Tarea 1: Configuración del sevidor de Apache

Programación Web

LOTHAR SOTO PALMA
LSOTPAL@CORREO.UGR.ES
Universidad de Granada
10 de marzo de 2017

Índice

1. Introducción

En este documento vamos a tratar de seleccionar las 10-15 directivas y las opciones más importantes que incluye apache para su correcto funcionamiento, conoceremos en que archivos incluir estas directivas y además los archivos de configuración que proporciona el servidor. También veremos que módulos usar en el sistema para mejorar su seguridad y desempeñar otros cometidos como la ejecución de páginas webs dinámicas usando lenguajes como php o python. Toda la documentación acerca de las directivas se ha tomado de la web oficial del servidor y la versión 2.4 del mismo.

2. ¿Qué es un servidor apache?

Se trata de un servidor web que sirve haciendo uso del protocolo HTTP, es de código abierto y es posible usarlo tanto en plataformas tipo Unix como en Microsoft Windows y Mac Os X. Comenzo su desarrollo en el año 1995 y se basó en el proyecto anterior NCSA aunque acabó siendo reescrito de forma completa. El proyecto es supervisado por la organización Apache Software Foundation, una organización sin ánimo de lucro creada para dar soporte software a los diversos proyectos que incluyen el nombre de Apache en ellos. [1]

Apache Software foundation es una organización descentralizada de desarrolladores que trabajan en una serie de proyectos de forma abierta (no son privativos).[2]

Es preciso mencionar que el uso de un servidor apache trae ciertas ventajas como que es modular, extensible (como consecuencia de la modularidad) y multi-plataforma.[1] Algunos de estos módulos son los siguientes:

- mod_ssl: Para las comunicaciones seguras vía TLS (HTTPS)
- mod_php: Para servir páginas dinámicas haciendo uso de PHP.
- mod_python: Para servir páginas dinámicas haciendo uso de Python.
- ...

3. Archivos de configuración

Los archivos de configuración del sistema son los siguientes: [3]

- **apache2.conf**: Se trata del archivo de configuración principal del servidor, normalmente incluye el archivo httpd.conf y no se debe ser modificado por el usuario. Se encuentra en */etc/apache2*.
- **httpd.conf**: Se trata del archivo de configuración principal del servidor a nivel de usuario, en este archivo es donde se encuentran las mayorías de directivas que permiten hacer funcionar apache de una determinada forma especificada normalmente por el usuario. Se encuentra en */etc/apache2*.
- **default-server.conf**: Se encuentra en */etc/apache2* y es archivo en el que se describen las directivas generales a aplicar a los hosts virtuales.
- ***.conf**: Se encuentra en Se encuentra en */etc/apache2/vhosts.d* e incluye todos los archivos de configuraciónes de cada host virtual que tengamos en nuestro servidor.

- **.htaccess**: Se trata del archivo de configuración distribuida, es un fichero especial que permite definir diferentes directivas de configuración para cada directorio sin necesidad de editar el archivo *httpd.conf*

4. Directivas httpd.conf

4.1. Directory

Características:

```
Sintaxis: <Directory directory-path> ... </Directory>
Módulo: Core
```

Descripción: Se trata de un conjunto de directivas que permiten aplicarse sobre determinados directorios y subdirectorios, suele usarse para restringir rutas en función del nivel de usuario en el sistema.

4.2. KeepAlive

Características:

```
Sintaxis: KeepAlive On | Off
Valor por defecto: On
Módulo: Core
```

Descripción: La directiva sirve para habilitar conexiones HTTP persistentes es decir tener sesiones HTTP abiertas de larga duración que permitan enviar múltiples peticiones a través de una misma conexión TCP, es decir las conexiones keepalive se mantien abiertas tras recibir una peticion del cliente y servirla de forma que la siguiente petición se aprovecha de la conexión que ya estaba abierta. Esto ocasiona que en algunos casos se consiga mejorar en casi un 50 % los tiempos de latencia de archivos html con muchas imágenes. Hay que diferenciar los clientes que usan HTTP/1.0 y HTTP/1.1, para los primeros solo se llevará a cabo una conexión que use keepalive si es solicitado por el mismo, en los segundos todas las conexiones se realizan usando keepalive por defecto a menos que se especifique lo contrario.

