Servidores web: Apache y Nginx

- Servidores web: Apache y Nginx
  - Apache vs Nginx
  - Servidor Apache
    - Instalación de Apache
    - Ajustar el Firewall
    - Comprobar el servicio Apache
    - Creando Virtual Hosts
    - Activando y desactivando el sitio web
    - Resolución de nombres
    - Directorios y archivos de configuración de Apache
      - Contenido
      - Configuración del servidor
      - Registros y logs
    - Configurando SSL
      - Certificados SSL
      - Creando un certificado Self-Signed SSL
        - 1. Creando una clave privada
        - 2. Creando un Certificate Signing Request (CSR)
        - 3. Generando el certificado SSL
        - 4. Configurando Apache con el certificado creado
        - 5. Configurando el Virtual Host que queremos proteger con SSL
        - 6. Reconfigurando el firewall
        - 7. Configurando los módulos de Apache
  - Instalando LAMP
    - Instalando mysql
    - Instalando PHP
    - Probando PHP
    - index.html e index.php
  - Ficheros .htaccess
    - Introducción a los ficheros .htaccess
    - Ejemplo de uso de ficheros .htaccess
  - Servidor Nginx
    - Instalación de Niginx
    - Ajustando el Firewall

        - Comprobar el servicio Nginx
        - Configurando Server Blocks
        - Comprobando la sintaxis y reiniciando el servicio
        - Editando el fichero /etc/hosts
        - Directorios y archivos de configuración de Nginx
            - Contenido
            - Configuración
            - Server Logs
        - Más sobre nginx
    - Fuentes

Apache vs Nginx

Apache es un servidor web de código abierto altamente personalizable y ampliamente utilizado que ofrece una **amplia gama de características**, incluido el soporte para una variedad de lenguajes de programación y plataformas.

Por otro lado, **nginx** es conocido por su **alta eficiencia** y capacidad de manejar una **gran cantidad de tráfico y conexiones simultáneas**. Es popular entre los desarrolladores web debido a su capacidad para manejar cargas dinámicas y estáticas con facilidad.

En resumen, **Apache** es una excelente opción si requieres una **gran cantidad de personalización y una amplia gama de características**, mientras que **nginx** es una excelente opción si estás buscando **alta eficiencia** y capacidad de manejo de tráfico.

- Comparativa del uso de Nginx y Apache

Servidor Apache

Instalación de Apache

- Web Oficial: https://httpd.apache.org/

1. Abrir un terminal y actualizar los paquetes existentes:

```
sudo apt update
```

2. Instalar apache

```
sudo apt install apache2 -y
```

3. Verificar si Apache está funcionando correctamente. Para ello, hay que abrir un navegador y teclear **http://localhost** o **http://127.0.0.1**. Debería de aparecer la página de bienvenida de Apache.

Ajustar el Firewall

Si queremos **permitir el acceso externo**, tenemos que configurar el **firewall**:

```
ubuntu@ubuntu-2204:~$ sudo ufw app list
Available applications:
  Apache
  Apache Full
  Apache Secure
  CUPS
  OpenSSH
```

Tal y como podemos observar, existen tres perfiles diferentes para Apache.

- **Apache:** este perfil abre solo el **puerto 80** para **tráfico web normal no cifrado**.
- **Apache Full:** este perfil abre el puerto 80 y además el puerto **443** para **tráfico TLS/SSL cifrado**.
- **Apache Secure**: este perfil abre solo el puerto **443**.

Es recomendable utilizar el **perfil más restrictivo**, para el tráfico que queramos manejar. Por ahora, como no hemos activado aún **SSL** pues solo necesitamos permitir tráfico en el puerto 80.

```
sudo ufw allow 'Apache'
```

Podemos verificar el estado tecleando:

```
root@ubuntu-2204:~# sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
Apache                     ALLOW       Anywhere
Apache (v6)                ALLOW       Anywhere (v6)
```

Si nos sale que está inactivo, podemos activarlo con:

```
sudo ufw enable
```

Comprobar el servicio Apache

Podemos chequear el estado de Apache con el siguiente comando:

```
root@ubuntu-2204:~# service apache2 status
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: >
     Active: active (running) since Sun 2023-02-05 17:54:43 EST; 12h ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 3720 (apache2)
      Tasks: 55 (limit: 4620)
     Memory: 5.2M
        CPU: 110ms
     CGroup: /system.slice/apache2.service
             ├─3720 /usr/sbin/apache2 -k start
             ├─3722 /usr/sbin/apache2 -k start
             └─3723 /usr/sbin/apache2 -k start

Feb 05 17:54:43 ubuntu-2204 systemd[1]: Starting The Apache HTTP Server...
Feb 05 17:54:43 ubuntu-2204 systemd[1]: Started The Apache HTTP Server.
```

Podemos obtener la IP del servidor con el comando:

```
hostname -I
```

E intentar entrar en la página web de Apache desde otra máquina virtual o equipo en red.

Para parar el servicio tecleamos:

```
sudo service apache2 stop
```

Para iniciar el servicio tecleamos:

```
sudo service apache2 start
```

Para parar y arrancar el servicio tecleamos:

```
sudo service apache2 restart
```

Por defecto, apache está configurado para comenzar automáticamente cuando el equipo arranca. Si no es el comportamiento que se quiere:

```
sudo service apache2 disable
```

Para reactivar el inicio en el arranque:

```
sudo service apache2 enable
```

Creando Virtual Hosts

Los **Virtual Hosts** nos permiten poder almacenar páginas de **varios dominios** en un **mismo servidor web**.

Hay tres tipos de Virtual Hosts:

1. **Virtual Hosts basados en IP**: Un servidor tiene varias IP y en cada una de ellas sirve las páginas de un dominio diferente. Por ejemplo, un servidor web de una empresa que sirva en una interfaz la web de la intranet y en otra la web de la compañía.
2. **Virtual Hosts basados en nombre**: Un servidor aloja varias páginas pertenecientes a varios dominios y, gracias a que el navegador le envía la cabecera **Host:nombreDeDominio**, el servidor sabe qué página servirle al cliente. En este caso el servidor sirve **diferentes sitios web** por el **mismo puerto y misma IP**. Es la configuración más habitual.
3. **Virtual Hosts basados en puertos**: Un servidor sirve páginas de distintos dominios en función del puerto.

Por defecto, el servidor sirve la web alojada en **/var/www/html**. Podemos visualizar su html en consola escribiendo:

```
cat /var/www/html/index.html
```

Esta página **se sirve por defecto**. Si, por ejemplo, tenemos varios Virtual Host basados en nombre y escribimos en el navegador la IP del servidor y no la URL con el nombre, el navegador no podrá enviar la cabecera **Host:nombreDeDominio** y el servidor le responderá sirviéndole esta página por defecto.

Ahora supongamos que quiero servir mi página personal bajo el nombre "**gestal.es**", pues creo el directorio donde almacenaré los archivos de la página:

```
sudo mkdir /var/www/gestal.es
```

El servidor web Apache se ejecutará inicialmente como root, pues en Linux sólo root puede abrir sockets en puertos por debajo del 1000. Apache por defecto, escucha en el puerto 80.

Al arrancar Apache, el proceso de root arrancará otros procesos hijos que se ejecutarán con otro usuario y grupo menos peligroso que root. Por defecto estos procesos hijos se ejecutan con el usuario **www-data** y el grupo **www.data**.

