

Trabajo Final Guillermo Piña Herrera



Indice

1.Apache.....	3
1.1 <i>Virtual hosting</i>	3
1.2Mapeo de URL.....	7
1.3Control de acceso.....	11
2.VSFTP.....	14
3.GIT.....	18
3.1Local.....	18
3.2Remoto.....	19

1.Apache

Apache es un servicio que nos provee de un servidor web HTTP el cual tiene como objetivo el desarrollar y mantener servidores HTTP de código libre en sistemas operativos modernos. Este fue lanzado en el año 1995 y en febrero de este año ha cumplido 25 años operativos.

Al ser apache de código libre y bastante liviano hace que este se convierta en nuestra opción principal para desarrollar los contenidos sobre virtual hosting y su control.

Antes de ponernos con la parte práctica, es buena idea definir antes en que consiste el virtual hosting.

1.1Virtual hosting

El virtual hosting consiste en el hospedaje (hosting) de varios nombres de dominio en un mismo servidor. Esto permite a un servidor compartir sus recursos como la memoria sin requerir que todos los servicios ofrecidos usen el mismo nombre de host. Este es usado por compañías que se dedican a alojar páginas webs de terceros pues gracias a este método se reducen mucho los costos de mantenimiento.

Existen dos tipos principales de virtual hosting, *basado en nombre* y *basado en IP*

- Basado en nombre: Este usa el nombre del host que presenta el cliente, esto guarda su dirección IP.
- Basado en IP: Este usa diferentes direcciones IP por cada nombre de host, esto permite usarlo sin ningún protocolo pero requiere por lo tanto una IP dedicada por cada nombre de dominio.

Y ahora, tras explicar todo vamos a realizar una práctica sobre cómo hacer un virtual hosting con apache y su posterior administración que incluye control de acceso, autenticación y otras restricciones.

Para empezar, deberenos dirigirnos a la carpeta *etc/apache2/sites-available* y realizaremos una copia del archivo *000-default.conf* con el comando *cp "000-default.conf" "nombre del archivo.conf"*.

En mi caso mis archivos se llamaran *pagina1.conf* y *pagina2.conf*

IMPORTANTE: asegurarse de que tenga la extension *".conf"* , si no no funcionara correctamente.

El resultado en la consola deberia ser el siguiente:

```
usuario@hobbit:/etc/apache2/sites-available$ ls
000-default.conf  default-ssl.conf  pagina1.conf  pagina2.conf
usuario@hobbit:/etc/apache2/sites-available$
```

Tras realizar las copias vamos a editarlas con el comando *"sudo nano pagina1.conf"*.

La estructura principal de un archivo de configuracion seria el siguiente:

```
<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 apache1.openwebinars.net

    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/pagina1

    # 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_pagina1.log
    CustomLog ${APACHE_LOG_DIR}/access_pagina1.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>

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

Para clarificar la captura voy a explicar que significa cada parte:

- VirtualHost *:80 => En la apertura podemos observar que esta el puerto por el que accede apache (puerto 80) y un asterisco que hace que al no tener una ip asignada la consiga automaticamente. Esta ip puede modificarse, aunque para este ejercicio al sernos irrelevante lo dejaremos con el asterisco.
- ServerName => Esta es la “url” con la que accederemos a nuestra pagina. Este link es simbolico y al ponerlo en el navegador es como si estuviésemos accediendo a la ip directamente.
- ServerAdmin => el nombre de quien es el creador del archivo
- DocumentRoot => Indica el directorio de donde va a coger los recursos
- Error y Custom Log => Los logs de errores y demas de la pagina. Es recomendable cambiarles el nombre para que no se junten con otros del mismo nombre y asi poder depurar mas facilmente la pagina en caso de errores

Tras terminar con la configuracion de los dos archivos, iremos a el directorio “/var/www” donde copiaremos la carpeta *html* con los directorios que pusimos en el documentRoot de los archivos de configuracion.

Deberia quedar de la siguiente manera:

```
usuario@hobbit:/var/www$ ls
html  pagina1  pagina2
usuario@hobbit:/var/www$
```

Dentro de cada carpeta, modificaremos sus html para ver la diferencia. Cada html tendra un *h1* saludando e indicando si es la pagina 1 o la 2.

Ahora nos dirigiremos a el directorio “/etc/apache2/sites-enabled” y ahi ejecutaremos el comando “a2ensite pagina1” y “a2ensite pagina2” con lo que activaremos las paginas. Tras esto nos saltara un mensaje de que necesitamos reiniciar el servidor apache, lo hacemos mediante el comando “systemctl reload apache2”. Deberia quedar como en la captura:

```
usuario@hobbit:/etc/apache2/sites-enabled$ ls
'dfoisda fushdf'  pagina1.conf  pagina2.conf
```

Para acabar iremos a la carpeta etc y añadiremos al archivo “*hosts*” nuestra ip seguida del enlace simbolico de la pagina de la siguiente manera “*192.168.xx.xx enlace*”. Tras terminar podremos acceder a nuestra pagina web.

IMPORTANTE: usaremos el navegador opera pues necesitamos usar protocolo http



Hola, estas en la pagina 1

1.2 Mapeo de URL

En este tutorial veremos maneras de modificar los ficheros con varias directivas. Vamos a ir paso por paso, viendo primero la directiva *Options* la cual cuenta con los siguientes modificadores:

- ALL: todas las opciones excepto MultiViews
- FollowSymLinks: permite seguir enlaces simbólicos
- Indexes: Cuando no encuentra el fichero indicado, muestra la lista de archivos
- MultiViews: Permite la negociación de contenidos
- SymLinkIfOwnerMatch: permite seguir enlaces simbólicos solo cuando el fichero del destino es del mismo usuario que el enlace
- ExecCGI: Permite ejecutar scripts cgi

Para mostraros cómo funcionan, vamos a modificar el archivo *apache2.conf* que se encuentra en */etc/apache2* y al bajar podemos ver la configuración default en cuando abrimos el directorio */var/www*

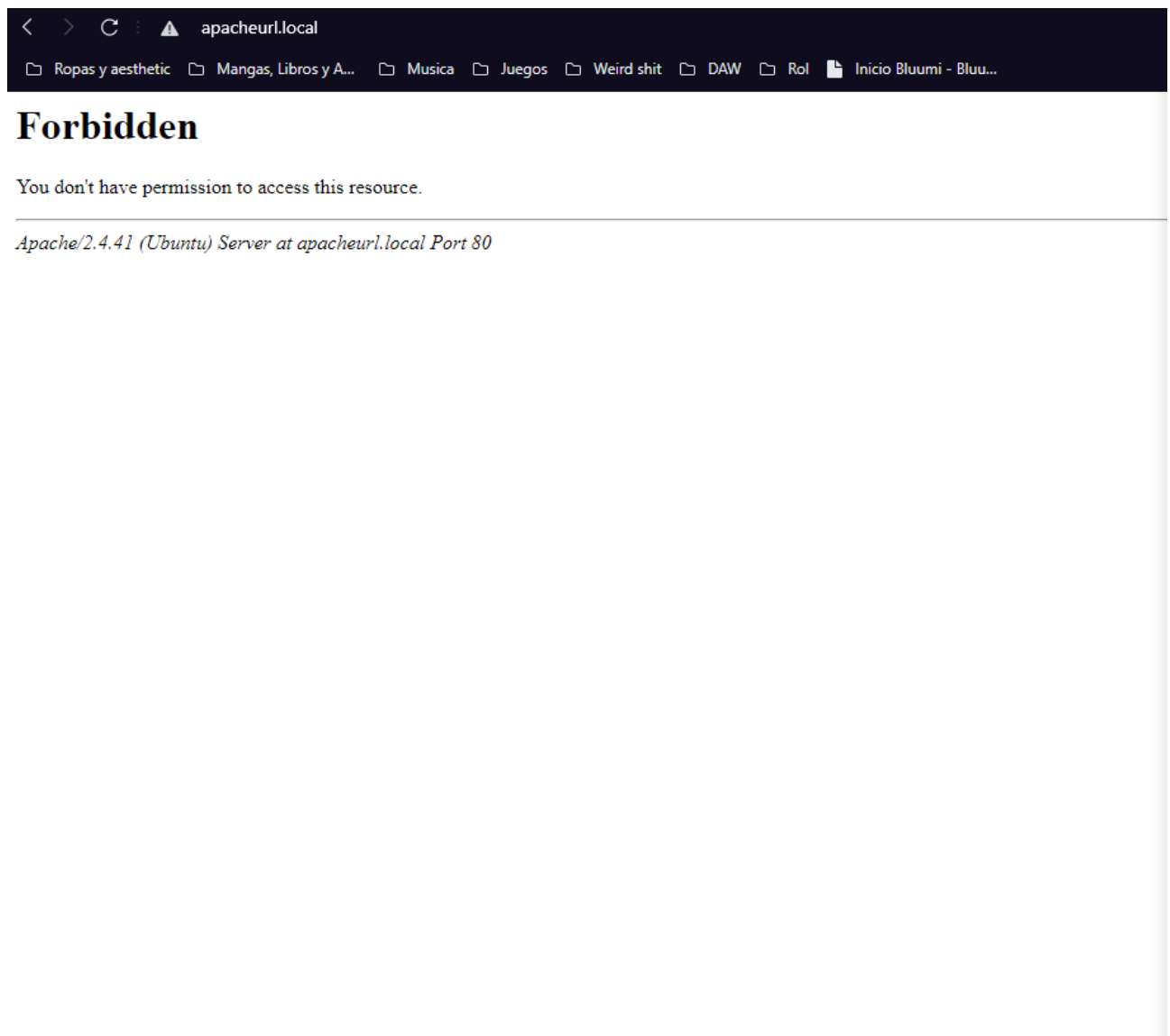
```
<Directory /var/www/>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
```

Pero esto es en la configuración default, si queremos aplicárselo a nuestras páginas, deberemos añadir estos modificadores dentro de su archivo de configuración. Tendremos que abrir una etiqueta *Directory* y dentro de la misma poner los modificadores. Para mostrar un ejemplo, quitare la opción de Indexes.

IMPORTANTE: Para añadir una opción usar + y para quitarla usar -. Si no pones ninguna, quitara el resto menos la escrita

