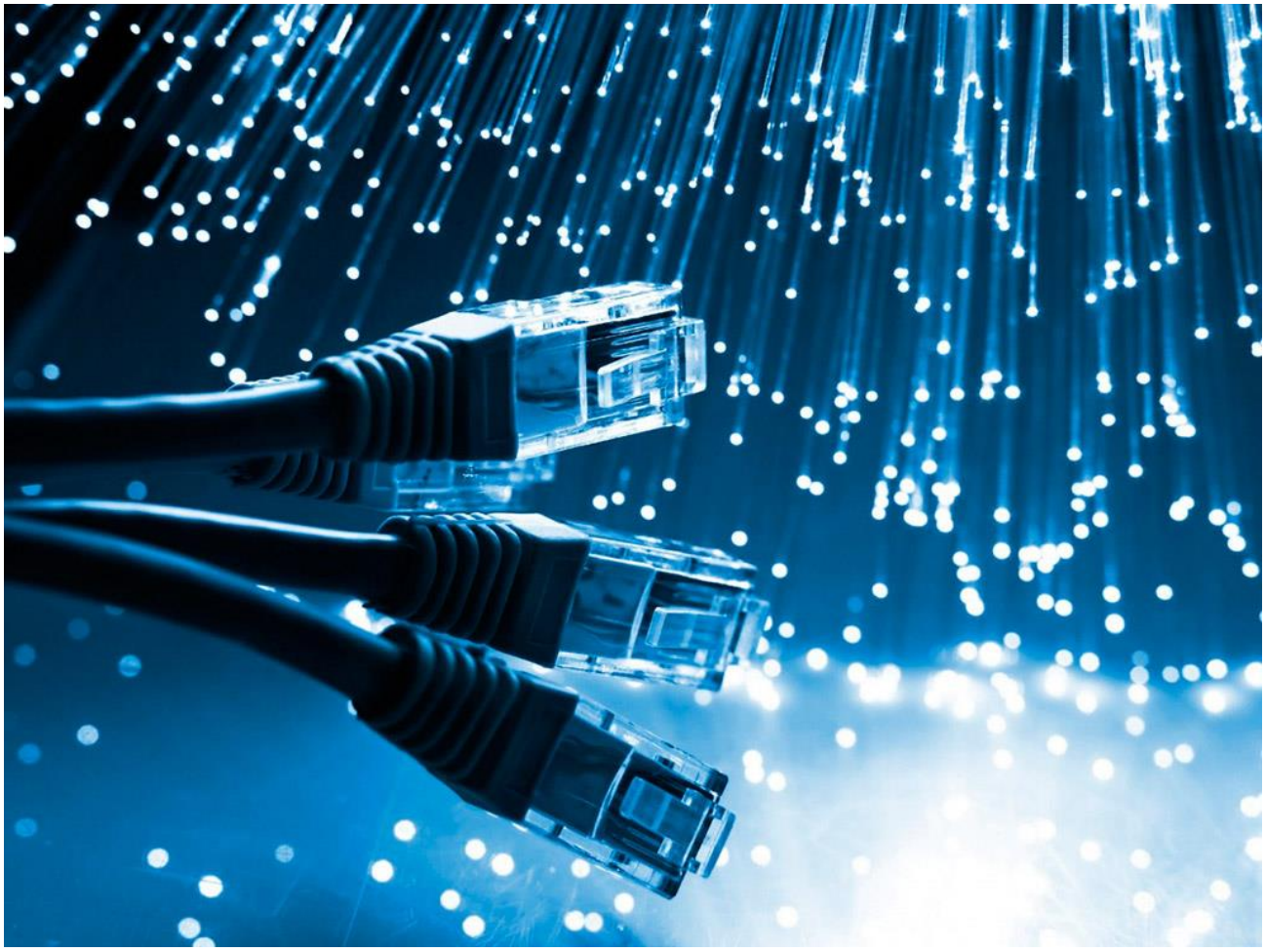
Alejandro López Arjona – Trabajo 1º Evaluación DAW



Índice

Apache virtual hosting

El término Virtual Hosting de Apache se usa para definir el hecho de tener varias páginas webs en una sola máquina que hará de servidor.

Vamos a configurar 2 páginas webs en Apache que funcionen de manera independiente en la misma página.

1) En la carpeta **/etc/apache/sites-available** se encuentra el archivo **000-default.conf** que es la página web por defecto cuando se accede por IP. Vamos a hacer dos copias del archivo, una para cada página web que queremos configurar.

Como vemos en la imagen, usamos esta secuencia de comandos:

cd /etc/apache2/sites-available
cp 000-default.conf pagina1.conf
cp 000-default.conf pagina2.conf

2) Editamos los archivos **pagina1.conf** y **pagina2.conf** con el comando nano y modificamos las propiedades **ServerName** (URL de la página web) y **DocumentRoot** (carpeta donde se almacena la página web).

```
  GNU nano 4.8                                pagina1.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.pagina1.byalejandro.com

        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.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>

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


                              [ Read 31 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line M-E Redo
```

```
  GNU nano 4.8                                pagina2.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.pagina2.byalejandro.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/pagina2

        # 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>

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


                              [ Read 31 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line M-E Redo
```

3) Tenemos que crear un enlace simbólico a los ficheros de configuración que acabamos de crear. Para ello, nos vamos a la carpeta **/etc/apache2/sites-enabled** y con el comando **a2ensite nombre-fichero** los activamos.

```
root@hobbit:/etc/apache2/sites-available# a2ensite pagina1
Enabling site pagina1.
To activate the new configuration, you need to run:
  systemctl reload apache2
root@hobbit:/etc/apache2/sites-available# a2ensite pagina2
Enabling site pagina2.
To activate the new configuration, you need to run:
  systemctl reload apache2
root@hobbit:/etc/apache2/sites-available# _
```

Si queremos desactivar una página web en el futuro, podemos hacerlo con el comando **a2dissite nombre-fichero**