Hay que asegurarse que ese usuario y ese grupo tengan permisos para leer los documentos que debe de servir el navegador web.

Esto podemos comprobarlo con el comando:

```
ps -aux
```

- Más información sobre permisos

Vamos a darle la propiedad del directorio al usuario de Apache (www-data), aunque quizá esta propiedad correspondería a un hipotético usuario "gestal" que sería quien administraría su página, y que podría tener también cuenta de FTP o SSH. Esto es un escenario común en servidores que sirven páginas de distintos usuarios. Pero vamos a simplificarlo.

Cambiamos la propiedad y grupo del directorio:

```
sudo chown -R www-data:www-data /var/www/gestal.es
```

Otorgamos permisos a Apache:

```
sudo chmod -R 755 /var/www/gestal.es
```

Y creamos un archivo html de prueba:

```
sudo nano /var/www/gestal.es/index.html
```

Con el contenido:

```
<html>
  <head>
    <title>Bienvenido a la web de gestal.es</title>
  </head>
  <body>
    <h1>Gestal.es</h1>
    <p>Lorem Ipsum dolor sit amet.
  </body>
</html>
```

La configuración por defecto de apache se encuentra en **/etc/apache2/sites-available/000-default.conf**:

```
root@ubuntu-2204:~# cat /etc/apache2/sites-available/000-default.conf
<VirtualHost *:80>
 # The ServerName directive sets the request scheme, hostname and port that
 # the server uses to identify itself. This is used when creating
 # redirection URLs. In the context of virtual hosts, the ServerName
 # specifies what hostname must appear in the request's Host: header to
 # match this virtual host. For the default virtual host (this file) this
 # value is not decisive as it is used as a last resort host regardless.
 # However, you must set it for any further virtual host explicitly.
 #ServerName www.example.com

 ServerAdmin webmaster@localhost
 DocumentRoot /var/www/html

 # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
 # error, crit, alert, emerg.
 # It is also possible to configure the loglevel for particular
 # modules, e.g.
 #LogLevel info ssl:warn

 ErrorLog ${APACHE_LOG_DIR}/error.log
 CustomLog ${APACHE_LOG_DIR}/access.log combined

 # For most configuration files from conf-available/, which are
 # enabled or disabled at a global level, it is possible to
 # include a line for only one particular virtual host. For example the
 # following line enables the CGI configuration for this host only
 # after it has been globally disabled with "a2disconf".
 #Include conf-available/serve-cgi-bin.conf
</VirtualHost>
```

En lugar de modificar el archivo de configuración por defecto, podemos crear uno:

```
sudo nano /etc/apache2/sites-available/gestal.es.conf
```

Con el siguiente contenido:

```
<VirtualHost *:80>
  ServerAdmin juan@gestal.es
  ServerName gestal.es
  ServerAlias www.gestal.es
  DocumentRoot /var/www/gestal.es
  ErrorLog ${APACHE_LOG_DIR}/error.log
  CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Hemos cambiado las directivas:

- ServerAdmin al email de administración.
- ServerName a gestal.es, el dominio base.
- ServerAlias a www.gestal.es, un nombre alternativo para nuestro dominio.
- DocumentRoot al directorio que creamos antes, donde almacenamos nuestra página web en html.

Activando y desactivando el sitio web

Para activar y desactivar sitios web utilizamos **a2ensite** y **a2dissite**.

Podemos activar el sitio de gestal.es:

```
root@ubuntu-2204:~# sudo a2ensite gestal.es.conf
Enabling site gestal.es.
To activate the new configuration, you need to run:
  systemctl reload apache2
```

O desactivar el sitio por defecto definido en 000-default.conf.

```
root@ubuntu-2204:~# sudo a2dissite 000-default.conf
Site 000-default disabled.
To activate the new configuration, you need to run:
  systemctl reload apache2
```

Y podemos comprobar si hay errores de configuración con el comando:

```
sudo apache2ctl configtest
```

Que nos tiene que devolver **Syntax OK** si no hay errores.

Ahora reiniciamos el servicio:

```
service apache2 restart
```

Y ahora, **Apache** estaría sirviendo la web que creamos. Si abro un navegador y tecleo:

```
http://127.0.0.1
```

Vería la página web que he creado para gestal.es, ya que es la única que está activada en el servidor. Pero si tecleo simplemente:

```
http://gestal.es
```

En lugar de visitar la página alojada en el servidor que he creado, me va a llevar a mi página web que tengo online en el servidor con IP 5.39.80.5.

Resolución de nombres

Lo que está ocurriendo es que no he creado ni configurado ningún **servidor de DNS** que sea capaz de resolver gestal.es a la IP de mi equipo.

Una opción es descargar y configurar bind9, y configurar el resolver DNS del cliente que quiera conectarse a la página para que le pregunte a este servidor por el nombre.

Otra opción más sencilla para esta practica y que evita tener que configurar bind9 ahora, consiste en editar el fichero **/etc/hosts** en el cliente y decirle que resuelva el nombre **gestal.es** a la **IP del servidor web**.

Editamos el fichero hosts desde la propia máquina virtual del servidor:

```
sudo nano /etc/hosts
```

Y añadimos la línea necesaria para que resuelva **gestal.es** a 127.0.0.1:

```
127.0.1.1       gestal.es
```

Si estamos usando otra máquina virtual para conectarnos a nuestro servidor, tendremos que configurar en esa máquina el fichero hosts con el nombre apuntando a la IP de la máquina virtual del servidor.

```
127.0.0.1       localhost
127.0.1.1       ubuntu-2204.linuxvmimages.local ubuntu-2204
127.0.1.1       gestal.es

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Es posible que si refrescamos la página, el servidor se intente conectar a **https://gestal.es** en el servidor **5.39.80.5**. Los navegadores tienen caché, para evitar consultar al resolver, así que vamos a la configuración del navegador y borramos su caché. Por ejemplo, para el Firefox sería:

1. Pinchar en el icono del menú (tipo hamburguesa) y seleccionar "Preferences".
2. Ir a la sección de "Privacy & security panel".
3. En la sección de "Cookies and Site Data" le damos a "Clear Data".
4. Marcamos "Cached Web Content" y le damos al botón "Clear".

Y ahora si tecleamos **gestal.es** en el navegador, nos debe llevar a la página que sirve nuestro Apache.

Directorios y archivos de configuración de Apache

Contenido

- **/var/www/html**: Es el contenido web por defecto, con la página predeterminada de Apache que hemos visto antes. Este directorio puede modificarse en los archivos de configuración.
- **/var/www/gestal.es**: Es el directorio donde almacenamos los archivos del sitio **gestal.es** que hemos creado antes.

