Guillermo Romero Escribano



Practica 2.2 Servidor Web Modulo

UD 2. Servicio Web

Contenido

[A) Módulos en Linux 2](#_Toc120105876)

[A.1) Módulos 3](#_Toc120105877)

[A.2) Módulo userdir 3](#_Toc120105878)

[A.3) Módulo userdir en el servidor de clase 4](#_Toc120105879)

[B) Control de acceso por IP y nombre de dominio 5](#_Toc120105880)

[C) Autenticación y autorización Basic y Digest 7](#_Toc120105881)

[C.1) Autenticación Basic 8](#_Toc120105882)

[C.2) Autenticación Digest 9](#_Toc120105883)

[D) Ficheros .htaccess (si no sale poner pantallazo de haberlo intentado) 10](#_Toc120105884)

[E) Ficheros de registros (logs) 12](#_Toc120105885)

[F) Módulos status e info 13](#_Toc120105886)

[G) Webalizer 14](#_Toc120105887)

[F) GitHub 16](#_Toc120105888)

# A) Módulos en Linux

El servidor HTTP Apache es **MODULAR**, lo cual quiere decir que se pueden añadir módulos para darle otras funcionalidades al servidor HTTP. En este apartado vamos a ver como se cargan nuevos módulos y como se descargan dichos módulos en Linux y le daremos uso.

Existen módulos estáticos, que se cargan al compilar el servidor y se pueden ver mediante el comando:

sudo apache 2ctl -l

También existen módulos dinámicos, los cuales pueden cargarse y descargarse de manera dinámica. En Linux, los módulos disponibles se encuentran en el directorio

/etc/apache2/mods-available/

Los archivos **.load** sirven para cargar el módulo y los **.conf** para configurarlo.

Mientras que los módulos que están cargados se encuentran en el directorio

/etc/apache2/mods-enabled/

Para habilitar y deshabilitar módulos se usan los comandos:

a2enmod nombre\_del\_modulo

a2dismod nombre\_del\_modulo

Cada vez que se carga/descarga un módulo, tendrás que reiniciar el servidor Apache.

Los módulos existentes se pueden consultar en: <http://httpd.apache.org/docs/2.2/mod/>

## A.1) Módulos

**PASO 1)** Comprueba los módulos estáticos que se han cargado al compilar el servidor ejecutando el comando correspondiente.

**PASO 2)** Comprueba los módulos que se han cargado dinámicamente al arrancar el servidor.

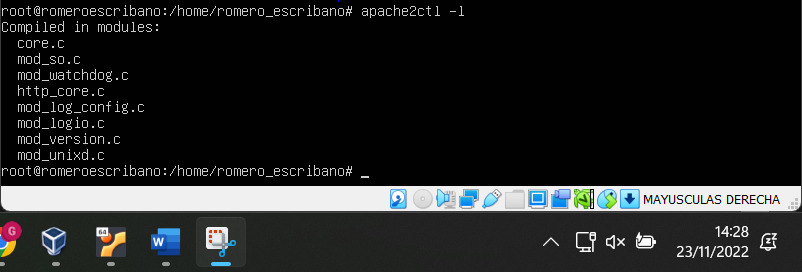
**PASO 3)** Edita uno de los archivos .load y observa cómo se usa la directiva LoadModule. ¿Qué extensión tienen los archivos donde está el código del módulo?

