



DESPLIEGUE DE  
APLICACIONES  
WEB



# Documentación sobre la instalación de un servidor web.

Recopilación de guías y documentos para la instalación y utilización del software de apache para que tu ordenador se pueda convertir en un servidor web.

**Por Ismael Sarrion  
y Gonzalo Martinez.**

20/11/2023



# ÍNDICE

## Primeros Pasos.....02

- Instalar ubuntu server
- Conectarse a una red wifi
- Datos de interés
- Instalación de apache2

## Ejercicio4..... 06

- 4.1
- 4.2
- 4.3
- 

## Ejercicio1..... 03

- 1.1
- 1.2
- 1.3
- 1.4

## Ejercicio5..... 05

- 3.1
- 3.2
- 3.3

## Ejercicio2..... 04

- 2.1
- 2.2

## Ejercicio6..... 10

- 6.1
  - 6.1.1
  - 6.1.2
  - 6.1.3
  - 6.1.4
- 6.2
- 6.3
- 6.4
  - 6.4.1
  - 6.4.2
  - 6.4.3
  - 6.4.4

## Ejercicio3..... 05

- 3.1
- 3.2
- 3.3

# INTRODUCCIÓN

Apache es un servidor web, pero en realidad es el software que se instala en un ordenador para que pueda convertirse en un servidor web.

Digamos que en tu ordenador personal tienes instalado un software, un sistema operativo que lo hace funcionar, como por ejemplo Windows o Linux.

Un servidor no es más que un ordenador, eso sí, con muchos recursos y conectados a una red con mucho ancho de banda, todos los días del año y a todas horas.

Para que esa máquina pueda convertirse en un servidor web necesita tener un software instalado y este software sería Apache.

De hecho, si accedes a la web de Apache, verás que en realidad se llama Apache HTTP Server.

Por otro lado, HTTP es el protocolo que se encarga de comunicar los navegadores web como Chrome, Safari o Firefox, con un servidor para mostrar páginas web.



# PRIMEROS PASOS

Antes de iniciar, es importante cambiar varias cosas de nuestro equipo para que podáis seguir nuestra documentación sin ningún problema.

## 1. INSTALAR UBUNTU SERVER

### Option 1: Manual server installation

USB or DVD image based physical install

- OS security guaranteed until April 2027
- Expanded security maintenance until April 2032
- Commercial support for enterprise customers



[Download Ubuntu Server 22.04.3 LTS](#)

[Alternative downloads](#) > [Alternative architectures](#)

Ya sea en una **maquina virtual o en un ordenador** que usaremos como servidor, deberemos instalar **ubuntu server**; esta versión de ubuntu no usa una interfaz grafica, ya que esta puede ralentizarnos en el proceso de crear nuestro servidor.

## 2. CONECTARSE A UNA RED WIFI

Al no tener interfaz grafica este paso es muy importante ya que deberemos conectarnos a una **red wifi** en el proceso de instalación de ubuntu server.

## 3. INSTALACIÓN DE APACHE2

Por ultimo, antes de empezar deberemos **instalar el servidor web de apache** con este comando.

***sudo apt-get install apache2***

```
tecmint@ubuntu-20-04:~$ sudo apt install apache2
[sudo] password for tecmint:
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser
The following NEW packages will be installed:
  apache2
```

Ahora que hemos completado la instalación de ubuntu ya podremos empezar a configurarlo.

# EJERCICIO 1

## Investigar sobre Hosting Virtual y crear un ejemplo en Apache con dos Hosting Virtuales en local

- 1.1** Lo primero que debemos hacer es crear dentro del directorio **/etc/apache2/sites-available** 2 archivos .conf llamados **Apache1.conf** y **Apache2.conf**. Esto lo haremos con los comando:

```
$sudo nano /etc/apache2/sites-available/Apache1.conf
```

```
$sudo nano /etc/apache2/sites-available/Apache2.conf
```

- 1.2** Seguidamente, deberemos copiar el contenido de **/etc/apache2/sites-available/000-default.conf** en los 2 archivos que hemos creado antes. Esto lo haremos con los comando:

```
$sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/Apache1.conf
```

```
$sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/Apache2.conf
```

- 1.3** Tenemos los ficheros creados, pero todavía **no están activos**, si nos vamos a site-enabled vemos que no están añadidos los enlaces simbólicos para que formen parte del servidor. Vamos a activarlos.

```
$a2ensite Apache1.conf
```

```
$a2ensite Apache2.conf
```

- 1.4** Por ultimo **reiniciaremos** el servidor para que se apliquen los cambios. Esto lo haremos con el comando:

```
$sudo systemctl status apache2
```

The screenshot shows a terminal window with the title 'usuario@usuario: /etc/apache2'. The file being edited is 'Apache1.conf'. The content of the file is as follows:

```
GNU nano 6.2                                         Apache1.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
    ServerName www.apache.gonzalo.com
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/apache1.es/public
</VirtualHost>
<VirtualHost *:443>
    ServerName www.tudominio.com
    SSLEngine on
    SSLCertificateFile /etc/apache2/ssl/apache.crt
    SSLCertificateKeyFile /etc/apache2/ssl/apache.key
    DocumentRoot /var/www/html
</VirtualHost>

    # 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
```

At the bottom of the terminal window, there is a menu bar with various keyboard shortcuts for nano editor, such as Help, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, Redo, Set Mark, To Bracket, Copy, and Where Was.