```
<Directory /var/www/paginaURL>
    Options -Indexes
</Directory>
```

Si quitamos la opción Indexes, si no encuentra el *index.html*, en vez de mostrarnos la lista de archivos como dice la configuración global, nos denegará el permiso y nos dará la siguiente pantalla:



Pero en cambio, si intentamos entrar a una pagina mediante un enlace simbolico, nos la mostrara sin problemas. Aqui el ejemplo accediendo a un enlace simbolico de hosts (el cual se encuentra en *etc/hosts*. */etc/hosts.conf*)

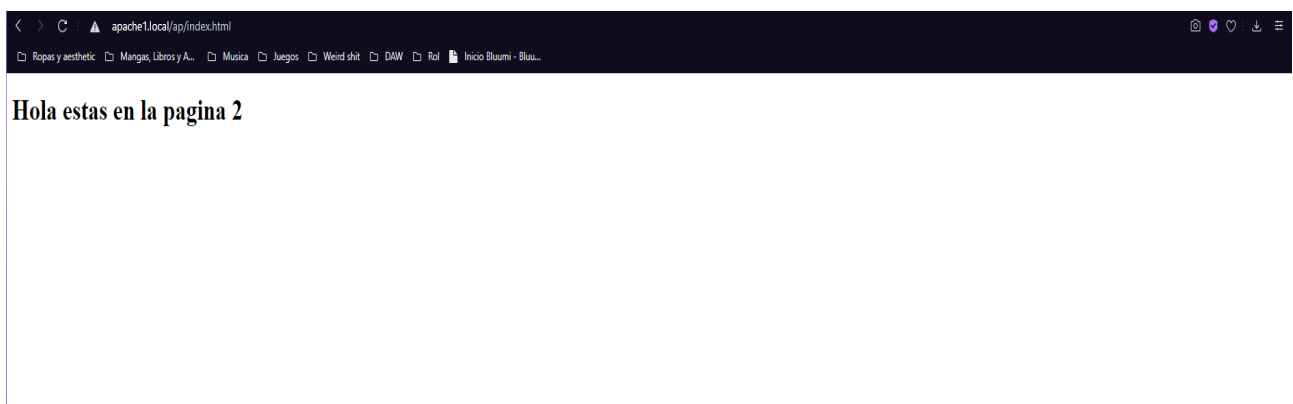

