RESUMEN TEMA 2. Configuración y administración de servidores Web.

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

## 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.