De manera opcional, podemos deshabilitar el fichero **000-default.conf** para que si nos acceden por IP no les cargue la web predeterminada. Para ello podemos usar **a2dissite 000-default.conf**

4) Creamos las carpetas pagina1 y pagina2 y configuramos un fichero index.html en su interior o lo que creamos conveniente para el sitio web.

```
root@hobbit:/var/www# mkdir pagina1
root@hobbit:/var/www# mkdir pagina2
root@hobbit:/var/www# ls
html  pagina1  pagina2
root@hobbit:/var/www#
```

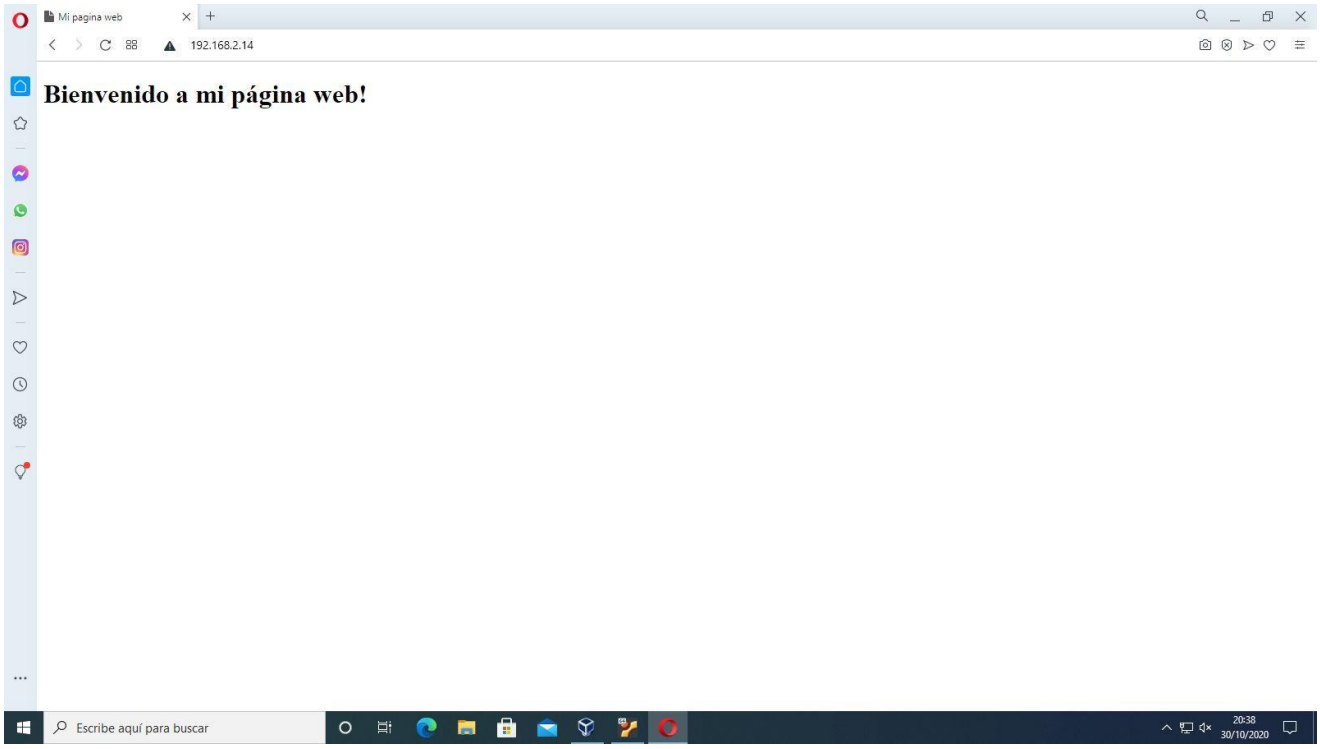
5) Le damos permisos a las carpetas en usuario y el servidor apache para que funcione correctamente.

```
root@hobbit:~# chown -R www-data:www-data /var/www/pagina1
root@hobbit:~# chown -R www-data:www-data /var/www/pagina2
root@hobbit:~# _
```

6) Finalmente, accedemos con nano al archivo **/etc/hosts/** y añadimos las URL de los sitios web que acabamos de crear.

```
  GNU nano 4.8                                    /etc/hosts
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
192.168.100.36 www.pagina1.byalejandro.com
192.168.100.36 www.pagina2.byalejandro.com
```

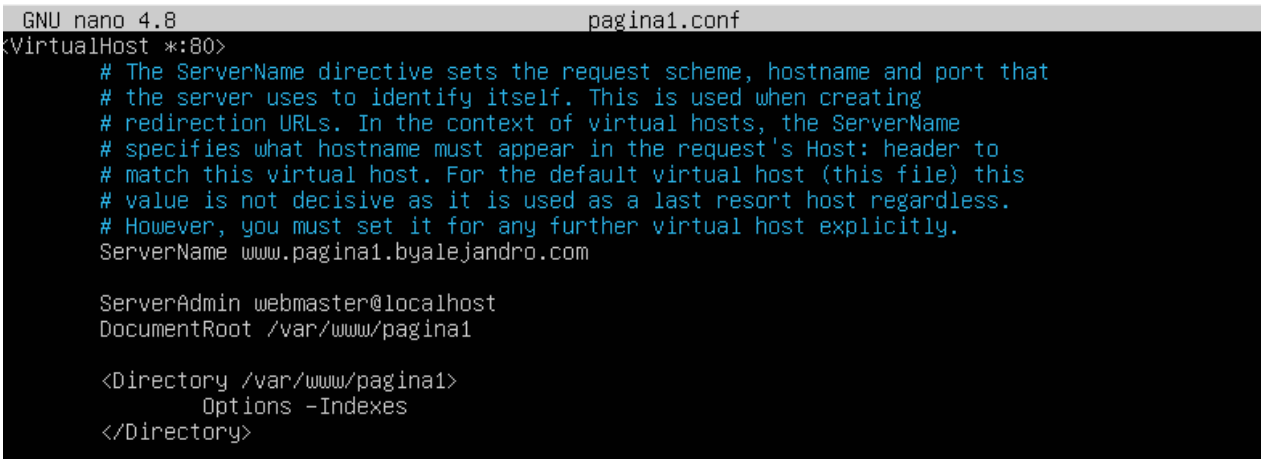
Para terminar, tan solo quedaría reiniciar el servidor apache con el comando **systemctl restart apache2**



Apache mapeo de URL

- Opciones de directorios

En el archivo de configuración de nuestro sitio web, podemos indicar una directiva **Directory** y configurarle algunas opciones. Para ello editamos el archivo **pagina1.conf** con el comando nano.