```
< > ↻ ⚠ apacheurl.local/hosts
📁 Ropas y aesthetic 📁 Mangas, Libros y A... 📁 Musica 📁 Juegos 📁 Weird shit 📁 DAW 📁 Rol 📄 Inicio

127.0.0.1 localhost
127.0.1.1 hobbit

# 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
192.168.18.91 apache1.openwebinars.net
192.168.18.91 apache2.openwebinars.net
```

Alias

Gracias al mapeo de URL, tambien disponemos de los alias, los cuales nos permitiran acceder a otros directorios con solo añadir a la direccion inicial una barra y la palabra elegida. Ejemplo: “*apache1.local/ap*” nos mostrara los archivos de la carpeta *apache2*



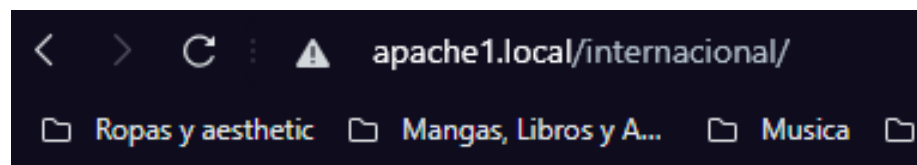
Negociacion de contenido

Gracias a la opcion Multiview, podemos permitir que el navegador cargue la pagina en un idioma u otro, el cual sera indicado con el navegador. Vamos a crear una carpeta dentro del directorio de pagina 1 llamada *Internacional* en la cual tendremos dos html *index.html.en* y *index.html.es*.

Para activar esto necesitaremos decir en la configuracion que active el *MultiView*

```
<Directory /var/www/pagina1/internacional>
    Options +Multiviews
</Directory>
```

Tras activar esto, si ponemos el navegador en Ingles, nos mostrara la pagina en ingles.



Welcome you're on page 1

La otra manera de gestionarlo es mediante un archivo *indexes.var*, en el cual indicaremos con *URI*: para determinar el idioma y el tipo de contenido. Esto nos permite tambien generar logs de errores en otros idiomas. Este es un ejemplo de español e ingles.

```
URI: index.html.en
Content-type: text/html
Content-language: en

URI: index.html.es
Content-type: text/html
Content-language: es
```

Otra faceta de la gestion de contenido son las redirecciones, las cuales nos

permiten que si cambiamos algo de sitio el cliente haga otra peticion para encontrar el recurso. Existen dos tipos, Permanente y Temporales.

- Temporales: Redirect “/traducir” “/internacional”
- Permanentes: Redirect permanent “web” “/docs”

1.3Control de acceso

El control de acceso, hace referencia a todos los medios que proporcionan una forma de controlar el acceso a cualquier recurso. Gracias a la directiva “require”, tendremos varias maneras para permitir o denegar el acceso a los recursos. Por ejemplo, si ponemos “*Required all denied*” haremos que nadie pueda entrar a los recursos. Aquí un ejemplo de código básico de esto;

