

**Instituto Politécnico Nacional**

**Escuela Superior de Cómputo**

**Soto Ramos Manuel Alejandro**

**Administración de Servicios en Red**

**Manual de Instalación y Configuración - Servidor HTTP/HTTPS**

Jiménez Maruri Pedro

Valencia Rodriguez Fernando  
Romero Serrano Luis

Grupo: 4CV4

# 

[**Requerimientos**](#_ynjeycim5ace) **4**

[**Sistema**](#_831edxnqsbxq) **4**

[Licencia](#_c6m9tj4v94be) 4

[Fabricante](#_sm8u2o43cjw9) 4

[Impacto en el mercado](#_bbe912bywhod) 4

[Lenguaje utilizado](#_k8n7qllas2oj) 4

[Sistemas Operativos](#_v0zkopxgvuy8) 4

[**Instalación**](#_831edxnqsbxq) **5**

[Requisitos previos](#_sbbs7i5mln5x) 5

[Instalar Apache](#_aotz7kv6rzrs) 5

[Paso 1: Instalar Apache](#_3znnyqcbih2f) 5

[Configurar Global ServerName para que suprima los avisos de sintaxis](#_2dai05wpy9wh) 5

[Paso 2: Ajustar el cortafuegos](#_4j1017surubn) 6

[Cómo encontrar la dirección IP pública de su servidor](#_atjh7zbp7cqn) 7

[Paso 3: Compruebe su servidor web](#_9h86y5jj0bkn) 7

[Gestionar el proceso de Apache](#_nw7w4ib8tyxj) 10

[Familiaricese con los archivos y directorios importantes de Apache](#_9ktev8c9hzka) 10

[Contenido](#_5pads09v5c4d) 10

[Configuración del servidor](#_ti2pw4tme0hu) 10

[Registros del servidor](#_gb7421rcncy9) 11

[Install MySQL](#_o0tu0uj8b4qw) 11

[Instalar PHP](#_nz7bu7xbx51p) 13

[Instalar módulos PHP](#_700sqej9v6pa) 14

[Pruebe el procesamiento de PHP en su servidor web](#_yqr6ong6e5xy) 15

[**Virtual Host**](#_kf23o714otna) **17**

[Paso uno: Crear la estructura de directorios](#_2pr1ol77i8oq) 17

[Paso Dos: Otorgar Permisos](#_oe84tfjooknz) 18

[Paso tres: Crear páginas de demostración para cada host virtual](#_5k8o05u2mw45) 18

[Paso cuatro: Crear nuevos archivos de host virtual](#_8an6itng8sen) 19

[Crear el primer archivo de host virtual](#_4sjd5bvfryc6) 19

[Copiar el primer host virtual y personalizarlo para el segundo dominio](#_kyl22nxlr65l) 20

[Alojamiento virtual basado en IP - Alternativa](#_5qutbe5de38) 21

[1.- Creación de múltiples direcciones para un único servidor.](#_ae3r83l30va2) 21

[2.- Configurar Múltiples instancias para el servidor Apache](#_j4lf13ucyqs1) 22

[3.- Modificar los archivos del Virtual Host](#_v049hxye76pv) 22

[Configuración del puerto de operación](#_ac141a9x9ej) 23

[1.- Activar la escucha del puerto](#_f0oi0w72alp3) 23

[2.- Modificar el puerto en el Virtual Host del sitio](#_hdq0s8faayz0) 24

[3.- Reiniciar el servidor](#_azyyx7vbp4y) 25

[Paso cinco: Habilitar los nuevos archivos del host virtual](#_t7rvg9cuqbe6) 25

[Paso seis: configurar el archivo de hosts locales (opcional)](#_72i2ox244h3a) 25

[Paso siete: Pruebe sus resultados](#_bx9ucdmkr964) 26

[**htaccess**](#_cle4mm18f5nk) **27**

[Habilitar .htaccess](#_g7jxufl5dhva) 27

[Restringir listados de directorios](#_cqeg4ynv97qj) 28

[Crear .htaccess](#_cpyosv8uu38) 28

[Restringir IPs](#_mb89wi4hhdgh) 28

[Bloquear IPs](#_ae7bq0acrhlx) 28

[Permitir IPs](#_qpfvsroba7tq) 29

[Redireccionamiento](#_tw2xvi5kszjb) 29

[Configurar la página de error 404 Error Page](#_wdpe3gdq33yh) 30

[**Control de acceso**](#_fjxnf4lrgt7i) **31**

[Configuración del control de acceso](#_x9eas8e0zq87) 31

[**Módulo mod\_authz\_host**](#_badj51l7xlj6) **31**

[Bloqueo por IP](#_2y5uq7s0gfui) 31

[Bloqueo por segmento de red](#_2y5uq7s0gfui) 32

[Bloqueo por nombre de usuario](#_2y5uq7s0gfui) 32

[mod\_authn\_core](#_d3l5i0nn13y2) 32

[htpasswd](#_s3dz59t373og) 33

[**Aplicación de Certificado para HTTPS**](#_q9j9e5re6y4j) **33**

[Certificado auto firmado](#_8bj9bbcig40p) 33

[Paso 1: Crear el certificado SSL](#_qrhodsdfz8m9) 34

[Paso 2: Configurar Apache para usar SSL](#_vputm7y7oc32) 35

[Crear un fragmento de configuración de Apache con una configuración de cifrado fuerte](#_rbn73hpi72ys) 35

[Modificar el archivo de host virtual SSL predeterminado de Apache](#_a71iuz8x4f9q) 37

[(Recomendado) Modificar el archivo de host virtual no cifrado para redirigirlo a HTTPS](#_dvbnez1rlx1k) 38

[Paso 3: Ajustar el cortafuegos](#_btjo2bgb1x15) 39

[Paso 4: Habilitar los cambios en Apache](#_9i3c5rbbr7ma) 40

[Paso 5: Comprobar el cifrado](#_dg9ubl7y82no) 41

[Paso 6: Cambiar a una redirección permanente](#_h1o5z31kmwnx) 42

[Certificado firmado por una autoridad](#_8bj9bbcig40p) 43

[Requisitos previos](#_tcjjwdbc8q2s) 43

[Paso 1 - Instalando Certbot](#_vztn0ytwion9) 44

[Paso 2: Configuración del certificado SSL](#_ebe5i32j8kjp) 44

[Paso 3 - Permitir HTTPS a través del cortafuegos](#_ecrb8f6ljjdx) 45

[Paso 4 - Obtención de un certificado SSL](#_p9ykpo1glkno) 46

[Paso 5 - Verificando la renovación automática de Certbot](#_6r3i6a5c3psd) 47

[**Personalización de páginas de error**](#_9yq66ev7vicu) **47**

[Páginas de errores](#_gr2qopuh3mgr) 47

[Tipos de errores](#_tcsv4js1eh6u) 47

[Configuración de las páginas de errores](#_nlrtox1mlt1z) 48

[**Bitácoras y mensajes de error**](#_6icp7e1si17m) **48**

[Configuración de bitácoras](#_2eckuikjnmhq) 48

[Configuración de formato](#_ncuvfs2vbr22) 48

[**Resumen de operación**](#_wy8idt1etos7) **49**

[Conexiones](#_ge84gr5jp6ju) 49

[Consumo de recursos](#_ge84gr5jp6ju) 49

[**Referencias**](#_gjzc2zjvyu1j) **49**

# 

# **Requerimientos**

* Sistema operativo: Unix, Unix-like o Windows 2000 o superior.
* Disco duro: Al menos 50 MB. La cantidad de disco duro necesario varía de acuerdo a los requerimientos de cada proyecto.
* Compilador ANSI-C.
* Sistema de sincronización de tiempo.
* Librerías:
  + openssl-0.9.7e (source)
  + httpd-2.0.49 (source)
  + gcc-3.3-sol9-sparc-local packages (for Solaris SPARC).
  + gcc-3.3-sol9-intel-local packages (for Solaris x86)
  + flex-2.5.4a-sol9-sparc-local packages (for Solaris SPARC)
  + flex-2.5.4a-sol9-intel-local packages (for Solaris x86)

# **Sistema**

## Licencia

*Apache* cuenta con una licencia permisiva. Este tipo de licencia permite el uso, distribución y modificación del software.

## Fabricante

*Apache Software Foundation* es la organización sin fines de lucro encargada del desarrollo y mantenimiento de los proyectos *Apache*, entre ellos *Apache HTTP Server.*

## Impacto en el mercado

Desde su desarrollo en 1995 se ha convertido en el servidor web más usado a nivel mundial, siendo usado en aproximadamente el 44.1% de los sitios, logrando esto desde abril de 1996[1].

## Lenguaje utilizado

Originalmente basada en el proyecto *NCSA HTTPd*, *Apache* fue posteriormente reestructurado. En la actualidad, el código fuente de *Apache* está escrito en C.

## Sistemas Operativos

*Apache* está diseñado para funcionar en sistemas operativos *Unix, Unix-Like* y Windows. Un sistema operativo *Unix-Like* es un sistema operativo que se comporta de manera similar a un sistema operativo *Unix*  sin la necesidad de ser uno.

# **Instalación**

El servidor HTTP Apache es el servidor web más utilizado del mundo. Proporciona muchas funciones potentes, incluyendo módulos que se pueden cargar dinámicamente, soporte de medios robusto e integración extensiva con otro software popular.

En esta guía, veremos cómo instalar un servidor web Apache en un equipo con Ubuntu 16.04 o superior.

## **Requisitos previos**

Antes de comenzar, debe tener un usuario regular, no root, con privilegios *sudo* configurados en su servidor.

## **Instalar Apache**

### **Paso 1: Instalar Apache**

Apache está disponible en los repositorios de software por defecto de Ubuntu, por lo que lo instalaremos utilizando las herramientas convencionales de gestión de paquetes.

Comenzaremos por actualizar el índice de paquetes locales para que refleje los últimos cambios. Después, podemos instalar el paquete apache2:

sudo apt-get update  
 sudo apt-get install apache2

Después de confirmar la instalación, apt-get instalará Apache y todas las dependencias necesarias.

#### **Configurar Global ServerName para que suprima los avisos de sintaxis**

A continuación, añadiremos una sola línea al archivo **/etc/apache2/apache2.conf** para suprimir un mensaje de advertencia. Si bien es inofensivo, si no establece **ServerName** globalmente, recibirá la siguiente advertencia cuando compruebe si hay errores de sintaxis en la configuración de Apache:

sudo apache2ctl configtest

**Output**  
 AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message  
 Syntax OK

Abra el archivo de configuración principal con su edición de texto:

sudo nano /etc/apache2/apache2.conf

Dentro, en la parte inferior del archivo, agregue una directiva **ServerName**, apuntando a su nombre de dominio primario. Si no tiene un nombre de dominio asociado con su servidor, puede utilizar la dirección IP pública de su servidor:

Si no conoce la dirección IP de su servidor, vaya a la sección sobre cómo encontrar la dirección IP pública de su servidor para encontrarla.

**/etc/apache2/apache2.conf**

. . .  
ServerName server\_domain\_or\_IP

Guarde y cierre el archivo cuando haya terminado.

A continuación, compruebe si hay errores de sintaxis escribiendo:

sudo apache2ctl configtest

Como hemos añadido la directiva global ServerName, todo lo que debería ver es:

**Output**

Syntax OK

Reinicie Apache para implementar sus cambios:

sudo systemctl restart apache2

Ahora puede empezar a ajustar el cortafuegos.

### **Paso 2: Ajustar el cortafuegos**

Antes de que podamos probar Apache, necesitamos modificar nuestro firewall para permitir el acceso externo a los puertos web por defecto.

Durante la instalación, Apache se registra en la UFW para proporcionar algunos perfiles de aplicaciones. Podemos usar estos perfiles para simplificar el proceso de habilitar o deshabilitar el acceso a Apache a través de nuestro cortafuegos.

Podemos listar los perfiles de la aplicación ufw escribiendo:

sudo ufw app list

Debería obtener una lista de los perfiles de la aplicación:

**Output**  
Available applications:  
 Apache  
 Apache Full  
 Apache Secure  
 OpenSSH

Como puede ver, hay tres perfiles disponibles para Apache:

* **Apache:** Este perfil sólo abre el puerto 80 (tráfico web normal, no encriptado)
* **Apache Full:** Este perfil abre tanto el puerto 80 (tráfico web normal no cifrado) como el puerto 443 (tráfico cifrado TLS/SSL).
* **Apache Secure:** Este perfil sólo abre el puerto 443 (tráfico cifrado TLS/SSL)

Para nuestros propósitos, permitiremos el tráfico entrante para el perfil completo de Apache escribiendo:

sudo ufw allow in "Apache Full"  
 sudo ufw enable

Puede verificar el cambio escribiendo:

sudo ufw status

Debería ver el tráfico HTTP permitido en la salida mostrada:

**Output**  
Status: active  
  
To Action From  
-- ------ ----  
OpenSSH ALLOW Anywhere   
Apache Full ALLOW Anywhere   
OpenSSH (v6) ALLOW Anywhere (v6)   
Apache Full (v6) ALLOW Anywhere (v6)

Como puede ver, el perfil ha sido activado para permitir el acceso al servidor web.

