RESUMEN TEMA 2. Configuración y administración de servidores Web.

*Albert Einstein: "Se debe hacer todo tan sencillo como sea posible, pero no más sencillo."*

*Confucio:* "*Me lo contaron y lo olvidé. Lo vi y lo entendí. Lo hice y lo aprendí.*"

## 1.- Funcionamiento de un servidor Web

## 1.1.- Servicio de ficheros estáticos

Debemos diferenciar cuando accedemos a una página web entre código ejecutable en el lado del servidor y en el lado del cliente.

Si al acceder a una página web no es necesaria la intervención de código en el lado del servidor, entonces entenderemos que la página es estática, si por el contrario la intervención en el lado servidor y/o en el lado del cliente entenderemos que la página es dinámica.

Ofrecer páginas estáticas es simple, puesto que solamente se necesita que el servidor web disponga de soporte html/xhtml/css o solo html/xhtml.

La configuración y administración del servidor es simple: soporte mínimo base de instalación del servidor Apache.

El rendimiento del servidor, es bajo, como no se ejecuta código en el lado servidor, el coste de CPU y memoria es bajo y se traduce en mayor rapidez.

Para ofrecer páginas estáticas mediante el servidor Apache simplemente copias la página en la ruta correspondiente donde quieres.

Estructura de ficheros de Apache:



En la instalación de Apache se crea una página web en **/var/www/html/index.html** referenciada a través del archivo **/etc/apache2/sites-avaliable/000-default.conf** (contiene la configuración por defecto.

Ejercicio:

Copiar el archivo .zip al servidor ubuntuserver en el directorio raíz del usuario Ubuntu.

Scp ruta.zip ubuntu@dirip-server:/home/usuario

El fichero nuevo, descomprimimos y copiamos el contenido del directorio web-estatica al directorio **/var/www/html.**

Comprobar el despliegue desde el navegador.

## 1.2.- Contenido dinámico

Una página dinámica, necesita más recursos del servidor web que una página estática, ya que consume más tiempo de CPU y más memoria.

La configuración y administración del servidor web será más compleja, cuantos más módulos tengamos que soportar, más tendremos que configurar y actualizar.

Repercusión en la seguridad del servidor web, cuántos más módulos más posibilidades de problemas de seguridad. Algunos módulos son: *mod\_actions, mod\_cgi\_ mod\_cgid, mod\_ext\_filter, mod\_include, mod\_ldap, mod\_perl, mod\_php, mod\_python.*

***ACTIVAR O DESACTIVAR MODULOS****:* a2enmod , a2ensite, a2dismod, a2dissite

Debemos controlar que es lo que se ejecuta en el servidor.

Ejercicio:

Copiar y descomprimir el fichero .zip en el directorio raíz.

Copiar el contenido de la web dinámica al directorio /var/www/html.

Comprobamos el funcionamiento accediendo desde un navegador a la URL.

*Composer: es una herramienta de administración de dependencias para PHP que facilita la gestión de bibliotecas y paquetes en tus proyectos utilizado por los desarrolladores PHP.*

* *Instalación de dependencias: permite definir las bibliotecas y paquetes que tu proyecto necesita y los instala automáticamente a partir de registros de paquetes en línea.*
* *Autocarga de clases: genera un archivo de autocarga que simplifica la inclusión de clases en tu código eliminando la necesidad de incluir manualmente archivos de clase.*
* *Resolución de conflictos: resuelve automáticamente conflictos entre diferentes versiones de dependencias para garantizar la compatibilidad de las bibliotecas en tu proyecto.*
* *Actualización de dependencias: puedes actualizar las dependencias de tu proyecto fácilmente con Composer, lo que te permite mantener tu código actualizado con las últimas versiones de las bibliotecas.*
* *Creación de paquetes personalizados: te permite crear y distribuir tus propias bibliotecas y paquetes PHP de manera sencilla.*
* [*https://packagist.org/*](https://packagist.org/)

|  |
| --- |
| * sudo apt update * sudo apt install php-cli unzip * cd ~ * curl -sS [https://getcomposer.org/installer -o /tmp/composer-setup.php](https://getcomposer.org/installer%20-o%20/tmp/composer-setup.php) * HASH=`curl -sS [https://composer.github.io/installer.sig`](https://composer.github.io/installer.sig%60) * echo $HASH * php -r "if (hash\_file('SHA384', '/tmp/composer-setup.php') === '$HASH') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP\_EOL;" * sudo php /tmp/composer-setup.php --install-dir=/usr/local/bin --filename=composer * composer |

## 1.3.- Protocolo Httpy Https

<https://howhttps.works/es/>

El protocolo de Tranferencia de Hipertexto (**http**) es un protocolo de comunicación utilizado en la WWW para la transferencia de datos entre un cliente y un servidor web. Consta de dos tipos principales:

**Solicitud (Request):** este mensaje es enviado por el cliente al servidor para solicitar un recurso web. Una solicitud http generalmente incluye métodos:

* Get: recuperar datos
* Post: enviar datos al servidor
* Put: actualizar recursos
* Delete: eliminar recursos

Además, contiene la URL del recurso solicitado y una serie de cabeceras que proporcionan información adicional.

**Respuesta (response):** el servidor responde a una solicitud http con un mensaje de respuesta que incluye un código de estado (200 OK, 404 Not Found), el tipo de contenido y el contenido real solicitado. También puede incluir cabeceras adicionales para proporcionar información adicional sobre la respuesta.

Estos mensajes http son la base de la comunicación entre los navegadores web y los servidores web, permitiendo la transferencia de datos y recursos a través de la WWW.

El protocolo https permite que la información viaje de forma segura entre el cliente y el servidor.

El protocolo **http** envía la información en texto claro, esto es, cualquiera que accediese a la información transferida entre el cliente y el servidor puede ver el contenido exacto y textual de 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 será perjudicado el rendimiento del servidor web.

Un servidor web, como Apache, puede emitir certificados, pero puede que en algún navegador sea interpretado como peligroso, esto suele ser debido a que 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 puedes crear tus certificados en un servidor web, de hecho muchas empresas lo hacen, sobre todo para sitios internos o externos en lo que solamente puede acceder personal autorizado por la propia empresa.

El protocolo https utiliza cifrado sobre SSL/TLS que proporciona autenticación y privacidad. Entonces, si necesitas que la información viaje cifrada debes emplear el protocolo https, en caso contrario el protocolo http. Hay que dejar claro que la utilización del protocolo https no excluye ni impide el protocolo http, los dos pueden convivir en un mismo dominio.

Bien, pero ¿Cómo funciona?

En el protocolo http cuando escribes una dirección url en el navegador, antes de ver la página en el este existe todo un juego de protocolos, sin profundizar en todos ellos básicamente lo que ocurre es lo siguiente:

* Se traduce el dominio dns por una ip, una vez obtenida la ip se busca en ella si un servidor web aloja la página solicitada en el puerto 80, puerto tcp asignado por defecto al protocolo http. Si el servidor web aloja la página ésta será transferida a tu navegador.

Sin embargo, cuando escribes en el navegador una dirección url con llamada al protocolo https, el procedimiento es similar al anterior pero un poco más complejo:

* Así se traduce el dominio dns con una ip, con la ip se busca el servidor web que aloja la página solicitada en el puerto 443, puerto tcp asignado por defecto al protocolo https, pero ahora antes de transferir la página a tu navegador se inicia una negociación ssl, en la que entre otras cosas el servidor envía su certificado. Si el certificado es firmado por una Entidad Certificadora de confianza se acepta el certificado y se cifra la comunicación con él, transfiriendo así la página web de forma cifrada.

## 1.4.- Tips 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 o cualquier archivo que no esté codificado en US\_ASCII. MIME está especificado en seis RFC (Request for Commentes)

¿Cómo funciona?

Cuando un 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. Si el archivo no tiene un tipo de MIME especificado el programa asociado puede suponer el tipo de archivo mediante la extensión del mismo.

