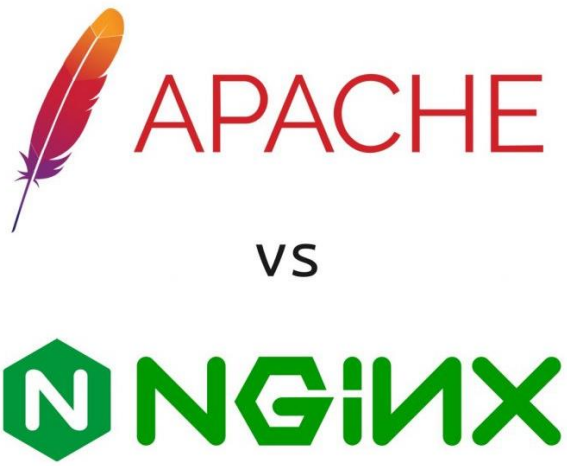
Índice

1. SERVIDORES WEB

Cuando vamos a **poner en marcha un servidor web**, lo primero que necesitamos es utilizar un **sistema operativo** sobre el cual vamos a ejecutar los diferentes servicios, sistema operativo que en más del 95% de las ocasiones suele ser un sistema **Linux**, así como un software que se encargue de la gestión de las **bases de datos**, **MySQL** habitualmente, y un software para gestionar el **contenido dinámicos** de las webs, que suele ser **PHP**. Además de este software esencial, otra de las partes **más importantes del servidor** suele ser la **elección del servidor web**, y aquí es donde entran las dudas.

Los **dos servidores más utilizados** para montar páginas web hoy en día son **Apache** y **Nginx**, sin embargo, es imposible decir que uno es mejor que otro ya que cada uno de ellos tiene sus propias fortalezas y debilidades y puede mejorar mejor bajo ciertas circunstancias o simplemente ser más sencillo de utilizar.



2. APACHE: INSTALACIÓN

Vamos a **instalar** y configurar el servidor **Apache**. Para ello:

```
sudo apt-get install apache2 apache2-utils
```

Sobre el servicio apache2 podemos realizar las **operaciones** de:

```
sudo service apache2 start|stop|restart|status
```

3. APACHE: CONFIGURACIÓN

3.1. Ficheros de configuración

Al igual que la mayoría de servidores en los sistemas Linux, **Apache se configura en base a ficheros de texto**. Todos los ficheros para configurar apache se encuentran en `/etc/apache2`. A continuación explicamos su estructura:

```
ls /etc/apache2

apache2.conf    conf-enabled  magic     mods-enabled  sites-available
conf-available  envvars       mods-available  ports.conf  sites-enabled
```

La configuración de Apache está muy estructurada, por lo que es necesario saber en qué fichero/directorio corresponde configurar cada parte. Realmente es mucho más cómodo disponer de **varios ficheros de configuración pequeños** que de uno muy grande. Eso también hace que sea más modular a la hora de activar o desactivar según qué características que iremos viendo más adelante.

- `apache2.conf`: El fichero de configuración principal de Apache, donde se pueden realizar cambios generales.
- `envvars`: Contiene la configuración de las variables de entorno.
- `ports.conf`: Contiene la configuración de los puertos donde Apache escucha.
- `conf-available`: Contiene ficheros de configuración adicionales para diferentes aspectos de Apache o de aplicaciones web como phpMyAdmin.
- `conf-enabled`: Contiene una serie de enlaces simbólicos a los ficheros de configuración adicionales para activarlos. Puede activarse o desactivarse con los comandos `a2enconf` o `a2disconf`.