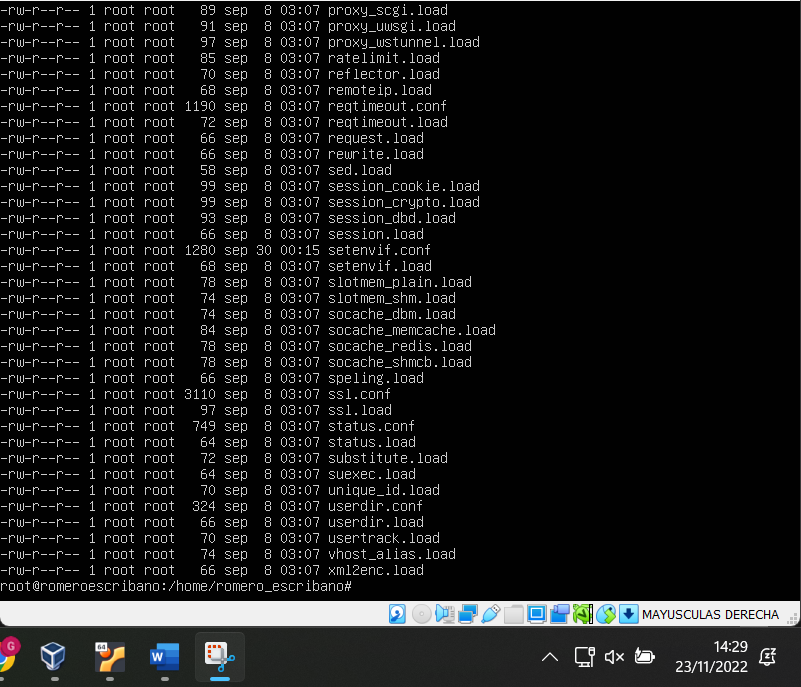
**PASO 4)** Edita uno de los archivos .conf y observa cómo se añaden directivas dentro del módulo.

¿Qué etiquetas se utilizan en estos archivos?

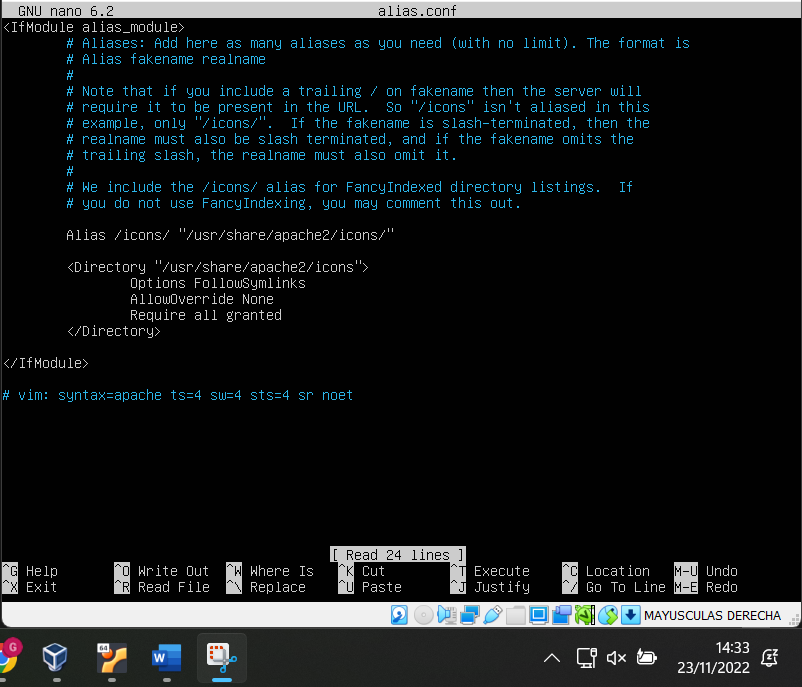
**PASO 5)** Consulta el directorio /usr/lib/apache2/modules/ ¿qué archivos contiene?

