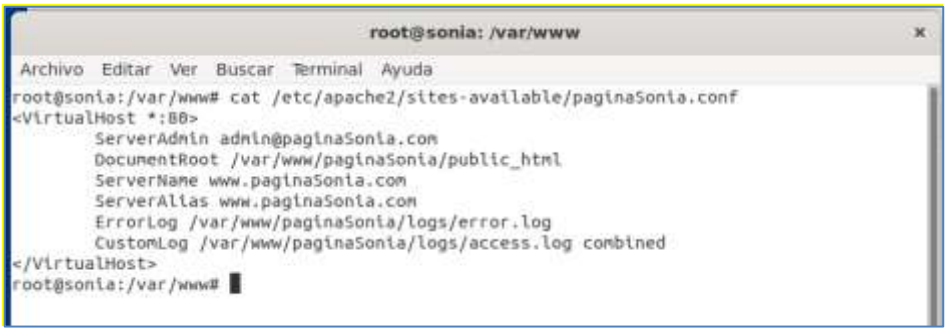
- Crea un host virtual en tu servidor tal y como se ha visto en este apartado.
- Crea un documento HTML (o coge uno que tuvieras ya), para que sea la página de inicio de este host virtual. Busca como hacerlo.
- Prueba con un navegador en línea a acceder a la página creada.
- Accede unas cuantas veces a la página desde el navegador, después comprueba que esas visitas realizadas se han ido registrando en el correspondiente fichero de log que hayas configurado.

Como crear el directorio para el sitio web

- Crea un directorio donde se alojará el sitio web. El nombre coincidirá con el dado en el fichero de configuración del sitio. Dentro de dicho directorio es común crear una carpeta para los ficheros del sitio en sí, otras para los ficheros de log y, si se considera necesario, otra para el backup.
- En esa carpeta, donde se incluirán todos los documentos del sitio, se alojará el fichero index.html:



- El directorio donde se aloja el sitio web deberá tener, para otros usuarios, permisos de lectura como mínimo, sino no será accesible.
- Modifica los datos necesarios del fichero de configuración del sitio para que concuerden con el directorio creado:



- Reinicia apache.
- Para probarlo desde la máquina Windows, simulamos un servidor DNS modificando el fichero hosts de Windows. En el Aula Virtual tienes un enlace a una página donde se explica cómo modificar dicho fichero.

7. Instalación y configuración de módulos

Un servidor web debe proporcionar estabilidad, disponibilidad y escalabilidad. Para que esta última característica sea sostenible en el tiempo, es muy importante poder dotar al servidor web de nuevas funcionalidades de forma sencilla, así como del mismo modo poder quitarlas o deshabilitarlas.

Apache ofrece, a través de sus módulos, extender su funcionalidad, suponiendo esto una gran escalabilidad y disponibilidad y por ello es uno de los servidores web más manejables y potentes. Por ejemplo:

- Soporte para lenguaje PHP.
- Soporte para Python.
- Autenticación HTTP.
- Control del ancho de banda.
- Reescritura de URLS.
- En el apartado de seguridad, configurar Apache para conexiones seguras HTTPS.
- etc.

Para trabajar con módulos en Apache hay dos comandos fundamentales, a2enmod y a2dismod.

Mediante el comando **a2enmod** se puede habilitar un módulo. Si no se incluye ningún parámetro, Apache preguntará que módulo se quiere habilitar. Los ficheros de configuración de los módulos disponibles están en /etc/apache2/mods-available/ y al habilitarlos se crea un enlace simbólico desde /etc/apache2/mods-enabled/.

Por su parte, el comando **a2dismod** permite deshabilitar un módulo de Apache. Sin ningún parámetro, al igual que el anterior, preguntará que módulo se va a deshabilitar. Al deshabilitar un módulo se elimina el enlace simbólico al mismo desde /etc/apache2/mods-enabled/.

La instalación o desinstalación de un módulo no implica la desinstalación de Apache o la nueva instalación de Apache perdiendo la configuración del servidor en el proceso, simplemente implica la posibilidad de poder trabajar en Apache con un nuevo módulo o no.