#### **Cómo encontrar la dirección IP pública de su servidor**

Si no sabe cuál es la dirección IP pública de su servidor, hay varias maneras de encontrarla. Normalmente, esta es la dirección que utiliza para conectarse a su servidor a través de SSH.

Desde la línea de comandos, puede encontrar esto de varias maneras. Primero, puede usar las herramientas de iproute2 para obtener su dirección escribiendo esto:

ip addr show eth0 | grep inet | awk '{ print $2; }' | sed 's/\/.\*$//'

Esto te dará dos o tres líneas de vuelta. Todas son direcciones correctas, pero es posible que su ordenador sólo pueda utilizar una de ellas, así que no dude en probar cada una de ellas.

Un método alternativo es usar la utilidad **curl** para contactar con un tercero y decirle cómo ve su servidor. Puede hacerlo preguntando a un servidor específico cuál es su dirección IP:

sudo apt-get install curl  
 curl http://icanhazip.com

Independientemente del método que utilice para obtener su dirección IP, puede escribirla en la barra de direcciones de su navegador web para llegar a su servidor.

### **Paso 3: Compruebe su servidor web**

Al final del proceso de instalación, Ubuntu 16.04 inicia Apache. El servidor web ya debería estar en funcionamiento.

Podemos verificar con *systemd* para asegurarnos de que el servicio está funcionando escribiendo:

sudo systemctl status apache2

**Output**  
● apache2.service - LSB: Apache2 web server  
 Loaded: loaded (/etc/init.d/apache2; bad; vendor preset: enabled)  
 Drop-In: /lib/systemd/system/apache2.service.d  
 └─apache2-systemd.conf  
 Active: active (running) since Fri 2017-05-19 18:30:10 UTC; 1h 5min ago  
 Docs: man:systemd-sysv-generator(8)  
 Process: 4336 ExecStop=/etc/init.d/apache2 stop (code=exited, status=0/SUCCESS)  
 Process: 4359 ExecStart=/etc/init.d/apache2 start (code=exited, status=0/SUCCESS)  
 Tasks: 55  
 Memory: 2.3M  
 CPU: 4.094s  
 CGroup: /system.slice/apache2.service  
 ├─4374 /usr/sbin/apache2 -k start  
 ├─4377 /usr/sbin/apache2 -k start  
 └─4378 /usr/sbin/apache2 -k start  
  
May 19 18:30:09 ubuntu-512mb-nyc3-01 systemd[1]: Stopped LSB: Apache2 web server.  
May 19 18:30:09 ubuntu-512mb-nyc3-01 systemd[1]: Starting LSB: Apache2 web server...  
May 19 18:30:09 ubuntu-512mb-nyc3-01 apache2[4359]: \* Starting Apache httpd web server apache2  
May 19 18:30:09 ubuntu-512mb-nyc3-01 apache2[4359]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message  
May 19 18:30:10 ubuntu-512mb-nyc3-01 apache2[4359]: \*  
May 19 18:30:10 ubuntu-512mb-nyc3-01 systemd[1]: Started LSB: Apache2 web server.

Como puede ver arriba, el servicio parece haber comenzado con éxito. Sin embargo, la mejor manera de probar esto es solicitar una página a Apache.

Puede acceder a la página de destino predeterminada de Apache para confirmar que el software funciona correctamente. Puede acceder a ella a través del nombre de dominio o de la dirección IP de su servidor (generalmente es <http://localhost/> o <http://127.0.0.1/> ).

Debería ver la página web predeterminada de Ubuntu 16.04 Apache, que debería ser algo como en la siguiente imagen:



Ejemplo de servidor Apache activo.

### **Gestionar el proceso de Apache**

Ahora que tiene su servidor web en funcionamiento, podemos repasar algunos comandos básicos de administración.

Para detener su servidor web, puede escribir:

sudo systemctl stop apache2

Para iniciar el servidor web cuando se detiene, escriba:

sudo systemctl start apache2

Para detenerse y luego volver a iniciar el servicio, escriba:

sudo systemctl restart apache2

Si simplemente está haciendo cambios de configuración, Apache a menudo puede recargar sin dejar caer las conexiones. Para ello, puede utilizar este comando:

sudo systemctl reload apache2

Por defecto, Apache está configurado para que se inicie automáticamente cuando el servidor arranque. Si esto no es lo que quieres, puedes desactivar este comportamiento escribiendo:

sudo systemctl disable apache2

Para volver a habilitar el servicio para que se inicie al arrancar, puede escribir:

sudo systemctl enable apache2

Ahora Apache debería iniciarse automáticamente cuando el servidor arranque de nuevo.

### **Familiaricese con los archivos y directorios importantes de Apache**

Ahora que ya sabe cómo administrar el servicio en sí, debería tomarse unos minutos para familiarizarse con algunos directorios y archivos importantes.

#### **Contenido**

* **/var/www/html:** El contenido web real, que por defecto sólo consiste en la página por defecto de Apache que ha visto antes, se sirve desde el directorio **/var/www/html**. Esto se puede cambiar alterando los archivos de configuración de Apache.

#### **Configuración del servidor**

* **/etc/apache2:** El directorio de configuración de Apache. Todos los archivos de configuración de Apache residen aquí.
* **/etc/apache2/apache2.conf:** El archivo de configuración principal de Apache. Esto puede ser modificado para hacer cambios en la configuración global de Apache. Este fichero es responsable de cargar muchos de los otros ficheros en el directorio de configuración.
* **/etc/apache2/ports.conf:** Este archivo especifica los puertos en los que Apache escuchará. Por defecto, Apache escucha en el puerto 80 y adicionalmente escucha en el puerto 443 cuando un módulo que proporciona capacidades SSL está habilitado.
* **/etc/apache2/sites-available/:** El directorio donde se pueden almacenar los "hosts virtuales" de cada sitio. Apache no usará los ficheros de configuración que se encuentran en este directorio a menos que estén enlazados al directorio habilitado para sitios (ver más abajo). Típicamente, toda la configuración de bloques de servidor se realiza en este directorio, y luego se habilita enlazando al otro directorio con el comando **a2ensite**.
* **/etc/apache2/sites-enabled/:** El directorio donde está habilitado cada sitio "Virtual Host" es almacenado. Por lo general, se crean enlazando a los archivos de configuración que se encuentran en el directorio **sites-available** con **a2ensite**. Apache lee los archivos de configuración y los enlaces que se encuentran en este directorio cuando se inicia o se recarga para compilar una configuración completa.
* **/etc/apache2/conf-available/**, **/etc/apache2/conf-enabled/**: Estos directorios tienen la misma relación que **sites-available** y **sites-enabled**, pero se utilizan para almacenar fragmentos de configuración que no pertenecen a un Virtual Host. Los archivos en el directorio disponible pueden ser habilitados con el comando **a2enconf** y deshabilitados con el comando **a2disconf**.
* **/etc/apache2/mods-available/**, **/etc/apache2/mods-enabled/**: Estos directorios contienen los módulos disponibles y habilitados, respectivamente. Los archivos que terminan en **.load** contienen fragmentos para cargar módulos específicos, mientras que los que terminan en **.conf** contienen la configuración de esos módulos. Los módulos se pueden habilitar y deshabilitar utilizando los comandos **a2enmod** y **a2dismod**.

#### **Registros del servidor**

* **/var/log/apache2/access.log:** Por defecto, cada petición a su servidor web se registra en este archivo de registro a menos que Apache esté configurado para hacer lo contrario.
* **/var/log/apache2/error.log:** Por defecto, todos los errores se registran en este archivo. La directiva **LogLevel** en la configuración de Apache especifica cuánto detalle contendrán los registros de errores.

## **Install MySQL**

Ahora que tenemos nuestro servidor web funcionando, es el momento de instalar MySQL. MySQL es un sistema de gestión de bases de datos. Básicamente, organizará y proporcionará acceso a bases de datos donde nuestro sitio puede almacenar información.

Una vez más, podemos usar **apt** para adquirir e instalar nuestro software. Esta vez, también instalaremos otros paquetes "auxiliares" que nos ayudarán a conseguir que nuestros componentes se comuniquen entre sí:

sudo apt-get install mysql-server

**Nota:** En este caso, no es necesario ejecutar sudo apt-get update antes del comando. Esto se debe a que recientemente lo hemos ejecutado en los comandos anteriores para instalar Apache. El índice de paquetes de nuestro ordenador ya debería estar actualizado.

Una vez más, se le mostrará una lista de los paquetes que se instalarán, junto con la cantidad de espacio en disco que ocuparán. Introduzca Y para continuar.

Durante la instalación, su servidor le pedirá que seleccione y confirme una contraseña para el usuario "root" de MySQL. Esta es una cuenta administrativa en MySQL que ha aumentado los privilegios. Piense que es similar a la cuenta root del propio servidor (sin embargo, la que está configurando ahora es una cuenta específica de MySQL). Asegúrese de que se trata de una contraseña segura y única, y no la deje en blanco.

Cuando la instalación esté completa, queremos ejecutar un sencillo script de seguridad que eliminará algunos valores predeterminados peligrosos y bloqueará un poco el acceso a nuestro sistema de base de datos. Inicie el script interactivo ejecutando:

mysql\_secure\_installation

Se le pedirá que introduzca la contraseña que ha establecido para la cuenta root de MySQL. A continuación, se le preguntará si desea configurar el **VALIDATE PASSWORD PLUGIN**.

**Advertencia:** Habilitar esta función es algo así como un juicio de valor. Si está habilitado, las contraseñas que no coincidan con los criterios especificados serán rechazadas por MySQL con un error. Esto causará problemas si utiliza una contraseña débil junto con un software que configura automáticamente las credenciales de usuario de MySQL, como los paquetes Ubuntu para phpMyAdmin. Es seguro dejar la validación desactivada, pero siempre debe usar contraseñas fuertes y únicas para las credenciales de la base de datos.

Responde **y** para que sí, o para que cualquier otra cosa continúe sin permitirlo.

VALIDATE PASSWORD PLUGIN can be used to test passwords

and improve security. It checks the strength of password

and allows the users to set only those passwords which are

secure enough. Would you like to setup VALIDATE PASSWORD plugin?

Press y|Y for Yes, any other key for No:

Se le pedirá que seleccione un nivel de validación de contraseña. Tenga en cuenta que si introduce 2, para el nivel más alto, recibirá errores al intentar establecer cualquier contraseña que no contenga números, letras mayúsculas y minúsculas y caracteres especiales, o que esté basada en palabras comunes del diccionario.

There are three levels of password validation policy:

LOW Length >= 8

MEDIUM Length >= 8, numeric, mixed case, and special characters

STRONG Length >= 8, numeric, mixed case, special characters and dictionary file

Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 1

Si habilitó la validación de contraseña, se le mostrará una contraseña segura para la contraseña de root existente y se le preguntará si desea cambiarla. Si está satisfecho con su contraseña actual, introduzca n para "no" en la solicitud:

Using existing password for root.

Estimated strength of the password: 100

Change the password for root ? ((Press y|Y for Yes, any other key for No) : n

Para el resto de las preguntas, pulse Y y pulse la tecla Intro en cada pregunta. Esto eliminará algunos usuarios anónimos y la base de datos de prueba, deshabilitará los inicios de sesión remotos y cargará estas nuevas reglas para que MySQL respete inmediatamente los cambios que hemos realizado.

En este punto, su sistema de base de datos ya está configurado y podemos seguir adelante.

## **Instalar PHP**

PHP es el componente de nuestra configuración que procesará el código para mostrar contenido dinámico. Puede ejecutar scripts, conectarse a nuestras bases de datos MySQL para obtener información y entregar el contenido procesado a nuestro servidor web para su visualización.

Una vez más, podemos aprovechar el sistema **apt** para instalar nuestros componentes. Vamos a incluir algunos paquetes de ayuda también, para que el código PHP pueda ejecutarse bajo el servidor Apache y hablar con nuestra base de datos MySQL:

sudo apt-get install php libapache2-mod-php php-mcrypt php-mysql

Esto debería instalar PHP sin ningún problema. Lo probaremos en un momento.