Una de las principales ventajas de habilitar esta opción es que no es necesario abrir una conexión para cada elemento de la pagina web, pero ¿Cuánto tiempo debemos mantener una conexión keepalive en el sistema? La respuesta que en mi opinión es más lógica para esto es el tiempo medio necesario para la carga de la web realizando diversas pruebas a través de diferentes velocidades de conexión y variando los clientes. Una web que no contenga muchos elementos multimedia se suele demorar actualmente pocos milisegundos en la carga por lo que podemos hacer el sistema más eficiente en tiempo a la hora de responder peticiones si ajustamos ese parámetro. El parámetro que ajusta esto se denomina KeepAliveTimeOut.

4.3. KeepAliveTimeOut

Características:

```
Sintaxis: KeepAlive num[s/ms]
Valor por defecto: 5 (2.4)
Módulo: Core
```

Descripción: Es la directiva que controla cuantos segundos el servidor de apache deberá esperar por las peticiones posteriores en una conexión persistente, controla cuanto tiempo estará abierta una conexión tipo keepalive, es posible la especificación del tiempo en milisegundos a partir de la versión 2.3.2 y posteriores añadiendo el prefijo ms, esto también puede hacerse en la directiva timeout. Establecer un valor muy elevado en esta directiva puede provocar problemas de rendimiento en servidores con una gran carga. Pero ahora surge otro problema no podemos mantener infinitas conexiones keepalive simultaneas en el servidor, ya que estas llega un momento en que ya se han usado y son inútiles y aportan más carga al servidor, por lo que se hace necesaria una directiva para controlar el número de conexiones persistentes.

4.4. MaxKeepAliveRequests

Características:

```
Sintaxis: MaxKeepAliveRequests num
Valor por defecto: 100
Módulo: Core
```

Descripción: Esta directiva nos va a permitir establecer un número limitado de conexiones persistentes entre clientes y el servidor, si se establece el valor 0 en esta directiva se abrirán conexiones de forma ilimitada, pero esto puede llegar a no ser lógico por posibles ataques que pretendan sobrecargar el servidor intentando realizar muchas conexiones simultaneas hasta que el servidor llega a su capacidad máximas (limitación por hardware) y comienza a no poder servir peticiones reales de otros clientes.

Es recomendable usar un valor elevado en esta directiva, se recomienda por ejemplo 500 a través de la web [4] en la sección que explica esta funcionalidad.

4.5. Protocol

Características:

```
Sintaxis: Protocol protocol
Módulo: Core
```

Descripción: Esta directiva especifica el protocolo usado para un determinado socket que esté a la escucha de peticiones, esta directiva sirve para conocer que módulo del servidor debe encargarse de gestionar una petición. Normalmente solo se necesita establecer el protocolo si se está escuchando por los puertos no estándares, por ejemplo el puerto 80 son para las peticiones HTTP y el puerto 443 para las peticiones HTTPS por convenio. Esta directiva puede servir para servir distintos protocolos mediante puertos que no son estándares, aunque también se puede especificar el protocolo haciendo uso de la directiva listen.

4.6. SetInputFilter

Características:

```
Sintaxis: SetInputFilter filter,...
Módulo: Core
```

Descripción: Se trata de una directiva muy interesante que nos va a permitir preveer salvarnos de distintos ataques como la inyección de código en nuestro servidor, ya que lo que va a hacer esta opción es establecer uno o más filtros que procesarán las solicitudes enviadas por el cliente además de lo que este envía por entrada con el procedimiento POST. Los filtros pueden definirse en cualquier lugar si se incluye en la directiva AddInputfilter. Pero en caso de que estos filtros sean burlados es posible establecer con la siguiente directiva un nivel más de seguridad.

4.7. SetOutputFilter

Características:

```
Sintaxis: SetOutputFilter filter,...
Módulo: Core
```

Descripción: Se trata de un proceso que en lugar de aplicar un filtro en las peticiones de entrada al servidor, trata de analizar la respuesta generada por el mismo mediante uno o más filtros, al igual que en el anterior es posible añadir los filtros a través de la directiva AddOutputFilter. Estos dos procesos de análisis de petición/respuesta podría llegar a descartar muchos de los ataques a los servidores más comunes, en el caso de que pase el primer filtro hay otra posibilidad analizando la salida proporcionada por nuestro servidor.