Un módulo puede estar instalado, pero no habilitado, por lo que habría que habilitarlo para que funcionara. A continuación, se ve como instalar, habilitar, deshabilitar y eliminar un módulo:

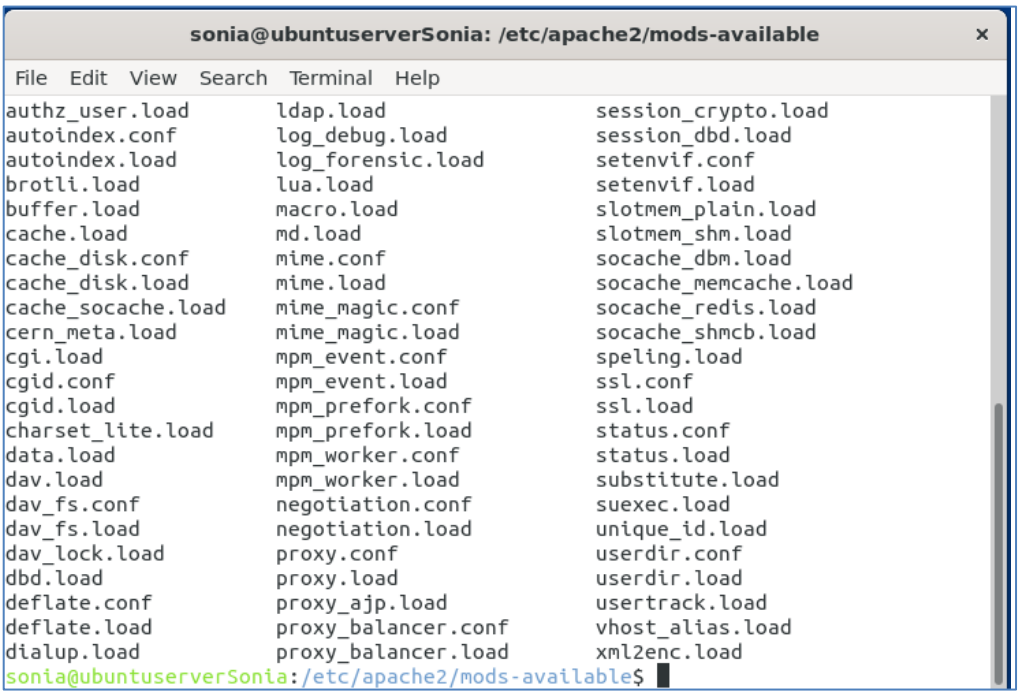
- apt-get install libapache2-mod-gnutls: instala el módulo en cuestión, en este caso ssl.
- a2enmod ssl: habilita el módulo ssl (crea el enlace).
- a2dismod ssl: deshabilita el módulo ssl (elimina el enlace).
- apt-get remove libapache2-mod-gnutls: elimina el módulo ssl.

Para que la habilitación de un módulo surta efecto se ha de recargar la configuración de apache.