Para indicarle las opciones, ponemos la etiqueta **<Directory nombrecarpeta> </Directory>** y dentro podemos poner **Options nombre-opcion**. Además, si le metemos un **-,** significa que deja todas las opciones por defecto y le quita esa, o si le ponemos un **+**, significa que le deja las opciones por defecto y le habilita esa.

Las opciones que tenemos disponibles son:

- **All**: Habilita todas las opciones excepto la de MultiViews.
- **FollowSymLinks**: Se pueden seguir los enlaces simbólicos.
- **Indexes**: Cuando accedemos a la página web y no se encuentra un archivo por defecto de los indicados en la directiva DirectoryIndex del módulo mod_dir, se muestra la lista de ficheros. Conviene desactivarlo para que no puedan ver nuestros archivos.
- **MultiViews**: permite mostrar una página distinta en función del idioma del navegador.

- **SymLinksIfOwnerMatch**: Se pueden seguir enlaces simbólicos, sólo cuando el fichero destino es del mismo propietario que el enlace simbólico.
- **ExecCGI**: Permite la ejecución de aplicaciones CGI en el directorio.

- Trabajando con alias

  Un alias sirve para indicar una ruta en la URL que te lleve a otro directorio distinto. Lo tenemos que configurar en el archivo de nuestra web, en nuestro caso, **pagina1.conf**

  Para ello, ponemos el comando **Alias "/nombrealias" "/directorio".** También tendremos que añadir un **Directory** y ponerle el comando **Require all granted**.

```
Alias "/registrar" "/ftp/pup/registro"

<Directory "/ftp/pup/registro">
        Require all granted
</Directory>

# 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
```

- Negociación de contenidos

  Hay 2 maneras de hacer que nuestra página web se muestre de manera diferente dependiendo del idioma que tenga seleccionado el navegador web.

  La primera manera es muy sencilla, copiamos el **Index.html** como **index.html.es** y también **index.html.en**. Cada archivo será para un idioma distinto y tendrá su contenido.

Para que funcione vamos a crear una carpeta llamada **internacional** y ahí meteremos los archivos que acabamos de crear.

Finalmente, configuramos **pagina1.conf** con la nueva carpeta y la opción de **Multiviews**, lo cual hará que automáticamente se cargue el index.html que corresponda.

```
<Directory "/var/www/pagina1/internacional">
        Options +Multiviews_
</Directory>
```

La otra manera de hacerlo es creando un archivo con extensión **.var**, en el cual se indicará los archivos que cargará la web en función del idioma del navegador.

Para ello, creamos el archivo **index.var** en la carpeta **internacional**, lo editamos con nano y le ponemos los siguientes comandos:

```
URI: index

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

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

Finalmente, configuramos el archivo **pagina1.conf**

```
<Directory "/var/www/pagina1/internacional">
        DirectoryIndex index.var
        AddHandler type-map .var_
</Directory>
```

- Redirecciones

Las redirecciones pueden ser temporales o permanentes.

Para configurarlas, nos tenemos que ir al archivo pagina1.conf y añadir el siguiente código para las temporales:

```
Redirect "/web2" "http://apache2.openwebinars.net"
Redirect "/traducir" "/internacional"
```

O el siguiente para las permanentes:

```
Redirect permanent "/web2" "http://apache2.openwebinars.net"
Redirect 301 "/traducir" "/internacional"
```

Como podemos apreciar, para realizar las redirecciones permanentes podemos hacerlo con la palabra **permanent** o con el código **301**.

Las redirecciones del citado código harán que cuando entremos a **/web2** nos lleve a la URL **http://apache2.openwebinars.net** y si entramos a **/traducir** nos llevará a **/internacional**.

- Páginas de errores personalizadas

Hay 2 maneras de configurar las páginas de errores en Apache.

La primera de ellas, es sencilla, tan solo hay que añadir un código en nuestro archivo **pagina1.conf** indicando la url del **index.html** que se lanzará cuando se produzca el error indicado.

```
ErrorDocument 404 /error/index.html
```

La otra opción es modificando el archivo **localized-error-pages.conf** que se encuentra en **/etc/apache2/conf-available**

Para ello hay que descomentar las siguientes líneas y ya estará funcionando

```
<IfModule mod_negotiation.c>
        <IfModule mod_include.c>
                <IfModule mod_alias.c>

                        Alias /error/ "/usr/share/apache2/error/"

                        <Directory "/usr/share/apache2/error">
                                Options IncludesNoExec
                                AddOutputFilter Includes html
                                AddHandler type-map var
                                Order allow,deny
                                Allow from all
                                LanguagePriority en cs de es fr it nl sv pt-br ro
                                ForceLanguagePriority Prefer Fallback
                        </Directory>