En la mayoría de los casos, queremos modificar la forma en que Apache sirve los ficheros cuando se solicita un directorio. Actualmente, si un usuario solicita un directorio desde el servidor, Apache buscará primero un archivo llamado **index.html**. Queremos decirle a nuestro servidor web que prefiera los archivos PHP, así que haremos que Apache busque primero un archivo **index.php**.

Para ello, escriba este comando para abrir el archivo **dir.conf** en un editor de texto con privilegios de root:

sudo nano /etc/apache2/mods-enabled/dir.conf

Se verá así:

**/etc/apache2/mods-enabled/dir.conf**

<IfModule mod\_dir.c>  
 DirectoryIndex index.html index.cgi index.pl index.php index.xhtml index.htm  
</IfModule>

Queremos mover el archivo índice PHP resaltado arriba a la primera posición después de la especificación DirectoryIndex, así:

**/etc/apache2/mods-enabled/dir.conf**

<IfModule mod\_dir.c>  
 DirectoryIndex index.php index.html index.cgi index.pl index.xhtml index.htm  
</IfModule>

Cuando haya terminado, guarde y cierre el archivo presionando **Ctrl-X**. Tendrá que confirmar el archivo guardado escribiendo **Y** y luego presionando **Enter** para confirmar la ubicación del archivo guardado.

Después de esto, necesitamos reiniciar el servidor web Apache para que nuestros cambios sean reconocidos. Puede hacerlo escribiendo esto:

sudo systemctl restart apache2

También podemos comprobar el estado del servicio **apache2** usando **systemctl**:

sudo systemctl status apache2

**Sample Output**  
● apache2.service - LSB: Apache2 web server  
 Loaded: loaded (/etc/init.d/apache2; bad; vendor preset: enabled)  
 Drop-In: /lib/systemd/system/apache2.service.d  
 └─apache2-systemd.conf  
 Active: active (running) since Wed 2016-04-13 14:28:43 EDT; 45s ago  
 Docs: man:systemd-sysv-generator(8)  
 Process: 13581 ExecStop=/etc/init.d/apache2 stop (code=exited, status=0/SUCCESS)  
 Process: 13605 ExecStart=/etc/init.d/apache2 start (code=exited, status=0/SUCCESS)  
 Tasks: 6 (limit: 512)  
 CGroup: /system.slice/apache2.service  
 ├─13623 /usr/sbin/apache2 -k start  
 ├─13626 /usr/sbin/apache2 -k start  
 ├─13627 /usr/sbin/apache2 -k start  
 ├─13628 /usr/sbin/apache2 -k start  
 ├─13629 /usr/sbin/apache2 -k start  
 └─13630 /usr/sbin/apache2 -k start  
  
Apr 13 14:28:42 ubuntu-16-lamp systemd[1]: Stopped LSB: Apache2 web server.  
Apr 13 14:28:42 ubuntu-16-lamp systemd[1]: Starting LSB: Apache2 web server...  
Apr 13 14:28:42 ubuntu-16-lamp apache2[13605]: \* Starting Apache httpd web server apache2  
Apr 13 14:28:42 ubuntu-16-lamp apache2[13605]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName’  
Apr 13 14:28:43 ubuntu-16-lamp apache2[13605]: \*  
Apr 13 14:28:43 ubuntu-16-lamp systemd[1]: Started LSB: Apache2 web server.

### **Instalar módulos PHP**

Para mejorar la funcionalidad de PHP, opcionalmente podemos instalar algunos módulos adicionales.

Para ver las opciones disponibles para los módulos y bibliotecas PHP, puede canalizar los resultados de la búsqueda apt-cache hacia less, un buscapersonas que le permite desplazarse a través de la salida de otros comandos:

apt-cache search php- | less

Utilice las teclas de flecha para desplazarse hacia arriba y hacia abajo, y **q** para salir.

Los resultados son todos los componentes opcionales que puede instalar. Le dará una breve descripción de cada uno de ellos:

**libnet-libidn-perl - Perl bindings for GNU Libidn  
 php-all-dev - package depending on all supported PHP development packages  
 php-cgi - server-side, HTML-embedded scripting language (CGI binary) (default)  
 php-cli - command-line interpreter for the PHP scripting language (default)  
 php-common - Common files for PHP packages  
 php-curl - CURL module for PHP [default]  
 php-dev - Files for PHP module development (default)  
 php-gd - GD module for PHP [default]  
 php-gmp - GMP module for PHP [default]  
 …  
 :**

Para obtener más información sobre lo que hace cada módulo, puede buscar en Internet o puede ver la descripción larga del paquete escribiendo:

apt-cache show package\_name

Habrá mucha salida, con un campo llamado **Description-en** que tendrá una explicación más larga de la funcionalidad que proporciona el módulo.

Por ejemplo, para saber qué hace el módulo **php-cli**, podríamos escribir esto:

apt-cache show php-cli

Junto con una gran cantidad de otra información, encontrará algo que se parece a esto:

**Output  
 …  
 Description-en: command-line interpreter for the PHP scripting language (default)  
 This package provides the /usr/bin/php command interpreter, useful for  
 testing PHP scripts from a shell or performing general shell scripting tasks.  
 .  
 PHP (recursive acronym for PHP: Hypertext Preprocessor) is a widely-used  
 open source general-purpose scripting language that is especially suited  
 for web development and can be embedded into HTML.  
 .  
 This package is a dependency package, which depends on Debian's default  
 PHP version (currently 7.0).  
 …**

Si, después de investigar, decide que le gustaría instalar un paquete, puede hacerlo usando el comando **apt-get install** como hemos estado haciendo con nuestro otro software.

Si decidimos que **php-cli** es algo que necesitamos, podríamos escribir a máquina:

sudo apt-get install php-cli

Si desea instalar más de un módulo, puede hacerlo listando cada uno, separado por un espacio, siguiendo el comando apt-get install, así:

sudo apt-get install package1 package2 ...

En este punto, su pila LAMP está instalada y configurada. Sin embargo, aún así deberíamos probar nuestro PHP.

### **Pruebe el procesamiento de PHP en su servidor web**

Para comprobar que nuestro sistema está configurado correctamente para PHP, podemos crear un script PHP muy básico.

Llamaremos a este script **info.php**. Para que Apache pueda encontrar el archivo y servirlo correctamente, debe ser guardado en un directorio muy específico, que se llama "web root".

En Ubuntu 16.04, este directorio se encuentra en **/var/www/html/**. Podemos crear el archivo en esa ubicación escribiendo:

sudo nano /var/www/html/info.php

Esto abrirá un archivo en blanco. Queremos poner el siguiente texto, que es código PHP válido, dentro del archivo:

info.php

<?php  
phpinfo();  
?>

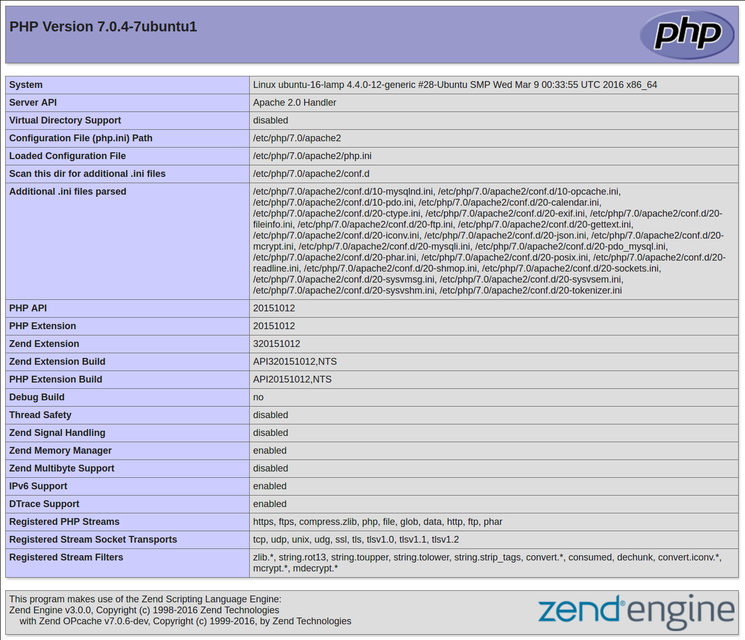
Cuando haya terminado, guarde y cierre el archivo.

Ahora podemos comprobar si nuestro servidor web puede mostrar correctamente el contenido generado por un script PHP. Para probar esto, sólo tenemos que visitar esta página en nuestro navegador web. Necesitará de nuevo la dirección IP pública de su servidor.

La dirección que desea visitar será:

[http://your\_server\_IP\_address/info.php](http://your_server_ip_address/info.php)

La página a la que llegas debe ser algo así:



Esta página básicamente le da información sobre su servidor desde la perspectiva de PHP. Es útil para depurar y para asegurarse de que la configuración se está aplicando correctamente.

Si esto fue exitoso, entonces su PHP está funcionando como se esperaba.

Es probable que desee eliminar este archivo después de esta prueba, ya que podría proporcionar información sobre su servidor a usuarios no autorizados. Para ello, puede escribirlo:

sudo rm /var/www/html/info.php

Siempre puede volver a crear esta página si necesita volver a acceder a la información más tarde.

# **Virtual Host**

El servidor web Apache es la forma más popular de servir contenido web en Internet. Representa más de la mitad de todos los sitios web activos en Internet y es extremadamente potente y flexible.

Apache divide su funcionalidad y componentes en unidades individuales que pueden ser personalizadas y configuradas independientemente. La unidad básica que describe un sitio o dominio individual se denomina *Virtual Host.*

Estas designaciones permiten al administrador utilizar un servidor para alojar múltiples dominios o sitios fuera de una sola interfaz o IP utilizando un mecanismo de correspondencia. Esto es relevante para cualquiera que desee alojar más de un sitio fuera de un solo VPS.

Cada dominio que se configure dirigirá al visitante a un directorio específico que contiene la información de ese sitio, nunca indicando que el mismo servidor también es responsable de otros sitios. Este esquema es expandible sin límite de software siempre y cuando su servidor pueda manejar la carga.

En esta guía, le explicaremos cómo configurar hosts virtuales Apache en un Ubuntu 16.04 VPS o superior. Durante este proceso, aprenderás a servir diferentes contenidos a diferentes visitantes dependiendo de los dominios que estén solicitando.

Para los propósitos de esta guía, nuestra configuración hará un host virtual por **ejemplo.com** y otro para **test.com**. Estos serán referenciados a lo largo de la guía, pero usted debe sustituir sus propios dominios o valores mientras los sigue.

Le mostraremos cómo editar el archivo de hosts locales más adelante para probar la configuración si está usando valores ficticios. Esto le permitirá probar su configuración desde su ordenador personal, aunque su contenido no estará disponible a través del nombre de dominio para otros visitantes.

## **Paso uno: Crear la estructura de directorios**

El primer paso que vamos a dar es crear una estructura de directorios que contenga los datos del sitio que vamos a servir a los visitantes.

Nuestra raíz de documento (el directorio de nivel superior que Apache busca para encontrar el contenido a servir) se establecerá en directorios individuales bajo el directorio **/var/www**. Aquí crearemos un directorio para los dos hosts virtuales que planeamos crear.

Dentro de cada uno de estos directorios, crearemos una carpeta **public\_html** que contendrá nuestros archivos actuales. Esto nos da cierta flexibilidad en nuestro alojamiento.

Por ejemplo, para nuestros sitios, vamos a hacer nuestros directorios así:

sudo mkdir -p /var/www/ejemplo.com/public\_html  
 sudo mkdir -p /var/www/test.com/public\_html

Las partes en rojo representan los nombres de dominio que queremos servir desde nuestro VPS.

## **Paso Dos: Otorgar Permisos**

Ahora tenemos la estructura de directorios para nuestros archivos, pero son propiedad de nuestro usuario root. Si queremos que nuestro usuario habitual pueda modificar los archivos de nuestros directorios web, podemos cambiar la propiedad de los mismos:

sudo chown -R $USER:$USER /var/www/ejemplo.com/public\_html  
 sudo chown -R $USER:$USER /var/www/test.com/public\_html

La variable **$USER** tomará el valor del usuario con el que está conectado actualmente. Al hacer esto, nuestro usuario regular ahora es dueño de los subdirectorios **public\_html** donde almacenaremos nuestro contenido.

También deberíamos modificar un poco nuestros permisos para asegurarnos de que se permite el acceso de lectura al directorio web general y a todos los archivos y carpetas que contiene para que las páginas se puedan servir correctamente:

sudo chmod -R 755 /var/www

Su servidor web debería tener ahora los permisos que necesita para servir contenido, y su usuario debería ser capaz de crear contenido dentro de las carpetas necesarias.

## **Paso tres: Crear páginas de demostración para cada host virtual**

Tenemos nuestra estructura de directorios en su lugar. Vamos a crear algún contenido para servir.