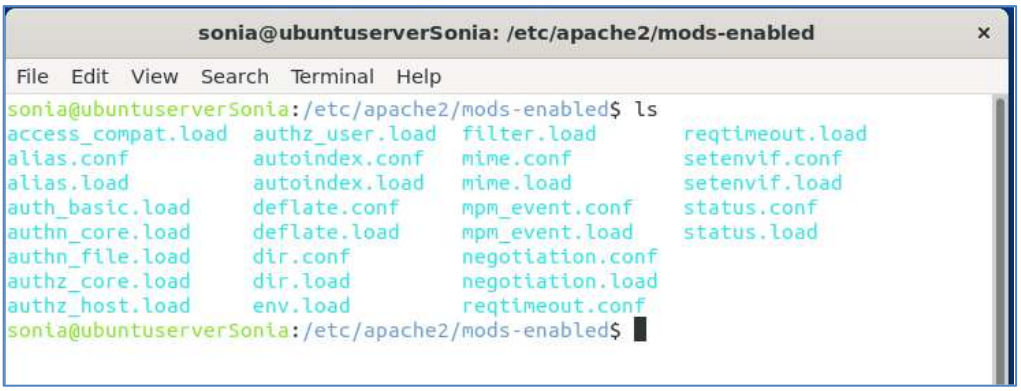
Ejercicio 8

- En el documento "Python en apache" puedes encontrar un ejemplo de despliegue de una aplicación desarrollada con Python.
- Dedica unos minutos a leer y entender los pasos que se dan. No te preocupes si no entiendes la parte de arquitectura y código Python. Después contesta a las siguientes preguntas:
  - ¿Cuál es el módulo que se instala y activa para poder desplegar aplicaciones desarrolladas con Python?
  - ¿Dónde se almacena el fichero de configuración del host virtual?
  - ¿Cuál es el directorio raíz de la aplicación a desplegar?
  - ¿Dónde se almacenarán los logs generados por la aplicación?

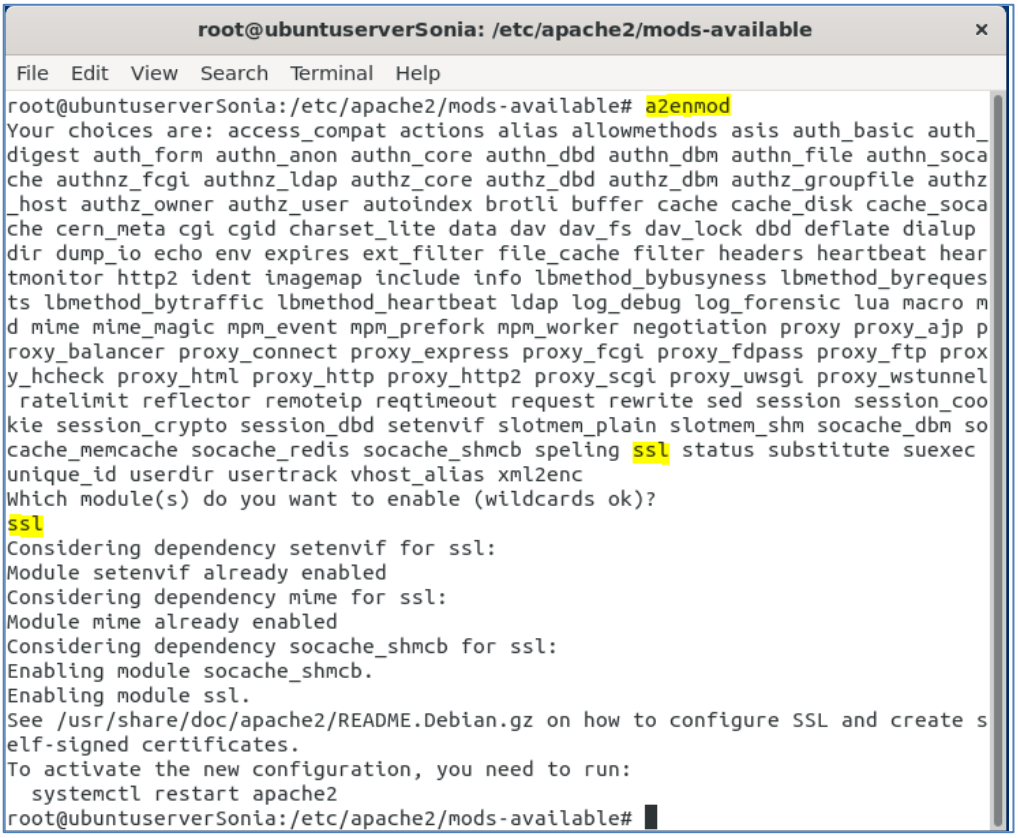
En la siguiente imagen se ven algunos de los módulos instalados en apache (el contenido del directorio mods-available):