                        ErrorDocument 400 /error/HTTP_BAD_REQUEST.html.var
                        ErrorDocument 401 /error/HTTP_UNAUTHORIZED.html.var
                        ErrorDocument 403 /error/HTTP_FORBIDDEN.html.var
                        ErrorDocument 404 /error/HTTP_NOT_FOUND.html.var
                        ErrorDocument 405 /error/HTTP_METHOD_NOT_ALLOWED.html.var
                        ErrorDocument 408 /error/HTTP_REQUEST_TIME_OUT.html.var
                        ErrorDocument 410 /error/HTTP_GONE.html.var
                        ErrorDocument 411 /error/HTTP_LENGTH_REQUIRED.html.var
                        ErrorDocument 412 /error/HTTP_PRECONDITION_FAILED.html.var
                        ErrorDocument 413 /error/HTTP_REQUEST_ENTITY_TOO_LARGE.html.var
                        ErrorDocument 414 /error/HTTP_REQUEST_URI_TOO_LARGE.html.var
                        ErrorDocument 415 /error/HTTP_UNSUPPORTED_MEDIA_TYPE.html.var
                        ErrorDocument 500 /error/HTTP_INTERNAL_SERVER_ERROR.html.var
                        ErrorDocument 501 /error/HTTP_NOT_IMPLEMENTED.html.var
                        ErrorDocument 502 /error/HTTP_BAD_GATEWAY.html.var
                        ErrorDocument 503 /error/HTTP_SERVICE_UNAVAILABLE.html.var
```

```
                    ErrorDocument 506 /error/HTTP_VARIANT_ALSO_VARIES.html.var
                </IfModule>
        </IfModule>
</IfModule>
```

Apache control de acceso, autenticación

- Control de acceso. Autorización

Con Apache 2.4, tenemos la directiva Require para controlar el acceso de los usuarios a nuestra página web. Esta directiva de opciones se coloca dentro de la etiqueta <Directory> en la que previamente se ha indicado el directorio al que queremos que haga efecto.

Las opciones que tenemos son:

Require all granted: El acceso es permitido incondicionalmente.

Require all denied: El acceso es denegado incondicionalmente.

Require user userid [userid]: El acceso es permitido sólo si los usuarios indicados se han autentificado.

Require group group-name [group-name]: El acceso es permitido sólo a los grupos de usuarios especificados.

Require valid-user: El acceso es permitido a los usuarios válidos.

Require ip 10 172.20 192.168.2: El acceso es permitido si se hace desde el conjunto de direcciones especificadas.

Require host dominio: El acceso es permitido si se hace desde el dominio especificado.

Require local: El acceso es permitido desde localhost.

A continuación, mostramos un ejemplo:

```
<Directory /var/www/servidor/interna>
        Require ip 172.22.0
</Directory>
```

Con este comando lo que estamos haciendo es que al directorio web interno solo se pueda acceder desde las IPs internas.

```
<Directory /var/www/servidor/interna>
        Require no ip 192.168.56
</Directory>
```

Con este comando hacemos lo mismo pero diciendo que no se permite desde la red externa

- Autenticación básica

Para configurar la autenticación básica, debemos indicar un **Directory** con la ruta del directorio de la web que queremos que tenga la autenticación y le indicamos los siguientes parámetros:

```
<Directory /var/www/pagina1/privado
        AuthUserFile "/etc/apache2/claves/passwd.txt"
        AuthName "Palabra de paso"
        AuthType Basic
        Require valid-user
</Directory
```

- **AuthUserFile**: Ruta del fichero donde se guardan los usuarios y contraseñas

- **AuthName**: Mensaje que aparece en el navegador al pedir la contraseña

- **AuthType**: Tipo de autenticación, que en este caso es la básica (**Basic**)

- **Require**: Con este comando le decimos cómo será el control de acceso, que en este caso es por usuarios (**valid-user**).

Para añadir usuarios al fichero de autenticación usamos el comando **htpasswd ruta_del_fichero nombre_usuario** y usamos el comando **-c** la primera vez que usemos el comando para crear el archivo.

```
root@hobbit:/etc/apache2# htpasswd -c /etc/apache2/claves/passwd.txt alejandro
New password:
Re-type new password:
Adding password for user alejandro
root@hobbit:/etc/apache2#
```

La autenticación básica debemos usarla lo menos posible y siempre en un entorno seguro https, ya que los datos no están cifrados con esta autenticación.

- Autenticación digest

La principal diferencia de esta autenticación con la autenticación básica es que trabajaremos con dominios, es decir, a cada usuario le corresponderá un dominio. De esta manera, la autenticación será más segura.

Antes de nada, hay que activar el módulo auth_digest en nuestro servidor para que funcione correctamente.

```
root@hobbit:~# a2enmod auth_digest
Considering dependency authn_core for auth_digest:
Module authn_core already enabled
Module auth_digest already enabled
root@hobbit:~# systemctl restart apache2
```

Seguidamente, configuramos el **Directory** como antes, pero cambiando el **AuthType** a **Digest** y teniendo en cuenta que en **AuthName** indicaremos el nombre del dominio de usuarios (grupo) con el que se realizará la autenticación.

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

Finalmente, para añadir usuarios usamos **htdigest -c ruta_fichero nombre_dominio nombre_usuario**

```
root@hobbit:/etc/apache2/claves# htdigest -c /etc/apache2/claves/digest.txt dominio alejandro
Adding password for alejandro in realm dominio.
New password:
Re-type new password:
root@hobbit:/etc/apache2/claves# _
```

- Implementación de políticas de autenticación y acceso

Para configurar el servidor con reglas de acceso más elaboradas en Apache 2.4, se emplea la directiva **RequireAll**, **RequireAny** o **RequireNone**, que van dentro de la directiva Directory.

- **RequireAll**: El acceso al directorio solamente se dará si se cumplen todas las condiciones.
- **RequireAny**: El acceso al directorio se dará cuando se cumpla alguna de las condiciones.
- **RequireNone**: El acceso al directorio se dará siempre y cuando no se cumplan las condiciones indicadas.

Las directivas se pueden anidar entre ellas, cumpliéndose siempre la lógica que tiene cada una de ellas.

Pongamos un ejemplo:

```
<Directory /var/www/pagina1/privado
        AuthUserFile "/etc/apache2/claves/passwd.txt"
        AuthName "Palabra clave"
        AuthType Basic
        <RequireAll>
                Require valid-user
                Require ip 172.22.0_
        </RequireAll>
</Directory
```

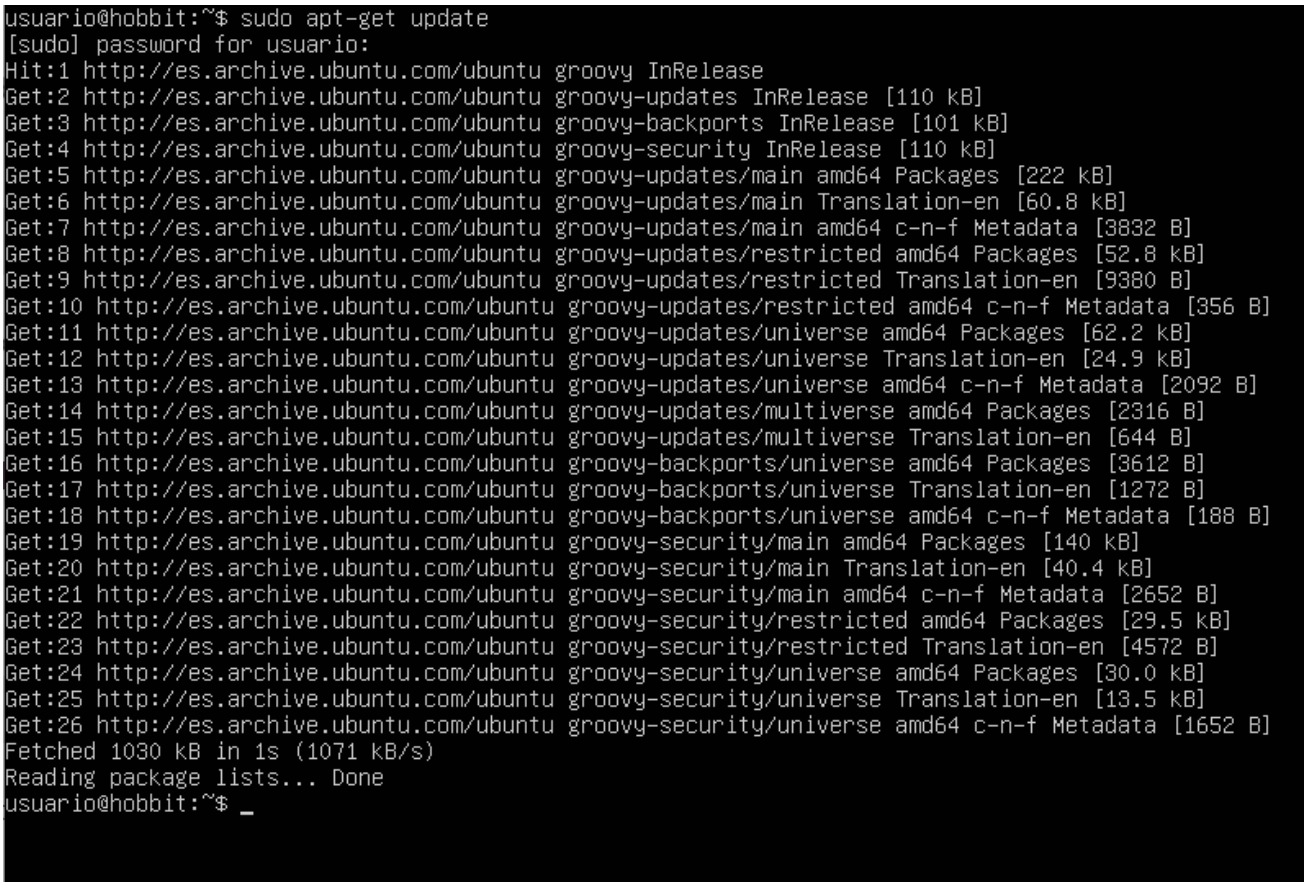
En este caso, se establece una autenticación básica de usuarios y además se debe cumplir que se pertenezca a la red interna.

Vsftpd

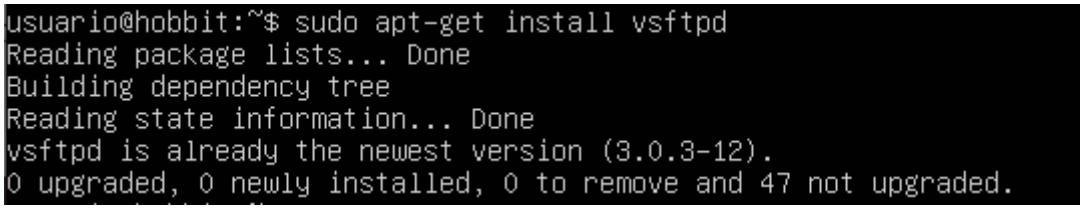
Antes de nada, es necesario que nos aseguremos de que tenemos instalada la última versión disponible de vsftpd en nuestro servidor.

Para ello, ejecutamos los comandos:

sudo apt-get update



sudo apt-get install vsftpd



Después de esto, es necesario que hagamos una copia del fichero de configuración vsftpd.conf para tenerla como backup en caso de que la necesitemos. Para ello escribimos el siguiente comando:

sudo cp /etc/vsftpd.conf /etc/vsftpd.conf.original

Una vez tenemos instalado el vsftpd, tenemos que asegurarnos de que el firewall está activado y funcionando correctamente.

Para ello miramos el estado del firewall con el comando **sudo ufw status** y en caso de que lo tengamos inactive, lo activamos con **sudo ufw enable**

```
usuario@hobbit:~$ sudo ufw status
Status: inactive
usuario@hobbit:~$ sudo ufw enable
Firewall is active and enabled on system startup
usuario@hobbit:~$ _
```

Es necesario abrir algunos puertos, por lo que usaremos estos comandos:

sudo ufw allow 20/tcp

sudo ufw allow 21/tcp

sudo ufw allow 990/tcp

sudo ufw allow 40000:50000/tcp

Después comprobamos con **sudo ufw status** y el resultado debería ser el siguiente:

```
usuario@hobbit:~$ sudo ufw allow 20/tcp
Rule added
Rule added (v6)
usuario@hobbit:~$ sudo ufw allow 21/tcp
Rule added
Rule added (v6)
usuario@hobbit:~$ sudo ufw allow 990/tcp
Rule added
Rule added (v6)
usuario@hobbit:~$ sudo ufw allow 40000:50000/tcp
Rule added
Rule added (v6)
usuario@hobbit:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
20/tcp                     ALLOW       Anywhere
21/tcp                     ALLOW       Anywhere
990/tcp                    ALLOW       Anywhere
40000:50000/tcp            ALLOW       Anywhere
20/tcp (v6)                ALLOW       Anywhere (v6)
21/tcp (v6)                ALLOW       Anywhere (v6)
990/tcp (v6)               ALLOW       Anywhere (v6)
40000:50000/tcp (v6)       ALLOW       Anywhere (v6)

usuario@hobbit:~$
```

Ahora vamos a crear nuestro directorio de usuarios ftp. En mi caso se llamará “alejandro”.

Para ello escribimos: **sudo mkdir /home/alejandro/ftp**

```
usuario@hobbit:~$ sudo adduser alejandro
Adding user `alejandro' ...
Adding new group `alejandro' (1001) ...
Adding new user `alejandro' (1001) with group `alejandro' ...
Creating home directory `/home/alejandro' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for alejandro
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
usuario@hobbit:~$ _
```

Y le damos algunos permisos al directorio:

sudo chown nobody:nogroup /home/alejandro/ftp

sudo chmod a-w /home/alejandro/ftp

sudo ls -la /home/alejandro/ftp

```
usuario@hobbit:~$ sudo mkdir /home/alejandro/ftp
usuario@hobbit:~$ sudo chown nobody:nogroup /home/alejandro/ftp
usuario@hobbit:~$ sudo chmod a-w /home/alejandro/ftp
usuario@hobbit:~$ sudo ls -la /home/alejandro/ftp
total 8
dr-xr-xr-x 2 nobody    nogroup   4096 Dec 13 23:02 .
drwxr-xr-x 3 alejandro alejandro 4096 Dec 13 23:02 ..
usuario@hobbit:~$ _
```

Finalmente creamos una carpeta para los archivos y le meteremos un archivo .txt dentro.

sudo mkdir /home/alejandro/ftp/files

sudo chown alejandro:alejandro /home/alejandro/ftp/files

echo "vsftpd sample file" | sudo tee /home/alex/ftp/files/sample.txt

```
usuario@hobbit:~$ sudo mkdir /home/alejandro/ftp/files
usuario@hobbit:~$ sudo chown alejandro:alejandro /home/alejandro/ftp/files
usuario@hobbit:~$ echo "vsftpd archivo de prueba" | sudo tee /home/alejandro/ftp/files/sample.txt
vsftpd archivo de prueba
usuario@hobbit:~$
```

Ya tenemos preparado nuestro servidor, ahora hay que configurarlo.

Para ello, nos meteremos al archivo de configuración con el comando:

sudo nano /etc/vsftpd.conf

Nos aseguramos de que tenemos **anonymous_enable=NO** y **local_enable=YES**, y descomentamos **write_enable=YES**

```
# Allow anonymous FTP? (Disabled by default).
anonymous_enable=NO
#
# Uncomment this to allow local users to log in.
local_enable=YES
#
# Uncomment this to enable any form of FTP write command.
write_enable=YES
#
```

También descomentamos **chroot_local_user=YES**

```
# You may restrict local users to their home directories.  See the FAQ for
# the possible risks in this before using chroot_local_user or
# chroot_list_enable below.
chroot_local_user=YES
#
```

A continuación, añadimos las siguientes líneas al final del fichero:

```
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
```

Teniendo en cuenta que en **userlist_file** estamos especificando el fichero donde se guardarán los usuarios del vsftpd.

Ahora solo falta añadir nuestro usuario al archivo (en mi caso “alejandro”):

echo "alejandro" | sudo tee -a /etc/vsftpd.userlist

```
usuario@hobbit:~$ echo "alejandro" | sudo tee -a /etc/vsftpd.userlist
alejandro
usuario@hobbit:~$ cat /etc/vsftpd.userlist
alejandro
usuario@hobbit:~$ sudo systemctl restart vsftpd
usuario@hobbit:~$
```

Finalmente, reiniciamos el vsftpd con el comando

sudo systemctl restart vsftpd

Ahora ya tenemos un servidor vsftpd funcional y operativo, pero vamos a dar un paso más allá para hacer que sea seguro. Para ello, usaremos SSL.

Antes de nada, creamos un certificado SSL

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

```
usuario@hobbit:~$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/
vsftpd.pem -out /etc/ssl/private/vsftpd.pem
Generating a RSA private key
...+++++
.....................................................................+++++
writing new private key to '/etc/ssl/private/vsftpd.pem'
-----
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]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:alejandro.lopez-arjona@iesruizgijon.com
usuario@hobbit:~$ _
```

Seguidamente, editamos nuevamente el archivo vsftpd.conf

sudo nano /etc/vsftpd.conf

Tenemos que cambiar las siguientes líneas:

rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem

rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key

Por estas:

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

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

Y activamos el SSL:

ssl_enable=YES