Sólo vamos a hacer una demostración, así que nuestras páginas serán muy sencillas. Vamos a hacer una página **index.html** para cada sitio.

Empecemos con **ejemplo.com**. Podemos abrir un archivo **index.html** en nuestro editor escribiendo:

nano /var/www/ejemplo.com/public\_html/index.html

En este archivo, cree un documento HTML simple que indique el sitio al que está conectado. Mi archivo tiene este aspecto:

/var/www/ejemplo.com/public\_html/index.html

<html>  
 <head>  
 <title>Bienvenido a Ejemplo.com!</title>  
 </head>  
 <body>  
 <h1>Correcto! El Virtual Host ejemplo.com esta funcionando!</h1>  
 </body>  
</html>

Guarde y cierre el archivo cuando haya terminado.

Podemos copiar este archivo para utilizarlo como base para nuestro segundo sitio escribiendo:

cp /var/www/ejemplo.com/public\_html/index.html /var/www/test.com/public\_html/index.html

Entonces podemos abrir el archivo y modificar la información relevante:

nano /var/www/test.com/public\_html/index.html

/var/www/test.com/public\_html/index.html

<html>  
 <head>  
 <title>Bienvenido a Test.com!</title>  
 </head>  
 <body> <h1>Correcto! El Virtual Host test.com esta funcionando!</h1>  
 </body>  
</html>

Guarde y cierre este archivo también. Ahora tiene las páginas necesarias para probar la configuración del host virtual.

## **Paso cuatro: Crear nuevos archivos de host virtual**

Los archivos de host virtual son los archivos que especifican la configuración real de nuestros hosts virtuales y dictan cómo el servidor web Apache responderá a las diferentes peticiones de dominio.

Apache viene con un archivo de host virtual por defecto llamado **000-default.conf** que podemos usar como punto de partida. Vamos a copiarlo para crear un archivo de host virtual para cada uno de nuestros dominios.

Comenzaremos con un dominio, lo configuraremos, lo copiaremos para nuestro segundo dominio y luego haremos los ajustes necesarios. La configuración predeterminada de Ubuntu requiere que cada archivo del host virtual termine en **.conf**.

### **Crear el primer archivo de host virtual**

Comience por copiar el archivo para el primer dominio:

sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/ejemplo.com.conf

Abra el nuevo archivo en su editor con privilegios de root:

sudo nano /etc/apache2/sites-available/ejemplo.com.conf

El archivo tendrá un aspecto parecido a este (he eliminado los comentarios aquí para que el archivo sea más accesible):

/etc/apache2/sites-available/ejemplo.com.conf

<VirtualHost \*:80>  
 ServerAdmin webmaster@localhost  
 DocumentRoot /var/www/html  
 ErrorLog ${APACHE\_LOG\_DIR}/error.log  
 CustomLog ${APACHE\_LOG\_DIR}/access.log combined  
</VirtualHost>

Como puede ver, no hay mucho aquí. Vamos a personalizar los elementos aquí para nuestro primer dominio y agregar algunas directivas adicionales. Esta sección del host virtual coincide con cualquier petición que se realice en el puerto **80**, el puerto HTTP predeterminado.

Primero, necesitamos cambiar la directiva **ServerAdmin** a un correo electrónico a través del cual el administrador del sitio pueda recibir correos electrónicos.

ServerAdmin [admin@ejemplo.com](mailto:admin@example.com)

Después de esto, tenemos que añadir dos directivas. El primero, llamado **ServerName**, establece el dominio base que debe coincidir con esta definición de host virtual. Lo más probable es que este sea su dominio. El segundo, llamado **ServerAlias**, define otros nombres que deberían coincidir como si fueran el nombre base. Esto es útil para emparejar los hosts que definió, como **www**:

ServerName ejemplo.com  
ServerAlias [www.ejemplo.com](http://www.ejemplo.com)

La única otra cosa que necesitamos cambiar para un archivo de host virtual básico es la ubicación de la raíz del documento para este dominio. Ya hemos creado el directorio que necesitamos, así que sólo necesitamos alterar la directiva **DocumentRoot** para que refleje el directorio que creamos:

DocumentRoot /var/www/ejemplo.com/public\_html

En total, nuestro archivo de host virtual debería tener este aspecto:

/etc/apache2/sites-available/ejemplo.com.conf

<VirtualHost \*:80>  
 ServerAdmin admin@ejemplo.com  
 ServerName ejemplo.com  
 ServerAlias www.ejemplo.com  
 DocumentRoot /var/www/ejemplo.com/public\_html  
 ErrorLog ${APACHE\_LOG\_DIR}/error.log  
 CustomLog ${APACHE\_LOG\_DIR}/access.log combined  
</VirtualHost>

Guarde y cierre el archivo.

### **Copiar el primer host virtual y personalizarlo para el segundo dominio**

Ahora que hemos establecido nuestro primer archivo de host virtual, podemos crear el segundo copiando ese archivo y ajustándose según sea necesario.

Empiece por copiarlo:

sudo cp /etc/apache2/sites-available/ejemplo.com.conf /etc/apache2/sites-available/test.com.conf

Abra el nuevo archivo con privilegios de root en su editor:

sudo nano /etc/apache2/sites-available/test.com.conf

Ahora necesita modificar todas las piezas de información para hacer referencia a su segundo dominio. Cuando haya terminado, puede parecerse a esto:

/etc/apache2/sites-available/test.com.conf

<VirtualHost \*:80>  
 ServerAdmin admin@test.com  
 ServerName test.com  
 ServerAlias www.test.com  
 DocumentRoot /var/www/test.com/public\_html  
 ErrorLog ${APACHE\_LOG\_DIR}/error.log  
 CustomLog ${APACHE\_LOG\_DIR}/access.log combined  
</VirtualHost>

Guarde y cierre el archivo cuando haya terminado.

### **Alojamiento virtual basado en IP - Alternativa**

Aquí, necesitará múltiples IPs que apuntarán a los dominios únicos en el servidor. Normalmente, usted tendría una sola IP para un servidor. Sin embargo para poder hacer uso de esta opción usted tiene que crear múltiples direcciones IP para un solo servidor que va a ser usado para alojar múltiples dominios. Vamos a seguir la técnica de Aliasing IP.

**¿Qué es el Aliasing IP?**

La creación de múltiples IP para una sola interfaz de red se llama IP aliasing Donde un solo nodo tiene las múltiples conexiones a la red. La ventaja es que el sistema único se puede mostrar como sistemas múltiples utilizando las diferentes direcciones IP.

#### **1.- Creación de múltiples direcciones para un único servidor.**

Para crear hosting virtual basado en IP, debe tener varias IPs para un único servidor.

Para crear múltiples IPs en una sola interfaz de red, debe editar la interfaz de red y añadir algunos códigos.

Primero, ejecute el siguiente comando.

sudo nano /etc/network/interfaces

Luego agregue las siguientes líneas en el archivo.

auto eth0  
 iface eth0 inet static  
 address 192.168.1.227  
 netmask 255.255.255.0  
 gateway 192.168.1.1  
 dns-nameservers 8.8.8.8  
  
 auto eth0:1  
 iface eth0:1 inet static  
 address 192.168.1.228  
 netmask 255.255.255.0

Guarde y cierre el archivo.

Ahora, tiene que reiniciar el servicio de red, por lo que la actualización del archivo tendrá efecto.

sudo /etc/init.d/networking restart  
 sudo ifup eth0:1

#### **2.- Configurar Múltiples instancias para el servidor Apache**

Ha creado varias IPs. Ahora tiene que hacer que todos ellos escuchen a puertos específicos.

Para ello, debe tener varios puertos.

Para añadir varios puertos, edite el archivo de configuración de Apache.

sudo nano /etc/apache2/ports.conf

Añada las líneas en el archivo

Listen 192.168.1.227:80  
 Listen 192.168.1.228:80  
 Listen 192.168.1.228:8080

En la última línea podemos ver que se comparte IP con un sitio, pero se utiliza un puerto distinto (8080). Si tuviéramos un tercer sitio podríamos asignarle a ese puerto, simplemente modificando el puerto en el archivo de configuración del host virtual como más adelante se muestra.

Guarda y cierra el archivo.

Ahora reinicie Apache

sudo systemctl apache2 restart

#### **3.- Modificar los archivos del Virtual Host**

Modificaremos los archivos .conf que ya tenemos de nuestras páginas de ejemplo para poder asignarles una IP

sudo nano /etc/apache2/sites-available/ejemplo.com.conf

Y procedemos a insertar la IP y puerto del Host Virtual.

/etc/apache2/sites-available/ejemplo.com.conf

<VirtualHost 192.168.1.227:80>  
 ServerAdmin admin@ejemplo.com  
 ServerName ejemplo.com  
 ServerAlias www.ejemplo.com  
 DocumentRoot /var/www/ejemplo.com/public\_html  
 ErrorLog ${APACHE\_LOG\_DIR}/error.log  
 CustomLog ${APACHE\_LOG\_DIR}/access.log combined  
</VirtualHost>

Guardamos y cerramos el archivo.

Hacemos lo mismo para nuestro segundo sitio.

sudo nano /etc/apache2/sites-available/test.com.conf

/etc/apache2/sites-available/test.com.conf

<VirtualHost 192.168.1.228:80>  
 #### ^... Aqui se puede modificar el puerto por el 8080  
 ServerAdmin admin@test.com  
 ServerName test.com  
 ServerAlias www.test.com  
 DocumentRoot /var/www/test.com/public\_html  
 ErrorLog ${APACHE\_LOG\_DIR}/error.log  
 CustomLog ${APACHE\_LOG\_DIR}/access.log combined  
</VirtualHost>

Guardamos y cerramos el archivo.

### **Configuración del puerto de operación**

A veces se requiere que Apache2 escuche y se comunique no a través de su puerto por defecto, sino a través de otro puerto.

Por ejemplo, si quiere que el servidor HTTP Apache2 se sitúe detrás de un servidor proxy, entonces el servidor proxy debe estar configurado para escuchar en el puerto por defecto 80. En este caso, Apache2 también debe estar configurado para escuchar en un puerto diferente al 80, ya que dos servicios no pueden ser asignados a un puerto para escuchar, o habría colisiones en la información.

A continuación se mostrará cómo cambiar fácilmente el puerto por defecto de Apache2 de 80 a otro, como el 8082. De esta manera, otros servicios pueden usar el puerto 80 para comunicarse.

Cuando esté listo para hacer este cambio, siga los pasos a continuación:

#### **1.- Activar la escucha del puerto**

Primero tenemos que hacer que Apache sepa por cual puerto queremos que esté a la escucha. Para poder hacer esto, introduce el siguiente comando en consola.

sudo nano /etc/apache2/ports.conf

A continuación se muestra el archivo con los puertos de configuración, aquí se puede modificar el puerto por default que seria el 80. Se puede modificar para cambiar el puerto o se puede agregar otro puerto de escucha.

/etc/apache2/ports.conf

Listen 81

Listen 8080  
<IfModule ssl\_module>  
 Listen 443  
</IfModule>  
<IfModule mod\_gnutls.c>  
 Listen 443  
</IfModule>

Esto hará que el servidor acepte las conexiones en los puertos 81 y 8080 en todas las interfaces. Para hacer que el servidor acepte conexiones en el puerto 8080 de una interfaz específica, solo tiene que incluir la dirección ip de la interfaz de red. Como se muestra a continuación.

/etc/apache2/ports.conf

Listen 81

Listen 192.168.1.228:8080  
<IfModule ssl\_module>  
 Listen 443  
</IfModule>  
<IfModule mod\_gnutls.c>  
 Listen 443  
</IfModule>

Esto es de utilidad si el servidor cuenta con varias direcciones IP de red.

Guarde y cierre el archivo.

#### **2.- Modificar el puerto en el Virtual Host del sitio**

Ahora que el servidor acepta conexiones por los puertos especificados, es necesario modificar los archivos .conf de los sitios actualmente habilitados, para esto abrimos el archivo de configuración de los sitios que deseemos editar.

sudo nano /etc/apache2/sites-available/ejemplo.com.conf

Y modificamos el puerto del host.

/etc/apache2/sites-available/ejemplo.com.conf

<VirtualHost \*:81>  
 ServerAdmin admin@ejemplo.com  
 ServerName ejemplo.com  
 ServerAlias www.ejemplo.com  
 DocumentRoot /var/www/ejemplo.com/public\_html  
 ErrorLog ${APACHE\_LOG\_DIR}/error.log  
 CustomLog ${APACHE\_LOG\_DIR}/access.log combined  
</VirtualHost>

Si queremos además de limitarlo a una interfaz de red podemos modificarlo de la siguiente manera:

sudo nano /etc/apache2/sites-available/test.com.conf

/etc/apache2/sites-available/test.com.conf

<VirtualHost 192.168.1.228:8080>  
 ServerAdmin admin@test.com  
 ServerName test.com  
 ServerAlias www.test.com  
 DocumentRoot /var/www/test.com/public\_html  
 ErrorLog ${APACHE\_LOG\_DIR}/error.log  
 CustomLog ${APACHE\_LOG\_DIR}/access.log combined  
</VirtualHost>

Guardamos y cerramos el archivo.