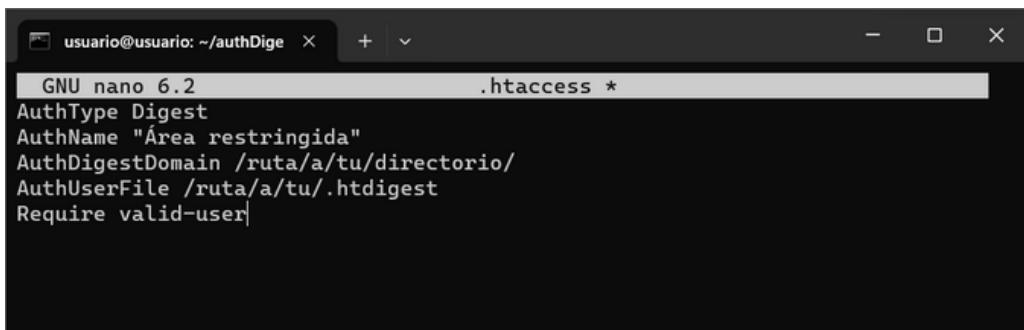
## EJERCICIO 2

**Experimenta con las opciones de directorios .htaccess y con los enlaces simbólicos.**

- 2.1** Los archivos **.htaccess** permiten cambiar algunas configuraciones del archivo que estamos mostrando en nuestro servidor. En nuestro caso, lo utilizaremos para que cuando nuestro servidor nos de error, no **revele toda la información** de la pagina que estamos mostrando.

```
$sudo nano /etc/apache1/.htaccess
```

```
usuario@usuario:~/authDigest$ sudo nano .htaccess
```



```
usuario@usuario: ~/authDigest X + - □ ×
GNU nano 6.2 .htaccess *
AuthType Digest
AuthName "Área restringida"
AuthDigestDomain /ruta/a/tu/directorio/
AuthUserFile /ruta/a/tu/.htdigest
Require valid-user
```

\*esta foto es de  
un .htaccess  
posterior\*

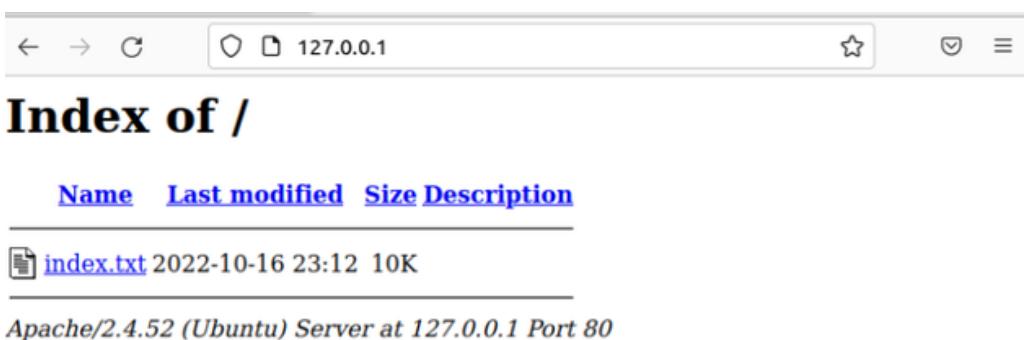
dentro de este archivo escribiremos:

**Options -Indexes**

- 2.2** Ahora crearemos un **enlace simbolico** que apunte al directorio /etc/apache1

```
$ln -s /etc/apache1 /etc/www/html/PaginaWeb
```

ahora si nuestro servidor tuviera algun problema no revelaria los archivos de nuestro proyecto, como ocurre en la siguiente foto



# EJERCICIO 3

**Qué son los módulos de apache, tipos, para que sirven. Instala el módulo userDir.**

- 3.1** En el contexto de **Apache**, los **módulos** son extensiones de software que se pueden cargar dinámicamente en el servidor web para agregar **funcionalidades** específicas. Estos módulos permiten ampliar las capacidades básicas del servidor Apache y ofrecen características adicionales según las necesidades del usuario.
- 3.2** ¡Hay 5 tipos de módulos, los cuales son: **Autenticación** y **autorización, manipulación** de contenido, **procesamiento** de lenguajes de programación, **seguridad** y módulos de **estadísticas**

## 1. Módulos de autenticación y autorización:

- **auth\_basic:** Proporciona la autenticación básica de HTTP, donde los usuarios deben ingresar un nombre de usuario y una contraseña.
- **auth\_digest:** Similar a mod\_auth\_basic, pero utiliza el método de autenticación digest.
- **authz\_host:** Ofrece control de acceso basado en direcciones IP y nombres de host.