```
# This option specifies the location of the RSA certificate to use for SSL
# encrypted connections.
#rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
#rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
rsa_cert_file=/etc/ssl/private/vsftpd.pem
rsa_private_key_file=/etc/ssl/private/vsftpd.pem
ssl_enable=YES
```

Una vez hecho esto, tan solo tenemos que añadir unas líneas más para darle más seguridad al FTP y que funcione correctamente

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

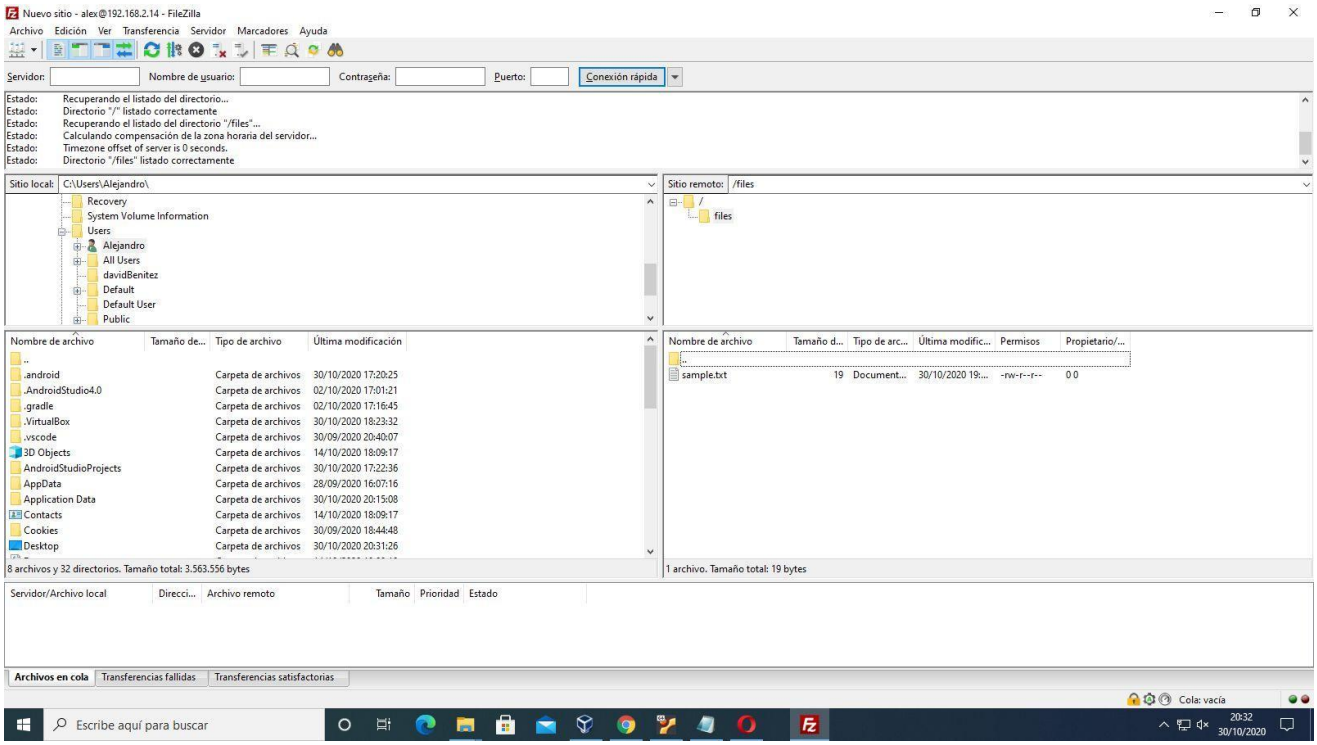
ssl_ciphers=HIGH

```
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
```

Finalmente, reiniciamos nuevamente el servidor:

sudo systemctl restart vsftpd

¡Y listo, ya hemos terminado! Ahora solo queda meternos en nuestro cliente favorito, en mi caso FileZilla y probar la conexión a nuestro servidor.



Si tenemos algún problema con nuestro servidor, podemos probar a ejecutar lo siguiente:

sudo ufw default deny

sudo iptables -L

Git

Configuración global inicial

Antes de nada, instalamos Git en nuestro servidor. Para ello empleamos los siguientes comandos:

sudo apt update

```
usuario@hobbit:~$ sudo apt update
[sudo] password for usuario:
Hit:1 http://es.archive.ubuntu.com/ubuntu groovy InRelease
Get:2 http://es.archive.ubuntu.com/ubuntu groovy-updates InRelease [110 kB]
Get:3 http://es.archive.ubuntu.com/ubuntu groovy-backports InRelease [101 kB]
Get:4 http://es.archive.ubuntu.com/ubuntu groovy-security InRelease [110 kB]
Fetched 321 kB in 1s (600 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
25 packages can be upgraded. Run 'apt list --upgradable' to see them.
usuario@hobbit:~$
```

sudo apt install git

```
usuario@hobbit:~$ sudo apt install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.27.0-1ubuntu1).
git set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 25 not upgraded.
usuario@hobbit:~$
```

Comprobamos que la instalación se ha realizado correctamente.

git –version

```
usuario@hobbit:~$ git --version
git version 2.27.0
usuario@hobbit:~$
```

Configuramos el usuario que usaremos con Git, para ello empleamos

git config --global user.name "nombre_del_usuario"

git config --global user.email "email_del_usuario "

Y comprobamos que lo hemos configurado bien

git config --global –list

```
usuario@hobbit:~$ git config --global user.name "alejandro"
usuario@hobbit:~$ git config --global user.name "alejandro.lopez-arjona@iesruizgijon.com"
usuario@hobbit:~$ git config --global --list
user.name=alejandro.lopez-arjona@iesruizgijon.com
usuario@hobbit:~$
```

Finalmente, tenemos que crear la carpeta con la que queremos trabajar en nuestro proyecto. Seguidamente nos metemos a ella e inicializamos el proyecto en Git.

mkdir nombre_carpeta

cd nombre_carpeta

git init

```
usuario@hobbit:~$ mkdir tareagit
usuario@hobbit:~$ cd tareagit
usuario@hobbit:~/tareagit$ git init
Initialized empty Git repository in /home/usuario/tareagit/.git/
usuario@hobbit:~/tareagit$
```

Creación de repositorio local y remoto

- Repositorio local:

Una vez hemos inicializado nuestro proyecto en la carpeta deseada, podemos agregar archivos y aprovechar las ventajas de Git.

Como ejemplo, creamos un archivo llamado prueba.txt

nano prueba.txt

Escribimos “Esto es una prueba” y damos CTRL+X para guardar y salir.

Este archivo todavía se encuentra en nuestra área de trabajo (working directory). Si hacemos un **git status**, veremos que efectivamente es así:

```
usuario@hobbit:~/tareagit$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        prueba.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Lo vamos a pasar al área de Staging

git add prueba.txt

```
usuario@hobbit:~/tareagit$ git add prueba.txt
usuario@hobbit:~/tareagit$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   prueba.txt

usuario@hobbit:~/tareagit$ _
```

Si modificamos el archivo, podemos ver que si hacemos un **git status**, nos informa de los cambios

```
usuario@hobbit:~/tareagit$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   prueba.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   prueba.txt

usuario@hobbit:~/tareagit$
```

En ese caso, tenemos que volver a hacer un **git add prueba.txt**

Una vez hayamos modificado los archivos de nuestro proyecto a nuestro gusto, podemos guardarlos como una versión concreta.

git commit -m "Primera revisión"

```
usuario@hobbit:~/tareagit$ git commit -m "Primera revision"
[master (root-commit) 843bd8b] Primera revision
 1 file changed, 1 insertion(+)
 create mode 100644 prueba.txt
usuario@hobbit:~/tareagit$
```

Con el siguiente comando podemos ver todos los cambios que hayamos hecho.

git log --oneline –color

```
usuario@hobbit:~/tareagit$ git log --oneline --color
843bd8b (HEAD -> master) Primera revision
usuario@hobbit:~/tareagit$
```

- Repositorio remoto:

Antes de nada, tenemos que crear una clave SSH.

Nos vamos a nuestra carpeta personal **/home/nombre_usuario**

Ejecutamos lo siguiente:

mkdir .ssh

cd ssh

ssh-keygen -t rsa -C "poner_nuestro_email"

```
usuario@hobbit:~$ mkdir .ssh
usuario@hobbit:~$ cd .ssh
usuario@hobbit:~/.ssh$ ssh-keygen -t rsa -C "alejandro.lopez-arjona@iesruizgijon.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/usuario/.ssh/id_rsa): clave
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in clave
Your public key has been saved in clave.pub
The key fingerprint is:
SHA256:JiX8PjN1pNzFC3k6OO7TLxm9ot+6eIijVKDoMxUOzEM alejandro.lopez-arjona@iesruizgijon.com
The key's randomart image is:
+---[RSA 3072]----+
|  +E             |
|  .+..     o     |
|   .o o . + +    |
| . ... = = = .   |
|. ..  o S *..    |
|. .  . = +...    |
| +  .  .*o o .   |
|  o.  o.o=*..    |
|    .. .+=+*o    |
+----[SHA256]-----+
usuario@hobbit:~/.ssh$ _
```

Si hacemos un **ls -al** veremos que se nos han creado un fichero con extensión .pup, que es el que contiene la clave.

```
usuario@hobbit:~/.ssh$ ls -al
total 16
drwxrwxr-x 2 usuario usuario 4096 Dec 14 02:28 .
drwxr-xr-x 7 usuario usuario 4096 Dec 14 02:27 ..
-rw------- 1 usuario usuario 2635 Dec 14 02:28 clave
-rw-r--r-- 1 usuario usuario  593 Dec 14 02:28 clave.pub
usuario@hobbit:~/.ssh$ _
```

Abrimos este archivo y copiamos la clave que hay en su interior. Ahora tan solo tenemos que irnos a nuestro servidor favorito, como por ejemplo GitHub e introducir la clave SSH. En este caso nos tendríamos que ir a GitHub > Settings > SSH Keys.

Para usar nuestro repositorio remoto, tenemos que agregar el proyecto que hayamos creado en GitHub

git remote add origin https://github.com/ usuario/proyecto.git

Y con esto estará configurado, ahora tan solo tenemos que añadir archivos con **git add** y **git commit** y cuando queramos mandarlo a nuestro repositorio online, escribimos:

git push -u origin master

Comandos básicos para la gestión de archivos usando git

- Eliminar archivos: **git rm nombre_archivo**

El archivo se elimina del Repository. Si hacemos un **git commit**, el archivo se elimina completamente.

- Mover archivos: **git mv nombre_archivo nueva_ruta**

Ejemplo: git mv estilos.css style

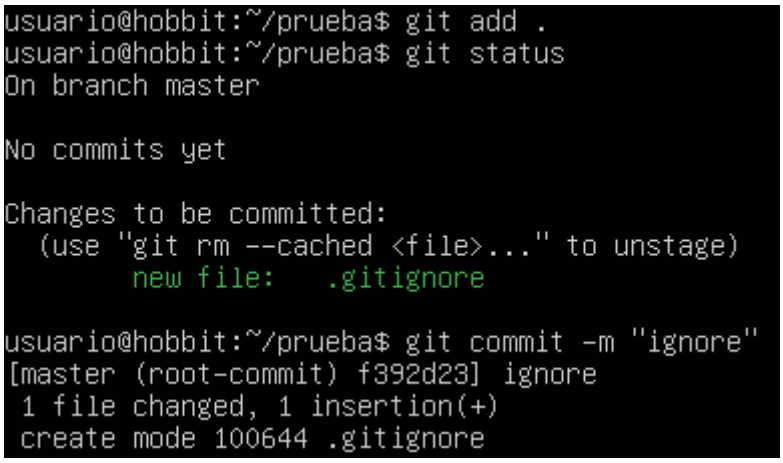
- Ignorar archivos: Si queremos que determinados archivos se ignoren de nuestro proyecto, tenemos que escribir **nano .gitignore** y dentro escribimos los ficheros que queremos ignorar. Si queremos ignorar los ficheros de un tipo de extensión en concreto, escribimos: *.extension

Por ejemplo: *.log

De esta manera ignoraremos todos los archivos .log. Tan solo tenemos que guardar el archivo y escribimos:

git add .

git commit -m "ignore"



- Actualizar Repository:

Para actualizar nuestro repository local escribimos

git pull origin master

git push origin master