#### **3.- Reiniciar el servidor**

Ahora ya solo falta reiniciar el servidor para hacer los cambios válidos.

sudo systemctl restart apache2

Podemos verificar los cambios realizados con el siguiente comando

sudo netstat -tulpn | grep :81

Mostrará una salida como la siguiente:

tcp6 0 0 :::8090 :::\* LISTEN 4066/apache2

## **Paso cinco: Habilitar los nuevos archivos del host virtual**

Ahora que hemos creado nuestros archivos de host virtual, debemos habilitarlos. Apache incluye algunas herramientas que nos permiten hacer esto.

Podemos usar la herramienta **a2ensite** para habilitar cada uno de nuestros sitios de esta manera:

sudo a2ensite ejemplo.com.conf

sudo a2ensite test.com.conf

A continuación, desactive el sitio por defecto definido en **000-default.conf**:

sudo a2dissite 000-default.conf

Cuando termine, debe reiniciar Apache para que estos cambios surtan efecto:

sudo systemctl restart apache2

En otra documentación, también puede ver un ejemplo utilizando el comando service:

sudo service apache2 restart

Este comando seguirá funcionando, pero puede no dar la salida que está acostumbrado a ver en otros sistemas, ya que ahora es una envoltura alrededor del systemd systemd **systemctl**.

## **Paso seis: configurar el archivo de hosts locales (opcional)**

Si no ha estado utilizando nombres de dominio reales de su propiedad para probar este procedimiento y ha estado utilizando algunos dominios de ejemplo en su lugar, puede al menos probar la funcionalidad de este proceso modificando temporalmente el archivo **hosts** en su equipo local.

Esto interceptará cualquier petición de los dominios que hayas configurado y los dirigirá a tu servidor VPS, tal y como haría el sistema DNS si estuvieras usando dominios registrados. Sin embargo, esto sólo funcionará desde su computadora, y es simplemente útil para propósitos de prueba.

Asegúrese de que está operando en su ordenador local para estos pasos y no en su servidor VPS. Necesitará conocer la contraseña administrativa del equipo o ser miembro del grupo administrativo.

Si se encuentra en un equipo Mac o Linux, edite el archivo local con privilegios administrativos escribiendo:

sudo nano /etc/hosts

**Para Windows 10 y 8**

1. Pulse la tecla Windows.
2. Escriba Bloc de notas en el campo de búsqueda.
3. En los resultados de la búsqueda, haga clic con el botón secundario en Bloc de notas y seleccione Ejecutar como administrador.
4. Desde el Bloc de notas, abra el siguiente archivo: c:\Windows\System32\Drivers\etc\hosts
5. Realice las modificaciones necesarias en el fichero.
6. Haga clic en Archivo > Guardar para guardar los cambios.

**Para Windows 7 y Vista**

1. Haga clic en Inicio > Todos los programas > Accesorios.
2. Haga clic con el botón derecho en Bloc de notas y seleccione Ejecutar como administrador.
3. Haga clic en Continuar en la ventana de UAC de Windows necesita su permiso.
4. Cuando se abra el Bloc de notas, haga clic en Archivo > Abrir.
5. En el campo Nombre de archivo, escriba C:\Windows\System32\Drivers\etc\hosts.
6. Haga clic en Abrir.
7. Realice las modificaciones necesarias en el fichero.
8. Haga clic en Archivo > Guardar para guardar los cambios.

Los detalles que necesita agregar son la dirección IP pública de su servidor VPS seguida del dominio que desea usar para llegar a ese VPS.

Para los dominios que he utilizado en esta guía, suponiendo que mi dirección IP VPS es 192.168.1.227, podría añadir las siguientes líneas en la parte inferior de mi archivo **hosts**:

/etc/hosts

127.0.0.1 localhost  
127.0.1.1 guest-desktop  
192.168.1.227 ejemplo.com  
192.168.1.228 test.com

Esto dirigirá cualquier solicitud, por **ejemplo.com** y **test.com** en nuestro ordenador y la enviará a nuestro servidor al número **192.168.1.227**. Esto es lo que queremos si no somos realmente los propietarios de estos dominios para probar nuestros hosts virtuales.

Save and close the file.

## **Paso siete: Pruebe sus resultados**

Ahora que tiene configurados sus hosts virtuales, puede probar su configuración fácilmente yendo a los dominios que configuró en su navegador web:

http://ejemplo.com

Debería ver una página que se vea así:



Del mismo modo, si puede visitar tu segunda página:

http://test.com

Verá el archivo que creó para su segundo sitio:



Sí ambos sitios funcionan bien, ha configurado correctamente dos hosts virtuales en el mismo servidor.

Si ha ajustado el archivo de hosts de su ordenador personal, es posible que desee eliminar las líneas que ha añadido ahora que ha comprobado que su configuración funciona. Esto evitará que el archivo de hosts se llene con entradas que no son realmente necesarias.

Si necesita acceder a este largo plazo, considere comprar un nombre de dominio para cada sitio que necesite y configurarlo para que apunte a su servidor VPS.

# **htaccess**

.htaccess es un archivo de configuración para el servidor web Apache. Es una herramienta extremadamente poderosa que puede ser utilizada para modificar la configuración de Apache sin necesidad de editar los archivos de configuración de Apache. Las siguientes secciones describen cómo crear esta configuración y utilizarla para restringir los listados de directorios y las direcciones IP, así como para gestionar los redireccionamientos.

## **Habilitar .htaccess**

Por defecto, .htaccess no está disponible. Para habilitarlo, deberá editar el archivo de configuración.

Utilice un editor de texto para abrir el archivo de configuración:

sudo nano /etc/apache2/sites-available/ejemplo.com.conf

Después del bloque VirtualHost () agregue:

**/etc/apache2/sites-available/ejemplo.com.conf**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 | .... </VirtualHost> <Directory /var/www/ejemplo.com/public\_html>  Options Indexes FollowSymLinks  AllowOverride All  Require all granted </Directory> |

Guarde el archivo, luego reinicie apache:

sudo service apache2 restart

## **Restringir listados de directorios**

De forma predeterminada, alguien que visite su sitio web puede ver el directorio y la estructura de archivos, y obtener acceso a los archivos en el servidor web. Es una buena práctica restringir el acceso al directorio, de modo que un visitante de example.com tenga que estar familiarizado con los archivos del servidor para poder verlos. Una forma de restringir esto es a través de .htaccess.

### **Crear .htaccess**

Los sistemas CMS como WordPress crean configuraciones.htaccess por defecto. Esta guía asume que no existe ningún archivo.htaccess, por lo que tendrá que crear uno manualmente. Navegue hasta el directorio raíz de su sitio:

cd /var/www/ejemplo.com/public\_html

Crea un archivo .htaccess:

**/var/www/ejemplo.com/public\_html/.htaccess**

|  |  |
| --- | --- |
| 1 | Options -Indexes |

Ahora, si navega a su sitio, verá un mensaje de Prohibido. Deberá indicar específicamente el archivo o directorio que desea ver.

## **Restringir IPs**

Esta sección le guiará a través de la restricción de IPs específicas para acceder a su sitio. Esto es útil si desea impedir que ciertos visitantes visiten su sitio. También puede configurar esto para evitar que ciertas IPs accedan a ciertas secciones de su sitio.

Los subdirectorios pueden heredar la configuración de los archivos.htaccess en sus directorios principales si no son sustituidos por un archivo.htaccess separado en el subdirectorio. Los ejemplos de esta guía continuarán trabajando con un archivo.htaccess en el directorio raíz del proyecto. Debe considerar cuidadosamente en qué directorio se deben colocar las diferentes directivas .htaccess.

### **Bloquear IPs**

Cree o edite el archivo.htaccess ubicado en el directorio donde Apache alojará el sitio web:

cd /var/www/ejemplo.com/public\_html/

sudo nano .htaccess

Elimine la línea **Options -Indexes** de la sección anterior (si corresponde) y agregue las siguientes líneas para bloquear las direcciones IP de destino:

**/var/www/ejemplo.com/public\_html**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 | order allow,deny  # Esto bloquea la IP 172.15.23.9 deny from 172.15.23.9  # esto bloque todas las IPs desde 172.15.23.0 hasta 172.15.23.255 deny from 172.15.23 |

### **Permitir IPs**

Cree o edite el archivo.htaccess ubicado en el directorio web donde desea que se aplique esta configuración.

Añada las siguientes líneas para denegar todas las IPs excepto la IP específica y el grupo de IPs mencionadas en el comando:

**/var/www/ejemplo.com/public\_html**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 10 | order deny,allow  # Bloque todas las Ips Deny from all  # TEsto permitira la IP 172.15.23.9 allow from 172.15.23.9  # Esto permitira todas las IPs desde 172.15.23.0 hasta 172.15.23.255 allow from 172.15.23 |

## **Redireccionamiento**

Puede redirigir el tráfico utilizando la configuración de .htaccess. En el siguiente ejemplo, actualizará el archivo .htaccess del directorio raíz de su sitio web para redirigir a un visitante a **http://ejemplo.com/test2/index.html** si intenta visitar [**http://ejemplo.com/main.html**](http://example.com/main.html).

Abra el archivo .htaccess en el directorio raíz de su proyecto. Elimine todas las configuraciones existentes en este archivo y añada la siguiente línea:

**/var/www/ejemplo.com/public\_html/.htaccess**

|  |  |
| --- | --- |
| 1 | Redirect 301 /main.html /test1/index.html |

El primer parámetro después del comando 'Redirect' es el código de estado HTTP. Especificar un código de estado es útil para que el navegador sepa que la página se ha movido a una nueva ubicación. Si deja este parámetro en blanco, por defecto es un código 302 que indica que la redirección es temporal. Si se especifica 301, queda claro que la página en la ubicación solicitada se ha movido permanentemente a una nueva ubicación.

El siguiente parámetro es la ruta de Unix al archivo que se solicita en la URL. Este parámetro requiere que sea una ruta Unix y no una URL. La ruta debe ser la ubicación del archivo.htaccess donde se configura la configuración de redireccionamiento. El parámetro final indica hacia dónde desea que se redirija el visitante. En este caso, el tráfico se redirige a **/index.html**; para este segundo parámetro se acepta una ruta Unix o una URL HTTP.

Navegue hasta **ejemplo.com/main.html** en un navegador. Debería ver la url redireccionada a **ejemplo.com/iindex.html** en la barra de direcciones, y su archivo html de prueba debería aparecer.

## **Configurar la página de error 404 Error Page**

Cuando un visitante intenta acceder a una página o recurso que no existe (por ejemplo, siguiendo un enlace roto o escribiendo una URL incorrecta), el servidor responderá con un código de error 404. Es importante que los usuarios reciban retroalimentación que explique el error. Por defecto, Apache mostrará una página de error en caso de que se produzca un error 404. Sin embargo, la mayoría de los sitios ofrecen una página de error personalizada. Puede usar la configuración de .htaccess para que Apache sepa qué página de error desea que se muestre cada vez que un usuario intente acceder a una página inexistente.

Esto redirigirá todas las solicitudes de documentos inexistentes a una página en el directorio raíz del proyecto llamada `404.html. Abra el archivo.htaccess y añada la siguiente línea:

**/var/www/ejemplo.com/public\_html/.htaccess**

|  |  |
| --- | --- |
| 1 | ErrorDocument 404 /404.html |

Cree el archivo 404.html:

**/var/www/ejemplo.com/public\_html/404.html**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 | <!doctype html> <html>  <body>  404 Error: Page not found  </body> </html> |

En un navegador, navegue a una página que no exista, como www.ejemplo.com/noexite.html. Se debe mostrar el mensaje 404 del html personalizado.

# **Control de acceso**

## **Configuración del control de acceso**

Establecer reglas para el control de acceso al servidor tiene como fin aumentar la seguridad del mismo. Sin embargo ninguna medida de seguridad es infalible, por lo que es recomendable implementar diversos sistemas de seguridad en caso de que alguno llegase a fallar.

En esta sección se describirán tres métodos para controlar el acceso al servidor:

1. Bloqueo de IP
2. Bloqueo por segmento de red
3. Bloqueo por usuario y contraseña

## **Módulo mod\_authz\_host**

El módulo mod\_authz\_host provee las herramientas necesarias para controlar el acceso por distintos medios a través de la directiva Require[REFERENCIA Documentación Apache].