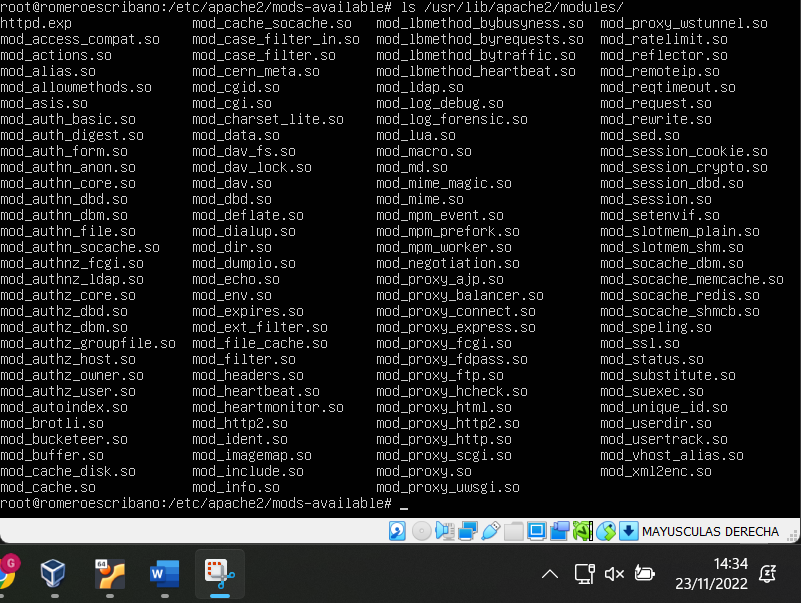
**Toma capturas de los pasos 1, 2, 3 y 4.**







******

******

A.2) Módulo userdir

El módulo **userdir** se utiliza para usar como directorio raíz del servidor HTTP el directorio home de un usuario.

Al utilizar este módulo, el usuario desde el que se va a usar, en el directorio raíz (/home/usuario) tendrá un directorio public\_html que hará las veces de raíz web para Apache2.

En el caso de directorios raíz de usuarios, para acceder a ellos habrá que usar el carácter “~”, o sea, la dirección será de la forma <http://hostname/~username/>

**PASO 1)** Comprueba si el módulo userdir está habilitado. ¿Lo está?

No está habilitado

**PASO 2)** Si no lo está, habilita el módulo userdir.

**PASO 3)** Verifica ahora si el módulo está habilitado.

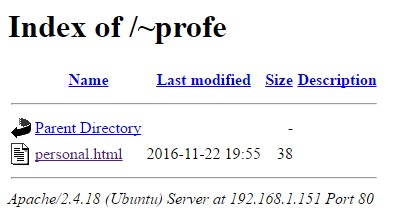
**PASO 4)** Reinicia el servidor para que los cambios tengan efecto.

**PASO 5)** Consulta el archivo **/etc/apache2/mods-enabled/userdir.conf**. ¿Cuál es el único usuario para el que está deshabilitado el uso de directorios personales? ¿Cuál es el subdirectorio que deben crear los usuarios en su carpeta home para poner sus páginas personales?

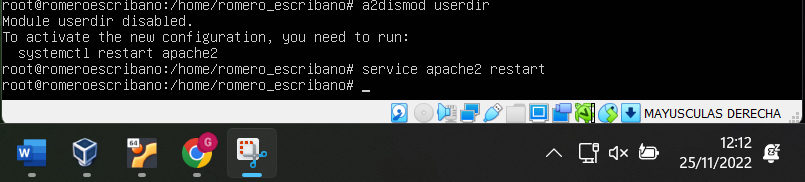
El usuario root

**PASO 6)** Crea el directorio necesario dentro de tu usuario y añade un fichero denominado **personal.html** con el contenido Tu nombre e indicando que es personal.

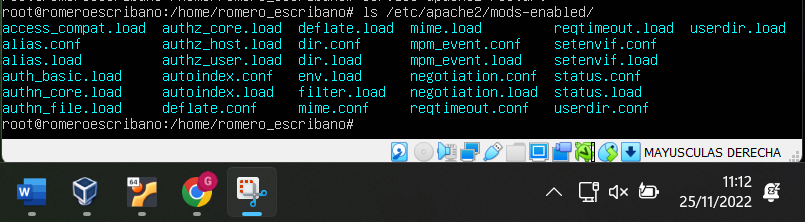
**PASO 7)** Desde la máquina física, abre un navegador y accede al directorio raíz de tu usuario Linux.