¿Cómo lo hace? El navegador solicita la página web y el servidor antes de transferirla confirma que la petición requerida existe y el tipo de datos que contiene. Esto último, mediante referencia al tipo MIME al que corresponde. Este diálogo, oculto al usuario, es parte de las cabeceras http, protocolo que se sigue en la web.

En ese diálogo, en las cabeceras respuestas del servidor existe el campo Content-type donde el servidor avisa del tipo de MIME de la página. Con esta información, el navegador sabe cómo debe presentar los datos que recibe.

**Tres** lugares distintos: el servidor web, la propia página web y el navegador.

* El servidor debe estar capacitado y habilitado para manejar diversos tipos MIME.
* En el código de la página web se referencia tipos MIME constantemente en etiquetas *link, script, object, form, meta, etc.*
* El enlace a un archivo hoja de estilo css.
* El enlace a un archivo código javascript.
* Con las etiquetas meta podemos hacer que la página participe en el diálogo servidor-cliente, especificando datos MIME.
* 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 de datos, informar que tipos MIME puede aceptar la cabecera http\_accept.

## 1.4.1.- Configurar el servidor para enviar los tipos MIME correctos

En un servidor web podemos especificar el tipo MIME por defecto para aquellos archivos que el servidor no pueda identificar automáticamente como pertenecientes a un tipo concreto, esto es, para aquellos los cuales no se resuelven según su extensión.

Dos directivas: DefaultType y ForceType

* DefaultType asigna la cabecera Content-Type a cualquier archivo cuya MIME no pueda determinarse desde la extensión del archivo.
* ForceType hace que todos los ficheros cuyos nombres tengan una equivalencia con lo que se especifique sean servidos como contenido del tipo MIME que se establezca.

Ejemplos:

* DefaultType text/plain : Esto significa que cuando el navegador web solicita y recibe ese archivo como respuesta, desplegará el contenido como un archivo de texto.
* DefaultType text/html : Desplegará el contenido como un archivo HTML.
* ForceType image/gif : Desplegará el contenido como un archivo de imagen gif.
* ForceType video/mp4 : Desplegará el contenido como un archivo de vídeo mp4.

En el servidor web Apache existe el archivo **/etc/apache2/mods-available/mime.conf** donde encontrarás una referencia al archivo **/etc/mime.types** el cual contiene la lista de tipos MIME reconocidos por el servidor.

## 2.- Servidores virtuales (VirtualHosts). Creación, configuración y utilización.

Cada servidor virtual es único e independiente de los demás, pero cada servidor virtual que no esté incluido heredará la configuración principal: apache2conf.

Existen dos tipos de servidores virtuales: basados en nombres de dominio del servidor y basados en IP. La diferencia radica en el dato que usa el servidor Apache para enviar las peticiones a uno u otro servidor virtual.

El proceso de creación de un servidor virtual consiste en crear un fichero de configuración en el directorio **/etc/apache2/sites-available.**

Para activar o desactivar los servidores virtuales se usan estos comandos:

* **A2ensite**: habilita el servidor virtual, crea un enlace simbólico desde **/etc/apache2/sites-available** a **/etc/apache2/sites-enabled.**
* A2dissite: deshabilita un servidor virtual. Elimina el enlace simbólico de **/etc/apache2/sites-available** a **/etc/apache2/sites-enabled.**

Linux: el archivo que se debe modificar para la resolución de nombres de dominio en direcciones IP es **/etc/hosts.**

Windows: el archivo que se debe modificar es **C:\Windows\System32\drivers\etc\|hosts.**

## 2.1.- Servidores virtuales basados en nombre.

Ejercicio: gatitos.net

1. Creamos el fichero gatitos.net.conf en la ruta **/etc/apache2/sites-available**
2. Añadimos la configuración para un servidor virtual.
3. **<VirtualHost IP\_Servidor\_Web:puerto>** : IP define el servidor web donde se aloja el sitio web. \* el servidor está escuchando en todos los interfaces que se haya definido en el servidor. Puerto por defecto 80 definido en la directiva **Listen**. Se puede usar varias.
4. **DocumentRoot**: ruta donde está alojada la página web en el servidor.
5. **ServerName**: indica el nombre del dominio que aparece en el URL de acceso al sitio web desde el navegador.
6. **ServerAlias**: permite asociar el servidor virtual a otros nombres de dominio.
7. Descargamos el archivo y habilitamos el sitio web.
8. Y por último comprobamos con un navegador web: [www.gatitos.net](http://www.gatitos.net)

## 2.2.- Servidores virtuales basados en IP.

Ejercicio: buscaminas.net

Ahora vamos a crear un nuevo servidor virtual basado en la IP a la que está asociado en la configuración. La directiva VirtualHost va a especificar la IP de uno de los interfaces con los que está configurado el servidor.

Es más difícil de mantener si las IP del servidor web se modifican con cierta frecuencia.

Para el despliegue seguimos el procedimiento similar utilizado para el servidor virtual basado en nombre.

Pero en vez de un asterisco colocaremos la dirección IP de nuestro servidor. Dicha configuración permite al servidor Apache redirigir todas las peticiones que llegan hacia esa dirección IP a este servidor virtual.

Copiamos y subimos el fichero de buscaminas al servidor.

Habilitamos el sitio y damos los permisos necesarios.

Comprobamos con un buscador el sitio dirigiéndonos a: [www.buscaminas.net](http://www.buscaminas.net)

## 2.3.- Directivas de configuración

Vamos a presentar algunas directivas de configuración adicionales que se pueden usar en a definición de los servidores web.

* **Directory**: se utiliza para aplicar configuraciones específicas a un directorio en el servidor web. Esta directiva permite definir reglas y restricciones para ese directorio, como opciones de autorización, límites de acceso, configuraciones de seguimiento de enlaces simbólicos y más.
* **DirectoryIndex**. Especifica el nombre del archivo que se debe buscar cuando se accede a un directorio. Por lo general, se establece en index.html, index.php, u otros archivos de índice.
* **Options**. Establecer varias opciones de configuración para un directorio o un servidor virtual específico. Estas opciones controlan el comportamiento y las características del servidor web para las solicitudes que se manejan en ese contexto. La directiva Options puede tener múltiples valores que se pueden combinar para personalizar la configuración.
  + **Indexes**: Permite o prohíbe la generación automática de listados de directorios cuando no se encuentra un archivo de índice en unndirectorio. Si se habilita, se mostrará una lista de archivos en el directorio si no hay un archivo de índice presente.
  + **FollowSymLinks**: Permite la navegación simbólica, lo que permite a los usuarios seguir enlaces simbólicos dentro del sistema de archivos. Esto habilita la capacidad de seguir enlaces simbólicos para acceder a recursos en otros directorios.
  + **MultiViews**: Habilita la búsqueda de archivos con nombres similares si no se encuentra un archivo con el nombre exacto en una solicitud. Por ejemplo, si se solicita "archivo" y no se encuentra, Apache buscará "archivo.html" o "archivo.php", si están presentes.
* **AllowOverride**. controla qué configuraciones se pueden anular o sobrescribir mediante archivos ` .htaccess` en un directorio específico. Los archivos `.htaccess` son archivos de configuración que permiten a los administradores de sitios web modificar la configuración de Apacheen un directorio determinado, lo que brinda flexibilidad para ajustar la configuración en un nivel más local. La directiva `AllowOverride` acepta varios valores para determinar qué configuraciones pueden ser anuladas en un directorio. Algunos de los valores más comunes incluyen:
  + **None**: No se permiten anulaciones en ningún nivel. Los archivos `.htaccess` se ignoran completamente.
  + **All**: Se permiten todas las anulaciones. Los archivos `.htaccess` pueden sobrescribir prácticamente todas las configuraciones de Apache, incluyendo la mayoría de las opciones de configuración.
  + **AuthConfig**: Permite la anulación de configuraciones relacionadas con la autenticación y la autorización de usuarios, como AuthType, AuthName, AuthUserFile, etc.
  + **FileInfo**: Permite la anulación de configuraciones relacionadas con la información del archivo, como AddType, AddOutputFilter, DefaultType, etc.
  + **Indexes**: Permite la anulación de configuraciones relacionadas con la generación de listados de directorios, como Options +Indexes, IndexOptions, etc.
  + **Limit**: Permite la anulación de configuraciones relacionadas con la limitación de acceso a través de directivas como Order, Deny,Allow, etc.
  + **Options**: Permite la anulación de configuraciones relacionadas con las opciones del servidor web, como Options, DirectoryIndex,AddHandler, etc.