La directiva Require hace una comprobación para permitir o bloquear el acceso de acuerdo a diferentes parámetros. Su sintaxis básica es:

**Require <not> <tipo de regla> <parámetro>**

Donde *tipo de regla*se refiere a la comprobación a aplicar y el *parámetro* es el elemento a comprobar. Esta directiva puede ser usada en <Directory>, <Files> o <Location> para controlar el acceso a partes distintas del servidor

Por ejemplo:

**Require ip 192.168.1.25**

La directiva anterior permite el acceso al servidor sólo al dispositivo con la ip192.168.1.25

En el caso en el que se desee el resultado opuesto, es decir, bloquear la ip indicada, es necesario agregar **not** antes **ip:**

**Require not ip 192.168.1.25**

Los tipos de regla que soporta Directive son:

1. ip: Bloquea al dispositivo con la IP indicada
2. host: Bloquea el acceso de acuerdo al nombre host del cliente que lo solicita **Require host <host-name>**

## **Bloqueo por IP**

El bloqueo por IP tiene como fin prevenir el acceso al servidor HTTP desde un dispositivo específico. Su sintaxis es:

**Require <not> ip <dirección ip>**

Donde <not> es el parámetro opcional para bloquear la ip en lugar de permitir y <dirección ip> es la dirección ip del dispositivo deseado.

Ejemplo:

**Require ip 192.168.1.25**

Lo opuesto (Bloquear la ip) se logra agregando **not** antes de **ip:**

**Require not ip 192.168.1.25**

## **Bloqueo por segmento de red**

El bloqueo por segmento de red tiene una sintaxis similar al del bloqueo por ip y nos proporciona un mayor alcance.

Sintaxis:

**Require <not> ip <x.x.x.x/x>**

Donde *x.x.x.x/x* es la subred con segmento se desea controlar

El siguiente ejemplo permite el acceso sólo al segmento de red indicado:

**Require ip 192.168.16.0/24**

Por el contrario, la siguiente línea bloquea el acceso al mismo segmento de red:

**Require not ip 192.168.16.0/24**

## **Bloqueo por nombre de usuario**

### **mod\_authn\_core**

El módulo mod\_authn\_core de Apache provee las funciones necesarias para implementar control de acceso por nombre de usuario/contraseña [https://httpd.apache.org/docs/2.4/mod/mod\_authn\_core.html].

Utiliza la directiva AuthnProviderAlias para buscar nombres de usuarios y contraseñas en archivo de texto haciendo uso de htpasswd. Ejemplo:

**<AuthnProviderAlias file file1>**

**AuthUserFile “www/conf/passwords”**

**</AuthnProviderAlias>**

El ejemplo anterior crea el alias *file1* para el archivo localizado en *www/conf/passwords.*

El alias se utiliza en la directiva *AuthBasicProvider* junto a *Require* de la siguiente manera:

**AuthBasicProvider <alias>**

**AuthName <mensaje>**

**Require <parámetro>**

Donde:

* AuthBasicProvider es la directiva que utiliza el *alias* creado por *AuthBasicProvider*
* AuthName permite enviar un mensaje en la pantalla de ingreso
* Requiere indica el tipo de validación a ejecutar

Ejemplo:

**<Directory "/var/web/pages/secure">  
 AuthBasicProvider file1   
 AuthType Basic  
 AuthName "Protected Area"  
 Require valid-user  
</Directory>**

### **htpasswd**

*htpasswd* se encarga de gestionar las credenciales para acceder al servidor. Almacena las contraseñas en un archivo de texto plano usando encriptación. El tipo de encriptación que usa depende el sistema operativo en el que se ejecute.

Sintaxis de htpasswd:

**htpasswd <parámetros> nombre-usuario contraseña**

Ejemplo:

**htpasswd -c usuario1 123456**

La línea anterior crea un archivo, si este no existe (-c) y crea un nuevo usuario (usuario1) con la contraseña 123456

# **Aplicación de Certificado para HTTPS**

TLS, o seguridad de la capa de transporte, y su predecesor SSL, que significa capa de sockets seguros, son protocolos web utilizados para envolver el tráfico normal en una envoltura protegida y encriptada.

Usando esta tecnología, los servidores pueden enviar tráfico de forma segura entre el servidor y los clientes sin la posibilidad de que los mensajes sean interceptados por terceros. El sistema de certificados también ayuda a los usuarios a verificar la identidad de los sitios con los que se están conectando.

## **Certificado auto firmado**

**¿Cuándo utilizar un certificado autofirmado?**

Nunca debe utilizar un certificado autofirmado en un sitio de comercio electrónico o en cualquier otro sitio que transfiera información personal valiosa como tarjetas de crédito, números de seguro social, etc.

* Los certificados autofirmados se pueden utilizar en un servidor de desarrollo Apache. No hay necesidad de gastar dinero extra comprando un certificado de confianza cuando está desarrollando o probando una aplicación.
* Los certificados autofirmados se pueden utilizar en una intranet. Cuando los clientes sólo tienen que pasar por una intranet local para llegar al servidor, no hay prácticamente ninguna posibilidad de un ataque de hombre en el medio.
* Los certificados autofirmados se pueden utilizar en sitios personales con pocos visitantes. Si tiene un sitio personal pequeño que transfiere información no crítica, hay muy pocos incentivos para que alguien ataque la conexión.

### **Paso 1: Crear el certificado SSL**

TLS/SSL funciona utilizando una combinación de un certificado público y una clave privada. La clave SSL se mantiene en secreto en el servidor. Se utiliza para encriptar el contenido enviado a los clientes. El certificado SSL se comparte públicamente con cualquier persona que solicite el contenido. Puede utilizarse para descifrar el contenido firmado por la clave SSL asociada.

Podemos crear un par de claves y certificados autofirmados con OpenSSL en un solo comando:

sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/apache-selfsigned.key -out /etc/ssl/certs/apache-selfsigned.crt

Se le harán una serie de preguntas. Antes de repasar eso, echemos un vistazo a lo que está sucediendo en el comando que estamos emitiendo:

* **openssl:** es la herramienta básica de la línea de comandos para crear y gestionar certificados, claves y otros archivos de OpenSSL.
* **req:** Este subcomando especifica que queremos utilizar la gestión de solicitud de firma de certificado (CSR) X.509. El "X.509" es un estándar de infraestructura de clave pública al que se adhieren SSL y TLS para la gestión de claves y certificados. Queremos crear un nuevo cert X.509, así que estamos usando este subcomando.
* **-x509:** Esto modifica aún más el subcomando anterior al indicar a la utilidad que queremos hacer un certificado autofirmado en lugar de generar una solicitud de firma de certificado, como ocurriría normalmente.
* **-nodes:** Esto le indica a OpenSSL que omita la opción de asegurar nuestro certificado con una frase de contraseña. Necesitamos que Apache sea capaz de leer el archivo, sin intervención del usuario, cuando el servidor se inicie. Una frase de contraseña evitaría que esto ocurriera porque tendríamos que introducirla después de cada reinicio.
* **-days 365:** Esta opción establece el período de tiempo durante el cual el certificado se considerará válido. Lo fijamos para un año aquí.
* **-newkey rsa:2048:** Esto especifica que queremos generar un nuevo certificado y una nueva clave al mismo tiempo. No creamos la clave necesaria para firmar el certificado en un paso anterior, por lo que debemos crearla junto con el certificado. La porción rsa:2048 le dice que haga una clave RSA de 2048 bits de longitud.
* **-keyout:** Esta línea le indica a OpenSSL dónde colocar el archivo de clave privada generado que estamos creando.
* **-out:** Indica a OpenSSL dónde colocar el certificado que estamos creando.

Como dijimos anteriormente, estas opciones crearán un archivo de clave y un certificado. Se nos harán algunas preguntas sobre nuestro servidor para poder insertar la información correctamente en el certificado.

Llene las instrucciones apropiadamente. La línea más importante es la que solicita el **Common Name (e.g. server FQDN or YOUR name)**. Necesita introducir el nombre de dominio asociado con su servidor o, más probablemente, la dirección IP pública de su servidor.

La totalidad de las indicaciones se parecerá a esto:

**Output  
 Country Name (2 letter code) [AU]:US  
 State or Province Name (full name) [Some-State]:New York  
 Locality Name (eg, city) []:New York City  
 Organization Name (eg, company) [Internet Widgits Pty Ltd]:Bouncy Castles, Inc.  
 Organizational Unit Name (eg, section) []:Ministry of Water Slides  
 Common Name (e.g. server FQDN or YOUR name) []:server\_IP\_address  
 Email Address []:admin@your\_domain.com**

Los dos archivos que ha creado se colocarán en los subdirectorios apropiados del directorio **/etc/ssl**.

Mientras usamos OpenSSL, también debemos crear un grupo fuerte de Diffie-Hellman, que se utiliza en la negociación de Perfect Forward Secrecy con los clientes.

Podemos hacer esto escribiendo:

sudo openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048

Esto puede tardar unos minutos, pero cuando esté hecho tendrás un grupo DH fuerte en **/etc/ssl/certs/dhparam.pem** que podemos usar en nuestra configuración.

### **Paso 2: Configurar Apache para usar SSL**

Hemos creado nuestros archivos de claves y certificados bajo el directorio **/etc/ssl**. Ahora sólo tenemos que modificar nuestra configuración de Apache para aprovecharlas.

Haremos algunos ajustes a nuestra configuración:

1. Crearemos un fragmento de configuración para especificar la configuración predeterminada de SSL.
2. Modificaremos el archivo SSL Apache Virtual Host incluido para que apunte a nuestros certificados SSL generados.
3. (Recomendado) Modificaremos el archivo no cifrado del Host Virtual para redirigir automáticamente las peticiones al Host Virtual cifrado.

Cuando terminemos, debemos tener una configuración SSL segura.

#### **Crear un fragmento de configuración de Apache con una configuración de cifrado fuerte**

Primero, crearemos un fragmento de configuración de Apache para definir algunos ajustes de SSL. Esto configurará Apache con un potente conjunto de cifrado SSL y habilitará algunas funciones avanzadas que ayudarán a mantener nuestro servidor seguro. Los parámetros que estableceremos pueden ser utilizados por cualquier Anfitrión Virtual que habilite SSL.

Cree un nuevo fragmento en el directorio **/etc/apache2/conf-available**. Nombraremos el archivo ssl-params.conf para dejar claro su propósito:

sudo nano /etc/apache2/conf-available/ssl-params.conf

Para configurar Apache SSL de forma segura, utilizaremos las recomendaciones de Remy van Elst en el sitio Cipherli.st. Este sitio está diseñado para proporcionar una configuración de cifrado fácil de usar para el software más popular.

Los ajustes sugeridos en el sitio vinculados a los anteriores ofrecen una gran seguridad. A veces, esto se produce a costa de una mayor compatibilidad con el cliente. Si necesita dar soporte a clientes mayores, hay una lista alternativa a la que puede acceder haciendo clic en el enlace de la página "Yes, give me a ciphersuite that works with legacy / old software" (Sí, dame una ciphersuite que funcione con software antiguo / antiguo). Esta lista puede sustituir a los elementos que se copian a continuación.

La elección de la configuración que utilice dependerá en gran medida de lo que necesite apoyar. Ambos proporcionarán una gran seguridad.

Para nuestros fines, podemos copiar la configuración proporcionada en su totalidad. Sólo haremos dos pequeños cambios.

Establezca la directiva **SSLOpenSSLConfCmd DHParameters** para que apunte al archivo Diffie-Hellman que generamos anteriormente. Además, tómese un momento para leer sobre HTTP Strict Transport Security, o HSTS, y específicamente sobre la funcionalidad de "precarga". La precarga de HSTS proporciona mayor seguridad, pero puede tener consecuencias de gran alcance si se activa accidentalmente o se activa incorrectamente. En esta guía, no preinstalaremos los ajustes, pero usted puede modificarlos si está seguro de que entiende las implicaciones:

/etc/apache2/conf-available/ssl-params.conf

# from https://cipherli.st/  
# and https://raymii.org/s/tutorials/Strong\_SSL\_Security\_On\_Apache2.html  
  
SSLCipherSuite EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH  
SSLProtocol All -SSLv2 -SSLv3  
SSLHonorCipherOrder On  
# Disable preloading HSTS for now. You can use the commented out header line that includes  
# the "preload" directive if you understand the implications.  
#Header always set Strict-Transport-Security "max-age=63072000; includeSubdomains; preload"  
Header always set Strict-Transport-Security "max-age=63072000; includeSubdomains"  
Header always set X-Frame-Options DENY  
Header always set X-Content-Type-Options nosniff  
# Requires Apache >= 2.4  
SSLCompression off   
SSLSessionTickets Off  
SSLUseStapling on   
SSLStaplingCache "shmcb:logs/stapling-cache(150000)"  
  
SSLOpenSSLConfCmd DHParameters "/etc/ssl/certs/dhparam.pem"

Guarde y cierre el archivo cuando haya terminado.