Configuración del servidor

- **/etc/apache2**: Es el directorio donde se encuentran los **archivos de configuración de Apache**.

```
root@ubuntu-2204:~# ls -l /etc/apache2
total 80
-rw-r--r-- 1 root root  7224 Jan 23 13:34 apache2.conf
drwxr-xr-x 2 root root  4096 Feb  5 17:54 conf-available
drwxr-xr-x 2 root root  4096 Feb  5 17:54 conf-enabled
-rw-r--r-- 1 root root  1782 Sep  7 23:07 envvars
-rw-r--r-- 1 root root 31063 Sep  7 23:07 magic
drwxr-xr-x 2 root root 12288 Feb  5 17:54 mods-available
drwxr-xr-x 2 root root  4096 Feb  5 17:54 mods-enabled
-rw-r--r-- 1 root root   320 Sep  7 23:07 ports.conf
drwxr-xr-x 2 root root  4096 Feb  7 14:03 sites-available
drwxr-xr-x 2 root root  4096 Feb  7 14:04 sites-enabled
```

- **/etc/apache2/apache2.conf**: Es el archivo principal de la configuración de Apache. Se puede modificar para realizar cambios en la configuración.

```
# This is the main Apache server configuration file.  It contains the
# configuration directives that give the server its instructions.
# See http://httpd.apache.org/docs/2.4/ for detailed information about
# the directives and /usr/share/doc/apache2/README.Debian about Debian specific
# hints.
#
#
# Summary of how the Apache 2 configuration works in Debian:
# The Apache 2 web server configuration in Debian is quite different to
# upstream's suggested way to configure the web server. This is because Debian's
# default Apache2 installation attempts to make adding and removing modules,
# virtual hosts, and extra configuration directives as flexible as possible, in
# order to make automating the changes and administering the server as easy as
# possible.

# It is split into several files forming the configuration hierarchy outlined
# below, all located in the /etc/apache2/ directory:
#
# /etc/apache2/
# |-- apache2.conf
# | `--  ports.conf
# |-- mods-enabled
# | |-- *.load
# | `-- *.conf
# |-- conf-enabled
# | `-- *.conf
#  `-- sites-enabled
#   `-- *.conf
#
#
# * apache2.conf is the main configuration file (this file). It puts the pieces
#   together by including all remaining configuration files when starting up the
#   web server.
#
# * ports.conf is always included from the main configuration file. It is
#   supposed to determine listening ports for incoming connections which can be
#   customized anytime.
#
# * Configuration files in the mods-enabled/, conf-enabled/ and sites-enabled/
#   directories contain particular configuration snippets which manage modules,
#   global configuration fragments, or virtual host configurations,
#   respectively.
#
#   They are activated by symlinking available configuration files from their
#   respective *-available/ counterparts. These should be managed by using our
#   helpers a2enmod/a2dismod, a2ensite/a2dissite and a2enconf/a2disconf. See
#   their respective man pages for detailed information.
#
# * The binary is called apache2. Due to the use of environment variables, in
#   the default configuration, apache2 needs to be started/stopped with
```

```
#   /etc/init.d/apache2 or apache2ctl. Calling /usr/bin/apache2 directly will not
#   work with the default configuration.


# Global configuration
#

#
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
#
# NOTE!  If you intend to place this on an NFS (or otherwise network)
# mounted filesystem then please read the Mutex documentation (available
# at <URL:http://httpd.apache.org/docs/2.4/mod/core.html#mutex>);
# you will save yourself a lot of trouble.
#
# Do NOT add a slash at the end of the directory path.
#
#ServerRoot "/etc/apache2"

#
# The accept serialization lock file MUST BE STORED ON A LOCAL DISK.
#
#Mutex file:${APACHE_LOCK_DIR} default

#
# The directory where shm and other runtime files will be stored.
#

DefaultRuntimeDir ${APACHE_RUN_DIR}

#
# PidFile: The file in which the server should record its process
# identification number when it starts.
# This needs to be set in /etc/apache2/envvars
#
PidFile ${APACHE_PID_FILE}

#
# Timeout: The number of seconds before receives and sends time out.
#
Timeout 300

#
# KeepAlive: Whether or not to allow persistent connections (more than
# one request per connection). Set to "Off" to deactivate.
#
KeepAlive On

#
# MaxKeepAliveRequests: The maximum number of requests to allow
```

```
# during a persistent connection. Set to 0 to allow an unlimited amount.
# We recommend you leave this number high, for maximum performance.
#
MaxKeepAliveRequests 100

#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection.
#
KeepAliveTimeout 5


# These need to be set in /etc/apache2/envvars
User ${APACHE_RUN_USER}
Group ${APACHE_RUN_GROUP}

#
# HostnameLookups: Log the names of clients or just their IP addresses
# e.g., www.apache.org (on) or 204.62.129.132 (off).
# The default is off because it'd be overall better for the net if people
# had to knowingly turn this feature on, since enabling it means that
# each client request will result in AT LEAST one lookup request to the
# nameserver.
#
HostnameLookups Off

# ErrorLog: The location of the error log file.
# If you do not specify an ErrorLog directive within a <VirtualHost>
# container, error messages relating to that virtual host will be
# logged here.  If you *do* define an error logfile for a <VirtualHost>
# container, that host's errors will be logged there and not here.
#
ErrorLog ${APACHE_LOG_DIR}/error.log

#
# LogLevel: Control the severity of messages logged to the error_log.
# Available values: trace8, ..., trace1, debug, info, notice, warn,
# error, crit, alert, emerg.
# It is also possible to configure the log level for particular modules, e.g.
# "LogLevel info ssl:warn"
#
LogLevel warn

# Include module configuration:
IncludeOptional mods-enabled/*.load
IncludeOptional mods-enabled/*.conf

# Include list of ports to listen on
Include ports.conf
```

```
# Sets the default security model of the Apache2 HTTPD server. It does
# not allow access to the root filesystem outside of /usr/share and /var/www.
# The former is used by web applications packaged in Debian,
# the latter may be used for local directories served by the web server. If
# your system is serving content from a sub-directory in /srv you must allow
# access here, or in any related virtual host.
<Directory />
 Options FollowSymLinks
 AllowOverride None
 Require all denied
</Directory>

<Directory /usr/share>
 AllowOverride None
 Require all granted
</Directory>

<Directory /var/www/>
 Options Indexes FollowSymLinks
 AllowOverride None
 Require all granted
</Directory>

#<Directory /srv/>
# Options Indexes FollowSymLinks
# AllowOverride None
# Require all granted
#</Directory>




# AccessFileName: The name of the file to look for in each directory
# for additional configuration directives.  See also the AllowOverride
# directive.
#
AccessFileName .htaccess

#
# The following lines prevent .htaccess and .htpasswd files from being
# viewed by Web clients.
#
<FilesMatch "^\.ht">
 Require all denied
</FilesMatch>



#
# The following directives define some format nicknames for use with
# a CustomLog directive.
#
```

```
# These deviate from the Common Log Format definitions in that they use %O
# (the actual bytes sent including headers) instead of %b (the size of the
# requested file), because the latter makes it impossible to detect partial
# requests.
#
# Note that the use of %{X-Forwarded-For}i instead of %h is not recommended.
# Use mod_remoteip instead.
#
LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" vhost_combined
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

# Include of directories ignores editors' and dpkg's backup files,
# see README.Debian for details.

# Include generic snippets of statements
IncludeOptional conf-enabled/*.conf

# Include the virtual host configurations:
IncludeOptional sites-enabled/*.conf

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

- **/etc/apache2/ports.conf**: Este archivo especifica los **puertos** por los que Apache escuchará peticiones. Por defecto, Apache escucha en el **puerto 80**. De forma adicional, lo hace en el **443** cuando se habilita el **módulo de SSL**, para HTTPS.