## 3.- Módulos

La importancia de un servidor web radica en su: estabilidad, disponibilidad y escalabilidad. Es muy importante poder dotar al servidor web de nuevas funcionalidades de forma sencilla, así como del mismo modo quitárselas.

Dos comandos:

* **A2enmod**: para habilitar un módulo en apache. Los ficheros de configuración están en **/etc/apache2/mods-available/** y se crea un enlace simbólico desde **/etc/apache2/mods-enabled/**
* **A2dismod**: deshabilita un módulo de apache. Los ficheros de configuración de los módulos están en **/etc/apache2/mods-available/**
* Si no se dispone de estos comandos simplemente tienes que crear enlaces simbólicos desde **/etc/apache2/mods-enabled/** hasta **/etc/apache2/mods-available/**

## 3.1.- Operaciones sobre módulos

Los módulos de Apache puedes instalarlos, desinstalarlos, habilitarlos o deshabilitarlos, así puedes tener un módulo instalado, pero no habilitado.



Para habilitar un módulo Apache, con el comando **a2enmod**, para deshabilitarlo con el comando **a2dismod**.

Para listar los módulos estáticos que se han cargado al compilar el servidor ejecutando el comando: **sudo apachectl -l**

Para consultar los módulos activados dinámicamente consultando el directorio: **/etc/apache2/mods-enabled.**

Una vez habilitado o deshabilitado los módulos Apache sólo reconocerá los cambios cuando recargues su configuración: **systemctl restart apache2.**

Para recargar la configuración: **systemctl reload apache2.**

Si no dispones de los comandos **a2enmod** y **a2dismod** puedes habilitar y deshabilitar módulos Apache creando los enlaces simbólicos correspondientes desde **/etc/apache2/mods-enabled/** hasta **/etc/apache2/mods-available/.**

Creas el enlace simbólico para habilitar el módulo: (siempre dos)

ln -s ../mods-enabled/ssl.conf ./ssl.conf

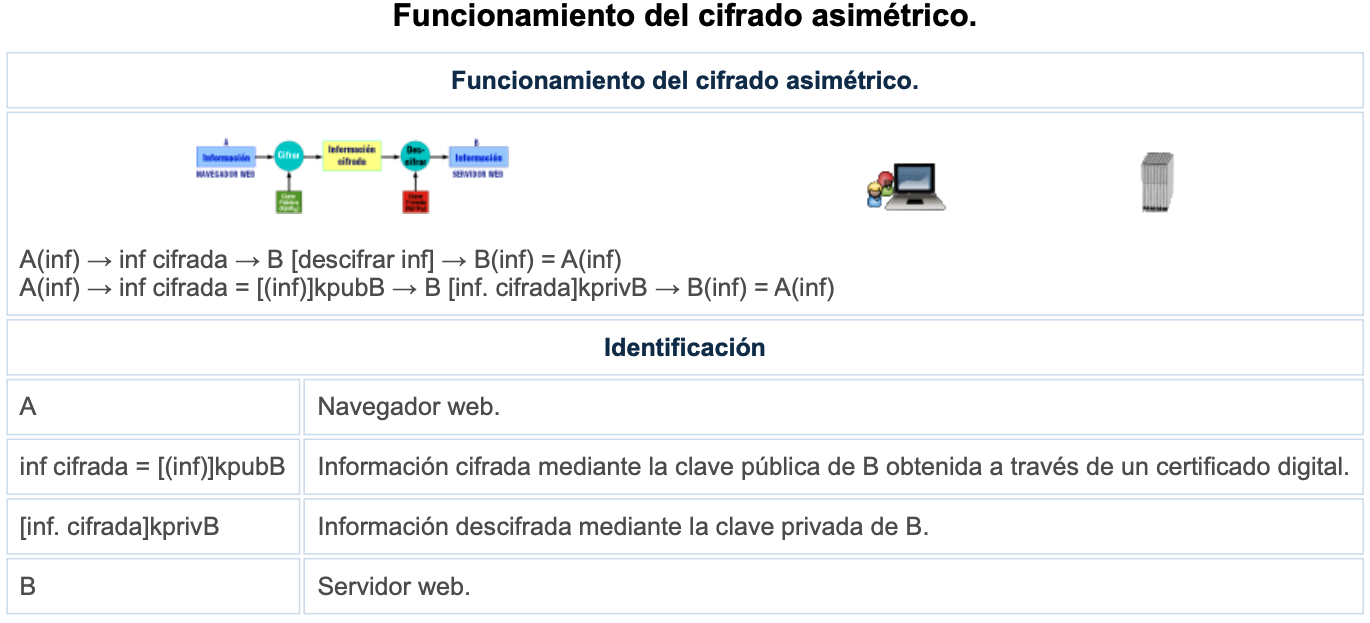
ln -s ../mods-enabled/ssl.load ./ssl.load

## 4.- Acceso a carpetas seguras

Existe la posibilidad de asegurar la información sensible que viaja entre el navegador y el servidor, pero esto repercutirá en un mayor consumo de recursos del servidor.

El cifrado de clave pública o asimétrico: clave pública y clave privada. La kpub interesa publicarla para que llegue a ser conocida por cualquiera, la kpriv no interesa que nadie la posea, solo el propietario de la misma.

En el cifrado asimétrico podemos estar hablando de individuos o de máquinas, en nuestro caso hablamos de máquinas y de flujo de información entre el navegador (A) y el servidor web (B).



En Internet existen autoridades de certificación **(CA ó AC)** que aseguran la autenticidad del certificado digital, y así la autenticidad de B, como: Verisign y Thawte. Pero, como ya hemos comentado el Servidor Web Apache permite ser **CA**, por lo que tienes la posibilidad de crear tus propios certificados digitales, ahora bien, ¿el navegador web(A) confiará en estos certificados? Pues, en principio no, por lo que los navegadores avisarán que la página a la cuál intentas acceder en el servidor web representa un peligro de seguridad, ya que no existe en su lista de autoridades certificadoras de confianza.

## 4.1.- Certificados digitales, AC y PKI

**Un certificado digital** es un documento electrónico que asocia una clave pública con la identidad de su propietario, individuo o máquina y emitido por una autoridad certificadora.

Estas certifican el documento de asociación entre clave pública e identidad de un individuo o máquina firmando dicho documento con su clave privada, mediante firma digital.

Dos extremos de una comunicación puedan confiar directamente entre sí, si ambos tienen relación con una tercera parte, que da fe de la fiabilidad de los dos, aunque en la práctica te suele interesar solamente la fiabilidad del servidor, para saber que te conectas con el servidor que quieres y no con otro servidor. Así la necesidad de una Tercera Parte Confiable (TPC o TTP Trusted Third Party) es fundamental en cualquier entorno de clave pública.

La TPC que se encarga de la firma digital de los certificados de los usuarios de un entorno de clave pública se conoce con el nombre de Autoridad de Certificación (AC).

El modelo de confianza basado en Terceras partes Confiables es la base de definición de las Infraestructuras de Clave Pública (ICP o PKI Public Key Infrastructures).

Una infraestructura de Clave Pública es un conjunto de protocolos, servicios y estándares que soportan aplicaciones basadas en criptografía de clave pública.

Servicios de ICP o PKI:

* **Registro de claves:** emisión de un nuevo certificado para una clave pública.
* **Revocación de certificados:** cancelación de un certificado previamente emitido.
* **Selección de claves:** publicación de la clave pública de los usuarios.
* **Evaluación de la confianza:** determinación sobre si un certificado es válido y qué operaciones están permitidas para dicho certificado.
* **Recuperación de claves:** posibilidad de recuperar las claves de un usuario.