- `mods-available`: Contiene los módulos disponibles para usar con Apache.
- `mods-enabled`: Contiene enlaces simbólicos a aquellos módulos de Apache que se encuentran activados en este momento.
- `sites-available`: Contiene los ficheros de configuración de cada uno de los hosts virtuales configurados y disponibles (activos o no). Se crean utilizando los comandos `a2enmod` y `a2dismod` que más adelante explicaremos con más detalle.
- `sites-enabled`: Contiene una serie de enlaces simbólicos a los ficheros de configuración cuyos hosts virtuales se encuentran activos en este momento. Se crean a través de los comandos `a2ensite` y `a2dissite` que más adelante explicaremos con más detalle.

Apache nos brinda una herramienta para comprobar **si la sintaxis de los ficheros es correcta**:

```
apache2ctl configtest
```

3.2. Hosts virtuales

Por defecto, Apache proporciona documentos desde el directorio `/var/www/html` al escribir la **dirección IP de nuestro servidor**. Esto se debe a que tiene configurado un **virtualhost por defecto** en el fichero `000-default.conf` del directorio `sites-available`.

La configuración de **host virtuales** permite **alojar páginas de distintos dominios** . Permiten que un mismo servidor web aloje múltiples dominios. **Cada sitio web** tendrá su **virtualhost único** e independiente de las demás.

Todo aquello que no esté incluido en la definición de cada virtualhost se heredará de la configuración principal: `apache2.conf` (`/etc/apache2/apache2.conf`)

Existen **dos tipos** de virtualhost:

- **Basados en nombre**: permite asociar nombres de dominio con directorios web. Probablemente es **el planteamiento más habitual**.
- **Basados en IP**: permite asociar las IPs de las diferentes interfaces de red de mi servidor con directorios web. Requiere que mi servidor tenga diferentes interfaces configuradas.

Virtual host basado en nombre:

En el directorio `/etc/apache2/sites-available` se definen los virtualhosts.

Cada virtualhost en un fichero de texto de configuración distinto de la forma:

empresa.com.conf

```
<VirtualHost 192.168.56.104>
      DocumentRoot "/var/www/empresa"
      ServerName empresa.com
      ServerAlias www.empresa.com empresa.es www.empresa.es
</VirtualHost>
```

prueba.com.conf

```
<VirtualHost 192.168.56.104>
      DocumentRoot "/var/www/prueba"
      ServerName prueba.com
      ServerAlias www.prueba.com prueba.es www.prueba.es
</VirtualHost>
```

Donde:

- **`VirtualHost [ip]:80`**: Indico la IP del servidor. Indicar el puerto 80 es opcional.
- **`ServerAdmin`**: correo electrónico al que pueda acceder el administrador.
- **`ServerName`**: dominio base que debería coincidir para esta definición de host virtual.
- **`ServerAlias`**: define más nombres de dominio que redirigen a este mismo sitio.
- **`DocumentRoot`**: Indicamos la ruta de nuestros documentos (debe estar dentro de los directorios habilitados, sino debo habilitarlos mediante **`<Directory>`**).
- **`ErrorLog`**: Ficheros para log de errores: Por defecto en **`/var/log/apache2/error.log`**.
- También podemos crear hosts virtuales asociados a un puerto diferente, de modo que si indicando diferentes puertos se permite acceder a diferentes sitios: **`VirtualHost [ip]:8080`**.

Para **habilitar el archivo** anterior:

```
a2ensite prueba.conf
```

Deshabilitamos el fichero de host virtual por defecto:

```
a2dissite 000-default.conf
```

Finalmente, **reiniciamos el servidor**.

Cuando realizamos configuraciones en **entornos de pruebas privados**, es probable que no tengamos acceso a ellos desde internet, por lo que **no podremos utilizar nombres de dominio** TLD. Sin embargo, **podemos asociar nombres de dominio a direcciones IPs sin necesidad de instalar y configurar servidores DNS**, mediante el fichero de **`hosts`**. Se puede considerar como el fichero **DNS local**.