```
# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 80

<IfModule ssl_module>
 Listen 443
</IfModule>

<IfModule mod_gnutls.c>
 Listen 443
</IfModule>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

- **/etc/apache2/sites-available**: Es el directorio en el que se pueden alamacenar **hosts por sitio**. Apache **no utilizará** los archivos de configuración de este directorio a menos que estén

vinculados al directorio **/etc/apache2/sites-enabled**. Normalmente toda la configuración de bloques de servidor se realiza en este directorio y luego se habilita al vincularse al otro directorio con el comando **a2ensite** que ya utilizamos.

```
root@ubuntu-2204:~# ls -l /etc/apache2/sites-available/
total 16
-rw-r--r-- 1 root root 1332 Sep  7 23:07 000-default.conf
-rw-r--r-- 1 root root 6338 Sep 29 20:15 default-ssl.conf
-rw-r--r-- 1 root root  237 Feb  7 13:58 gestal.es.conf
```

- **/etc/apache2/sites-enabled/**: Es el directorio donde se almacenan los **hosts virtuales por sitio habilitados**. Normalmente, se crean vinculando los archivos de configuración del directorio **/etc/apache2/sites-avilable** con el comando **a2ensite** y se desvinculan con **a2dissite**. Apache lee los sitios de este directorio cuando se inicia o se vuelve a cargar.

```
root@ubuntu-2204:~# ls /etc/apache2/sites-enabled/
gestal.es.conf
```

Actualmente sólo tengo habilitado **gestal.es**.

- **/etc/apache2/conf-available/** y **/etc/apache2/conf-enabled/**: estos directorios tienen la misma relación que los directorios sites-available y sites-enabled, pero se utilizan para **almacenar fragmentos de configuración que no pertenecen a un host virtual**. Los archivos del directorio conf-available pueden habilitarse con el comando **a2enconf** y deshabilitarse con el comando **a2disconf**.

```
root@ubuntu-2204:~# ls -l /etc/apache2/conf-available/
total 20
-rw-r--r-- 1 root root  315 Sep  7 23:07 charset.conf
-rw-r--r-- 1 root root 3224 Sep  7 23:07 localized-error-pages.conf
-rw-r--r-- 1 root root  189 Sep  7 23:07 other-vhosts-access-log.conf
-rw-r--r-- 1 root root 2174 Sep  7 23:07 security.conf
-rw-r--r-- 1 root root  455 Sep  7 23:07 serve-cgi-bin.conf
root@ubuntu-2204:~# ls -l /etc/apache2/conf-enabled/
total 0
lrwxrwxrwx 1 root root 30 Feb  5 17:54 charset.conf -> ../conf-available/charset.conf
lrwxrwxrwx 1 root root 44 Feb  5 17:54 localized-error-pages.conf -> ../conf-available/localized-error-
lrwxrwxrwx 1 root root 46 Feb  5 17:54 other-vhosts-access-log.conf -> ../conf-available/other-vhosts-a
lrwxrwxrwx 1 root root 31 Feb  5 17:54 security.conf -> ../conf-available/security.conf
lrwxrwxrwx 1 root root 36 Feb  5 17:54 serve-cgi-bin.conf -> ../conf-available/serve-cgi-bin.conf
```

- **/etc/apache2/mods-available/** y **/etc/apache2/mods-enabled/**: Son directorios que contienen los **módulos disponibles** y los **módulos habilitados**, respectivamente.

- Los archivos que terminan en **.load** contienen fragmentos para cargar módulos específicos.
- Los archivos que terminan en **.conf** contienen la configuración para esos módulos.
- Los módulos pueden **habilitarse y deshabilitarse** con los comandos **a2enmod** y **a2dismod**.

```
root@ubuntu-2204:~# ls -l /etc/apache2/mods-available/
total 568
-rw-r--r-- 1 root root  100 Sep  7 23:07 access_compat.load
-rw-r--r-- 1 root root  377 Sep  7 23:07 actions.conf
-rw-r--r-- 1 root root   66 Sep  7 23:07 actions.load
-rw-r--r-- 1 root root  843 Sep  7 23:07 alias.conf
-rw-r--r-- 1 root root   62 Sep  7 23:07 alias.load
-rw-r--r-- 1 root root   76 Sep  7 23:07 allowmethods.load
-rw-r--r-- 1 root root   76 Sep  7 23:07 asis.load
-rw-r--r-- 1 root root   94 Sep  7 23:07 auth_basic.load
-rw-r--r-- 1 root root   96 Sep  7 23:07 auth_digest.load

(... continúa)


root@ubuntu-2204:~# ls -l /etc/apache2/mods-enabled/
total 0
lrwxrwxrwx 1 root root 36 Feb  5 17:54 access_compat.load -> ../mods-available/access_compat.load
lrwxrwxrwx 1 root root 28 Feb  5 17:54 alias.conf -> ../mods-available/alias.conf
lrwxrwxrwx 1 root root 28 Feb  5 17:54 alias.load -> ../mods-available/alias.load
lrwxrwxrwx 1 root root 33 Feb  5 17:54 auth_basic.load -> ../mods-available/auth_basic.load
lrwxrwxrwx 1 root root 33 Feb  5 17:54 authn_core.load -> ../mods-available/authn_core.load

(...continúa)
```

Registros y logs

- **/var/log/apache2/access.log**: Por defecto cada solicitud enviada se registra en este archivo, a menos que se configure a Apache para no hacerlo.

```
root@ubuntu-2204:~# cat /var/log/apache2/access.log
127.0.0.1 - - [07/Feb/2023:14:07:10 -0500] "GET / HTTP/1.1" 200 469 "-" "Mozilla/5.0 (X11; Ubuntu; Linu
127.0.0.1 - - [07/Feb/2023:14:07:10 -0500] "GET /favicon.ico HTTP/1.1" 404 487 "http://127.0.0.1/" "Moz
127.0.0.1 - - [07/Feb/2023:14:07:20 -0500] "GET /favicon.ico HTTP/1.1" 404 488 "http://127.0.0.1/" "Moz
127.0.0.1 - - [07/Feb/2023:14:07:21 -0500] "GET / HTTP/1.1" 200 468 "-" "Mozilla/5.0 (X11; Ubuntu; Linu
127.0.0.1 - - [08/Feb/2023:11:03:08 -0500] "GET / HTTP/1.1" 200 469 "-" "Mozilla/5.0 (X11; Ubuntu; Linu
127.0.0.1 - - [08/Feb/2023:11:03:08 -0500] "GET /favicon.ico HTTP/1.1" 404 487 "http://gestal.es/" "Moz
```

- **/var/log/apache2/error.log**: Por defecto, todos los **errores** se registran en este archivo. La directiva **LogLevel** de la configuración de Apache especifica el nivel de detalle de los registros

de log.

```
root@ubuntu-2204:~# cat /var/log/apache2/error.log
[Tue Feb 07 13:22:17.266339 2023] [mpm_event:notice] [pid 7563:tid 140277510547328] AH00489: Apache/2.4
[Tue Feb 07 13:22:17.266362 2023] [core:notice] [pid 7563:tid 140277510547328] AH00094: Command line: '
[Tue Feb 07 14:06:02.709426 2023] [mpm_event:notice] [pid 7563:tid 140277510547328] AH00492: caught SIG
[Tue Feb 07 14:06:02.774924 2023] [mpm_event:notice] [pid 56488:tid 139972950738816] AH00489: Apache/2.
[Tue Feb 07 14:06:02.775064 2023] [core:notice] [pid 56488:tid 139972950738816] AH00094: Command line:
```

Configurando SSL

Para que nuestro servidor sea seguro, debemos activar **HTTPS**, es decir, HTTP sobre una conexión SSL (Secure Sockets Layer). Así agregaremos un protocolo de **cifrado asimétrico** al HTTP común. Esto, en principio, sería solo necesario en los sitios que traten con **autenticación** o en apps que **intercambien datos**, pero hoy en día es necesario configurarlo en cualquier página web disponible online, puesto que navegadores como **Chrome** utilizan este protocolo por defecto para mejorar la **privacidad y seguridad** de los usuarios. Google penalizará nuestro sitio web si no ofrecemos **HTTPS**.