Las ICP o PKI están compuestas por:

* **Autoridad de certificación (AC):** realiza la firma de los certificados con su clave privada y gestiona la lista de certificados revocados.
* **Autoridad de Registro (AR):** es la interfaz hacia el mundo exterior. Recibe las solicitudes de los certificados y revocaciones, comprueba los datos de los sujetos que hacen las peticiones y traslada los certificados y revocaciones a la AC par que los firme.

El más común de los formatos de certificados digitales es el estándar UIT-T X.509 y contiene una serie de campos:

* Versión
* Nº de serie del certificado
* Identificador del algoritmo de firmado
* Nombre del emisor
* Periodo de validez
* Nombre del sujeto
* Información de clave publica del sujeto
* Identificador único del emisor
* Identificador único del sujeto y extensiones.

## 4.2.- Módulo SSL para Apache

El método de cifrado **SSL/TLS** utiliza un método de cifrado de clave pública (cifrado asimétrico) para la autenticación del servidor.

El módulo SSL es quien permite cifrar la información entre navegador y servidor web. **A2enmod** ssl para activarlo.

Este módulo proporciona **SSL v2/v3 y TLS v1** para el Servidor Apache Http, y se basa en **OpenSsl** para proporcionar el motor de la criptografía.

**TLS** (Transport Layer Security) es un protocolo de seguridad diseñado para garantizar la confidencialidad, integridad y autenticación de la comunicación en Internet.

|  |
| --- |
| **1. Inicio de la comunicación:**  Cuando un cliente (como un navegador web) desea establecer una conexión segura con un servidor (como un sitio web), se inicia el  proceso TLS. El cliente y el servidor negocian el uso de TLS y comienzan un apretón de manos (handshake).  **2. Negociación de protocolo y parámetros de cifrado:**  Durante el apretón de manos, el cliente y el servidor acuerdan qué versión de TLS utilizarán y los algoritmos de cifrado que se emplearán para proteger la comunicación. Esto incluye algoritmos para el cifrado simétrico (como AES), algoritmos de intercambio de claves asimétricas (como RSA o Diffie-Hellman), y algoritmos de resumen (hash) para garantizar la integridad de los datos.  **3. Intercambio de claves:**  El servidor envía su clave pública al cliente como parte del apretón de manos. El cliente utiliza esta clave para cifrar una clave de sesión que será compartida entre el cliente y el servidor. Esto permite que la comunicación se cifre de manera segura sin necesidad de compartir la clave de sesión a través de la red.  **4. Autenticación del servidor:**  El servidor presenta su certificado digital al cliente, que contiene su clave pública y está firmado por una entidad de certificación confiable (CA). El cliente verifica la autenticidad del servidor asegurándose de que el certificado sea válido y provenga de una fuente confiable. Esto previene los ataques de intermediarios maliciosos.  **5. Cifrado y autenticación de la comunicación:**  Una vez que se ha completado el apretón de manos y se ha compartido una clave de sesión segura, todos los datos intercambiados entre el cliente y el servidor se cifran con esa clave de sesión. También se utilizan códigos de autenticación de mensajes (MAC) para garantizar la integridad de los datos.  **6. Comunicación segura:**  A partir de este punto, la comunicación entre el cliente y el servidor está cifrada y protegida de escuchas no autorizadas y manipulación de datos. Los datos se envían de manera segura a través de Internet.  **7. Cierre de la conexión segura:**  Al finalizar la comunicación, se cierra la conexión de manera segura, lo que evita cualquier intento de interceptación o modificación de los datos restantes. |

**TLS** es esencial para garantizar la seguridad de la comunicación en Internet, y es ampliamente utilizado en protocolos como HTTPS (HTTP seguro) para proteger la transmisión de datos en la web, así como en otros servicios y aplicaciones que requieren seguridad en la comunicación.

## 4.3.- Crear un servidor virtual seguro en Apache.

En Ubuntu, Apache posee por defecto en su instalación el fichero **/etc/apache2/sites-available/default-ssl** (0.03 KB), que contiene la configuración por defecto de SSL. En su contenido podemos ver las siguientes líneas:

**SSLEngine on :** Activa o desactiva SSL

**SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem :** Certificado digital del propio servidor Apache

**SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key :** Clave privada del servidor Apache.

Cuando activaste el módulo ssl, mediante el comando a2enmod ssl permitiste que Apache atienda el protocolo SSL.

Lo que indica que el certificado digital del servidor no viene firmado por una **AC** contenida en la lista que posee el navegador, sino por el mismo Apache.

Ahora tienes dos opciones: Confiar en el certificado o no.

* Si confías haces clic en Entiendo los riesgos y Añadir excepción...Una vez que confías puedes, antes de Confirmar excepción de seguridad, ver el contenido del certificado. Si estás de acuerdo la comunicación se establece y la información viaja cifrada.
* Si no confías haces clic en !Sácame de aquí!

Hay que tener en cuenta que la negociación SSL es **dependiente** totalmente **de la IP,** no del nombre del sitio web, así no puedes servir distintos certificados en una misma IP.

## 4.3.1.- Crear un servidor virtual seguro en Apache

Para crear el certificado autofirmado haremos lo siguiente:

* Instalación del paquete openssl: (Este paso no es necesario en Ubuntu 22.04 si se ha instalado openssl con el sistema operativo)
  + sudo apt-get install openssl
* Genera la clave privada del servidor para generar su certificado y autofirmarlo.
  + sudo mkdir /etc/apache2/tus-ssl/
  + cd /etc/apache2/tus-ssl/
  + sudo openssl genrsa -out seguro.key 4096
* Genera la solicitud de certificado digital del servidor.
  + sudo openssl req -new -key seguro.key -out seguro.csr
* Firma el certificado con su misma clave privada. Para que el servidor fuera seguro habría que firmarlo con la clave privada de una autoridad certificadora (CA) reconocida.
  + sudo openssl x509 -req -days 365 -in seguro.csr -signkey seguro.key -out seguro.crt
* Editar la configuración SSL por defecto en el archivo /etc/apache2/sites-available/default-ssl.conf para indicar el certificado del servidor y su respectiva clave privada asignando los siguientes valores a los parámetros:
  + SSLEngine On
  + SSLCertificateFile /etc/apache2/tus-ssl/seguro.crt
  + SSLCertificateKeyFile /etc/apache2/tus-ssl/seguro.key
* Asegúrate que el fichero /etc/apache2/ports.conf incluya el valor Listen 443
* Si no lo has hecho previamente, habilita el módulo ssl y la configuración SSL por defecto.
  + a2enmod ssl
  + a2ensite default-ssl
  + systemctl restart apache2
* Si lanzas el navegador desde tu equipo:
  + Indicar https://www.dawdistancia.net en la barra de direcciones.

## 4.4.- Comprobar el acceso seguro al servidor.

A continuación, una serie de actuaciones que te servirán para comprobar que el acceso seguro que estableces con el servidor seguro es el esperado:

|  |
| --- |
| * Siempre que te conectes mediante SSL a un página web y el certificado no sea admitido, debes ver los campos descriptivos del certificado antes de generar la excepción que te permita visitar la página. * Debes comprobar en el certificado si la página a la que intentas acceder es la misma que dice el certificado. * Típicamente en los navegadores, si no está configurado lo contrario, cuando accedes mediante cifrado SSL a una página web puedes ver en algún lugar del mismo un icono: un candado, por lo cual debes verificar su existencia para asegurarte que estás accediendo por https. * Incluso si el certificado pertenece a alguna AC que el navegador posee en su lista de AC puedes ver en la barra de direcciones indicaciones del tipo de certificado con el que se cifra la comunicación. * Revisar la lista de certificados admitidos que posee tu navegador. En **Firefox**, versión > 3.x , donde x es el número de revisión de la versión 3, puedes verlas dirigiéndote por las pestañas a: **Editar → Preferencias → Avanzado → Cifrado → Ver certificados** * Revisar la lista de revocaciones que posee tu navegador. En **Firefox**, versión > 3.x , donde x es el número de revisión de la versión 3, puedes verlas dirigiéndote por las pestañas a: **Editar → Preferencias → Avanzado → Cifrado → Listas de revocación** |

Puedes Importar/Exportar certificados en los navegadores, con lo cual los puedes llevar a cualquier máquina. Esto es muy útil cuando necesitas un certificado personal en máquinas distintas.