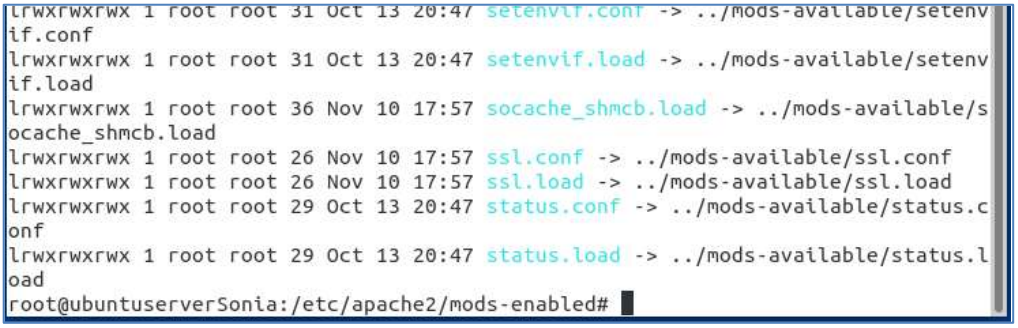
Y a continuación los módulos que están habilitados (el contenido de mods-enabled):



Si, por ejemplo, se quiere habilitar el módulo visto anteriormente, ssl, que ya está instalado:



Y el enlace se ha creado:



EJERCICIO 9

- Dentro de mods-available, visualiza algún fichero .load y observa cómo se utiliza la directiva LoadModule para cargar el módulo en cuestión. Busca cual es la funcionalidad de LoadModule.
- Edita, después, algún fichero .conf(por ejempl dir.cong) y observa cómo se añaden directivas dentro de una declaración IfModule. Busca cual es la funcionalidad de IfModule y, también, que indica la directiva DirectoryIndex.
- Habilita (y si no está disponible instala antes) el módulo userdir. Verifica que se ha habilitado. Busca para que sirve ese módulo y alguna forma de probarlo.

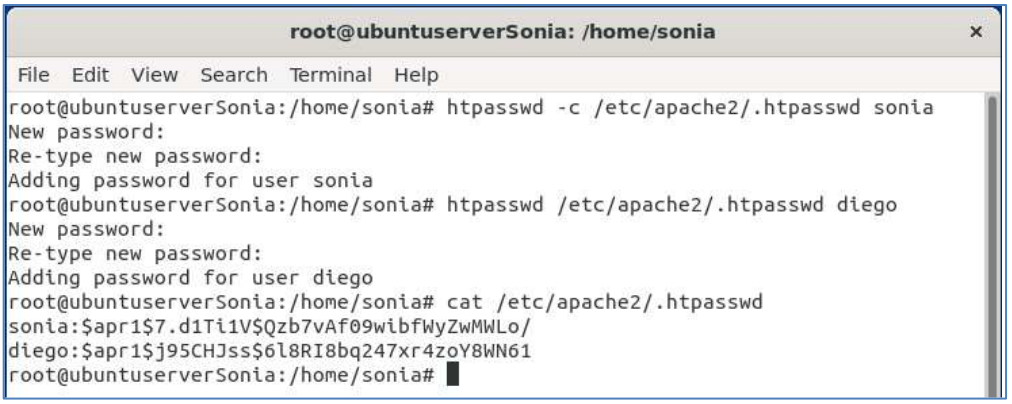
7.1. Autenticación HTTP

La autenticación HTTP permite proteger carpetas personales dentro del servidor web (en este caso Apache). Para hacer uso de ella, lo primero es instalar (si no lo está) el paquete apache2-utils.

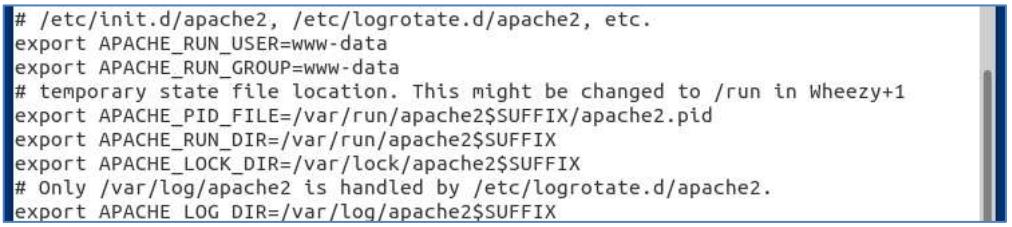
Una de las utilidades que ofrece es manipular archivos de autenticación (con el comando htpasswd) que serán los ficheros a través de los cuales se configura el acceso, mediante usuario y contraseña, a esas carpetas protegidas dentro del servidor web.

