El servidor web Apache

1. Servidor Web Apache.

El servidor web Apache es una de las aplicaciones estrella del mundo Linux. Es el servidor web más implantado entre los distintos servidores que ofertan servicios web en Internet.

Entre las características más significativas destacamos:

- Es modular
- Permite crear servidores virtuales

- Permite crear servidores seguros https
- Permite crear sitios privados

En este documento haremos uso de éstas y otras características de Apache.

1.1 Organización del sitio web

Se pretende implantar un servidor web. La organización que realizaremos de nuestro servidor Apache, será la clásica en los sistemas Unix:

- la **página web** de la **intranet** se almacenará en la **carpeta raíz** del servidor web
- las **páginas de los usuarios** se almacenarán en la **carpeta home** de cada usuario
- para albergar las **páginas web** de los distintos **departamentos** del centro, lo más práctico es **crear nuevos usuarios** con el nombre del departamento.

Espacio web para la Intranet

******NOTA: EL DIRECTORIO RAIZ ES /var/www/html/**

Por defecto, la **carpeta raíz del servidor** web es la carpeta **/var/www/html/.** Todos los documentos que se encuentren dentro de la carpeta raíz del servidor web, serán accesibles vía web. Dentro del raíz de documentos crearemos la página web de nuestra intranet.

Carpeta principal del servidor web (**DocumentRoot)**:

- Carpeta raíz del servidor web: **/var/www/html/**
- Acceso a la web principal: **http://ip-del-servidor ó http://nombre-del-servidor**

Para acceder vía web a la página almacenada en la carpeta raíz del servidor, **desde un navegador** debemos acceder directamente con la dirección IP a: **http://ip-del-servidor** o bien utilizando el nombre del mismo **si tenemos el DNS** funcionando: **http://nombre-del-servidor.** Si **no tenemos el DNS** funcionando, podemos añadir el nombre y la IP en **/etc/hosts** para resolver localmente.

Espacio web seguro

Además crearemos un **sitio web virtual** seguro en el servidor web Apache para poder tener **acceso vía SSL** a contenidos que deseamos que sean seguros, es decir, accesibles

en el navegador mediante el **protocolo "https"**, será la carpeta **/var/www/websegura/**

Carpeta web segura

Carpeta web segura: **/var/www/html/websegura/**

Acceso a la web segura: **https://ip-del-servidor/websegura/**

Dentro de esta estructura la mayoría de los contenidos serán públicos y cualquier usuario podrá acceder a ellos. Sin embargo, algunas de las carpetas serán privadas y solo se tendrá acceso a ellas identificándose con nombre de usuario y contraseña.

1.2 Instalación de Apache2

Antes de instalar nada: ¿dónde lo instalo?. Los servicios ftp y web van unidos. Es decir **instala el servidor web Apache en la misma máquina donde instalaste el servicio ftp.**

Instalación desde un servidor Ubuntu:

// Instalación de apache2

$sudo apt-get install apache2

Con lo cual se instalarán los archivos necesarios para que funcione nuestro servidor web. Se instalará apache v2.

Desde un cliente podemos acceder con el navegador a http://ip-del-servidor ó http://nombre-del-servidor (siempre que tengamos configurado nuestro servidor de nombres **DNS**) y nos confirma que el servidor apache funciona:

It works!

Pruébalo.

Para que los PCs de la red local sepan que ***www.sercamp.org*** es nuestro servidor web, debemos crear una entrada ***'www'*** hacia su dirección IP en el servidor ***DNS.***

Podemos personalizar nuestra página modificando el archivo ***index.html*** que hay dentro de la carpeta ***/var/www/html/*** del servidor web:

$sudo nano /var/www//html/index.html

Fíjate en la estructura que usa el HTML. Lo que pongas entre las etiquetas <body> y </body> es lo que aparecerá en el documento.

Para que aparezcan correctamente los signos de acentuación, accediendo al fichero ***"index.html"*** se introduce entre las etiquetas de "head" el siguiente código:

<META http-equiv="Content-Type"content="text/html; charset=UTF-8">

Nota: El usuario no tiene porqué (no debe) dedicarse a crear/modificar los ficheros de la página web en el servidor, como acabamos de hacer. Lo que se hace es que el usuario crea la página web en su máquina de cliente con entorno gráfico y herramientas de ayuda a a creación de páginas web. Por el otro lado el servidor web también será normalmente un servidor ftp. El usuario con algún programa gráfico de copia de archivos cliente ftp, por ejemplo FileZilla (disponible en linux y windows) se conecta a su carpeta personal y copia los archivos necesarios al servidor. Ya veremos esto en una tarea.