Así que vamos a **habilitar la conexión SSL en nuestro servidor**.

Certificados SSL

Para poder establecer una **conexión segura**, nuestro servidor Apache necesitará un **certificado SSL** que puede ser obtenido a través de una **Autoridad de Certificación (CE)**.

Por conveniencia y dado que nuestro servidor se encuentra en una máquina virtual en nuestro equipo, creamos nosotros este certificado, que puede ser utilizado en **entornos de prueba**.

- Si se quiere obtener un certificado de una **Autoridad de Certificación**, se puede seguir esta guía para obtener un certificado de Let's Encrypt.

Creando un certificado Self-Signed SSL

Los certificados **SSL Self-Signed** son certificados **autofirmados** que se utilizan en nuestra máquina local o nuestro servidor remoto cuando **no hay un certificado disponible** de una **autoridad de certificación externa**.

Estos certificados **no se pueden utilizar en sistemas en producción** porque no garantizan el **nivel adecuado de confiabilidad** ya que no están verificados por una **Autoridad de Certificación**.

1. Creando una clave privada

Primero hay que crear una **clave privada** para generar el **certificado público**.

Con este comando especificamos la creación de una **clave privada** con una longitud de **2048 bits** que se guardará en el archivo **private.key**.

```
sudo openssl genrsa -aes128 -out private.key 2048
```

Se nos solicitará una clave por consola:

```
root@ubuntu-2204:~# sudo openssl genrsa -aes128 -out private.key 2048
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

Yo le he puesto como clave **patata**, pero en teoría se debería de utilizar una clave segura.

2. Creando un Certificate Signing Request (CSR)

Tras generar la **clave privada**, debemos crear un **certificate signing request (CSR)** que especificará los detalles del certificado:

```
sudo openssl req -new -days 365 -key private.key -out request.csr
```

Y se nos solicitarán ciertos datos:

```
root@ubuntu-2204:~# sudo openssl genrsa -aes128 -out private.key 2048
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
root@ubuntu-2204:~# openssl req -new -days 365 -key private.key -out request.csr
Ignoring -days without -x509; not generating a certificate
Enter pass phrase for private.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Galicia
Locality Name (eg, city) []:Santiago
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Gestal
Organizational Unit Name (eg, section) []:Mi casa
Common Name (e.g. server FQDN or YOUR name) []:gestal.es
Email Address []:juan@gestal.es

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:patata
An optional company name []:Blah
```

Entre los datos, se nos dice que protejamos nuestro certificado con una clave. Yo he vuelto a poner **patata**, ya que esto es una práctica.

3. Generando el certificado SSL

Ahora ya podemos proceder a generar el certificado, donde se nos pedirá la contraseña:

```
sudo openssl x509 -in request.csr -out certificate.crt -req -signkey private.key -days 365
```

- El parámetro **-in** contiene el **certificate signing request.
- El parametro **-out** especifica el nombre del fichero que contendrá el certificado.
- Para **-signkey** se especifica el fichero **private.key**.
- Para **-days** se especifica el número de días de validez del certificado. En nuestro caso, un año.

4. Configurando Apache con el certificado creado

Tras obtener el certificado, creamos el diectorio **/etc/certificate** y copiamos el **certificado** y la **clave privada** allí:

```
sudo mkdir /etc/certificate
sudo mv private.key /etc/certificate/
sudo mv certificate.crt /etc/certificate/
```

Y ahora toca **configurar las directivas de Apache** para la **conexión segura**. Para ello, creamos el archivo **ssl-params.conf** en el directorio de Apache **/etc/apache2/conf-available/**.

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

Y le añadimos el siguiente contenido:

```
SSLCipherSuite EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH

    SSLProtocol All -SSLv2 -SSLv3 -TLSv1 -TLSv1.1

    SSLHonorCipherOrder On


    Header always set X-Frame-Options DENY

    Header always set X-Content-Type-Options nosniff

    # Requires Apache >= 2.4

    SSLCompression off

    SSLUseStapling on

    SSLStaplingCache "shmcb:logs/stapling-cache(150000)"


    # Requires Apache >= 2.4.11

    SSLSessionTickets Off
```

Salvamos y cerramos el fichero.

5. Configurando el Virtual Host que queremos proteger con SSL

Ahora debemos **modificar la configuración** del **virtual host** que queremos proteger con la conexión SSL.

```
sudo nano /etc/apache2/sites-available/gestal.es.conf
```

La configuración actual es esta:

```
<VirtualHost *:80>
  ServerAdmin juan@gestal.es
  ServerName gestal.es
  ServerAlias www.gestal.es
  DocumentRoot /var/www/gestal.es
  ErrorLog ${APACHE_LOG_DIR}/error.log
  CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Y lo dejamos así:

```
<VirtualHost *:80>
  ServerAdmin juan@gestal.es
  ServerName gestal.es
  ServerAlias www.gestal.es
  DocumentRoot /var/www/gestal.es
  ErrorLog ${APACHE_LOG_DIR}/error.log
  CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>

<IfModule mod_ssl.c>
            <VirtualHost _default_:443>

                    ServerAdmin juan@gestal.es
                    ServerName gestal.es
                    ServerAlias www.gestal.es
                    DocumentRoot /var/www/gestal.es
                    ErrorLog ${APACHE_LOG_DIR}/error.log
                    CustomLog ${APACHE_LOG_DIR}/access.log combined

                    SSLEngine on

                    SSLCertificateFile /etc/certificate/certificate.crt
                    SSLCertificateKeyFile /etc/certificate/private.key

                    <FilesMatch "\.(cgi|shtml|phtml|php)$">
                                    SSLOptions +StdEnvVars
                    </FilesMatch>

                    <Directory /usr/lib/cgi-bin>
                                    SSLOptions +StdEnvVars
                    </Directory>

            </VirtualHost>
</IfModule>
```

6. Reconfigurando el firewall

En el caso de que tengamos el equipo detrás de un firewall, debemos configurarlo para que permita el tráfico **HTTP** y **HTTPS**.

Así vemos los perfiles del firewall:

```
sudo ufw app list
```

Que nos devolverá:

```
Available applications:
  Apache
  Apache Full
  Apache Secure
  CUPS
  OpenSSH
```

Y para permitir el tráfico HTTP (puerto **80**) y HTTPS (puerto **443**) debemos usar el perfil **Apache Full**.

Esto podemos comprobarlo pidiendo la descripción del perfil:

```
sudo ufw app info "Apache Full"
```

Obteniendo:

```
Profile: Apache Full
Title: Web Server (HTTP,HTTPS)
Description: Apache v2 is the next generation of the omnipresent Apache web
server.

Ports:
  80,443/tcp
```

Así que activamos este perfil:

```
sudo ufw allow in "Apache Full"
```

7. Configurando los módulos de Apache

Debemos **activar los módulos mod_ssl** y **mod_headers** en Apache. Lo hacemos con el comando **a2enmod**:

```
sudo a2enmod ssl
sudo a2enmod headers
```

Ahora **activamos la ocnfiguración** creada anteriormente:

```
sudo a2enconf ssl-params
```

Activamos el **virtual host** con SSL.

```
sudo a2ensite gestal.es
```

Verificamos que no haya errores de sintaxis:

```
sudo apache2ctl configtest
```

Y reiniciamos el servicio:

```
sudo service apache2 restart
```

Al reiniciar el servicio se nos solicitará la clave (patata):

```
root@ubuntu-2204:~# service apache2 restart
🔐 Enter passphrase for SSL/TLS keys for gestal.es:443 (RSA): (press TAB for no ******
```

Y ya tendríamos activado **HTTPS** en nuestro servidor.