#### **Modificar el archivo de host virtual SSL predeterminado de Apache**

A continuación, modifiquemos **/etc/apache2/sites-available/default-ssl.conf**, el archivo predeterminado del Host Virtual Apache SSL. Si está utilizando un archivo de bloqueo de servidor diferente, sustituya su nombre en los siguientes comandos.

Antes de seguir adelante, hagamos una copia de seguridad del archivo original del Host Virtual SSL:

sudo cp /etc/apache2/sites-available/default-ssl.conf /etc/apache2/sites-available/default-ssl.conf.bak

Ahora, abra el archivo SSL Virtual Host para hacer los ajustes:

sudo nano /etc/apache2/sites-available/default-ssl.conf

Dentro, con la mayoría de los comentarios eliminados, el archivo del Host Virtual debería tener un aspecto parecido a este por defecto:

/etc/apache2/sites-available/default-ssl.conf

<IfModule mod\_ssl.c>  
 <VirtualHost \_default\_:443>  
 ServerAdmin webmaster@localhost  
  
 DocumentRoot /var/www/html  
  
 ErrorLog ${APACHE\_LOG\_DIR}/error.log  
 CustomLog ${APACHE\_LOG\_DIR}/access.log combined  
  
 SSLEngine on  
  
 SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem  
 SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key  
  
 <FilesMatch "\.(cgi|shtml|phtml|php)$">  
 SSLOptions +StdEnvVars  
 </FilesMatch>  
 <Directory /usr/lib/cgi-bin>  
 SSLOptions +StdEnvVars  
 </Directory>  
  
 # BrowserMatch "MSIE [2-6]" \  
 # nokeepalive ssl-unclean-shutdown \  
 # downgrade-1.0 force-response-1.0  
  
 </VirtualHost>  
</IfModule>

Haremos algunos ajustes menores al archivo. Estableceremos las cosas normales que querríamos ajustar en un archivo de Host Virtual (dirección de correo electrónico de ServerAdmin, ServerName, etc.), ajustaremos las directivas SSL para que apunten a nuestros archivos de certificados y claves, y descomentaremos una sección que proporciona compatibilidad para navegadores más antiguos.

Después de hacer estos cambios, el bloque del servidor debería tener un aspecto similar a este:

/etc/apache2/sites-available/default-ssl.conf

<IfModule mod\_ssl.c>  
 <VirtualHost \_default\_:443>  
 ServerAdmin your\_email@example.com  
 ServerName server\_domain\_or\_IP  
  
 DocumentRoot /var/www/html  
  
 ErrorLog ${APACHE\_LOG\_DIR}/error.log  
 CustomLog ${APACHE\_LOG\_DIR}/access.log combined  
  
 SSLEngine on  
  
 SSLCertificateFile /etc/ssl/certs/apache-selfsigned.crt  
 SSLCertificateKeyFile /etc/ssl/private/apache-selfsigned.key  
  
 <FilesMatch "\.(cgi|shtml|phtml|php)$">  
 SSLOptions +StdEnvVars  
 </FilesMatch>  
 <Directory /usr/lib/cgi-bin>  
 SSLOptions +StdEnvVars  
 </Directory>  
  
 BrowserMatch "MSIE [2-6]" \  
 nokeepalive ssl-unclean-shutdown \  
 downgrade-1.0 force-response-1.0  
  
 </VirtualHost>  
</IfModule>

Guarde y cierre el archivo cuando haya terminado.

##### **(Recomendado) Modificar el archivo de host virtual no cifrado para redirigirlo a HTTPS**

En su estado actual, el servidor proporcionará tanto tráfico HTTP no cifrado como HTTPS cifrado. Para mayor seguridad, se recomienda en la mayoría de los casos redirigir HTTP a HTTPS automáticamente. Si no desea o necesita esta funcionalidad, puede omitir esta sección con seguridad.

Para ajustar el archivo de Host Virtual sin encriptar para redirigir todo el tráfico a ser encriptado SSL, podemos abrir el archivo **/etc/apache2/sites-available/000-default.conf**:

sudo nano /etc/apache2/sites-available/000-default.conf

En el interior, dentro de los bloques de configuración de **VirtualHost**, sólo tenemos que añadir una directiva **Redirect**, apuntando todo el tráfico a la versión SSL del sitio:

/etc/apache2/sites-available/000-default.conf

<VirtualHost \*:80>  
 . . .  
  
 Redirect "/" "https://your\_domain\_or\_IP/"  
  
 . . .  
</VirtualHost>

Guarde y cierre el archivo cuando haya terminado.

### **Paso 3: Ajustar el cortafuegos**

Si tiene activado el cortafuegos **ufw**, tal y como recomiendan las guías de requisitos previos, es posible que deba ajustar la configuración para permitir el tráfico SSL. Afortunadamente, Apache registra algunos perfiles con **ufw** en el momento de la instalación.

Podemos ver los perfiles disponibles escribiendo:

sudo ufw app list

Debería ver una lista como esta:

**Output**

Available applications:

Apache

Apache Full

Apache Secure

OpenSSH

Puede ver la configuración actual escribiendo:

sudo ufw status

Si antes sólo permitía el tráfico HTTP normal, su salida podría tener este aspecto:

**Output**

Status: active

To Action From

-- ------ ----

OpenSSH ALLOW Anywhere

Apache ALLOW Anywhere

OpenSSH (v6) ALLOW Anywhere (v6)

Apache (v6) ALLOW Anywhere (v6)

Para permitir la entrada de tráfico HTTPS adicionalmente, podemos permitir el perfil "Apache Full" y luego eliminar la tolerancia de perfil "Apache" redundante:

sudo ufw allow 'Apache Full'

sudo ufw delete allow 'Apache'

Su estado debería ser el siguiente:

sudo ufw status

**Output**

Status: active

To Action From

-- ------ ----

OpenSSH ALLOW Anywhere

Apache Full ALLOW Anywhere

OpenSSH (v6) ALLOW Anywhere (v6)

Apache Full (v6) ALLOW Anywhere (v6)

### **Paso 4: Habilitar los cambios en Apache**

Ahora que hemos hecho nuestros cambios y ajustado nuestro firewall, podemos habilitar los módulos SSL y cabeceras en Apache, habilitar nuestro Host Virtual preparado para SSL y reiniciar Apache.

Podemos habilitar **mod\_ssl**, el módulo SSL de Apache, y **mod\_headers**, necesarios para algunas de las configuraciones de nuestro snippet SSL, con el comando **a2enmod**:

sudo a2enmod ssl

sudo a2enmod headers

A continuación, podemos habilitar nuestro Host Virtual SSL con el comando **a2ensite**:

sudo a2ensite default-ssl

También necesitaremos habilitar nuestro archivo **ssl-params.conf**, para leer los valores que establezcamos:

sudo a2enconf ssl-params

En este punto, nuestro sitio y los módulos necesarios están habilitados. Deberíamos comprobar que no hay errores de sintaxis en nuestros archivos. Podemos hacer esto escribiendo:

sudo apache2ctl configtest

Si todo tiene éxito, obtendrá un resultado parecido a este:

**Output**

Syntax OK

La primera línea es sólo un mensaje que indica que la directiva **ServerName** no está definida globalmente. Si desea deshacerse de ese mensaje, puede configurar **ServerName** al nombre de dominio o dirección IP de su servidor en **/etc/apache2/apache2.conf**. Esto es opcional ya que el mensaje no hará daño

Si su salida tiene **Sintaxis OK**, su archivo de configuración no tiene errores de sintaxis. Podemos reiniciar Apache de forma segura para implementar nuestros cambios:

sudo systemctl restart apache2

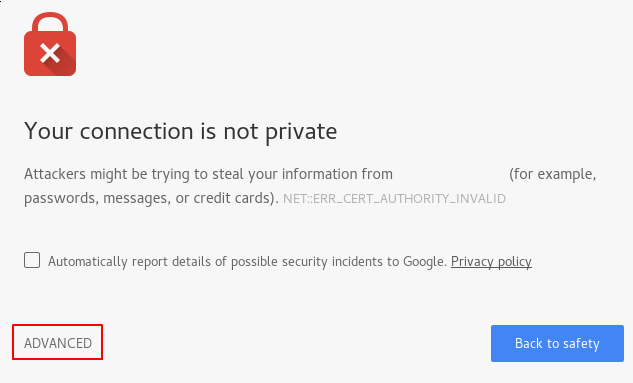
### **Paso 5: Comprobar el cifrado**

Ahora, estamos listos para probar nuestro servidor SSL.

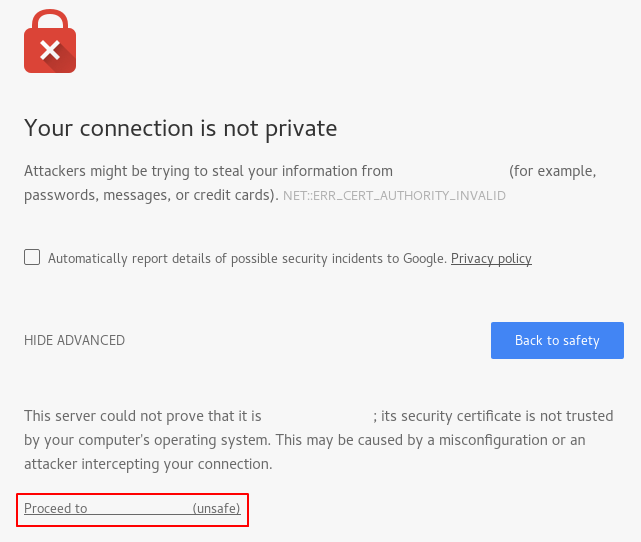
Abra su navegador web y escriba **https://** seguido del nombre de dominio o IP de su servidor en la barra de direcciones:

https://server\_domain\_or\_IP

Debido a que el certificado que creamos no está firmado por una de las autoridades de certificación de confianza de su navegador, es probable que vea una advertencia de aspecto aterrador como la que se muestra a continuación:



Esto es lo esperado y normal. Sólo nos interesa el aspecto de cifrado de nuestro certificado, no la validación de la autenticidad de nuestro host por parte de terceros. Haga clic en "AVANCED" y, a continuación, en el enlace que se proporciona para continuar con su anfitrión de todos modos:



Deberían llevarte a tu sitio. Si miras en la barra de direcciones del navegador, verás un candado con una "x" sobre él. En este caso, esto sólo significa que el certificado no puede ser validado. Sigue encriptando tu conexión.

Si configuró Apache para redirigir HTTP a HTTPS, también puede comprobar si el redireccionamiento funciona correctamente:

http://server\_domain\_or\_IP

Si el resultado es el mismo icono, significa que la redirección ha funcionado correctamente.

### **Paso 6: Cambiar a una redirección permanente**

Si su redirección ha funcionado correctamente y está seguro de que sólo desea permitir tráfico cifrado, debería modificar el Host Virtual Apache no cifrado de nuevo para que la redirección sea permanente.

Abra de nuevo el archivo de configuración de bloqueo del servidor:

sudo nano /etc/apache2/sites-available/000-default.conf

Encuentra la línea **Redirect** que añadimos antes. Agregar **permanent** a esa línea, lo que cambia la redirección de una redirección temporal de 302 a una redirección permanente de 301:

/etc/apache2/sites-available/000-default.conf

<VirtualHost \*:80>  
 . . .  
  
 Redirect permanent "/" "https://your\_domain\_or\_IP/"  
  
 . . .  
</VirtualHost>

Guarde y cierre el archivo.

Compruebe si hay errores de sintaxis en la configuración:

sudo apache2ctl configtest

Cuando esté listo, reinicie Apache para que la redirección sea permanente:

sudo systemctl restart apache2

## **Certificado firmado por una autoridad**

Let's Encrypt es una autoridad de certificación (CA) que proporciona una forma sencilla de obtener e instalar certificados TLS/SSL gratuitos, permitiendo así HTTPS cifrados en los servidores web. Simplifica el proceso al proporcionar un cliente de software, Certbot, que intenta automatizar la mayoría (si no todos) de los pasos requeridos. Actualmente, todo el proceso de obtención e instalación de un certificado está totalmente automatizado tanto en Apache como en Nginx.

En este tutorial, utilizará Certbot para obtener un certificado SSL gratuito para Apache en Ubuntu 18.04 y configurar su certificado para que se renueve automáticamente.