1.3 Configuración de Apache

Los archivos de configuración de apache2 se encuentran en la carpeta ***/etc/apache2.***

El archivo principal de configuración es ***/etc/apache2/apache2.conf.*** Antes de realizar cualquier cambio en este archivo, es conveniente realizar una copia de seguridad del mismo ya que si Apache encuentra algún error en el archivo de configuración, no arrancará. Se pueden configurar infinidad de parámetros. Los valores que hay por defecto funcionan bien en la mayoría de situaciones. Las líneas que empiezan por #, se consideran comentarios y no se procesan por Apache.

Arranque y parada del servidor web apache

El servidor web apache2, al igual que todos los servicios en Ubuntu, dispone de un script de arranque y parada en la carpeta /etc/init.d o bien como servicio según la versión

Para Ubuntu (14.04)

// Arrancar o reiniciar el servidor apache2: $sudo /etc/init.d/apache2 restart

// Parar el servidor Apache: $sudo /etc/init.d/apache2 stop

Para Ubuntu (16.04)

$sudo service apache2 restart

$sudo service apache2 stop

O bien con scripts de arranque

sudo systemctl start apache2.service

2. Servidores virtuales

Vamos a preparar un servidor Apache2 para que funcionase con varios dominios diferentes desde la misma IP, es decir, que **un único servidor Apache aloje y resuelva 2 hosts del mismo dominio, por separado**. Es lo que se conoce como **configurar VirtualHosts basados en nombres.**

Una vez instalado Apache, toda la configuración de Apache está en ***/etc/apache2/.***

Mirando el contenido de este directorio, observamos distintos archivos y directorios, entre los cuales están: ***sites-available y sites-enabled.***

En ***sites-available*** tendremos los VirtualHosts disponibles (**pero no "activados"**) para ser usados, y en ***sites-enabled*****,** tendremos los VirtualHosts que tenemos **funcionando**. La forma de trabajar entre estos directorios es a través de un **enlace simbólico;** así el ***default*** de ***sites-enabled*** será un enlace simbólico del ***default de sites-available.***

Ubuntu incluye sus propias herramientas para activar y desactivar tanto sitios web como módulos:

- **a2ensite**-> Activa un sitio web.
- **a2dissite**-> Desactiva un sitio web
- **a2enmod**-> Activa un módulo de Apache disponible en mods-available.
- **a2dismod**->Desactiva un módulo.

Ejemplo: Queremos construir en nuestro servidor web Apache dos sitios web con las siguientes características:

A) El nombre de dominio del primero será ***clientes.sercamp.org,*** su directorio base será ***/var/www/html/clientes/*** y contendrá una página llamada ***index.html***, donde sólo se verá una bienvenida a la página de los clientes.

B) En el segundo sitio vamos a crear una página donde se pondrán noticias por parte de los departamento, el nombre de este sitio será ***departamentos.sercamp.org***, y su directorio base será ***/var/www//html/departamentos/***. En este sitio sólo tendremos una página inicial ***index.htm***l, dando la bienvenida a la página de los departamentos de la

empresa.

Hay que tener en cuenta lo siguiente:

- Cada sitio web tendrá **nombres distintos.**
- Cada sitio web **compartirán la misma dirección IP y el mismo puerto (80).**

Para conseguir estos dos sitios virtuales debes seguir los siguientes pasos:

1. Los ficheros de configuración de los sitios webs se encuentran en el directorio /***etc/apache2/sites-available***, por defecto hay dos ficheros, uno se llama **000-*default.conf*** que es la configuración del sitio web por defecto. Necesitamos tener dos ficheros para realizar la configuración de los dos sitios virtuales, para ello vamos a ***copiar el fichero 000-default.conf:***

```
$cd /etc/apache2/sites-available
$sudo cp 000-default.conf clientes.conf
$sudo cp 000-default.conf departamentos.conf
```

De esta manera tendremos un fichero llamado *clientes.conf* para realizar la configuración del sitio web **clientes.sercamp.org**, y otro llamado *departamentos.conf* para el sitio web **departamentos.sercamp.org**

2. Modificamos el fichero *clientes.conf*, donde vamos a añadir la siguiente línea:

ServerName clientes.sercamp.org

Y vamos a modificar la línea:

DocumentRoot /var/www/html/clientes/

..