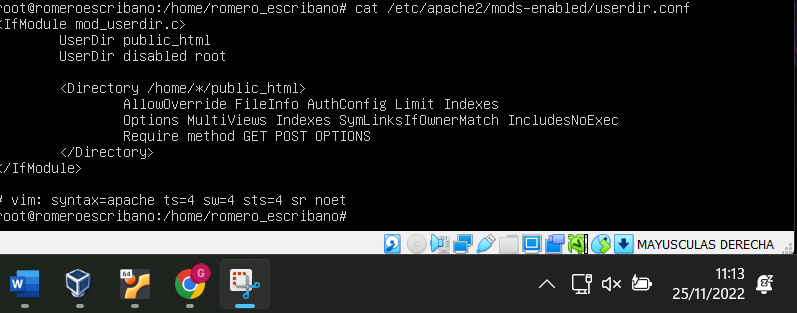


**PASO 8)** Descarga el módulo y reinicia el servidor para que los cambios tengan efecto.

******

**Toma una captura de los pasos 3,5 y 7 (en esta última, donde se vea la barra de direcciones del navegador)**

******

******



## A.3) Módulo userdir en el servidor de clase

En el servidor del aula todos tenéis un usuario y una contraseña para entrar.

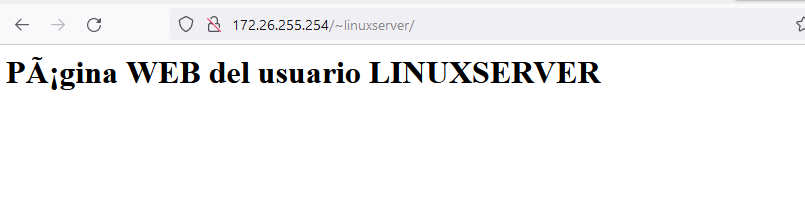
Recordad que es la inicial del primer nombre y el primer apellido.

Ejemplo: Amapola Gutiérrez de la Vega, sería agutierrez. La contraseña es alumno.

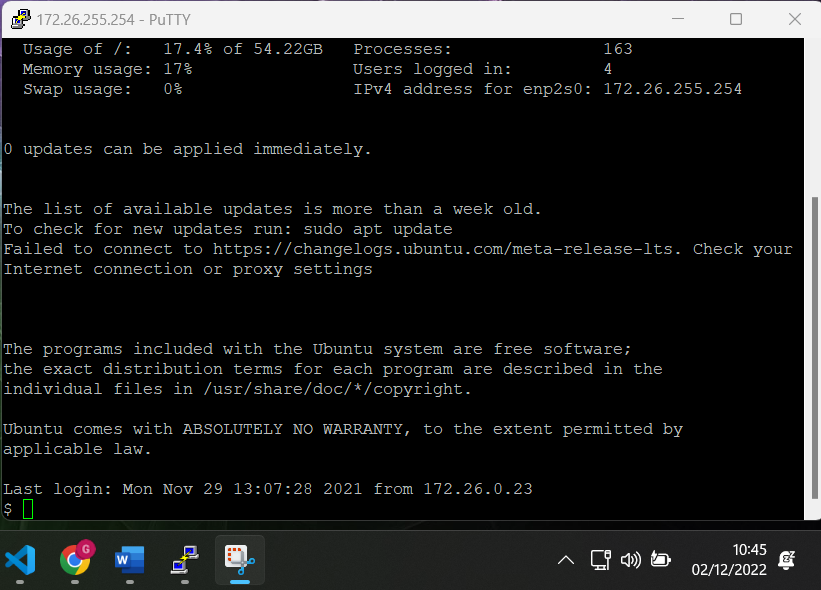
**PASO 1)** Accede al servidor a través de Putty. IP:

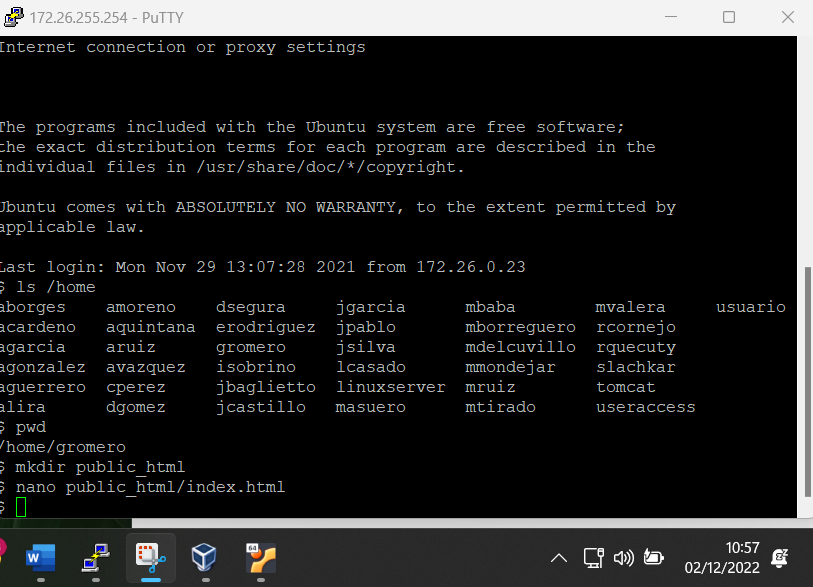
**PASO 2)** Da los pasos necesarios para qué al acceder a http.//172.26.255.254/~agutierrez se vea tu página web en el servidor.

La página debe contener la IP de servidor y tu nombre completo

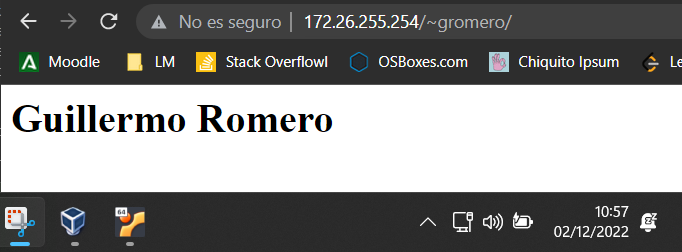


**Detalla los pasos seguidos para conseguirlo.**





Creamos la carpeta public\_html y dentro el archivo index.html



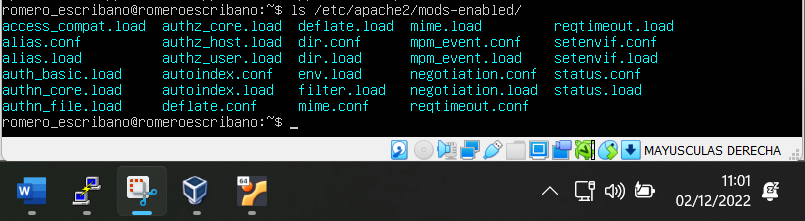
# B) Control de acceso por IP y nombre de dominio

Para poder controlar el acceso a diferentes recursos dentro de nuestro servidor web podemos hacer uso del módulo **authz\_host**. Este módulo puede permitir o denegar el acceso a un recurso por parte de un host a partir de su dirección IP o su nombre de dominio.

Más información del módulo en:<https://httpd.apache.org/docs/2.4/mod/mod_authz_host.html>

Vamos a controlar el acceso a un recurso de Apache en nuestro servidor Linux para que la máquina física tenga acceso, y la máquina de un compañero no:

**PASO 1)** Comprueba si está habilitado el módulo **authz\_host**. ¿Lo está?



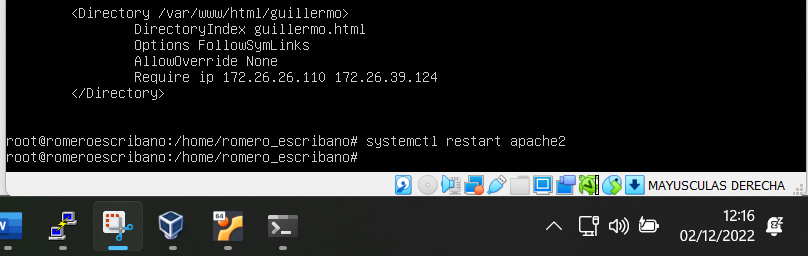
Si esta activado.

**PASO 2)** Crea un directorio ***/var/www/html/tuNombre/***. Dentro del directorio crea un archivo y llámalo **tuNombre.html** y añade el contenido que quieras.