## 2. Módulos de manipulación de contenido:

- **rewrite:** Permite la manipulación de URL y redirecciones.
- **headers:** Permite la manipulación de encabezados HTTP en las solicitudes y respuestas.

## 3. Módulos de procesamiento de lenguajes de programación:

- **php:** Permite la ejecución de scripts PHP.
- **python, perl, ruby:** Permiten la ejecución de scripts en Python, Perl y Ruby, respectivamente.

## 4. Módulos de seguridad:

- **security:** Ofrece capacidades de firewall de aplicación web (WAF) para proteger contra ataques comunes.

## 5. Módulos de log y estadísticas:

- **log\_config:** Permite la configuración avanzada de registros.
- **status:** Proporciona información en tiempo real sobre el estado del servidor.

- 3.3** El módulo **userDir** permite a los usuarios publicar sus sitios web personales en **subdirectorios** dentro de sus directorios de inicio. Por ejemplo, si tenemos un directorio llamado **public\_html** en nuestro directorio de inicio, los archivos colocados allí serán accesibles a través de **http://apache.gonzalo/~gonzalo**. Para instalar el modulo **userDir** usaremos estos comandos.

```
$sudo a2enmod userdir  
$sudo systemctl restart apache2
```



**hola gonzalo y jaime**

# EJERCICIO 4

## Autenticación de usuario en Apache, sobre autenticación básica y con el módulo 'auth\_digest'.

- 4.1** La autenticación de usuarios, permite a los administradores de sitios web controlar el acceso a ciertas áreas de su sitio.

La autenticación de usuario en Apache se puede dividir en dos tipos principales: autenticación básica y autenticación digest.

### 4.2 Autenticación Básica.

Este es el método más simple de autenticación. Cuando un usuario intenta acceder a un recurso protegido, el servidor solicita un nombre de usuario y una contraseña. Estos detalles se envían al servidor en texto plano, lo que significa que pueden ser interceptados fácilmente. Por lo tanto, la autenticación básica se utiliza mejor en combinación con una conexión segura, como HTTPS.

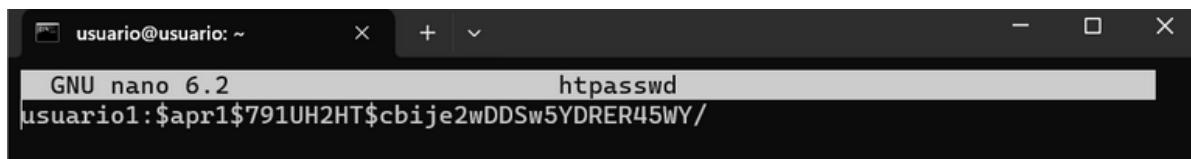
Para llevar a cabo esta autenticación, estos son los pasos a seguir:

- Crea un archivo .htpasswd: Este archivo almacenará los nombres de usuario y las contraseñas. Puedes crearlo en cualquier lugar, pero debe ser accesible por el servidor Apache. Para crear un usuario, puedes usar el comando htpasswd. Por ejemplo:

```
htpasswd -c /ruta/a/tu/.htpasswd usuario1
```

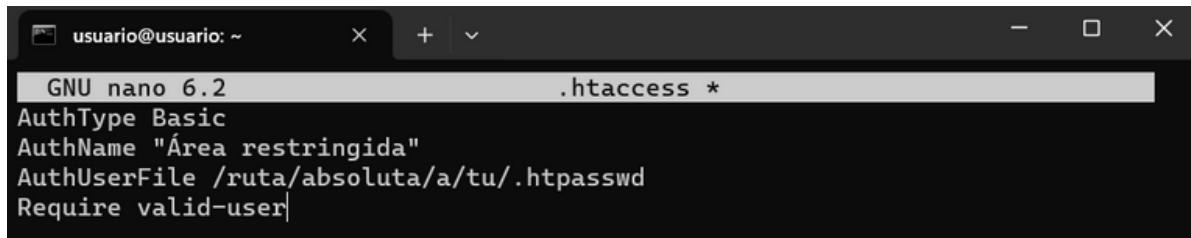
Este comando te pedirá que ingreses y confirmes la contraseña para usuario1. El argumento -c se usa para crear el archivo si no existe. No debes usarlo si ya tienes un archivo .htpasswd y solo quieres agregar un nuevo usuario.

```
usuario@usuario:~$ htpasswd -c ./htpasswd usuario1
New password:
Re-type new password:
Adding password for user usuario1
usuario@usuario:~$ ls
htpasswd  public_html
```



A screenshot of a terminal window titled "usuario@usuario: ~". The window shows the command "htpasswd -c ./htpasswd usuario1" being run. It prompts for a new password and asks to re-type it. After confirming, it adds a password for the user "usuario1". Finally, it lists the contents of the current directory, which includes "htpasswd" and "public\_html".