```
<Directory "/var/www/pagina2">
    Require all denied
</Directory>
```

También se pueden usar expresiones como Allow, Deny y Order, pero estas pertenecen a versiones anteriores de apache y están pendientes a ser eliminadas en futuras versiones.

Autenticación

Apache nos proporciona de varias opciones para poder autenticar a los usuarios. Existen varias formas, así que las iremos explicando una a una:

1. Autenticación básica

Esta viene de serie instalada con apache. Que se encuentra en el fichero “*mod_auth_basic.conf*”.

Un ejemplo de esto sería tal que así:

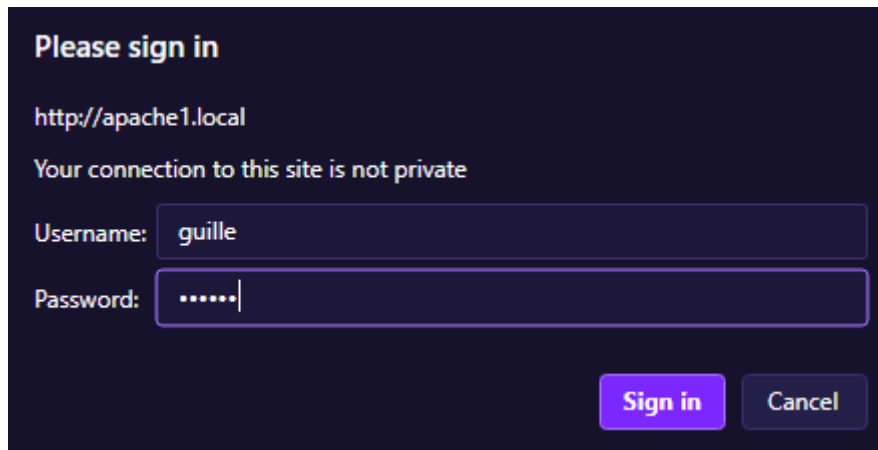
```
<Directory "/var/www/pagina1/privi">
    AuthUserFile "/etc/apache2/passwords/passwords.txt"
    AuthName "acceso"
    AuthType Basic
    Require valid-user
</Directory>
```

- AuthUserFile: se encarga de guardar la información de los usuarios y sus contraseñas
- Authname: Mensaje que saldrá cuando se pida el login
- Authtype: el tipo de autenticación

Vamos a añadir un nuevo usuario con mi nombre, con el comando `htpasswd`. Debería de darnos este resultado:

```
usuario@hobbit:/etc/apache2$ sudo htpasswd -c /etc/apache2/passwords/passwords.txt guille
New password:
Re-type new password:
Adding password for user guille
```

Si intentamos abrir el directorio ahora, nos pedirá un login básico:



Please sign in

http://apache1.local

Your connection to this site is not private

Username: guille

Password:

Sign in Cancel

2. Autenticación con Digest

La autenticación de tipo *digest* nos soluciona el problema de la transferencia de contraseñas sin necesidad de usar SSL. Esto al aplicar una función hash a la contraseña antes de enviarla a la red, hace el proceso mucho más seguro.

Para empezar a usarlo, primero tendremos que activarlo mediante el comando

`a2enmod auth_digest`. La cualidad de digest en cuanto a la básica también es que ahora también identifica un nombre de dominio el cual también debe coincidir. La configuración sería la siguiente:

```
<Directory "/var/www/pagina1/privi">
    AuthType Digest
    AuthName "dominio"
    AuthUserFile "/etc/apache2/passwords/digest.txt"
    Require valid-user
</Directory>
```

Tras poner a punto la configuración, añadimos al fichero digest como añadimos la contraseña anterior, pero añadiéndole antes del nombre de usuario el nombre del dominio.

Tras esto intentamos acceder y tras poner la contraseña y comprobar que el dominio es el mismo accederemos correctamente.

Finalmente, para explorar las opciones mas completa que nos ofrece apache para configurar el acceso, veremos las etiquetas Require all, RequireAny y Require none. Estas opciones nos permiten anidar varias opciones de manera compacta. Este seria un ejemplo complejo del mismo:

```
<RequireAny>
  <RequireAll>
    Require user root Require ip 123.123.123.123
  </RequireAll>
  <RequireAll>
    <RequireAny>
      Require group sysadmins
      Require group useraccounts
      Require user anthony
    </RequireAny>
    <RequireNone>
      Require group restrictedadmin
      Require host bad.host.com
    </RequireNone>
  </RequireAll>
</RequireAny>
```

2.VSFTP

En este tutorial mostrare como configurar un server ftp en linux server. En este caso usaremos vsftpd pues es considerado el mas rapido y seguro. Gracias a el FTP podremos enviar y recibir archivos mediante red entre varios ordenadores en una misma red.

Para empezar, instalaremos vsftpd con el comando *"sudo apt-get install vsftpd"* y esperamos que se instale.

Tras terminar la instalacion, iremos a modificar el archivo de configuracion. Por si acaso y para tener una copia de seguridad por si cometemos algun error, le haremos una copia con el comando cp:

"sudo cp etc/vsftpd.conf etcvsftpd.conf.old". Tras ello, configuraremos el firewall para poder añadir algunas reglas al mismo para que no nos de errores. Para añadir una regla al firewall se usa el comando *"ufw allow"* Añadiremos las siguientes reglas:

```
sudo ufw allow 20/tcp

sudo ufw allow 21/tcp

sudo ufw allow 990/tcp

sudo ufw allow 40000:50000/tcp
```

y miramos si todo se ha añadido correctamente con el comando *"ufw status"*:

To	Action	From
--	-----	----
20/tcp	ALLOW	Anywhere
21/tcp	ALLOW	Anywhere
990/tcp	ALLOW	Anywhere
4000:5000/tcp	ALLOW	Anywhere
80	ALLOW	Anywhere
20/tcp (v6)	ALLOW	Anywhere (v6)
21/tcp (v6)	ALLOW	Anywhere (v6)
990/tcp (v6)	ALLOW	Anywhere (v6)
4000:5000/tcp (v6)	ALLOW	Anywhere (v6)
80 (v6)	ALLOW	Anywhere (v6)

Como podemos ver, todas las reglas se han añadido correctamente, por lo que podemos proseguir con el siguiente paso que ser ala creacion del directorio de usuarios.

Para utilizar el ftp tendremos que elegir el usuario que va a utilizar el acceso FTP. Para mostrarlo vamos a crearlo desde 0:

Primero creamos el usuario con el comando *“adduser usuario_ftp”* y dentro de su carpeta home crearemos una carpeta ftp. Dentro de esa carpeta, con el comando *chown* y *chmod* quitaremos los permisos de escritura y le pondremos de propietario. Finalmente. Crearemos un archivo txt para ver que todo funciona correctamente mas adelante.

Tras crear el usuario vamos a configurar el archivo de configuracion. El archivo es grande y tiene muchas opciones y tendremos tambien que agregar algunas que no estan de base. Como consejo sugerimos usar *control + w* para buscar por palabras y asi hacerlo todo mas ligero.

Primero descomentaremos las siguientes opciones:

```
local_enable=YES  
write_enable=YES  
chroot_local_user=YES
```

Tras descomentar esas opciones, necesitaremos añadir al final del archivo las siguientes:

```
user_sub_token=$USER  
local_root=/home/$USER/ftp  
pasv_min_port = 40000  
pasv_max_port = 50000  
userlist_enable=YES  
userlist_file=/etc/vsftpd.userlist  
userlist_deny=NO
```

Tras ello, añadiremos el usuario creado a el archivo *“vsftpd.userlist”* y realizaremos un *cat* para ver que se ha añadido correctamente