**PASO 3)** Edita el fichero de configuración **/etc/apache2/sites-available/000-default.conf** y añade la directiva Directory para el recurso creado anteriormente.

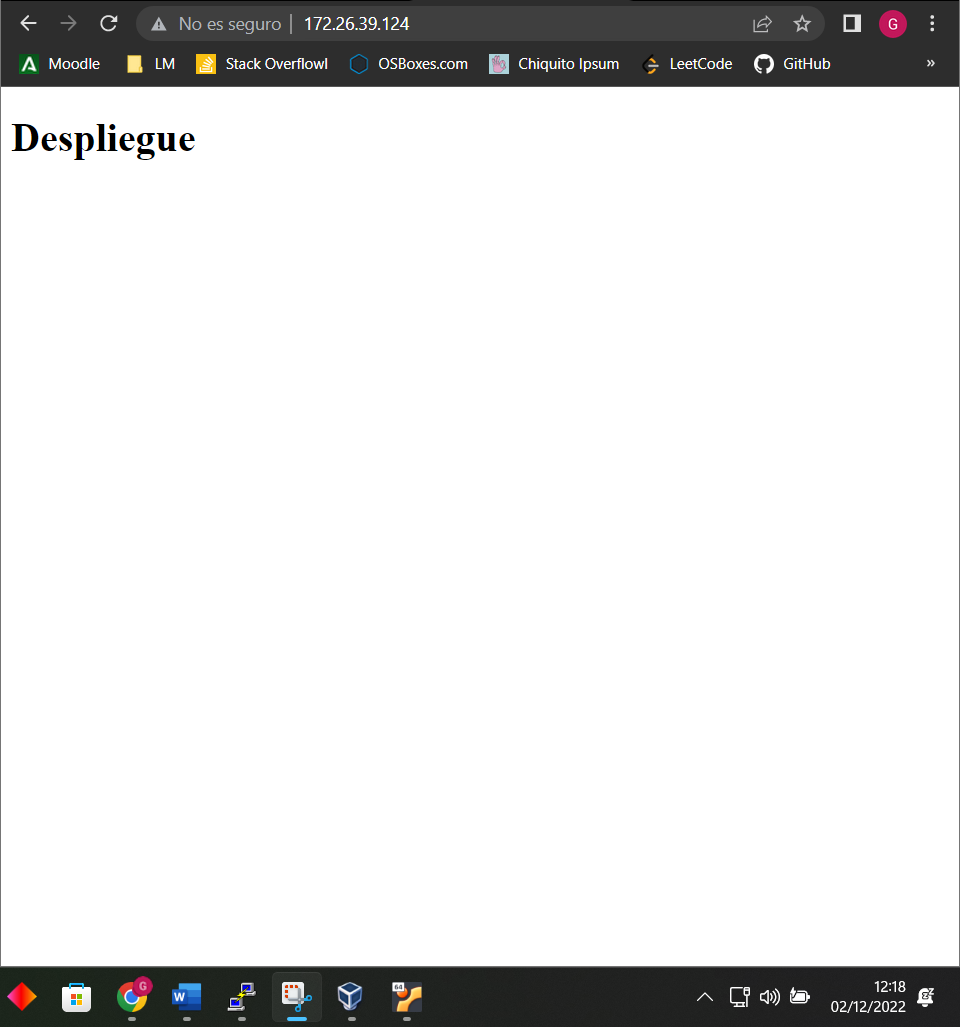


**PASO 4)** Añade dentro de la directiva anterior las directivas de acceso necesarias para que la máquina física, a partir de su dirección IP, pueda acceder a este recurso pero no la máquina del compañero (échale un vistazo al enlace informativo del módulo authz\_host que hay más arriba).