- Crea o edita el archivo .htaccess: Este archivo le dice al servidor Apache que requiere autenticación para acceder a un directorio. Aquí hay un ejemplo de cómo se vería:

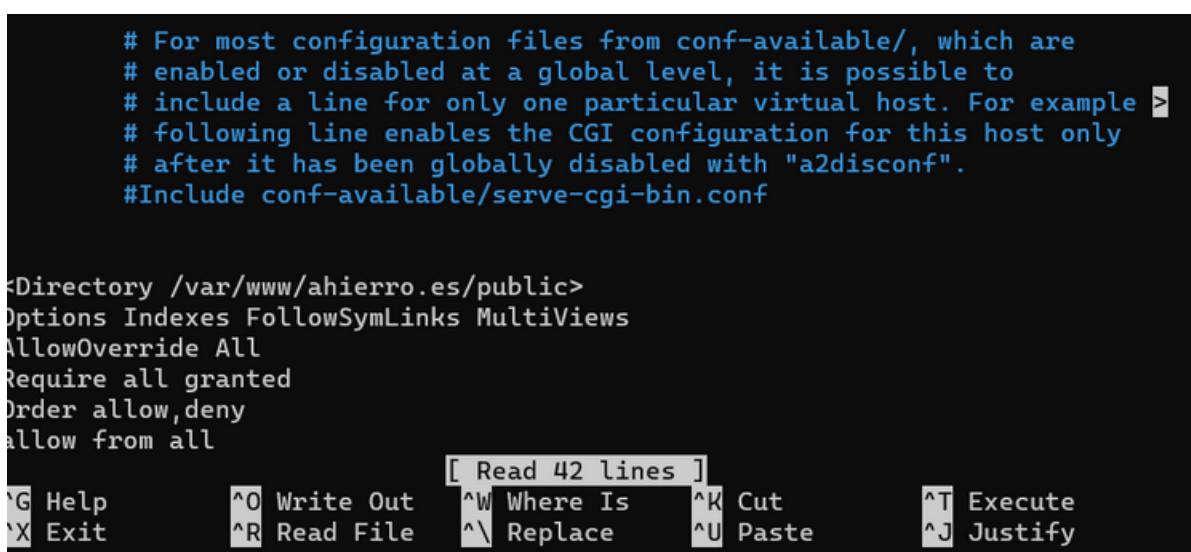


```
GNU nano 6.2           .htaccess *
AuthType Basic
AuthName "Área restringida"
AuthUserFile /ruta/absoluta/a/tu/.htpasswd
Require valid-user|
```

- Por ultimo, configura los permisos de Apache: Necesitas asegurarte de que Apache tiene permisos para leer el archivo .htaccess y el archivo .htpasswd.

```
usuario@usuario:~$ sudo chmod 644 .htaccess
usuario@usuario:~$ sudo chmod 644 htpasswd
```

- También necesitas habilitar el uso de archivos .htaccess editando el archivo de configuración de Apache y cambiando la directiva AllowOverride a All para el directorio que deseas proteger.



```
# 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 >
# 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

<Directory /var/www/ahierro.es/public>
Options Indexes FollowSymLinks MultiViews
AllowOverride All
Require all granted
Order allow,deny
allow from all
[G] Help      [^O] Write Out   [^W] Where Is    [^K] Cut        [^T] Execute
[X] Exit      [^R] Read File   [^\\] Replace     [^U] Paste      [^J] Justify
```



### 4.3 Autenticación Digest.

Este es un método más seguro de autenticación. Al igual que la autenticación básica, el servidor solicita un nombre de usuario y una contraseña. Sin embargo, en lugar de enviar estos detalles en texto plano, se envían como un hash MD5. Esto significa que incluso si los detalles son interceptados, no pueden ser utilizados para acceder al recurso protegido.

Para llevar a cabo esta autenticación, estos son los pasos a seguir:

- Habilita el módulo mod\_auth\_digest: Puedes hacer esto usando el comando a2enmod y después reiniciar Apache para que los cambios surtan efecto.

```
tarío@usuario:~$ sudo a2enmod auth_digest
considering dependency authn_core for auth_digest:
module authn_core already enabled
Enabling module auth_digest.
To activate the new configuration, you need to run:
systemctl restart apache2
tarío@usuario:~$ systemctl restart apache2
-- AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ===
Authentication is required to restart 'apache2.service'.
Authenticating as: usuario
Password:
-- AUTHENTICATION COMPLETE ===
tarío@usuario:~$ |
```