Si no tenemos configurado un servidor DNS con las entradas de dominio necesarias, puedes generar estas entradas modificando el archivo **`/etc/hosts`**.

```
# IP nombre-dominio

ip_servidor     nombre1.dominio nombre2.mi.dominio
```

Cada campo de cada entrada puede ir **separado por espacios o por tabulados**. Estas entradas solamente serán efectivas en el equipo en el que se modifique el archivo **`/etc/hosts`**. Así que, debes modificar el archivo**`/etc/hosts`** en cada equipo que quieres que se resuelvan esas entradas.

Cuando accedemos al servidor por un nombre que no coincide con ningún virtual host, o mediante la IP, se mostrará el contenido del primer Virtual Host cargado (alfabéticamente).

Preparamos ahora una web de bienvenida para el nuevo sitio web:

var/www/prueba/index.html

```
<!DOCTYPE>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Bienvenido a prueba.com</title>
</head>
<body>
    <p>Estás en prueba.com</p>
</body>
</html>
```

3.3. Control de acceso a directorios

Si un usuario se autentica en cualquier página, es necesario validarse. Para ello el servidor web necesita de una base de datos o servicio de directorio.

- Base de datos, puede ser Oracle, Mysql. Mon-goDB.
- Servicio de directorio, puede ser LDAP.

Por un lado tenemos la autenticación y por otro el control de acceso:

- **Control de acceso**: El servicio Apache determina si se tiene acceso a un recurso o no mediante una directiva determinada.
- **Autenticación**: Apache dar acceso a un recurso mediante la validación de un usuario y contraseña.

El control de acceso a directorios se realiza a la hora de ofrecer recursos mediante el servidor Apache, y se aplica sobre directorios mediante la directiva **`Require`**:

```
# Todos los clientes acceden
<Directory /var/www/directorio>
      Require all granted
</Directory>
```

```
# Solo el cliente con dicha ip accede
<Directory /var/www/directorio>
      Require all denied
      Require ip 192.168.1.45
</Directory>
```

Estas configuraciones se pueden aplicar en los diferentes fichero de configuración, como podría ser el de un host virtual.

El control de acceso permite controlar el acceso por parte de un dispositivo de la red que desea acceder a un recurso determinado del servidor Apache. Este control se aplica sobre ficheros y directorios.

Este acceso se aplica dentro de la directiva de `<Directory>`. y dentro de ella se aplicará la directiva `Require` con sus diferentes opciones. A continuación, se mostrará una lista de las directivas más usadas:

Directiva	Descripción
Require all granted	Se permite el acceso a todo el mundo.
Require all denied	Se deniega el acceso a todo el mundo.
Require env env-var	Se permite el acceso solo a las variables de entorno definidas.
Require method http-method	Acceso a los métodos HTTP definidos.
Require expr expression	Acceso permitido al resultado verdadero de evaluar la expresión definida.
Requiere user userid	Solamente tienen acceso los usuarios definidos.
Require group group-name	Solamente tienen acceso los grupos definidos.
Require valid-user	Todos los usuarios validados tienen acceso al recurso.
Require ip	Los clientes que tengan como IP o rangos de IP tienen acceso al recurso.

3.4. Autenticación

La autenticación de usuarios en Apache puede hacerse mediante:

- **Autenticación básica con ficheros**: el mecanismo más simple para implementar el control de acceso a recursos de una sede web es utilizar archivos de usuarios y grupos propios del servidor. Apache proporciona herramientas para crearlos. La ventaja principal de este método es la facilidad de administración. El inconveniente es que conlleva una gestión diferenciada de los usuarios del servicio web y los del sistema. De hecho, esto puede ser un inconveniente o una ventaja, si lo que interesa es tenerlos segregados.
- **Autenticación mediante PAM** : en los sistemas GNU / Linux actuales la autenticación de los usuarios se hace vía PAM (Pluggable Authentication Module). El PAM comprueba el directorio / etc / passwd, el LDAP, el Kerberos, las huellas dactilares o lo que sea necesario. Usar el vínculo con el módulo del PAM es un buen mecanismo para validar los usuarios del servicio web al igual que se validan los usuarios del sistema.
- **Autenticación mediante LDAP** : uno de los mecanismos más populares actualmente para la autenticación es el LDAP. El módulo del LDAP permite pasar la validación de los

usuarios al encargado de gestionar la autenticación LDAP de los usuarios del sistema. También se puede tener en funcionamiento un servicio LDAP específico para las validaciones del servicio web.