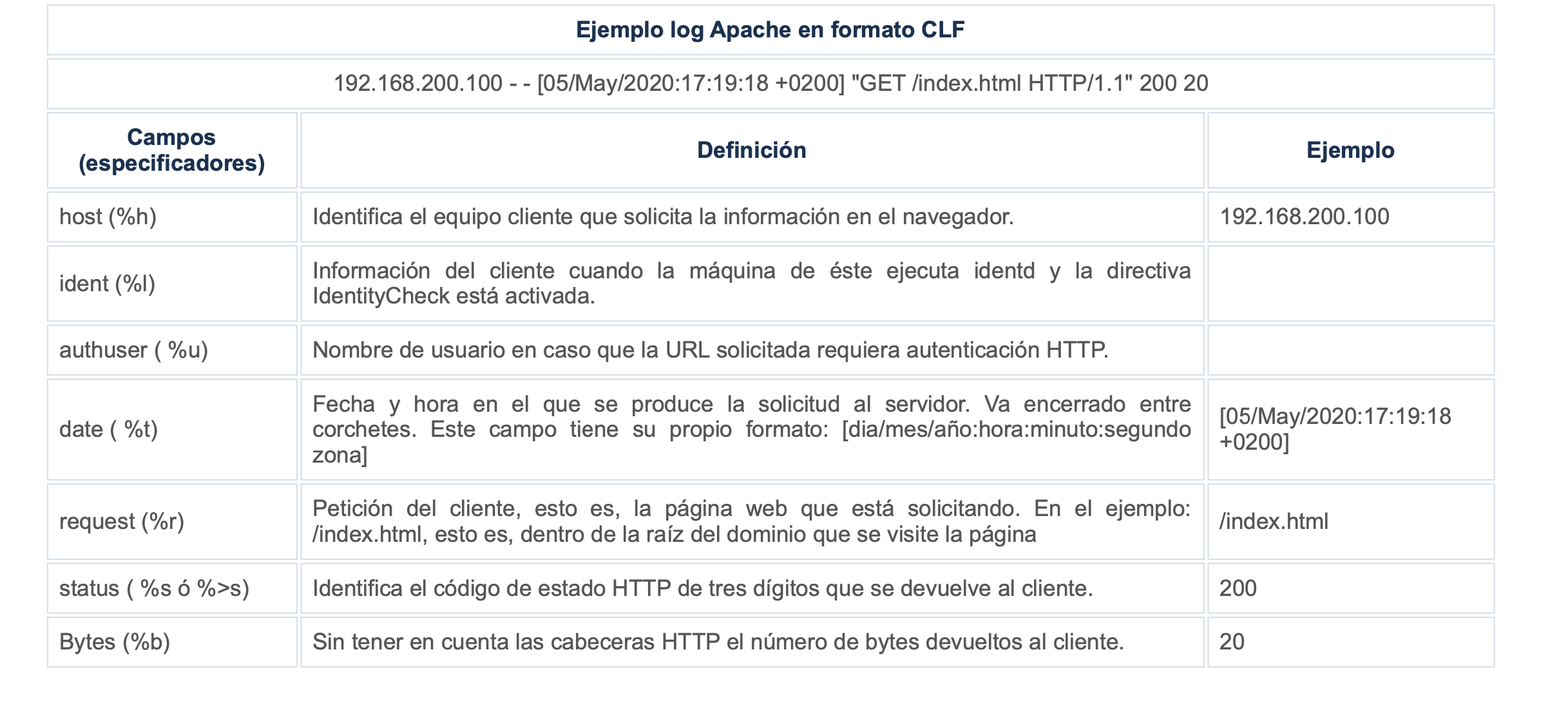
<https://www.youtube.com/watch?v=a99Qorfotv4&list=PLV5qVSPJeeG0SiCGHmMhcp8PG46HX1Wpo&index=1>

## 5.- Monitorización del acceso: Archivos de registro (logs)

Tan importante como es configurar un servidor web lo es mantener y comprobar su correcto funcionamiento, y para ello debes ayudarte de los logs o archivos de registro que te permite revisar y estudiar su funcionamiento.

Los archivos de registro que guardan la información correspondiente a las conexiones con el servidor. Se guarda en formato CLF (Common Logon Format) por defecto. Esta es una especificación utilizada por los servidores web para hacer que el análisis de registros entre servidores sea mucho más sencillo.

Cada línea identifica una solicitud al servidor web. Esta línea contiene varios campos separados con espacios. Cada campo sin valor es identificado con un guión “-“.



Cada campo tiene su especificador, el cual se emplea en las directivas de Apache para indicar que campo queremos registrar.

## 5.1.- Directivas para archivos de registro.

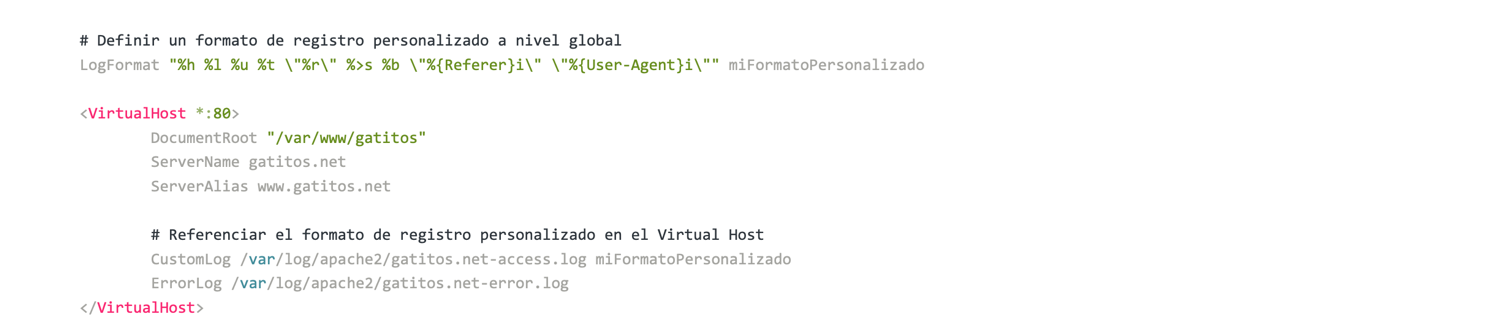
Configuración general.