Al visitar desde un navegador nuestro sitio, se nos notificará que el certificado no es confiable puesto que no procede de una **Autoridad Certificadora**, pero podemos darle a "aceptar el riesgo y continuar". **¡Y ya tendríamos HTTPS activado!**

```
https://gestal.es
```

Instalando LAMP

LAMP son las siglas de **Linux**, **Apache**, **MySQL** y **PHP**. Es muy común tener una configuración de este estilo en un servidor, por ejemplo, para correr **Wordpress**, el CMS más famoso.

Instalando mysql

Para poder guardar y manipular datos necesitamos un **sistema gestor de base de datos** y **MySQL** es uno muy utilizado en entornos en los que también se utiliza **PHP**.

Instalamos MySQL:

```
sudo apt install mysql-server -y
```

Al terminar la instalación, es recomendable correr un script que viene preinstalado con Mysql. Este script eliminará algunos **ajustes inseguros** que vienen en la **configuración por defecto y bloqueará el acceso a la base de datos para que solo se acceda desde el propio sistema**. Este script también nos permitirá configurar el **nivel de fortaleza de las claves**.

Yo voy a optar por **no correr el script**, ya que no me preocupa la seguridad en esta práctica, pero consistiría en ejecutar este comando y seguir los pasos:

```
sudo mysql_secure_installation
```

Tras instalar **mysql** podemos entrar en su consola tecleando:

```
sudo mysql
```

Y deberíamos ver:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.32-0ubuntu0.22.04.2 (Ubuntu)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Para salir tecleamos **exit**:

```
exit
```

Instalando PHP

Disponemos de **Apache** para servir el contenido y de **Mysql** para guardar y manejr los datos. Nos queda instalar **PHP** para poder servir webs con **contenido dinámico** al usuario.

Necesitamos instalar tres cosas:

1. **php**: Para instalar php.
2. **libapache2-mod-php**: Un **módulo de Apache** para manejar archivos php.
3. **php-mysql**: Un módulo que permite a **PHP** comunicarse con bases de datos **mysql**.

Por lo tanto escribimos:

```
sudo apt install php libapache2-mod-php php-mysql -y
```

Cuando la instalación finalice, podemos comprobar la versión de **php** con el comando:

```
php -v
```

Lo que nos devolverá algo parecido a:

```
PHP 8.1.2-1ubuntu2.10 (cli) (built: Jan 16 2023 15:19:49) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.1.2, Copyright (c) Zend Technologies
    with Zend OPcache v8.1.2-1ubuntu2.10, Copyright (c), by Zend Technologies
```

Ahora reiniciamos el servicio y comprobamos que esté activo con:

```
sudo service apache2 restart
```

```
sudo service apache2 status
```

Probando PHP

Ahora creamos en el directorio **/var/www/gestal.es/** un archivo de prueba de PHP llamado **index.php**:

```
nano /var/www/gestal.es/prueba.php
```

Con el contenido:

```
<?php
phpinfo();
```

Salvamos el fichero y visitamos https://gestal.es/prueba.php con un navegador y deberíamos ver la **página por defecto de PHP**.

¡Nuestro PHP ya funciona!

- Si quisiéramos testear la conexión a la **base de datos desde PHP** podríamos seguir a partir del paso 6 de este tutorial.

index.html e index.php

Ahora renombramos el archivo /var/www/gestal.es/prueba.php a /var/www/gestal.es/index.php y visitamos https://gestal.es.

```
sudo mv /var/www/gestal.es/prueba.php /var/www/gestal.es/index.php
```

Veremos que el archivo que se nos muestra por defecto es **index.html** y no **index.php**.

El motivo de que se muestre **index.html** es que tiene precedencia sobre el **index.php** en la configuración por defecto del Apache. Esto es útil por si queremos crear un **index.html** con información temporal mientras tenemos una página en mantenimiento. Una vez finalicemos el mantenimiento podemos renombrar o borrar el archivo **index.html**.

En el caso de que quisiéramos cambiar este comportamiento, tendríamos que editar el fichero **/etc/apache2/mods-enabled/dir.conf**:

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

Y modificar el orden de precedencia en la directiva **DirectoryIndex**:

```
<IfModule mod_dir.c>
        DirectoryIndex index.php index.html index.cgi index.pl index.xhtml index.htm
</IfModule>
```

Salvar los cambios y recargar el servidor:

```
sudo service apache2 restart
```

Ficheros .htaccess

Introducción a los ficheros .htaccess

Los ficheros **.htaccess** son **ficheros de configuración distribuída** y facilitan una forma de hacer **cambios en la configuración en un contexto de directorio**.

Son ficheros que **se colocan en determinados directorios** y las directivas que contienen **se aplican a ese directorio y a todos sus subdirectorios**.

¿Por qué son útiles? Pues imaginemos que tenemos un servidor web en el que hay varias páginas de varios clientes. Los clientes **no van a tener acceso al fichero de configuración principal de Apache**, solo el administrador tendrá acceso a ese fichero. Pero **pueden crear ficheros .htaccess** y realizar sus **configuraciones**.

Se debería **evitar el uso** de ficheros **.htaccess** completamente si se tiene **acceso al fichero de configuración principal de Apache**, puesto que usar ficheros .htaccess **ralentiza el servidor**. Cualquier directiva que pueda incluír un fichero .htaccess estará mejor configurada dentro de una sección **Directory**, tendrá el **mismo efecto y mejor rendimiento**.

También hay que sopesar desde el punto de vista de la **seguridad** qué privilegios se dan a los usuarios si se les permite usar ficheros **.htaccess**, ya que serían cambios sobre los que el administrador no tendría ningún control. Lo que se puede o no utilizar en estos ficheros viene determinado por la directiva de Apache **AllowOverride**.

- Más información: Ficheros .htaccess

Ejemplo de uso de ficheros .htaccess

Vamos a crear un fichero **.htaccess** que permita listar los ficheros de un directorio.

Lo primero es crear un directorio en nuestro sitio con varios ficheros vacíos:

```
mkdir /var/www/gestal.es/directorio_listado
touch /var/www/gestal.es/directorio_listado/file1.txt
touch /var/www/gestal.es/directorio_listado/file2.txt
touch /var/www/gestal.es/directorio_listado/file3.txt
```

Si accedemos a la dirección http://gestal.es/directorio_listado vamos a ver una lista de ficheros en el navegador.

Puede ser que no nos interese que los visitantes de nuestra web puedan ver estos ficheros listados, así que vamos a crear un archivo .htaccess que prohíba esta opción.

```
nano /var/www/gestal.es/directorio_listado/.htaccess
```

Con el contenido:

```
Options -Indexes
```

Que va a prohibir los listados de ficheros.