HTTP proporciona un **método de autenticación básico** de usuarios: **basic**. Este método ante una petición del cliente (navegador web) al servidor cuando se solicita una URL mostrará un diálogo pidiendo usuario y contraseña.

Una vez autenticado el usuario, el cliente volverá a hacer la petición al servidor pero ahora enviando el usuario y contraseña, en texto claro (sin cifrar) proporcionados en el diálogo. Es recomendable entonces si empleamos este método combinemos con conexión SSL (HTTPS).

Para crear un sistema de autenticación que controle el acceso a un directorio tenemos que **crear un fichero** `.htaccess` en el **directorio que queremos proteger**. En el siguiente ejemplo vamos a utilizar un sistema de autenticación basado en fichero, pero podríamos autenticar contra una base de datos, u otros métodos.

```
AuthType Basic

AuthName "Acceso Restringido"

AuthUserFile /var/www/directorio/.htpasswd

Require valid-user
```

A continuación, **daré de alta un usuario junto con su contraseña** y lo guardaré en el fichero `AuthUserFile` cuya ruta he definido en la configuración anterior: `AuthUserFile /var/www/directorio/.htpasswd`. Si quiero añadir **más usuarios** posteriormente, uso el comando **sin la opción -c**:

```
htpasswd -c /var/www/directorio/.htpasswd nombre_usuario
```

Los ficheros .htaccess sobrescriben las configuraciones por defecto de los sitios web habilitados en mi servidor. Debo añadir las siguientes líneas al fichero de configuración de mi virtual host:

```
<Directory /var/www/directorio/>

      AllowOverride All

</Directory>
```

Otra forma es incorporar el contenido de `.htaccess` a `<Direcory>` del host virtual y la ruta que queremos proteger. Por ejemplo:

```
<VirtualHost *:80>
    DocumentRoot "/var/www/html/prueba.com"
    ServerName prueba.com
    ServerAlias www.prueba.com
    <Directory "/var/www/html/prueba.com/privado">
        AuthType Basic
        AuthName "Acceso Restringido a Usuarios"
        AuthUserFile /vaw/www/prueba.com/.htpasswd
        Require valid-user
    </Directory>
</VirtualHost>
```

Para terminar, **reiniciamos** el servicio de Apache

Ahora, nos preguntará por credenciales de usuario a la hora de navegar al recurso.

Podemos también impedir, por ejemplo, que los ficheros de un directorio puedan ser listados por cualquier visitante. Basta con crear un fichero `.htaccess` y añadir la siguiente línea que indica que no puede listarse el directorio.

```
Options -Indexes
```

3.5. Módulos

Apache tiene una configuración basada en módulos: permite cargar diferentes funcionalidades para el servidor según las necesidades de cada sitio web. Algunos módulos habituales que podemos necesitar pueden ser el **módulo php** para que el servidor permita ejecutar scripts, o el **modulo ssl o tls**, para permitir transferencias cifradas.

MÓDULO SSL

El primer paso para **configurar SSL en Apache** será crear el **certificado y la clave**, que se quedarán almacenados en `/etc/apache2/certs`. Para eso primero creamos la carpeta y luego el certificado y su correspondiente clave.

Hay que tener en cuenta que estamos creando **un certificado autofirmado**. Este tipo de certificados sólo se deben usar con el propósito de enseñar o hacer una demostración puesto que en la práctica **no son válidos**. El navegador no confiará en él porque somos nosotros quienes lo hemos firmado. Los certificados, para que sean válidos, deben ser validados por una entidad certificadora.