3. Realiza los cambios similares al fichero departamentos.

4. No es suficiente crear los ficheros de configuración de cada sitio web, es necesario crear un **enlace simbólico** a estos ficheros dentro del directorio ***/etc/apache2/sites-enabled,*** para ello:

```
$sudo a2ensite clientes.conf
$sudo a2ensite departamentos.conf
```

Esto creará el enlace simbólico de estos archivos entre ***sites-available y sites-enabled.*** Ahora sólo tendremos que recargar la configuración para que Apache coja los cambios.

Ubuntu(14.04)

$sudo /etc/init.d/apache2 reload

Para Ubuntu (16.04)

$sudo service apache2 reload

5. Creamos los directorios y los ficheros ***index.html*** necesarios en ***/var/www*** y reiniciamos el servicio:

$sudo mkdir /var/www/html/clientes/

$sudo mkdir /var/www/html/departamentos/

Copiamos el fichero ***index.html de /var/www/html/,*** a los distintos directorios:

$sudo cp /var/www/html/index.html /var/www/html/clientes/

6. Modificamos el fichero ***index.html*** para que dé un mensaje de bienvenida apropiado: ***Bienvenidos a la web CLIENTES.***

$sudo nano /var/www/html/clientes/index.html

```
<html>
<head>
<title>Ejemplo de página web</title>
</head>
<body>
<h1> Bienvenidos a la web de CLIENTES</h1>
</body>
</html>
```

Y lo mismo con departamentos...

7. Reiniciamos el servicio: **$sudo /etc/init.d/apache2 restart (** Ubuntu 14.04) **$sudo service apache2 restart** (Ubuntu 16.04)

8. Para terminar lo único que tendremos que hacer es cambiar el fichero de

configuración DNS de bind9, para que resuelva los nuevos nombres de host:

clientes y departamentos, tienen la misma ip que la del host www, asi que usaremos la cláusula CNAME (una para clientes y otra para departamentos), para crear un alias de www.

Con esto, ya podemos comprobar si realmente funcionan nuestros **VirtualHosts,** a través del navegador del cliente, siempre y cuando el cliente tenga configurado su servidor DNS como el que tenemos configurado.

Clientes.conf

<VirtualHost *:80>

ServerAdmin webmaster@localhost

ServerName clientes.sercamp.org

DocumentRoot /var/www/html/clientes/

</VirtualHost>

3. Establecimiento de conexiones seguras.

3.1 Introducción.

Una página web segura o un sitio web seguro es un sitio web que utiliza el protocolo **https** en lugar de utilizar el protocolo **http.**

El protocolo https es idéntico al protocolo http, con la excepción de que la transferencia de información entre el cliente (navegador web) y el servidor (servidor web) **viaja a través de Internet cifrada** utilizando robustos algoritmos de cifrado de datos proporcionados por el paquete **OpenSSL.**

Los **algoritmos de cifrado** utilizados reúnen las características necesarias para garantizar que la información que sale **desde el servidor hacia el cliente**, esté **cifrada** y solamente pueda ser descifrada por el cliente, y que la información que sale **desde el cliente hacia el servidor**, esté cifrada y solamente pueda ser descifrada por el servidor. Si durante la transferencia de la información un 'hacker' hiciera copia de los paquetes de datos e intentara descifrarlos, los algoritmos garantizarían que no podría hacerlo por fuerza bruta (probando todas las claves posibles) en un plazo mínimo de varios años.

3.2 Módulo ssl para apache2

Al instalar apache2 se instala también el módulo ssl para apache2, por lo que no es necesario instalar ningún paquete adicional. Tan solo debemos **generar un certificado** para el servidor y activar el módulo ssl.

3.3 Generar el certificado

Para que nuestro servidor pueda servir páginas seguras con el protocolo https, necesita un certificado. Dicho certificado permitirá que nuestro servidor utilice **cifrado asimétrico** para intercambiar las claves de cifrado con los clientes, antes de iniciar una trasmisión segura de información.

Inicialmente, el cliente deberá aceptar el certificado del servidor, ya que generaremos un **certificado autofirmado**. Si queremos evitarlo, deberíamos contratar un certificado a una entidad certificadora confiable, pero tiene un coste que no merece la pena soportar en un entorno educativo. **Para generar nuestro certificado autofirmado, ejecutaremos el comando:**

// Generar certificado autofirmado

$sudo make-ssl-cert /usr/share/ssl-cert/ssleay.cnf /etc/ssl/certs/apache2.pem

Durante la ejecución de comando ***make-ssl-cert***, quizás nos pregunte algunas sencillas preguntas como el nombre del servidor, etc.