Este tutorial utilizará un archivo de host virtual Apache separado en lugar del archivo de configuración predeterminado. Recomendamos crear nuevos archivos de host virtual Apache para cada dominio porque ayuda a evitar errores comunes y mantiene los archivos por defecto como una configuración de emergencia.

### **Requisitos previos**

Para seguir este tutorial, necesitará:

* Un servidor Ubuntu 18.04 configurado siguiendo esta configuración inicial del servidor para el tutorial de Ubuntu 18.04, incluyendo un usuario sudo no root y un firewall.
* Un nombre de dominio completamente registrado. Este tutorial utilizará **ejemplo.com** en todo momento. Usted puede comprar un nombre de dominio en Namecheap, obtener uno gratis en Freenom, o utilizar el registrador de dominios de su elección.
* Ambos de los siguientes registros DNS configurados para su servidor. Puede seguir esta introducción a DigitalOcean DNS para obtener más detalles sobre cómo agregarlos.
  + Un registro A con **ejemplo.com** apuntando a la dirección IP pública de su servidor.
  + Un registro A con **www.ejemplo.com** apuntando a la dirección IP pública de su servidor.
* Apache se instala siguiendo el procedimiento de instalación de Apache en Ubuntu 18.04. Asegúrese de que tiene un archivo de host virtual para su dominio. Este tutorial usará **/etc/apache2/sites-available/ejemplo.com.conf** como ejemplo.

### **Paso 1 - Instalando Certbot**

El primer paso para usar Let's Encrypt para obtener un certificado SSL es instalar el software Certbot en su servidor.

Certbot está en un desarrollo muy activo, por lo que los paquetes de Certbot proporcionados por Ubuntu tienden a ser anticuados. Sin embargo, los desarrolladores de Certbot mantienen un repositorio de software de Ubuntu con versiones actualizadas, así que usaremos ese repositorio en su lugar.

Primero, añada el repositorio:

sudo add-apt-repository ppa:certbot/certbot

Tendrá que pulsar **ENTER** para aceptar.

Instale el paquete Apache de Certbot con **apt**:

sudo apt install python-certbot-apache

Certbot está ahora listo para usar, pero para que pueda configurar SSL para Apache, necesitamos verificar parte de la configuración de Apache.

### **Paso 2: Configuración del certificado SSL**

Certbot necesita ser capaz de encontrar el host virtual correcto en su configuración de Apache para que éste configure automáticamente SSL. Específicamente, lo hace buscando una directiva **ServerName** que coincida con el dominio para el que solicita un certificado.

Si ha seguido el paso de configuración del host virtual en el tutorial de instalación de Apache, debería tener un bloque VirtualHost para su dominio en **/etc/apache2/sites-available/ejemplo.com.conf** con la directiva ServerName ya configurada adecuadamente.

Para comprobarlo, abra el archivo de host virtual de su dominio utilizando **nano** o su editor de texto favorito:

sudo nano /etc/apache2/sites-available/ejemplo.com.conf

Busque la línea **ServerName** existente. Debería verse así:

/etc/apache2/sites-available/ejemplo.com.conf

...  
**ServerName** ejemplo.com;  
…

Si es así, salga de su editor y pase al siguiente paso.

Si no lo hace, actualícelo para que coincida. A continuación, guarde el archivo, salga del editor y verifique la sintaxis de las modificaciones de configuración:

sudo apache2ctl configtest

Si obtiene un error, vuelva a abrir el archivo del host virtual y compruebe si hay errores tipográficos o si faltan caracteres. Una vez que la sintaxis de su archivo de configuración sea correcta, vuelva a cargar Apache para cargar la nueva configuración:

sudo systemctl reload apache2

Certbot ahora puede encontrar el bloque correcto de VirtualHost y actualizarlo.

A continuación, actualicemos el firewall para permitir el tráfico HTTPS.

### **Paso 3 - Permitir HTTPS a través del cortafuegos**

Si tiene activado el cortafuegos **ufw**, como recomiendan las guías de requisitos previos, deberá ajustar la configuración para permitir el tráfico HTTPS. Afortunadamente, Apache registra algunos perfiles con **ufw** en el momento de la instalación.

Puede ver la configuración actual escribiendo:

sudo ufw status

Probablemente se verá así, lo que significa que sólo se permite el tráfico HTTP al servidor web:

Output

Status: active

To Action From

-- ------ ----

OpenSSH ALLOW Anywhere

Apache ALLOW Anywhere

OpenSSH (v6) ALLOW Anywhere (v6)

Apache (v6) ALLOW Anywhere (v6)

Para dejar entrar tráfico HTTPS adicionalmente, permita el perfil completo de Apache y elimine la tolerancia de perfil redundante de Apache:

sudo ufw allow 'Apache Full'

sudo ufw delete allow 'Apache'

Su estado debería ser como el siguiente:

sudo ufw status

**Output**

Status: active

To Action From

-- ------ ----

OpenSSH ALLOW Anywhere

Apache Full ALLOW Anywhere

OpenSSH (v6) ALLOW Anywhere (v6)

Apache Full (v6) ALLOW Anywhere (v6)

A continuación, ejecutemos Certbot y busquemos nuestros certificados.

### **Paso 4 - Obtención de un certificado SSL**

Certbot proporciona una variedad de formas de obtener certificados SSL a través de los plugins. El plugin de Apache se encargará de reconfigurar Apache y de recargar la configuración cuando sea necesario. Para usar este plugin, escriba lo siguiente:

sudo certbot --apache -d ejemplo.com -d www.ejemplo.com

Esto ejecuta **certbot** con el plugin **--apache**, usando **-d** para especificar los nombres para los que desea que el certificado sea válido.

Si es la primera vez que ejecuta **certbot**, se le pedirá que introduzca una dirección de correo electrónico y acepte los términos del servicio. Después de hacerlo, **certbot** se comunicará con el servidor Let's Encrypt, y luego ejecutará un desafío para verificar que usted controla el dominio para el que está solicitando un certificado.

Si esto tiene éxito, **certbot** le preguntará cómo desea configurar su configuración HTTPS:

**Output**

Please choose whether or not to redirect HTTP traffic to HTTPS, removing HTTP access.

-------------------------------------------------------------------------------

1: No redirect - Make no further changes to the webserver configuration.

2: Redirect - Make all requests redirect to secure HTTPS access. Choose this for

new sites, or if you're confident your site works on HTTPS. You can undo this

change by editing your web server's configuration.

-------------------------------------------------------------------------------

Select the appropriate number [1-2] then [enter] (press 'c' to cancel):

Seleccione su elección y pulse INTRO. La configuración se actualizará y Apache recargará para recoger la nueva configuración. certbot se cerrará con un mensaje que le indicará que el proceso se ha realizado correctamente y dónde están almacenados sus certificados:

**Output**

**IMPORTANT NOTES:**

**- Congratulations! Your certificate and chain have been saved at:**

**/etc/letsencrypt/live/ejemplo.com/fullchain.pem**

**Your key file has been saved at:**

**/etc/letsencrypt/live/ejemplo.com/privkey.pem**

**Your cert will expire on 2018-07-23. To obtain a new or tweaked**

**version of this certificate in the future, simply run certbot again**

**with the "certonly" option. To non-interactively renew \*all\* of**

**your certificates, run "certbot renew"**

**- Your account credentials have been saved in your Certbot**

**configuration directory at /etc/letsencrypt. You should make a**

**secure backup of this folder now. This configuration directory will**

**also contain certificates and private keys obtained by Certbot so**

**making regular backups of this folder is ideal.**

**- If you like Certbot, please consider supporting our work by:**

**Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate**

**Donating to EFF: https://eff.org/donate-le**

Sus certificados se descargan, instalan y cargan. Intente recargar su sitio web utilizando **https://** y observe el indicador de seguridad de su navegador. Debe indicar que el sitio está correctamente protegido, generalmente con un icono de candado verde. Si prueba su servidor utilizando la prueba de servidor de SSL Labs, obtendrá una calificación de **A**.

Terminemos probando el proceso de renovación.

### **Paso 5 - Verificando la renovación automática de Certbot**

Los certificados de Let's Encrypt sólo son válidos durante noventa días. Esto es para animar a los usuarios a automatizar su proceso de renovación de certificados. El paquete **certbot** que instalamos se encarga de esto añadiendo un script de renovación a **/etc/cron.d**. Este script se ejecuta dos veces al día y renovará automáticamente cualquier certificado que se encuentre dentro de los treinta días de su expiración.

Para probar el proceso de renovación, puede hacer un ensayo con **certbot**:

sudo certbot renew --dry-run

Si no ve ningún error, está todo listo. Cuando sea necesario, Certbot renovará sus certificados y recargará Apache para recoger los cambios. Si el proceso de renovación automatizado falla, Let's Encrypt enviará un mensaje al correo electrónico que usted especificó, advirtiéndole cuando su certificado está a punto de expirar.

# **Personalización de páginas de error**

## **Páginas de errores**

El principal propósito de las páginas de errores es mostrar al usuario cuando algo sucedió de manera incorrecta. Éstos pueden ir desde el intento de acceso a un recurso que no existe o al que no está autorizado, hasta un error del servidor.

Sin embargo, las páginas de errores por defecto suelen contener información sensible (Como la versión de Apache que estamos usando o el sistema operativo en el que se encuentra el servidor) lo cual puede conllevar un riesgo a la seguridad. No sólo eso, hablando de experiencia de usuario, este tipo de pantallas genéricas rompe el diseño de página web.

## **Tipos de errores**

Los errores de HTTP se definen usando un código de 3 números usando el formato XXX, donde X es un dígito cualquiera[1]. El primer número nos muestra la clasificación del error. Por ejemplo, un error que comience con el número 4 nos indica un error de cliente. El más conocido de estos errores es el 404, quel cual nos indica que el recurso al que tratamos de acceder no se encontró. A continuación se muestran ejemplos de clasificación de servidores.

|  |  |
| --- | --- |
| 403 | Acceso no autorizado |
| 404 | El recurso solicitado no se encontró |
| 500 | Error interno del servidor |
| 503 | Servicio no disponible |

## **Configuración de las páginas de errores**

La personalización de errores en *Apache* se define en el archivo de configuración usando la siguiente directiva:

**ErrorDocument <código en 3 dígitos> <acción a tomar>**

La acción a tomar puede ser[3]:

1. Una URL local
2. Una URL externa
3. Texto

Por ejemplo:

**ErrorDocument 404 /errors/not-found.html**

El comando anterior usa el recurso *not-found.html* que se encuentra dentro de la carpeta local */errors* cada vez que se presente un error 404.

La configuración para las páginas de errores pueden tener diferentes alcances. Por ejemplo, se pueden definir páginas de errores globales o para un virtualhost específico.

# **Bitácoras y mensajes de error**

Las bitácoras cumplen la función de registrar y almacenar los datos relacionados al servidor. Estos van desde interacciones normales, hasta errores en la operación.

Las bitácoras en Apache se almacenan en un archivo de texto plano con la extensión .log dentro del directorio */var/log/apache2/*. Las principales bitácoras de Apache son los siguientes:

**Access Log**

El access log registra los accesos hechos al servidor.

**Error Log**

Registra los errores ocurridos durante la operación del servidor.

## Configuración de bitácoras

Apache permite configurar diversos aspectos de las bitácoras del servidor desde el formato en el que son guardados, hasta el tiempo de rotación.

### Configuración de formato

El formato de las bitácoras se define usando la directiva *LogFormat.* Por ejemplo:

**LogFormat "%h %l %u %t \"%r\" %>s %b" common**

**CustomLog logs/access\_log common**

La primera línea define el formato de la bitácora y le asigna el apodo *common.* La segunda crea la bitácora haciendo uso del formato guardado en el apodo, en este caso, *common*.

La directiva *LogFormat* hace uso de directivas con el símbolo *%* para especificar la información que se requiere.

|  |  |
| --- | --- |
| %h | Dirección IP del cliente |
| %u | User Id determinado por autenticación |
| %t | Hora y fecha en la que fue recibida la petición |
| %r | Método usado, recurso solicitado y protocolo usado |
| %s | Status de la petición |
| %b | Tamaño del recurso solicitado |
| %T | Tiempo que tarda en responder el servidor en segundos. |
| %p | El puerto donde atiende la solicitud. |
| %v | Nombre del servidor. |

# **Referencias**

1. https://w3techs.com/technologies/details/ws-apache/all/all
2. <https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html> HTTP Status Codes
3. https://httpd.apache.org/docs/2.4/custom-error.html
4. <https://httpd.apache.org/docs/2.4/custom-error.html> Custom Error Responses
5. DigitalOcean 24/02/2019, <https://www.digitalocean.com/community/tutorials?q=apache+ubuntu>
6. PowerupHosting 24/02/2019, <https://poweruphosting.com/blog/ip-virtual-hosting-apache-ubuntu/>