4.8. Timeout

Características:

```
Sintaxis: Timeout num [s/ms]
Valor por defecto: 60 s
Módulo: Core
```

Descripción: Es una de las directivas en mi opinión de mayor relevancia puesto que establece un tiempo máximo en el que el servidor debe a realizar diversas tareas antes de descartarlas, algunás de estas tareas son las siguientes:

- Cuando se lleva a cabo una lectura de datos enviados por parte del cliente, el tiempo máximo de espera hasta que llegue el paquete TCP si el buffer está vacio.
- Cuando se está escribiendo la respuesta a el cliente por parte del servidor, el tiempo máximo de espera hasta que se reconoce un paquete si el buffer está lleno.
- Diversos módulos para establecer el tiempo máximo de espera para los procesos de filtrado o salida de scripts.

4.9. VirtualHost

Características:

```
Sintaxis: <VirtualHost address[:port], ...> [s/ms]
Valor por defecto: 60 s
Módulo: Core
```

Descripción: Se trata de un grupo de directivas relacionadas con los host virtuales, que sirven para almacenar y servir distintas aplicaciones web desde una misma dirección IP, esta directiva te

permite establecer el nombre de usuario, la ruta, el nombre del servidor, etc, y además junto con la directiva NameVirtualHost permite hacer la correspondencia entre la dirección IP y los hosts virtuales.

Un ejemplo obtenido de la web de apache [4] es el siguiente:

```
<VirtualHost 10.1.2.3>
ServerAdmin webmaster@host.example.com
DocumentRoot /www/docs/host.example.com
ServerName host.example.com
ErrorLog logs/host.example.com-error_log
TransferLog logs/host.example.com-access_log
</VirtualHost>
```

4.10. Listen

Características:

```
Sintaxis: Listen [IP][:port] [protocol]
Módulo: MPM
```

Descripción: Se trata de una ”lista” de puertos a través de los que el servidor realizará su escucha, como se ha mencionado antes es posible establecer el protocolo por el que se regiran las peticiones y respuestas entre servidor y cliente por ejemplo con esta directiva podemos habilitar puertos diferentes al 443 que escuchen por peticiones HTTPS, está misma directiva podría usarse en lugar de la directiva protocol.

4.11. MaxClients/MaxRequestWorkers

Características:

```
Sintaxis: MaxRequestWorkers number
Módulo: MPM
```

Descripción: Es una directiva que se encarga de establecer un límite en el número de peticiones simultaneas que serán procesadas, pero que ocurre con las solicitudes si llegan cuando ya se ha cubrido el límite, estas son puestas en cola una cola cuya longitud se basa en la directiva ListenBackLog. Esto ocurre para servidores hebrados pero para los no hebrados esta directiva se traduce en el máximo número de procesos hijos que pueden ser lanzados para servir las solicitudes. MaxRequestWorkers era llamado MaxClients antes de la versión 2.3.13 aunque aun se sigue pudiendo usar la definición antigua.

4.12. MinSpareServers/MaxSpareServers

Características:

```
Sintaxis: MaxSpareThreads number
Módulo: MPM
```

Descripción: Es la directiva que controla el máximo/mínimo número de hebras del servidor que estarán vacentes o a la espera, el valor por defecto para los módulos worker y event es de 250, para el módulo netware es de 100 y para os2 es de 10.

4.13. MaxConnectionsPerChild

Características:

```
Sintaxis: MaxConnectionsPerChild number
Valor por defecto: 0
Módulo: MPM
```

Descripción: Con esta directiva podemos establecer un límite en el número de conexiones que un único proces hijo del servidor puede gestionar durante su vida, anteriormente se denominaba MaxRequestsPerChild pero este nombre todavía puede usarse. Después de que se gestionen todas las conexiones una vez alcanzado el límite el proceso hijo se elimina, en el caso de que se establezca por defecto como 0 el proceso hijo nunca muere. Establecer a esta directiva un valor diferente de 0 puede producir problemas de consumo de memoria y posible pérdida de memoria en el sistema.

Referencias

1. https://es.wikipedia.org/wiki/Servidor_HTTP_Apache
2. https://es.wikipedia.org/wiki/Apache_Software_Foundation
3. https://es.opensuse.org/SDB:Configuración_archivos_Apache
4. http://httpd.apache.org/docs/2.4/mod/core.html#timeout
5. http://systemadmin.es/2008/11/conexiones-keepalive-de-apache
6. http://httpd.apache.org/docs/2.4/es/mod/
7. http://httpd.apache.org/docs/2.4/es/mod/mpm_common.html