Primero habilitamos el módulo: `sudo a2enmod ssl`

```
sudo mkdir /etc/apache2/certs
```

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/apache2/certs/apache2.key -out /etc/apache2/certs/apache2.crt
```

Ahora ya tenemos el certificado y su clave. Activamos entonces el módulo SSL de Apache.

Reiniciamos el servicio.

Y ahora **configuramos un host virtual para que soporte conexión segura**. En este caso he seleccionado el mismo sitio que antes hemos configurado como ejemplo de host virtual y he realizado los cambios necesarios para que ahora soporte conexión segura (HTTPS). Básicamente es **cambiar el puerto donde escucha** (ahora es 443, donde escuchan las conexiones seguras), **activar el soporte para SSL e indicar donde están el certificado y la clave**.

```
<VirtualHost *:443>
    DocumentRoot /var/www/html/prueba.com
    ServerName prueba.com
    ServerAlias www.prueba.com

    SSLEngine On
    SSLCertificateFile /etc/apache2/certs/apache2.crt
    SSLCertificateKeyFile /etc/apache2/certs/apache2.key
    SSLProtocol All -SSLv3
</VirtualHost>
```

Para probarlo, abrimos cualquier navegador e introducimos la dirección **https://prueba.com**. Automáticamente, al usar HTTPS, se conectará al puerto 443 y el navegador intentará comprobar la validez del certificado. Si fuera válido se establecería la conexión segura y veríamos como el

símbolo del candado en el navegador nos indica que la web es segura y podremos navegar sin problemas.

Llegados a este punto nos puede interesar **que todo el tráfico de la web se vea forzado a utilizar el protocolo seguro HTTPS**. Incluso aunque el usuario introduzca la URL directamente y decide navegar utilizando HTTP (http://…..) nosotros podemos redirigirle hacia la opción de utilizar la opción segura.

En ese caso podemos incluso **configurar el host virtual no seguro con las opciones mínimas para redirigirlo al seguro**.

```
<VirtualHost *:80>
   ServerName www.prueba.com
   Redirect / https://www.prueba.com/
</VirtualHost>
```

```
<VirtualHost *:443>
  ServerName prueba.com
  DocumentRoot . . .
  . . .
</VirtualHost>
```

Finalmente podemos optar por **una redirección permanente** (de esta forma así se notificará a los buscadores) modificando la orden Redirect por la siguiente:

```
Redirect permanent / https://www.prueba.com/
```

3.6. Registro y monitorización

En el servidor web Apache se tienen principalmente los siguientes **archivos de registro**:

Registro de errores: `/var/log/apache2/error.log`.

Registro de accesos: `/var/log/apache2/access.log`.

Para cada servidor virtual hay un registro de acceso. Estos registros de acceso tienen nombres `other_vhosts_Access`

```
<VirtualHost *:80>
    ServerName prueba.com
    ServerAlias www.prueba.com prueba.es
    DocumentRoot /var/www/prueba/
    # Para log de errores y acceso
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Puede ser el de la configuración principal del servidor así como el de la configuración de los host virtuales.

- **`TransferLog`**: Directiva que define el nombre del archivo de registro o al programa al que se envía la información de registro. Emplea los especificadores asignados por la directiva **`LogFormat`**.
- **`LogFormat`**: Directiva que define el formato del archivo de registro asignado con la directiva **`TransferLog`**
- **`ErrorLog`**: Directiva que permite registrar todos los errores que encuentre Apache. Permite guardar la información en un archivo de registro o bien en **`syslog`**
- **`ErrorLog ${APACHE_LOG_DIR}/web-error.log`**
- **`CustomLog:`** Directiva similar a la directiva **`TransferLog`**, pero con la particularidad que permite personalizar el formato de registro empleando los especificadores anteriormente vistos.
- **`CookieLog`**: Directiva que define el nombre del archivo de registro donde registrar información sobre cookies