Por defecto, el archivo **.htaccess** no está activado, así que vamos a editar la configuración de nuestro sitio:

```
sudo nano /etc/apache2/sites-available/gestal.es.conf
```

Dejando el archivo de la siguiente manera:

```
<VirtualHost *:80>
  ServerAdmin juan@gestal.es
  ServerName gestal.es
  ServerAlias www.gestal.es
  DocumentRoot /var/www/gestal.es
  <Directory /var/www/gestal.es/>
            AllowOverride All
            Order allow,deny
            allow from all
  </Directory>
  AccessFileName .htaccess
  ErrorLog ${APACHE_LOG_DIR}/error.log
  CustomLog ${APACHE_LOG_DIR}/access.log combined

</VirtualHost>

<IfModule mod_ssl.c>
            <VirtualHost _default_:443>

                    ServerAdmin juan@gestal.es
                    ServerName gestal.es
                    ServerAlias www.gestal.es
                    DocumentRoot /var/www/gestal.es
                    <Directory /var/www/gestal.es/>
              AllowOverride All
              Order allow,deny
              allow from all
        </Directory>
        AccessFileName .htaccess
                    ErrorLog ${APACHE_LOG_DIR}/error.log
                    CustomLog ${APACHE_LOG_DIR}/access.log combined

                    SSLEngine on

                    SSLCertificateFile /etc/certificate/certificate.crt
                    SSLCertificateKeyFile /etc/certificate/private.key

                    <FilesMatch "\.(cgi|shtml|phtml|php)$">
                                    SSLOptions +StdEnvVars
                    </FilesMatch>

                    <Directory /usr/lib/cgi-bin>
                                    SSLOptions +StdEnvVars
                    </Directory>

            </VirtualHost>
</IfModule>
```

También activamos el módulo **rewrite** con el comando:

```
sudo a2enmod rewrite
```

Recargamos el apache

```
sudo service apache2 restart
```

Y veremos que si accedemos de nuevo a http://gestal.es/directorio_listado obtenemos un código de estado **443** de parte del servidor.

```
Forbidden

You don't have permission to access this resource.
Apache/2.4.52 (Ubuntu) Server at gestal.es Port 443
```

Vamos a crear otro fichero .htaccess en el directorio **/var/www/gestal.es/** para personalizar nuestra página **404**, la que se muestra a los clientes cuando solicitan una URL de nuestro sitio que no existe.

Creamos el fichero:

```
sudo nano /var/www/gestal.es/.htaccess
```

Con el contenido:

```
ErrorDocument 404 /404.html
```

Ahora creamos también un fichero llamado **404.html**:

```
sudo nano /var/www/gestal.es/404.html
```

Con el contenido en HTML:

```
<html>
  <head>
    <title>404</title>
  </head>
  <body>
    <h1>Mi error 404 personalizado</h1>
  </body>
</html>
```

Y ahora visitamos una URL que no exista, por ejemplo: http://gestal.es/tortilla.html y se verá nuestra página personalizada.

Por último, vamos a **forzar las conexiones SSL**, para que los usuarios que visiten http://gestal.es sean redirigidos a https://gestal.es. Para ello añadimos al fichero **/var/www/gestal.es/.htaccess** las siguientes líneas de código:

```
RewriteEngine On
RewriteCond %{HTTPS} off
RewriteRule ^(.*)$ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]
```

Y ahora, si visitamos http://gestal.es seremos redirigidos a https://gestal.es.

Servidor Nginx

Nginx es actualmente **el servidor web más utilizado en el mundo**, debido a que a pesar de que no es tan configurable como Apache, es mucho más rápido. Además se puede utilizar como servidor web y como proxy inverso.

Instalación de Niginx

- Web Oficial: https://www.nginx.com

Para installar **nginx** simplemente abrimos el terminal y tecleamos:

```
sudo apt update
sudo apt install nginx –y
```

Ajustando el Firewall

Activamos el firewall:

```
sudo ufw enable
```

Para permitir acceso al servicio debemos configurarlo. Podemos ver las aplicaciones que **ufw** conoce con el comando:

```
sudo ufw app list
```

Y obtendremos:

```
Available applications:
  CUPS
  Nginx Full
  Nginx HTTP
  Nginx HTTPS
  OpenSSH
```

Existen tres perfiles disponibles:

- **Nginx Full**: Abre el puerto 80 (HTTP, tráfico sin encriptar) y el puerto 443 (HTTPS, tráfico encriptado).
- **Nginx HTTP**: Abre solo el puerto **80** (HTTP, tráfico sin encriptar).
- **Nginx HTTPS**: Abre solo el puerto **443** (tráfico encriptado TLS/SSL).

Es recomendable utilizar el perfil más restrictivo. Por ahora solo necesitaremos abrir el **puerto 80**:

```
sudo ufw allow 'Nginx HTTP'
```

Podemos verificar el cambio con:

```
sudo ufw status
```

Obteniedo la salida:

```
Status: active

To                         Action      From
--                         ------      ----
Nginx HTTP                 ALLOW       Anywhere
Nginx HTTP (v6)            ALLOW       Anywhere (v6)
```

Comprobar el servicio Nginx

Podemos chequear el estado de Nginx con el siguiente comando:

```
ubuntu@ubuntu-2204:~$ service nginx status
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset:>
     Active: active (running) since Wed 2023-02-08 18:58:25 EST; 7min ago
       Docs: man:nginx(8)
   Main PID: 3382 (nginx)
      Tasks: 3 (limit: 4620)
     Memory: 3.9M
        CPU: 36ms
     CGroup: /system.slice/nginx.service
             ├─3382 "nginx: master process /usr/sbin/nginx -g daemon on; master>
             ├─3384 "nginx: worker process" "" "" "" "" "" "" "" "" "" "" "" "">
             └─3385 "nginx: worker process" "" "" "" "" "" "" "" "" "" "" "" "">

Feb 08 18:58:25 ubuntu-2204 systemd[1]: Starting A high performance web server >
Feb 08 18:58:25 ubuntu-2204 systemd[1]: Started A high performance web server a>
lines 1-15/15 (END)
```

Podemos obtener la IP del servidor con el comando:

```
hostname -I
```

E intentar entrar en la página web de Apache desde otra máquina virtual o equipo en red.

Para parar el servicio tecleamos:

```
sudo service nginx stop
```

Para iniciar el servicio tecleamos:

```
sudo service nginx start
```

Para parar y arrancar el servicio tecleamos:

```
sudo service nginx restart
```

Si solo estás haciendo cambios de configuración, Nginx puede recargarlos sin descargar las conexiones de los clientes:

```
sudo service nginx reload
```

Por defecto, nginx está configurado para comenzar automáticamente cuando el equipo arranca. Si no es el comportamiento que se quiere:

```
sudo service nginx disable
```

Para reactivar el inicio en el arranque:

```
sudo service nginx enable
```

Si abrimos con un navegador la página web http://127.0.0.1/ en la máquina virtual del servidor, o utilizando su IP, nos encontraremos con la **web de bienvenida de Nginx**:

```
Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.
```

Configurando Server Blocks

Los **server blocks** son similares a los **virtual hosts** en Apache. Se pueden utilizar para encapsular detalles de configuración de más de un dominio en un mismo servidor. Yo utilizaré como dominio **gestal.es**.

Nginx tiene un server block activado por defecto para servir los documentos del directorio **/var/www/html**. Esto funciona para un único sitio, pero lo normal es que un servidor pueda servir diferentes páginas de diferentes dominios.