Para comprobar en tiempo real los logs que se guardan: **tail -f nombre\_archivo\_de\_registro.log**

Para configurar los logs asociados al servidor virtual editamos el fichero de configuración en el directorio **sites-available.**



|  |
| --- |
| En este ejemplo, el formato miFormatoPersonalizado incluye:  %h: La dirección IP del cliente.  %l: El identificador remoto (generalmente '-').  %u: El userID del cliente si la autenticación HTTP está en uso.  %t: La fecha y hora de la solicitud.  \"%r\": La línea de solicitud exacta recibida del cliente.  %>s: El estado de la respuesta HTTP.  %b: El tamaño del objeto devuelto al cliente, no incluye encabezados de respuesta HTTP.  \"%{Referer}i\": La página que refirió al cliente al recurso actual.  \"%{User-Agent}i\": El agente de usuario del cliente. |

## 5.2.- Rotación de los archivos de registro (I).

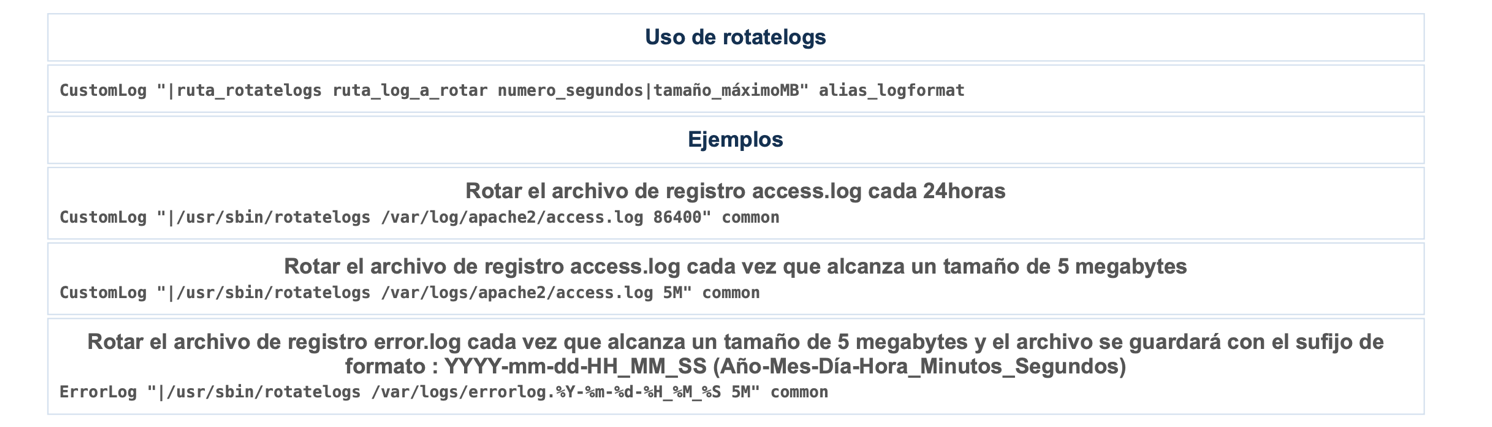
A medida que pasa el tiempo los archivos de registro van incrementando su tamaño, y para ello debe existir una política de mantenimiento de registros que no consuma recursos.

Es conveniente rotar los archivos de registro esto es, depurarlos, comprimirlos y gurardalos.

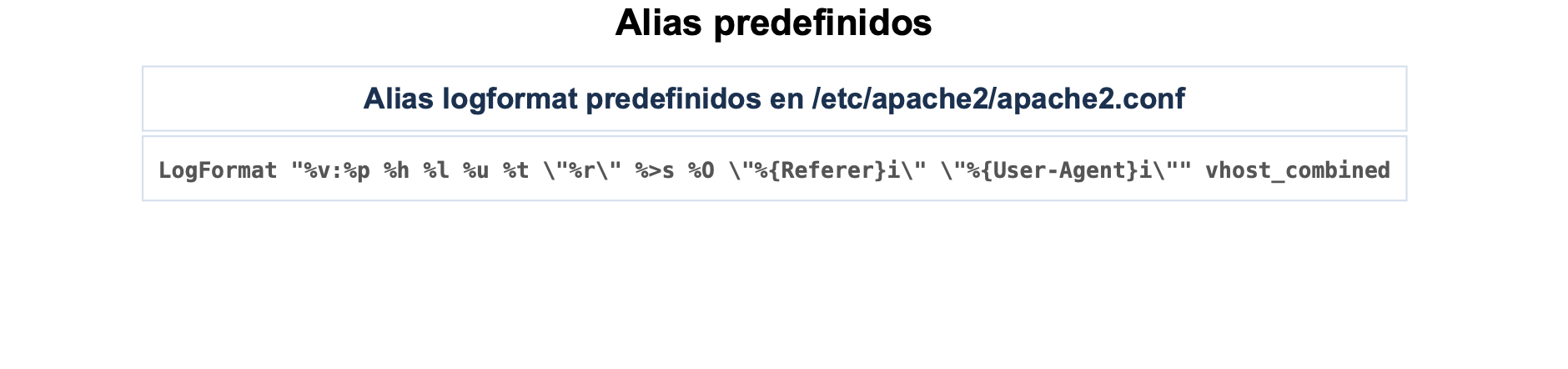
Dos opciones:

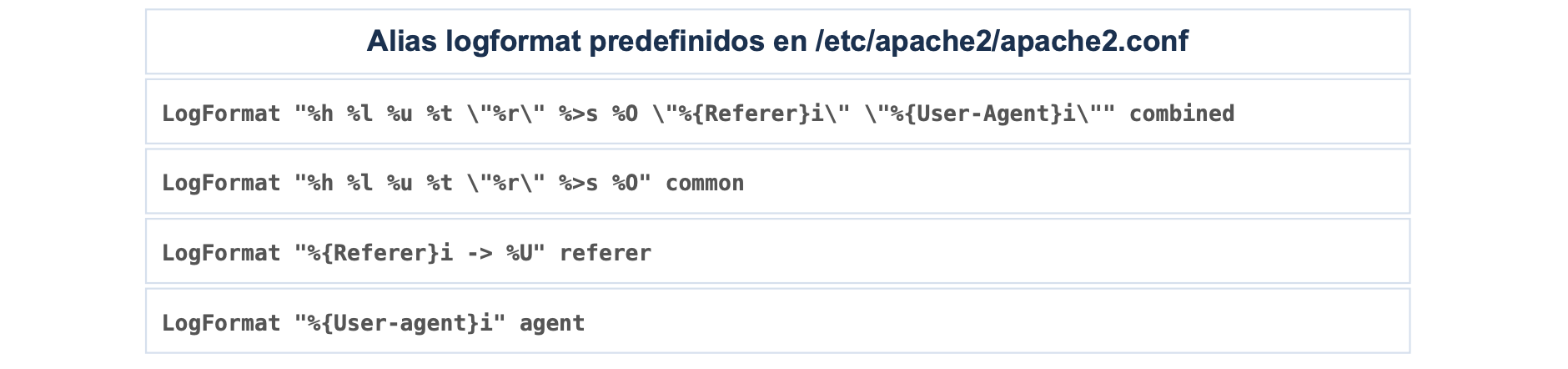
* Rotatelogs
* Logrotate

La información recopilada en los ficheros log se debe conservar al menos durante 1 año por eventuales necesidades legales (rotarlos y comprimirlos).



El alias **logformat** es muy interesante, porque permite definir un grupo de modificadores en una palabra, de tal forma que incorporando esa palabra en la directiva log correspondiente estás activando todo un grupo de modificadores.