```
usuario@hobbit:/etc/apache2/sites-available$ cat /etc/vsftpd.userlist
usuario_ftp
```

Con toda esto, el servidor ftp ya debería funcionar perfectamente, pero FTP por defecto no tiene ningún tipo de encriptación de datos, lo que lo hace muy vulnerable. Por ello usaremos Tls/Ssl para aumentar su seguridad.

Para empezar con ello necesitaremos usar este comando:

IMPORTANTE: el comando es muy largo, si no tienes portapapeles en tu máquina, asegurate de escribirlo correctamente

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/vsftpd.pem -out  
/etc/ssl/private/vsftpd.pem
```

Tras utilizar el comando, volveremos al archivo de configuración y cambiaremos estas dos líneas a los valores que tienen a continuación:

```
rsa_cert_file=/etc/ssl/private/vsftpd.pem  
rsa_private_key_file=/etc/ssl/private/vsftpd.pem
```

También activaremos el `ssl_enable` poniéndole `YES` de valor

Tras todo añadiremos estas líneas:

```
allow_anon_ssl=NO  
force_local_data_ssl=YES  
force_local_logins_ssl=YES  
ssl_tlsv1=YES  
ssl_sslv2=NO  
ssl_sslv3=NO  
require_ssl_reuse=NO  
ssl_ciphers=HIGH
```


Reiniciaremos el vsftp e iremos al filezilla para comprobar que todo va correctamente

Nuevo sitio - usuario_ftp@192.168.18.91 - FileZilla

Archivo Edición Ver Transferencia Servidor Marcadores Ayuda

Servidor: Nombre de usuario: Contraseña: Puerto: Conexión rápida

Estado: El servidor no permite caracteres no ASCII.
 Estado: Registrado en
 Estado: Recuperando el listado del directorio...
 Estado: Calculando compensación de la zona horaria del servidor...
 Estado: Timezone offset of server is 0 seconds.
 Estado: Directorio "/" listado correctamente

Sitio local: C:\Users\guill\ Sitio remoto: /

Nombre de archivo	Tamaño de...	Tipo de archivo	Última modificación
..			
.android		Carpeta de archivos	26/11/2020 18:14:48
.codeintel		Carpeta de archivos	28/11/2020 15:13:30
.gradle		Carpeta de archivos	20/11/2020 16:27:56
.VirtualBox		Carpeta de archivos	08/12/2020 15:22:57
3D Objects		Carpeta de archivos	16/11/2020 19:59:10
AndroidStudioProjects		Carpeta de archivos	04/12/2020 15:37:26
AppData		Carpeta de archivos	16/11/2020 19:57:46
Configuración local		Carpeta de archivos	08/12/2020 14:25:07
Contacts		Carpeta de archivos	16/11/2020 19:59:10
Cookies		Carpeta de archivos	16/11/2020 19:59:13
Datos de programa		Carpeta de archivos	07/12/2020 18:47:33
Desktop		Carpeta de archivos	06/12/2020 22:48:26
Documents		Carpeta de archivos	07/12/2020 18:16:37
Downloads		Carpeta de archivos	07/12/2020 18:47:41

8 archivos y 30 directorios. Tamaño total: 4.362.276 bytes

Nombre de archivo	Tamaño d...	Tipo de arc...	Última modific...	Permisos	Propietario/...
..					
actividad_autonomo_...	45.381	Microsoft ...	27/11/2020 20:...	-rw-----	1002 1002
ACTIVIDAD_Carta_Mir...	91.776	Document...	27/11/2020 20:...	-rw-----	1002 1002

2 archivos. Tamaño total: 137.157 bytes

Servidor/Archivo local	Direcci...	Archivo remoto	Tamaño	Prioridad	Estado
------------------------	------------	----------------	--------	-----------	--------

3.GIT

Git es una herramienta que nos permite controlar varias versiones de los archivos e incluso varias ramas de trabajo, permitiendonos así trabajar con mas seguridad pues siempre podemos volver a acceder a las versiones anteriores si lo necesitamos.