Creamos el directorio. La opción **-p** de **mkdir** nos permite crear cualquier padre de directorio necesario. Así con un comando, creamos el directorio /var/www/gestal.es/ y /var/www/gestal.es/html a la vez.

```
sudo mkdir -p /var/www/gestal.es/html
```

Asignamos la propiedad del directorio al usuario, con la variable de entorno $USER.

```
sudo chown -R $USER:$USER /var/www/gestal.es/html
```

Los permisos deberían de estar bien si no se ha modificado el valor por defecto de **umask**, pero para asegurarse:

```
sudo chmod -R 755 /var/www/gestal.es
```

Y ahora creamos el archivo **/var/www/gestal.es/index.html**:

```
sudo nano /var/www/gestal.es/html/index.html
```

Con un contenido básico en HTML:

```
<html>
  <head>
      <title>Web de gestal.es</title>
  <body>
    <h1>Mi web es genial</h1>
  </body>
</html>
```

Y para que Nginx comience a servir esta página web es **necesario crear un server block** con las **directivas necesarias**.

El archivo de configuración por defecto se encuentra en **/etc/nginx/sites-available/default**:

```
##
# You should look at the following URL's in order to grasp a solid understanding
# of Nginx configuration files in order to fully unleash the power of Nginx.
# https://www.nginx.com/resources/wiki/start/
# https://www.nginx.com/resources/wiki/start/topics/tutorials/config_pitfalls/
# https://wiki.debian.org/Nginx/DirectoryStructure
#
# In most cases, administrators will remove this file from sites-enabled/ and
# leave it as reference inside of sites-available where it will continue to be
# updated by the nginx packaging team.
#
# This file will automatically load configuration files provided by other
# applications, such as Drupal or Wordpress. These applications will be made
# available underneath a path with that package name, such as /drupal8.
#
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.
##

# Default server configuration
#
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        # SSL configuration
        #
        # listen 443 ssl default_server;
        # listen [::]:443 ssl default_server;
        #
        # Note: You should disable gzip for SSL traffic.
        # See: https://bugs.debian.org/773332
        #
        # Read up on ssl_ciphers to ensure a secure configuration.
        # See: https://bugs.debian.org/765782
        #
        # Self signed certs generated by the ssl-cert package
        # Don't use them in a production server!
        #
        # include snippets/snakeoil.conf;

        root /var/www/html;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;

        server_name _;

        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
```

```
                try_files $uri $uri/ =404;
        }

        # pass PHP scripts to FastCGI server
        #
        #location ~ \.php$ {
        #       include snippets/fastcgi-php.conf;
        #
        #       # With php-fpm (or other unix sockets):
        #       fastcgi_pass unix:/run/php/php7.4-fpm.sock;
        #       # With php-cgi (or other tcp sockets):
        #       fastcgi_pass 127.0.0.1:9000;
        #}

        # deny access to .htaccess files, if Apache's document root
        # concurs with nginx's one
        #
        #location ~ /\.ht {
        #       deny all;
        #}
}


# Virtual Host configuration for example.com
#
# You can move that to a different file under sites-available/ and symlink that
# to sites-enabled/ to enable it.
#
#server {
#       listen 80;
#       listen [::]:80;
#
#       server_name example.com;
#
#       root /var/www/example.com;
#       index index.html;
#
#       location / {
#               try_files $uri $uri/ =404;
#       }
#}
```

Pero en lugar de modificar este archivo de configuración por defecto directamente, crearemos uno nuevo en **/etc/nginx/sites-available/gestal.es** con el siguiente contenido:

```
server {
        listen 80;
        listen [::]:80;

        root /var/www/gestal.es/html;
        index index.html index.htm index.nginx-debian.html;

        server_name gestal.es www.gestal.es;

        location / {
                try_files $uri $uri/ =404;
        }
}
```

Y ahora vamos a activar el fichero creando un **link** hacia el fichero desde el directorio **sites-enabled**, que Nginx lee cuando inicia. Esta es una práctica común para poder **activar y desactivar sitios**. Para desactivar un sitio simplemente hay que **borrar el link del directorio sites-enabled**, y se mantiene el **server block** en el directorio **sites-available** por si más tarde queremos activarlo de nuevo.

```
sudo ln -s /etc/nginx/sites-available/gestal.es /etc/nginx/sites-enabled/
```

Comprobando la sintaxis y reiniciando el servicio

Para comprobar si la sintaxis de los archivos de configuración es correcta, ejecutamos el comando:

```
sudo nginx -t
```

Y si nos dice que está bien, reiniciamos el servicio:

```
sudo service nginx restart
```

Editando el fichero /etc/hosts

Ahora necesitamos que el nombre **gestal.es** se resuelva a la IP del servidor.

Una opción es instalar y configurar bind9 para que resuelva **gestal.es** a nuestra máquina servidor. Otra opción es editar el fichero **/etc/hosts** para que el equipo resuelva "gestal.es" a la IP que le digamos. Como voy a probar desde la propia máquina virtual:

Edito el fichero hosts:

```
sudo nano /etc/hosts
```

Y añado la línea:

```
127.0.0.1       gestal.es
```

Y si ahora abro el navegador y tecleo http://gestal.es, debería de ver la página que he creado. Es posible que tenga que limpiar los datos del navegador ("Cookies and Site Data", en Firefox, dentro de la sección "Privacy"), si me he olvidado de hacer esto y el ordenador tenga cacheada la dirección IP de gestal.es que está en Internet.

Directorios y archivos de configuración de Nginx

Contenido

- **/var/www/html**: Es el directorio web por defecto, que contiene la página por defecto de nginx. Se puede cambiar en los archivos de configuración.
- **/var/www/gestal.es/html**: Es el directorio web que hemos creado para nuestro sitio.

Configuración

- **/etc/nginx/nginx.conf**: Es el directorio de configuración principal de nginx. Puede ser modificado para configuraciones globales.
- **/etc/nginx/sites-available/**: Es el directorio donde se almacenan los bloques para cada sitio. Nginx no usará los archivos de configuración de este directorio a menos que estén enlazados al directorio **/etc/nginx/sites-enabled/**. Típicamente la configuración de los server blocks se hace en este directorio y luego se enlaza.
- **/etc/nginx/snippets:** Este directorio contiene fragmentos que pueden ser incluídos en cualquier lugar de la configuración de **nginx**. Los fragmentos son trozos de configuración que se pueden repetir en varios sitios.

Server Logs

- **/var/log/nginx/access.log**: Todas las peticiones al servidor son registradas en este archivo, a menos que configuremos nginx para que las guarde en otro sitio.
- **/var/log/nginx/error.log: Todos los errores estarán registrados en este archivo.

Más sobre nginx

Para securizar el servicio, o para instalar PHP y MySQL se pueden seguir los pasos detallados en los siguientes tutoriales:

- Tutorial: Securizar un servidor nginx con SSL
- Tutorial: Instalar una pila LEMP

Fuentes

- How to install the apache web server on Ubuntu 20.04
- How to install Linux Apache Mysql PHP LAMP stack on Ubuntu 20.04
- How to enable HTTPs protocol with Apache2 on Ubuntu 20.04
- Apache: Permisos de usuario y grupo
- How to create a self signed SSL certificate on Ubuntu 18.04
- Tutorial del Servidor Apache HTTP: Ficheros .htaccess
- Ejemplos de uso de .htaccess
- How to install nginx on Ubuntu 20.04