> Podemos poner nombre de servidor websegura.sercamp.org y dejar en blanco el nombre alternativo

Después se creará el archivo ***/etc/ssl/certs/apache.pem*** que contiene las claves que permitirán al servidor utilizar cifrado asimétrico (compruebalo).

El siguiente paso será configurar un servidor virtual para que utilice dicho certificado.

3.4 Crear servidor virtual seguro en apache2

1. Primero crearemos una carpeta de nombre ***'websegura'*** dentro de '/var/www/html'.

> $sudo mkdir /var/www/html/websegura/

Dentro de esta carpeta crea un i***ndex.html*** dando un mensaje de bienvenida a una zona segura.

Dicha carpeta será el raíz de documentos ***(DocumentRoot)*** de nuestro servidor virtual seguro, de modo que todo lo que coloquemos en dicha carpeta deba ser accedido **vía**

'https'. Eso lo indicaremos más adelante mediante el parámetro **SSLRequireSSL.**

2. El protocolo **https** utiliza el **puerto 443**, por lo tanto, tendremos que habilitar dicho puerto para que Apache lo utilice. Si ya está habilitado el puerto 443, no hacer nada.

// Habilitar puerto 443. Añadir en ***/etc/apache2/ports.conf***

Listen 443

3. Posteriormente debemos **habilitar el módulo ssl** del servidor apache:

// Habilitar el módulo ssl: **$sudo a2enmod ssl**

4. Después debemos crear el servidor virtual en Apache. Dicho servidor virtual dispondrá de **una url de acceso diferente a la de nuestra web principal** y será accesible mediante https, por tanto tendremos que habilitar SSL e **indicar la ruta del archivo que contiene el certificado.**

Todo ello lo haremos creando un nuevo **host virtual** tal y como sabemos, pero esta vez en lugar de copiarnos el archivo 000-default.conf nos copiaremos el default-ssl.conf de sites-availables

Recordad haced copia de seguridad del fichero ***default-ssl,*** para poder recuperarlo si fuera necesario:

En el archivo de configuración .conf de nuestro nuevo sitio configuraremos:

ServerAdmin

ServerName

DocumentRoot

SSLCertificateFile., aquí indicaremos cómo se llama el certificado que acabamos de crear:

SSLCertificateFile /etc/ssl/certs/*apache2.pem*

Comentamos con una # la línea:

#SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key

5. Habilitamos websegura: **$sudo a2ensite default-ssl.conf**

6. Y recargamos apache:

$sudo /etc/init.d/apache2 reload (Ubuntu 14.04)

$sudo service apache2 reload (Ubuntu 16.04)

7. Finalmente reiniciamos el servidor apache:

$sudo /etc/init.d/apache2 restart (Ubuntu 14.04)

$sudo service apache2 restart (Ubuntu 16.04)

8. Sólo nos queda configurar nuestro **servidor dns** para que asocie la dirección ***websegura.sercamp.org*** con la ip de nuestro servidor web.

3.5 Probando el acceso a la página web segura

Para acceder a las páginas seguras de nuestro servidor web, tecleamos desde el navegador *'**https://websegura.sercamp.org**'*.

Lo primero que se muestra es la alerta de seguridad que nos indica que el certificado no está emitido por una CA en la que confiamos.

Si pulsamos sobre *'**Technical Details**'* veremos la información tanto del certificado como de la entidad certificadora que lo firma. Si aceptamos el certificado significa que, a pesar de estar firmado por una entidad certificadora que no es de confianza para el navegador (lo hemos firmado nosotros mismos), lo aceptamos.

Para continuar debemos ir a '***I Understand the Risks'***. ***Add exception***. *Pulsando en* ***View*** vemos más detalles del certificado.

Tendremos que indicar al navegador si aceptamos el certificado para siempre o solo para ahora. Como tenemos la seguridad de que el certificado es bueno porque acabamos de crearlo nosotros mismos, podemos **aceptarlo para siempre,** y así el navegador no volverá a preguntarnos más sobre él ya que hemos indicado manualmente que confiamos en este certificado. Así que pulsamos en: ***Confirm Security Exception***

Ahora ya tenemos acceso a la web segura mediante el protocolo https lo que nos garantiza que la información de la página segura, antes de salir del servidor, ha sido cifrada y por tanto la trasferencia de datos desde el servidor a nuestro navegador se ha producido de forma segura. Al llegar a nuestro navegador, se han descifrado los datos. El candado cerrado que aparece abajo a la derecha en el navegador, indica que la trasferencia de datos se ha realizado de forma segura.