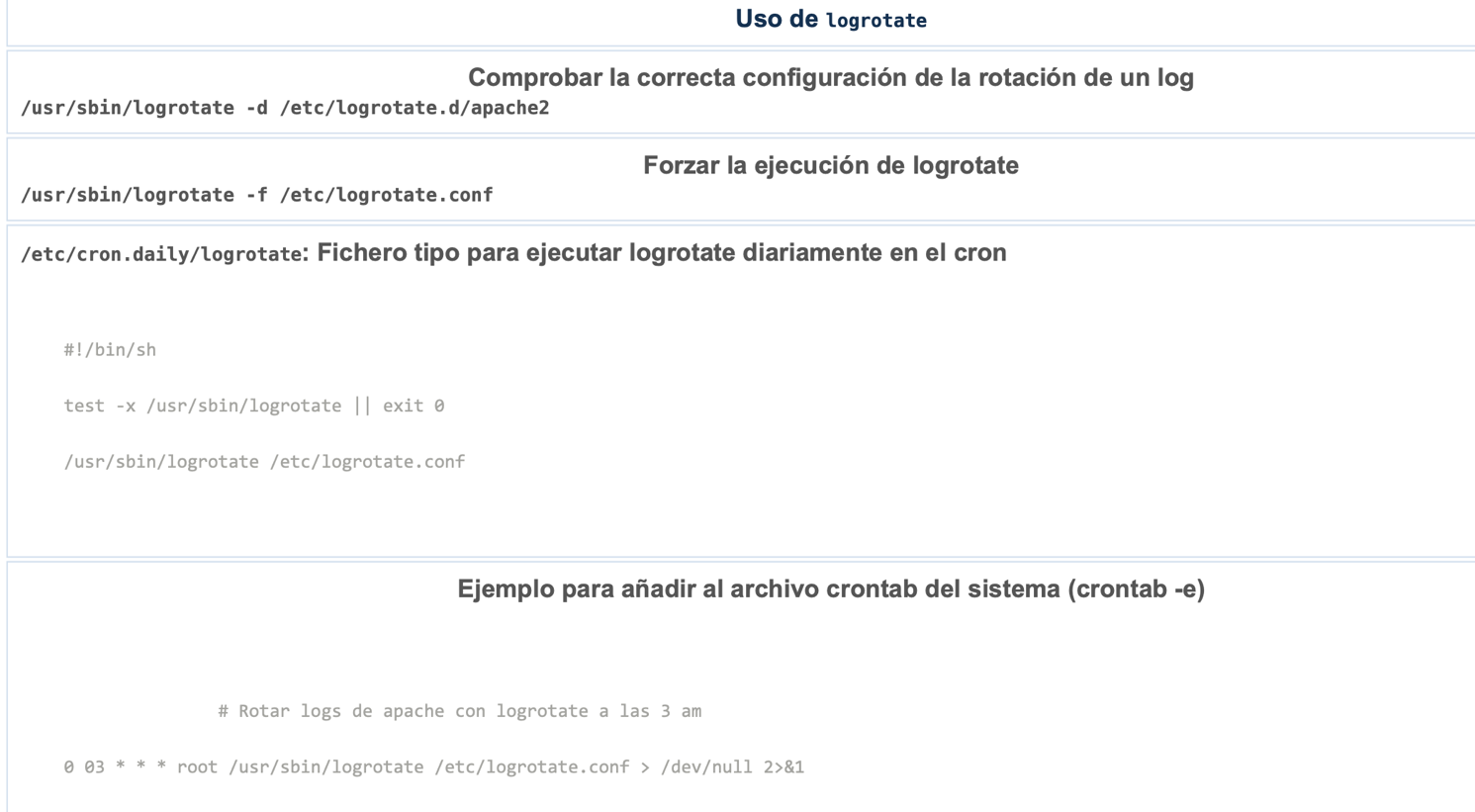




## 5.2.1.- Rotación de los archivos de registro (II)

El programa **logrotate** rota, comprime y envía archivos de registro a diario, semanalmente, mensualmente o según el tamaño del archivo. Suele emplearse en una tarea diaria del **cron**.

* **/etc/logrotate.conf:** define los parámetros globales.
* **/etc/logrotate.d/apache2:** define para apache2 el rotado de logs, todos aquellos parámetros que no se encuentren aquí recogen su valor del fichero **/etc/logrotate.conf.**



## 6.- Autenticación y control de acceso.

Cuando nos autenticamos en una web suele transferirse la información de autenticación a una base de datos, que puede existir en la misma máquina que el servidor web o en otra totalmente diferente. Suelen emplearse bases de datos SQL o LDAP para la autenticación de usuarios, siendo **OpenLDAP** una de las alternativas más empleadas.

**AAA** (Autenthication, Autoritation, Accouting) + A (Auditoria).

**HTTP** proporciona un método de autenticación básico de usuarios: **basic**. Este método ante una petición del cliente al servidor cuando se solicita una URL mostrará un diálogo pidiendo usuario y contraseña. Una vez autenticado el usuario, el cliente volverá a hacer la petición al servidor pero ahora enviando el usuario y contraseña, en texto claro (sin cifrar) proporcionados en el diálogo. Es recomendable entonces si empleas este método que lo hagas combinado con conexión SSL (**HTTPS**).

Configuración:

- Crear el archivo de contraseñas: Primero, necesitas crear un archivo que contenga los nombres de usuario y contraseñas. Esto se hace con la herramienta **htpasswd**.

sudo htpasswd -c /etc/apache2/.htpasswdgatitos admin

- Configurar el archivo de configuración del servidor: Necesitas decirle a Apache qué directorio quieres proteger. Esto se puede hacer en el archivo de configuración principal de Apache (por ejemplo, **httpd.conf** o apache2.conf) o en un archivo .**htaccess** en el directorio que deseas proteger.



En la autenticación HTTP Basic es muy típico utilizar archivos .htaccess en los directorios que queremos controlar el acceso. Los archivos .htaccess en Apache son archivos de configuración distribuida que permiten gestionar la configuración de Apache a nivel de directorio. Se utilizan para modificar la configuración sin necesidad de alterar los archivos de configuración principales del servidor web. Un uso común de los archivos .htaccess es para implementar la autenticación básica, que restringe el acceso a determinados recursos en tu sitio web.

En este caso, es necesario crear un fichero .htaccess en el directorio raíz del sitio de gatitos, /var/www/gatitos con el siguiente contenido:



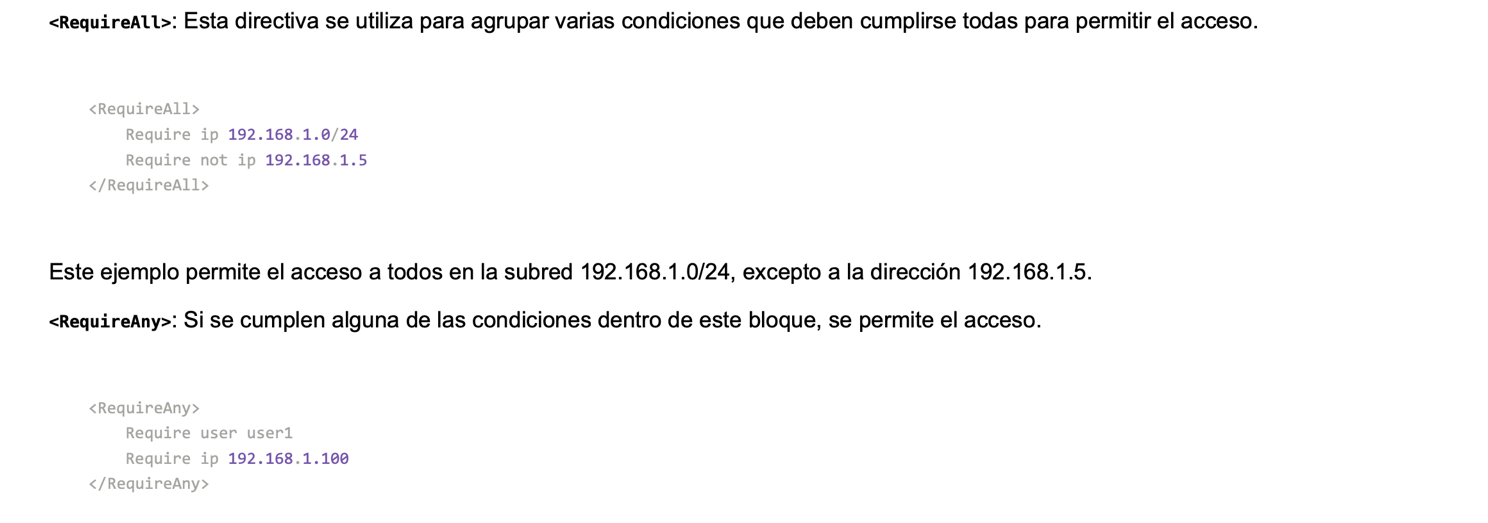
Para que Apache lea y respete las configuraciones en .**htaccess**, la directiva **AllowOverride** en el archivo de configuración principal del servidor debe estar establecida en All o al menos incluir **AuthConfig** para el directorio en cuestión. La configuración del archivo de configuración gatitos.net.conf en el directorio /etc/apache2/sites-available sería:

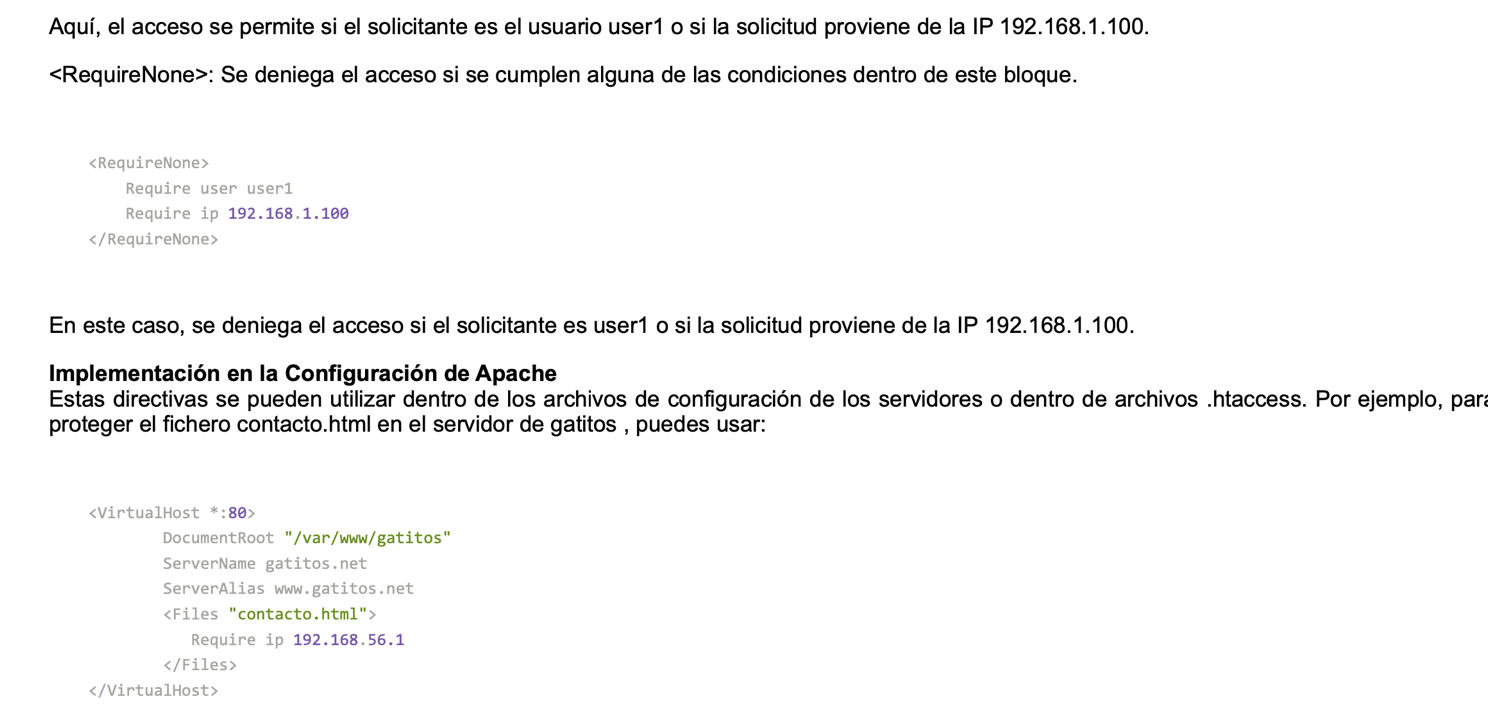


Para usar archivos .**htaccess**, necesitas tener una configuración en el servidor que permita poner directivas de autenticación en estos archivos, mediante la directiva **AllowOverride**, así: **AllowOverride** **AuthConfig.**

En Apache 2.4, las directivas de control de acceso se utilizan para restringir el acceso a los recursos del servidor web. Estas directivas se han simplificado y mejorado en comparación con las versiones anteriores de Apache.

|  |
| --- |
| **Directivas Principales de Control de Acceso**  Require: Esta directiva se usa para especificar qué usuarios, grupos o condiciones deben cumplirse para permitir el acceso. Ejemplos de uso:  Require all granted: Permite el acceso a todos.  Require all denied: Niega el acceso a todos.  Require user [nombre\_de\_usuario]: Permite el acceso solo al usuario especificado.  Require group [nombre\_del\_grupo]: Permite el acceso solo a los miembros del grupo especificado.  Require all granted: Permite el acceso a todos.  Require ip [dir\_ip]: Permite el acceso solo a peticiones provenientes de una dirección IP.  Order, Allow, Deny (Desaconsejadas en Apache 2.4): Estas directivas eran comunes en Apache 2.2, pero en Apache 2.4 se desaconseja su uso en favor de la nueva sintaxis Require. Si aún las encuentras, funcionan así:  Order: Define el orden de procesamiento de las directivas Allow y Deny.  Allow: Especifica una condición bajo la cual se permite el acceso.  Deny: Especifica una condición bajo la cual se deniega el acceso. |





## 7.- Despliegue de aplicaciones sobre servidores Web.

Las aplicaciones web necesitan de soporte php y soporte sql.

Procedimiento:

|  |
| --- |
| 1. Se descarga la aplicación.  2. Se configura para que sea visible a través del servidor web.  3. Suele traer una página de instalación que verifique si el servidor web cumple los requisitos para la instalación de la aplicación.  4. Es necesaria antes de finalizar el proceso de instalación autenticarse al servidor sql con un usuario con permisos para crear/modificar una  base de datos. Puede que previamente se tenga que crear la base de datos para que el proceso de instalación genere las tablas  necesarias en la misma.  5. Se pide un usuario y contraseña para poder acceder a la aplicación web.  6. Fin de la instalación.  En la siguiente sección se puede ver un ejemplo de despliegue de la aplicación OpenCart. (**Anexo I.- Despliegue aplicación OpenCart.**)  Se deberá tener instalado un servidor LAMP, en la unidad didáctica "Implantación de arquitecturas web", se vio la instalación del entorno Apache, MySQL y PHP, en Ubuntu.  Otra buena opción sería instalar el paquete XAMMP para GNU/Linux: |

<https://www.digitalocean.com/community/tutorials/how-to-install-mysql-on-ubuntu-20-04-es>

<https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-phpmyadmin-on-ubuntu-20-04-es>

<https://developer.wordpress.org/advanced-administration/before-install/howto-install/>

Pasos instalación de Mysql:

* sudo apt update
* sudo apt install mysql-server
* sudo mysql\_secure\_installation
* sudo mysql
* SELECT user,authentication\_string,plugin,host FROM mysql.user;
* ALTER USER 'root'@'localhost' IDENTIFIED WITH caching\_sha2\_password BY 'password';
* ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY 'password';
* FLUSH PRIVILEGES;
* SELECT user,authentication\_string,plugin,host FROM mysql.user;
* Exit
* mysql -u root -p
* CREATE USER 'dbdaw'@'localhost' IDENTIFIED BY 'password';
* GRANT ALL PRIVILEGES ON \*.\* TO 'dbdaw'@'localhost' WITH GRANT OPTION;
* Exit
* systemctl status mysql.service
* sudo mysqladmin -p -u root version

Pasos instalación de Phpmyadmin:

* sudo apt update
* sudo apt install phpmyadmin php-mbstring php-zip php-gd php-json php-curl
* sudo mysql
* mysql -u root -p
* UNINSTALL COMPONENT "file://component\_validate\_password";
* Exit
* sudo apt install phpMyAdmin
* mysql > INSTALL COMPONENT "file://component\_validate\_password";
* sudo phpenmod mbstring
* sudo systemctl restart apache2
* <https://your_domain_or_IP/phpmyadmin>

Pasos de instalación de Wordpress:

* Descargar el .zip de wordpress.org
* Subirlo al servidor y colocarlo en la ruta /var/www
* Crear el sitio web
* Editar el archivo hosts para www.wordpress.net
* rename wp-config-sample.php to wp-config.php
* Editar wp-config.php con los datos
* Instalar wordpress con el asitente: http://www.wordpress.net/wp-admin/install.php
* Listo