- Crea un archivo .htdigest: Este archivo es similar al archivo .htpasswd utilizado para la autenticación básica. Puedes crearlo usando el comando htdigest. Este comando te pedirá que ingreses y confirmes la contraseña para usuario1. El argumento -c se usa para crear el archivo si no existe.

```
usuario@usuario:~/authDigest$ htdigest -c ./htdigest "Área restringida" usu
ario1
Adding password for usuario1 in realm Área restringida.
New password:
Re-type new password:
usuario@usuario:~/authDigest$ |
```

```
GNU nano 6.2          htdigest
usuario1:Área restringida:ea8914f68daf6a896183b95553750e02
```



- Crea o edita el archivo .htaccess: Este archivo es similar al utilizado para la autenticación básica, pero con algunas diferencias. Aquí hay un ejemplo de cómo se vería:

```
GNU nano 6.2          .htaccess *
AuthType Digest
AuthName "Área restringida"
AuthDigestDomain /ruta/a/tu/directorio/
AuthUserFile /ruta/a/tu/.htdigest
Require valid-user|
```

- Editar el archivo de configuración de Apache: El archivo de configuración de Apache generalmente se encuentra en uno de los siguientes lugares, dependiendo de tu sistema operativo y configuración:

- /etc/apache2/apache2.conf (Ubuntu)
- /etc/httpd/httpd.conf (CentOS)
- /usr/local/etc/httpd/httpd.conf (macOS con Homebrew)
- C:/Program Files (x86)/Apache Group/Apache2/conf/httpd.conf (Windows)

Para editar el archivo, puedes usar un editor de texto como nano, vi o emacs. Por ejemplo:

```
sudo nano /etc/apache2/apache2.conf
```

```
<Directory /var/www/ahierro.es/public>
Options Indexes FollowSymLinks MultiViews
AllowOverride All
Require all granted
Order allow,deny
allow from all
```

[ Read 42 lines ]

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute  
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify



- Cambiar la directiva AllowOverride: Dentro del archivo de configuración, busca el bloque <Directory> que corresponde al directorio que deseas proteger. Dentro de este bloque, cambia la directiva AllowOverride a All.

```
<Directory /var/www/ahierro.es/public>
Options Indexes FollowSymLinks MultiViews
AllowOverride All
Require all granted
Order allow,deny
allow from all
[ Read 42 lines ]
^G Help      ^O Write Out   ^W Where Is    ^K Cut
^X Exit      ^R Read File   ^\ Replace     ^U Paste
                                                ^T Execute
                                                ^J Justify
```

Esto permitirá que Apache lea el archivo .htaccess en ese directorio.

- Guardar y cerrar el archivo: Una vez que hayas hecho los cambios, guarda y cierra el archivo. Si estás usando nano, puedes hacer esto presionando Ctrl+X, luego Y para confirmar, y finalmente Enter para guardar y cerrar.
- Cambiar los permisos de los archivos .htaccess y .htpasswd o .htdigest: Necesitas asegurarte de que Apache tiene permisos para leer estos archivos. Puedes cambiar los permisos usando el comando chmod. Por ejemplo:

```
usuario@usuario:~/authDigest$ sudo chmod 644 .htaccess
usuario@usuario:~/authDigest$ sudo chmod 644 .htdigest
chmod: cannot access '.htdigest': No such file or directory
usuario@usuario:~/authDigest$ sudo chmod 644 htdigest
usuario@usuario:~/authDigest$ |
```

```
usuario@usuario:~$ sudo chmod 644 htpasswd
```

- Reiniciar Apache: Finalmente, necesitas reiniciar Apache para que los cambios surtan efecto. Puedes hacer esto con uno de los siguientes comandos, dependiendo de tu sistema operativo:
  - sudo systemctl restart apache2 (Ubuntu)
  - sudo systemctl restart httpd (CentOS)
  - sudo apachectl restart (macOS con Homebrew)
  - httpd.exe -k restart (Windows, desde la línea de comandos con privilegios de administrador)

# EJERCICIO 5

## Prepara Apache para poder ver el portal web de forma segura con el protocolo HTTPS.

Para configurar Apache para servir contenido a través de HTTPS, necesitarás un certificado SSL. Aquí te dejo los pasos para hacerlo:

1. Instala el módulo mod\_ssl: Este módulo proporciona soporte para SSL en Apache. Puedes instalarlo con el siguiente comando:

```
usuario@usuario:~$ sudo a2enmod ssl
[sudo] password for usuario:
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and crea
te self-signed certificates.
To activate the new configuration, you need to run:
  systemctl restart apache2
usuario@usuario:~$ |
```

2. Crea un directorio para tus certificados SSL: Este directorio almacenará tus certificados SSL. Puedes crearlo con el siguiente comando:

```
usuario@usuario:~$ sudo mkdir /etc/apache2/ssl
```

3. Obtén un certificado SSL: Puedes obtener un certificado SSL de una Autoridad de Certificación (CA) como Let's Encrypt, o puedes crear un certificado autofirmado para pruebas.