Para empezar deberemos instalar git en nuestra maquina:

```
sudo apt install git
```

Tras instalarlo, configuraremos el email y usuario de la maquina de la siguiente manera:

```
git config --global user.name "usuario"  
git config --global user.email "email@email.com"
```

Tras configurarlo todo, Crearemos una carpeta que usaremos de repositorio. Tras ello, nos metemos en ella e iniciamos el comando que iniciara git

```
git init
```

Ahora ya tenemos git configurado y listo para usarse. Existen dos maneras de utilizar git. Manera Local y manera Remota:

3.1Local

De este modo, todos los cambios y versiones se quedaran guardadas dentro del mismo sistema y su sesion. Vamos a ver un ejemplo de ello.

Vamos a crear en el repositorio que tenemos un archivo cualquiera y usaremos el siguiente comando para ver en que rama se encuentra y si hay algun archivo pendiente en git

```
git status
```

Tras ello añadiremos el archivo a la siguiente fase para guardar sus cambios

```
git add arch.txt
```

```
No commits yet  
  
Changes to be committed:  
  (use "git rm --cached <file>..." to unstage)  
    new file:   arch.txt
```

Tras esto modificaremos de nuevo el archivo y le haremos su primer commit, lo cual lo guardara como la primera version de nuestro proyecto.

```
git commit -m "Primera version"
```

```
usuario@hobbit:~/git$ sudo git commit -m "primera version"
[master (root-commit) 5b8a3b9] primera version
1 file changed, 1 insertion(+)
create mode 100644 arch.txt
```

Ahora vamos a crear otro archivo nuevo y modificaremos el ya creado. Repetiremos el mismo proceso y haremos otro commit con otro mensaje. Tras ello miraremos el historial para ver los distintos commits

```
git log --oneline --color
```

```
usuario@hobbit:~/git$ sudo git log --oneline --color
9ed3e98 (HEAD -> master) segundo commit con otro archivo
5b8a3b9 primera version
```





3.2 Remoto

Git no solo es util para trabajar en tu mismo sistema, gracias a paginas como **github** o **gitlab** podemos subir nuestros proyectos mediante git para que sean almacenados en la nube. Esto combina todo lo positivo de git junto a las ventajas de la tecnologia de el trabajo en la nube

Lo basico sigue siendo lo mismo, usaremos los mismos comandos para crear y añadir. La diferencia principal es que el commit lo realizaremos al servidor de la pagina de nuestra eleccion, que en mio caso es github. Usaremos los siguientes comandos.

```
git remote add origin https://github.com/guille230/gitDespliegueTuto.git
git branch -M main
git push -u origin main
```

Git remote add nos añade nuestro repositorio de github para poder trabajar con el desde nuestro linux server. El branch y push lo que hacen es elegir la rama que quieres y la sube. Si vamos a github veremos que tenemos el archivo ahi

 guille segundo commit con otro archivo		9ed3e98 27 minutos ago	 2 commits
	arch.txt	segundo commit con otro archivo	27 minutos ago
	archivo2.txt	segundo commit con otro archivo	27 minutos ago

Si queremos descargar lo de github para nuestro equipo local usaremos el siguiente comando y ya tendremos todos los cambios de vuelta en el ordenador.

```
Sudo git pull origin master
```

Comandos basicos

Aqui vamos a hacer una lista de los comandos basicos de git para que los tengas a mano siempre que los necesites

- git init
- git add
- git commit -m "mensaje"
- git pull
- git branch
- git log --oneline --color
- git push
- git rm
- git status
- git fetch
- git merge