También podemos configurar la ubicación de los logs en un host virtual o definir páginas de error personalizadas.

```
<VirtualHost *:80>
    DocumentRoot "/var/www/html/prueba.com"
    ServerName prueba.com
    ServerAlias www.prueba.com

    ErrorLog "prueba.com-error.log"
    CustomLog "prueba.com-access.log" combined
```

```
    ErrorDocument 404 /error_404.html
    ErrorDocument 500 /error_500.html
</VirtualHost>
```

Monitorización.

Webalizer es una aplicación que podemos instalar para procesar el fichero log de Apache y generar un documento HTML con las estadísticas de nuestro sitio web.

```
sudo apt-get install webalizer
```

Y a continuar utilizar el comando webalizer para procesar el fichero log que queramos (en este caso el del sitio raíz) y éste generará una carpeta en `/var/www/webalizer` con un sitio web donde podremos visitar diferentes estadísticas sobre el uso del servidor.

```
sudo webalizer /var/log/apache2/access.log
```

Puesto que ahora por defecto la carpeta raíz para Apache es `/var/www/html`, tendremos que hacer un enlace simbólico dentro de dicha carpeta que apunte a la que Webalizer genera para poder visualizar el informe. También podría ser interesante proteger esa carpeta de alguna manera para que no fuera visible para cualquier visitante.

```
/var/www/html/$ sudo ln -s ../webalizer webalizer
```

Ahora podemos visitar la página web generada donde podremos observar las estadísticas ya preparadas para nuestro sitio web (en este caso en http://IP/webalizer, donde la IP será la de la máquina donde tenemos desplegado el sitio, o bien el dominio si lo tenemos)

GoAccess es una analizador visual de logs de Apache en tiempo real. De esa manera permite monitorizar el acceso y uso al servidor por parte de los usuarios en cada momento con numerosas métricas.

```
sudo echo "deb http://deb.goaccess.io/ stretch main" >> /etc/apt/sources.list
```

```
sudo apt-get update
```

```
sudo apt-get install goaccess
```

Podemos usarla de dos maneras, visualizando los resultados por consola o en formato HTML como una web más.

Para visualizarlos desde la consola, basta localizar el fichero log de Apache que queremos monitorizar (en este caso el fichero general, pero podríamos pasar los ficheros log de los diferentes hosts virtuales que hayamos configurado) y ejecutamos:

```
sudo goaccess /var/log/apache2/access.log -c
```

También podemos pedirle a GoAccess que prepare un documento HTML en tiempo real donde podremos ver las estadísticas en tiempo real desde el navegador.

```
sudo goaccess /var/log/apache2/access.log -o /var/www/html/report.html
--log-format=COMBINED --real-time-html
```

3.7. Cortafuegos

Del mismo modo que tendremos que hacer con todos los servicios de nuestro sistema, debemos controlar los puertos de acceso a nuestros servidores. Mediante el siguiente comando podemos ver las aplicaciones disponibles:

```
ufw app list
```

Respecto a Apache, tenemos **3 posibilidades**:

- **WWW**: abre solo el puerto 80 (tráfico web normal no cifrado, con http).
- **WWW Full**: abre el puerto 80 (tráfico web normal no cifrado, con http) y el puerto 443 (tráfico TLS/SSL cifrado, con https)
- **WWW Secure**: abre solo el puerto 443 (tráfico TLS/SSL cifrado, con https)

Dado que nos interesa permitir el tráfico mediante Http y Https, vamos a añadir la regla Apache Full.

```
ufw allow "WWW Full"
```

También podemos abrir los puertos o servicios concretos

```
ufw allow http

ufw allow 80

ufw allow https

ufw allow 443
```

Para comprobar el estado del Firewall:

```
ufw status verbose
```