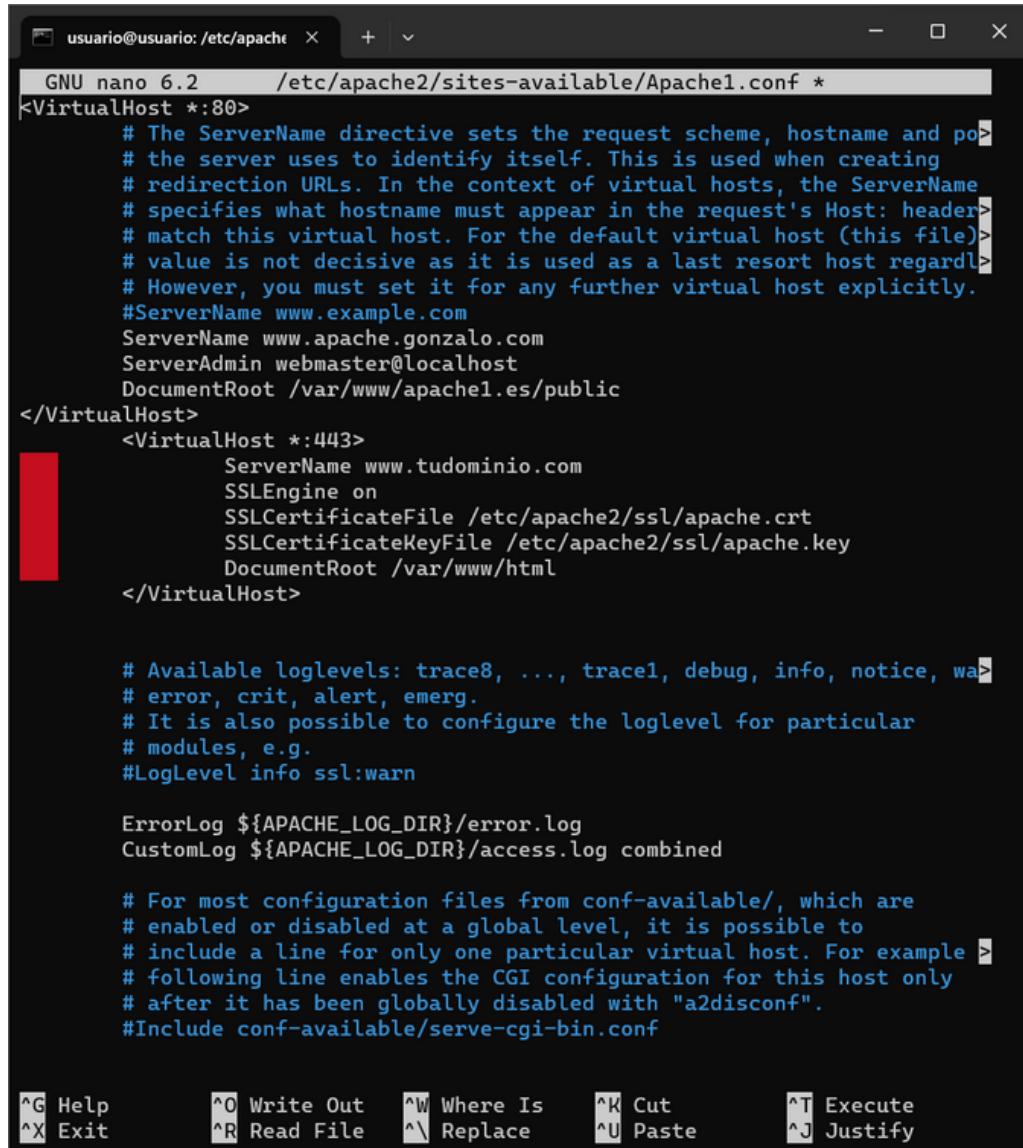
Para obtener un certificado de Let's Encrypt, puedes usar Certbot. Para crear un certificado autofirmado, puedes usar el siguiente comando:

```
usuario@usuario:~$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048
-keyout /etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt
```

Este comando te pedirá que ingreses información sobre tu sitio web. El certificado será válido por 365 días.

4. Configura Apache para usar SSL: Necesitas editar el archivo de configuración de tu sitio en Apache para que use SSL.

Este archivo generalmente se encuentra en /etc/apache2/sites-available. Aquí hay un ejemplo de cómo se vería:



The screenshot shows a terminal window titled "usuario@usuario: /etc/apache2/sites-available/Apache1.conf \*". The window contains the Apache configuration file "Apache1.conf". The configuration includes two virtual hosts: one for port 80 and one for port 443. The port 443 host is configured to use SSL with certificates located at /etc/apache2/ssl/apache.crt and /etc/apache2/ssl/apache.key. The configuration also specifies log levels and log files. At the bottom of the window, there is a menu bar with various keyboard shortcuts for file operations like Help, Exit, Read File, Replace, Cut, Paste, Where Is, and Justify.

```
GNU nano 6.2      /etc/apache2/sites-available/Apache1.conf *
<VirtualHost *:80>
    # The ServerName directive sets the request scheme, hostname and port
    # 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)
    # 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
    ServerName www.apache.gonzalo.com
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/apache1.es/public
</VirtualHost>
<VirtualHost *:443>
    ServerName www.tudominio.com
    SSLEngine on
    SSLCertificateFile /etc/apache2/ssl/apache.crt
    SSLCertificateKeyFile /etc/apache2/ssl/apache.key
    DocumentRoot /var/www/html
</VirtualHost>

# Available loglevels: trace8, ..., trace1, debug, info, notice, warning,
# 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 >
# 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
```

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute  
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify



```
GNU nano 6.2      /etc/apache2/sites-available/Apache1.conf *
  DocumentRoot /var/www/apache1.es/public
</VirtualHost>
<VirtualHost *:443>
  ServerName www.tudominio.com
  SSLEngine on
  SSLCertificateFile /etc/apache2/ssl/apache.crt
  SSLCertificateKeyFile /etc/apache2/ssl/apache.key
  DocumentRoot /var/www/html
</VirtualHost>

# 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:
# 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

<Directory /var/www/ahierro.es/public>
Options Indexes FollowSymLinks MultiViews
AllowOverride All
Require all granted
Order allow,deny
allow from all
</Directory>

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

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute  
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify

Este archivo redirige todas las solicitudes HTTP a HTTPS y configura Apache para usar tu certificado SSL.

5. Habilita el sitio y reinicia Apache: Finalmente, necesitas habilitar el sitio y reiniciar Apache para que los cambios surtan efecto. Puedes hacer esto con los siguientes comandos:

```
usuario@usuario:/etc/apache2/sites-available$ sudo a2ensite Apache1.conf
Site Apache1 already enabled
usuario@usuario:/etc/apache2/sites-available$ cd
usuario@usuario:~$ sudo systemctl restart apache2
```

# EJERCICIO 6

**Investiga los conceptos anteriores para el servidor NGINX. Cuál de los dos servidores elegirías, razonalo**

## 6.1.1 NGINX:

### Características principales:

- Diseñado para ser **ligero y eficiente**.
- Altamente **escalable** y capaz de manejar un gran número de conexiones concurrentes.
- Mejor **rendimiento** para servir contenido estático y actuar como proxy inverso.

### Uso común:

- Buen **rendimiento** para servir archivos estáticos y actuar como servidor proxy.
- Adecuado para implementaciones donde la eficiencia y la velocidad son fundamentales.

## APACHE:

### Características principales:

- Mayor **modularidad**.
- Enfoque históricamente centrado en la compatibilidad y la flexibilidad.
- Más configuración y ajustes posibles en comparación con NGINX.

### Uso común:

- Ampliamente utilizado en una variedad de entornos debido a su **flexibilidad** y soporte para una amplia gama de módulos.
- Ideal cuando se requiere una **configuración detallada y personalizada**.

### Elección del servidor:

Si el proyecto es principalmente **estático** o requiere un servidor que sea eficiente en la gestión de conexiones concurrentes, **NGINX** podría ser la elección preferida.

Si se necesita un servidor más **flexible** con una amplia gama de módulos y configuraciones, y el proyecto implica una mezcla de contenido estático y dinámico, **Apache** podría ser más adecuado.



**6.1.2** Para instalar **NGINX** localmente y configurar un proyecto con Angular, puedes seguir estos pasos generales:

**Instalación** de NGINX:

```
$sudo apt-get install nginx
```

**Configuración** de NGINX:

El archivo de configuración principal suele estar en **/etc/nginx/nginx.conf**.

Configuración del Proyecto **Angular**:

Asegúrate de que la **configuración** de producción de Angular está lista (**ng build -prod**).

Copia los archivos generados en la carpeta de compilación a la ubicación adecuada definida en la configuración de NGINX.

**Reinicia** NGINX:

```
$sudo systemctl restart nginx
```

Después de realizar cambios en la configuración, **reinicia** NGINX para que los cambios surtan efecto:

```
$sudo systemctl restart nginx
```

**6.1.3** En **NGINX, los enlaces simbólicos** pueden ser utilizados, pero debes tener en cuenta que NGINX sigue el enlace simbólico hasta llegar al archivo real.

Digamos que tienes tu proyecto Angular en **/etc/www/proyecto/dist/**.

Configuración de NGINX:

Puedes crear un **enlace simbólico** en el directorio html de NGINX que apunte a la carpeta **dist** de tu proyecto Angular.