El comando htpasswd mencionado antes, se utiliza tanto para crear el fichero por primera vez y añadir un usuario a la zona protegida que se va a crear, como para añadir otros usuarios una vez creado.

A continuación, se ve un ejemplo de creación. Al no existir el fichero, la primera vez hay que hacer uso de la opción -c. Para añadir más tarde otros usuarios ya no habría que incluirla, ya que si se hiciera se sobre escribiría el fichero y se perderían los datos anteriormente almacenados:

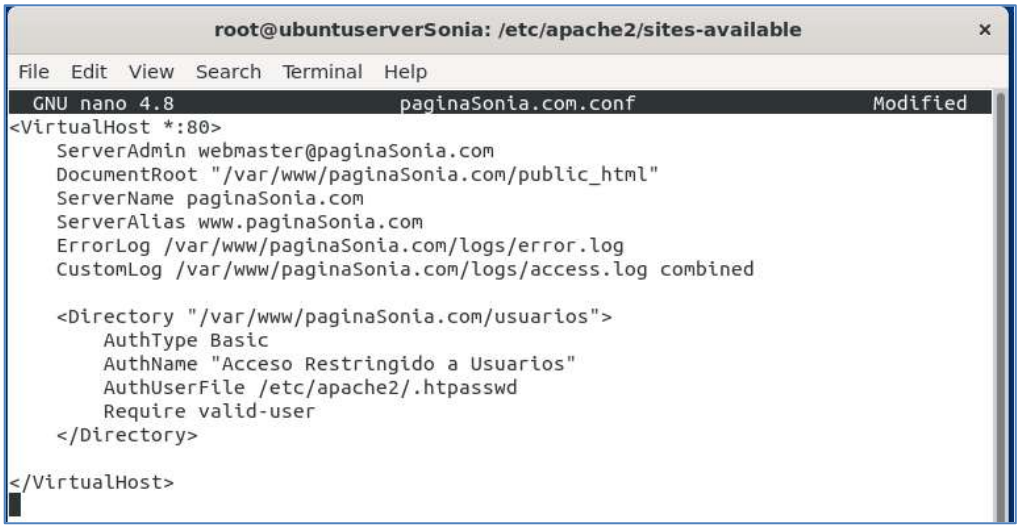


Una vez configurado usuario y contraseña, hay que asignar esa propiedad al usuario y grupo de apache. En el fichero de variables de entorno de apache se pueden ver (/etc/apache2/envvars). A continuación, se ve un fragmento del contenido del fichero envvars en el que se ve que, tanto usuario como grupo de apache, son www-data y www-data:



Entonces, se asigna:

El siguiente paso sería proteger el/los directorios, de algún sitio web, que se necesite. Para ello se incluye la directiva Directory en el fichero de configuración del sitio web. En el caso de ejemplo que se muestra a continuación, se está protegiendo el directorio usuarios que, supongamos, tiene información sensible que no se quiere poner al acceso de cualquiera:

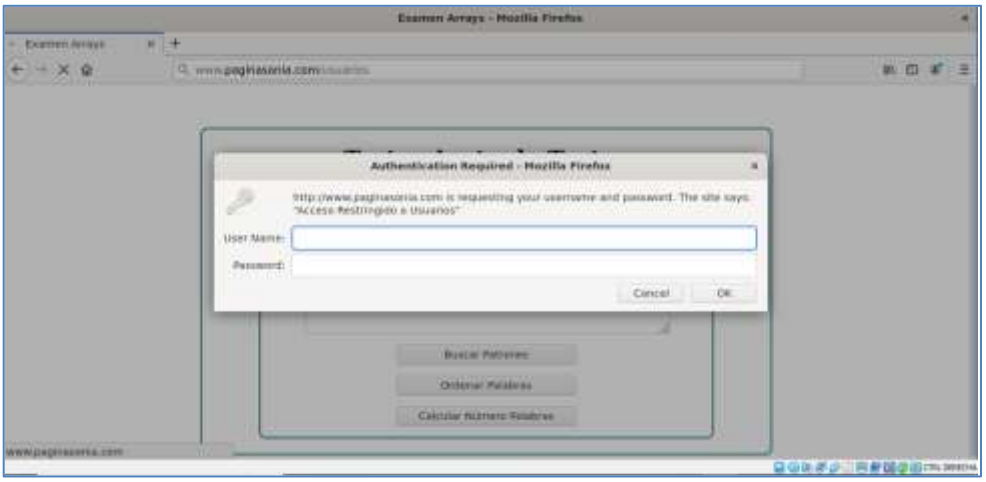


Esas directivas que se incluyen son:

- 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 se ha creado con los usuarios/contraseña que tienen permitido el acceso.
- Require valid-user: Indica que sólo se podrá acceder con un usuario válido. Si, de los usuarios que haya configurados, se pretende que solo tenga permisos de acceso uno, se puede indicar como sigue: Require user sonia.

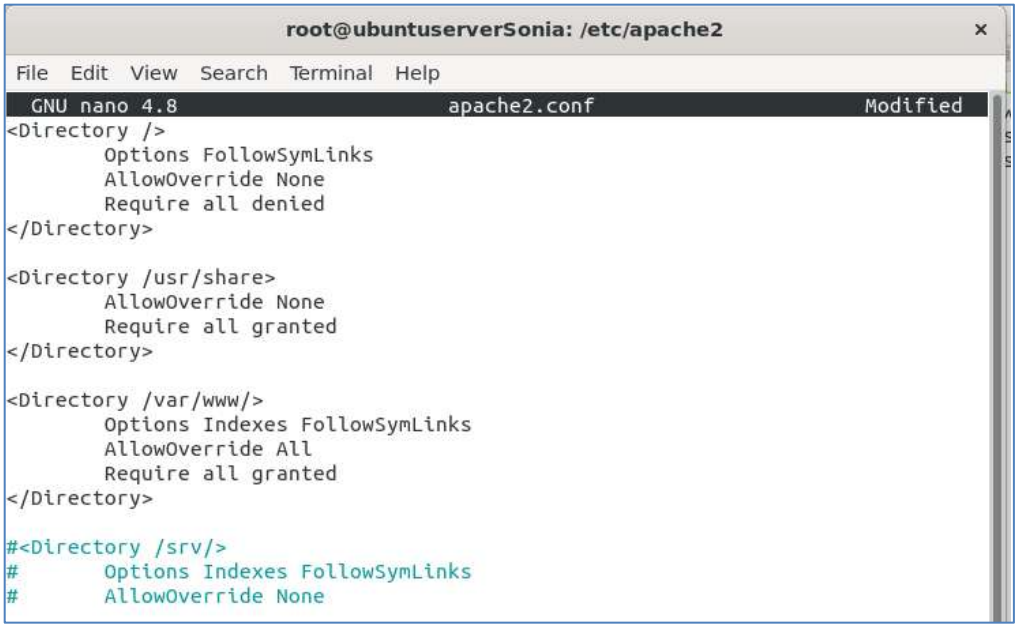
Por último, como siempre que se modifica cualquier dato de configuración, hay que reiniciar Apache o recargar su configuración.

Para comprobar que, efectivamente, se ha establecido la seguridad configurada, se puede probar a acceder desde el navegador y éste pedirá los datos de autenticación:



7.2. Autenticación HTTP con .htaccess

Otra forma de configurar la autenticación HTTP es creando el fichero .htaccess que se almacena dentro de la carpeta que se quiera proteger. Esta opción supone tener que modificar la configuración general de Apache para permitir que las propiedades sobre el directorio raíz puedan ser modificadas. El cambio que hay que realizar es sobre la directiva AllowOverride, que se establecerá a All, permitiendo que la política sobre dicho directorio sea diferente si así se define con algún fichero .htaccess.



Y, ya en el directorio que se quiere proteger, se incluye un fichero .htaccess con el siguiente contenido:

Y, como siempre, reiniciar Apache.

EJERCICIO 10

- Para algún host virtual de los que tengas creados, asigna autenticación (de alguna de las formas que se ha visto) a dos carpetas diferentes del directorio (si no tienes crea alguna) y da permisos de acceso a sendos usuarios.