4. Acceso autentificado a nuestro servidor web.

4.1 Carpetas privadas.

Una vez que nuestro sitio web está funcionando, vamos a crear una carpeta a la que sólo daremos acceso a usuarios autenticados empleando la autenticación básica proporcionada por Apache.

Para ello vamos a tener en nuestro directorio base del sitio web una carpeta (que

podemos llamar privado) para acceder a ella será necesario una **autentificación.**

- En primer lugar crearemos la carpeta (privado) en la que queramos restringir el acceso a determinados usuarios:

$sudo mkdir /var/www/html/departamentos/privado

Crearemos dentro del directorio **privado** un fichero ***index.html*** con un mensaje: **Bienvenidos a la zona PRIVADA**

Ahora hay que seguir los siguientes pasos:

4.2 El fichero de configuración del sitio

En el fichero de configuración del sitio ***departamentos.sercamp.org*** -> ***/etc/apache2/sites-available/departamentos.conf*** crea las siguientes líneas:

<Directory /var/www/departamentos/privado/>

AuthType basic

AuthName "Página privada sólo para personas autorizadas"

AuthUserFile /etc/apache2/password

<Limit GET POST>

Require valid-user

</Limit>

</Directory>

Esto quiere decir:

- Para acceder al directorio ***/var/www/html/departamentos/privado/*** se necesita autenticación básica.
- ***AuthName*** es el mensaje que se va a presentar al usuario.
- El fichero ***/etc/apache2/password*** contiene usuarios y contraseñas.
- Por último exigimos que se nos pregunte por un usuario válido.

Reiniciamos el servidor apache

$sudo /etc/init.d/apache2 reload (Ubuntu 14.04)

$sudo service apache2 reload (Ubuntu 16.04)

4.3 El fichero de contraseñas

Tenemos que crear el fichero de contraseña para los usuarios que tienen permiso. Para ello:

$cd /etc/apache2

$sudo htpasswd -c password jefe

Esta instrucción pide la contraseña para el **usuario jefe** y la guarda en el fichero password, la opción -c crea el fichero.

Para añadir un nuevo usuario al fichero operamos así:

$sudo htpasswd password carolina

Para crear el fichero de contraseñas con la introducción del primer usuario tenemos que añadir la **opción -c** (create) al comando anterior. Si por error la seguimos usando al incorporar nuevos usuarios borraremos todos los anteriores, así que cuidado con esto. El argumento **-c** solamente se pone al crear al primer usuario.

Las contraseñas, como podemos ver a continuación, no se guardan en claro. Lo que se almacena es el resultado de aplicar una **función hash:**

$cat password

pepe:rOUetcAKYaliE

carolina:hmO6V4bM8KLdw

Para denegar el acceso a algún usuario basta con que borremos la línea correspondiente al mismo. No es necesario que le pidamos a Apache que vuelva a leer su configuración ***(/etc/init.d/apache2 reload)*** cada vez que hagamos algún cambio en este fichero de contraseñas, pero si lo es después de hacer los cambios en el fichero de definición del ***Virtual Host.***

La principal ventaja de este método de autenticación es su sencillez. Sus inconvenientes: lo incómodo de delegar la generación de nuevos usuarios en alguien que no sea un

administrador de sistemas o de hacer un front-end para que sea el propio usuario quien cambie su contraseña. Y, por supuesto, que **dichas contraseñas y nombres de usuario viajan en texto plano por la red (sin encriptar)** a no ser que empleemos **https**, o sea, crear una instancia **Apache con SSL.**

Si ahora intentamos acceder a nuestra carpeta desde el navegador cliente:

http://departamentos.sercamp.org/privado

veremos que nos pide un usuario y contraseña y tras introducir el usuario y contraseña correcto nos dará acceso a la carpeta: ***privado.***

5. Apache HTTP Server Logs

La instalación por defecto de Apache2 crea l

el subdirectorio de log: ***/var/log/apache2.*** En el cual se encuentran dos ficheros de log con distinto propósito:

- **/var/log/apache2/access.log** – registra cada página servida y cada fichero descargado por el web server. Por defecto cada vez que Apache accede a un fichero o a una página, se registra la dirección IP, fecha y hora, browser, el texto de la petición que ha generado la consulta.

- **/var/log/apache2/error.log** – registra todas las condiciones de error informadas por el HTTP server. Cada vez que ocurre un error Apache añade una linea en el log de error.

Por ejemplo para ver los últimos errores: ***$sudo cat /var/log/apache2/error.log***

Para recorrer el fichero: ***$sudo less /var/log/apache2/error.log***

Con las teclas arriba y abajo lo recorro, pulsar una **q** para salir.