```
server {
    listen 8080;
    server_name tu-dominio.com;

    location / {
        root /usr/share/nginx/html;
        index index.html index.htm;
    }

    location /proyecto-angular {
        alias /usr/share/nginx/html/dist;
        try_files $uri $uri/ /proyecto/index.html;
    }
}
```



**6.1.4** NGINX no utiliza archivos **.htaccess**, pero puedes lograr funcionalidades similares utilizando **directivas** en el archivo de configuración principal. Por ejemplo, si quieras habilitar la reescritura de URL, puedes usar la directiva **location** para manejar esto:

```
server {  
    listen 80;  
    server_name tu-dominio.com;  
  
    location / {  
        root /usr/share/nginx/html;  
        index index.html index.htm;  
  
        # Redirigir todas las solicitudes a index.html (para Angular SPA)  
        try_files $uri $uri/ /index.html;  
    }  
  
    location /otro-directorio {  
        alias /ruta/a/otro-directorio;  
        index index.html index.htm;  
  
    }  
}
```

**6.2** NGINX no tiene **modulos** como tal pero puedes lograr algo similar utilizando la directiva **location** y la variable **\$user** para representar el nombre de usuario.

Editar el **archivo de configuración** de NGINX:

```
server {  
    listen 80;  
    server_name tu-dominio.com;  
  
    location ~ ^/~((^/)+)(.*)$ {  
        alias /home/$1/public_html/$2;  
        index index.html index.htm;  
        try_files $uri $uri/ /~$1/index.html;  
    }  
}
```

-La expresión regular **^/~((^/)+)(.\*)\$** captura el nombre de usuario y el resto de la ruta después de **/~nombre-de-usuario/**.

-La directiva **alias** apunta al directorio personal del usuario, por ejemplo, **/home/nombre-de-usuario/public\_html/**.

-La directiva **try\_files** intenta servir el archivo solicitado.

### 6.3 La autenticación

básica en **NGINX** se logra utilizando el módulo **ngx\_http\_auth\_basic**.

Este módulo permite **proteger áreas específicas** de tu sitio web con un nombre de usuario y una contraseña, y es útil cuando necesitas agregar una capa de seguridad simple a ciertas partes de tu aplicación.

Primero, necesitas **crear** un archivo que almacene los nombres de usuario y las contraseñas.

Puedes utilizar la herramienta **htpasswd** para esto. Si no está instalada, puedes instalarla.

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

Luego, crea el archivo de contraseñas.

```
$htpasswd -c /etc/nginx/.htpasswd usuario
```

#### Configurar NGINX:

A continuación, configura NGINX para utilizar la **autenticación básica**. Modifica la configuración del servidor NGINX para incluir las siguientes líneas:

```
server {  
    listen 80;  
    server_name tu-dominio.com;  
  
    location /area-restringida {  
        auth_basic "Área restringida";  
        auth_basic_user_file /etc/nginx/.htpasswd;  
    }  
}
```

**auth\_basic:** Define el mensaje que se mostrará en la ventana emergente de autenticación.

**auth\_basic\_user\_file:** Especifica la ubicación del archivo de contraseñas creado con htpasswd

**6.4** El protocolo **HTTPS** (Hypertext Transfer Protocol Secure) en Nginx funciona mediante el uso de **certificados SSL/TLS** para cifrar la comunicación entre el cliente y el servidor. Aquí te dejo un resumen de los pasos básicos para configurar HTTPS en Nginx:

**6.4.1. Obtener un certificado SSL/TLS:** Puedes obtener un certificado de una Autoridad de Certificación (CA) como Let's Encrypt, Comodo, etc. Algunas de estas autoridades ofrecen certificados gratuitos.

**6.4.2. Instalar el certificado SSL/TLS en tu servidor Nginx:** Una vez que hayas obtenido tu certificado, debes instalarlo en tu servidor. Esto generalmente implica copiar los archivos del certificado a tu servidor y luego referenciarlos en tu configuración de Nginx.

**6.4.3. Configurar Nginx para usar HTTPS:** Esto se hace editando el archivo de configuración de Nginx para tu sitio web. Aquí hay un ejemplo de cómo podría verse esto:

```
server {  
    listen 80;  
    server_name misitio.com www.misitio.com;  
    return 301 https://$host$request_uri;  
}  
  
server {  
    listen 443 ssl;  
    server_name misitio.com www.misitio.com;  
    ssl_certificate /etc/nginx/ssl/misitio.com.crt;  
    ssl_certificate_key /etc/nginx/ssl/misitio.com.key;  
    location / {  
        proxy_pass http://localhost:8000;  
    }  
}
```

En este ejemplo, cualquier tráfico que llegue al puerto 80 (HTTP) se redirige al puerto 443 (HTTPS). Luego, para el puerto 443, especificamos la ubicación de nuestros archivos de certificado SSL con **ssl\_certificate** y **ssl\_certificate\_key**.

**6.4.4. Reiniciar Nginx:** Finalmente, debes reiniciar Nginx para que los cambios surtan efecto. Esto se puede hacer con el comando **\$sudo systemctl restart nginx**



**Muchas Gracias**