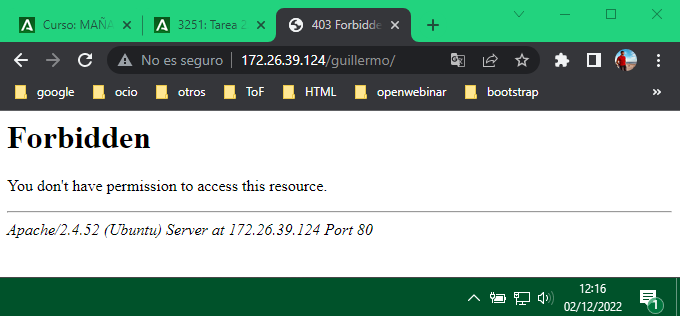


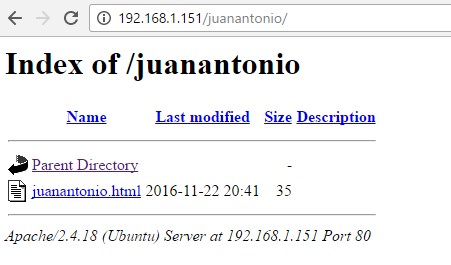
**PASO 5)** Reinicia el servidor para que los cambios tengan efecto.

**PASO 6)** Abre un navegador desde tu máquina física e intenta acceder al recurso **/tuNombre/** y comprueba que se puede.



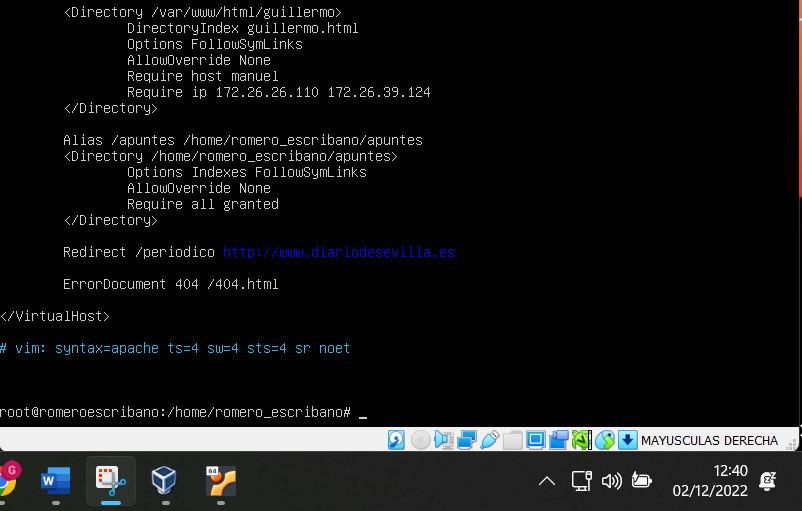
**PASO 7)** Abre un navegador desde la máquina del compañero e intenta acceder al recurso **/tuNombre/** y comprueba que no se puede.

Desde mi máquina física: Desde la máquina del compañero:



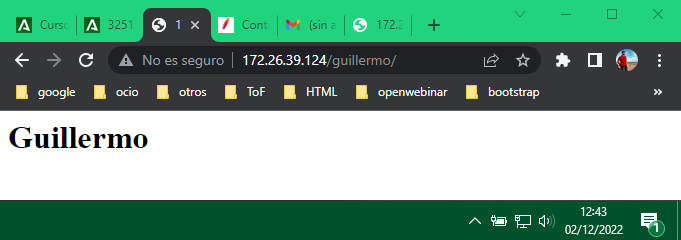
**Toma una captura de los pasos 3,4,5 y 6.**

**PASO 8)** Añade el acceso al recurso de tu carpeta para la máquina del compañero pero **usando su nombre de host en vez de su IP**.

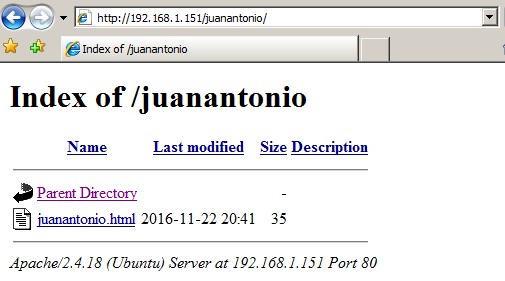


**PASO 9)** Reinicia el servidor para que los cambios tengan efecto.

**PASO 10)** Abre un navegador desde la máquina del compañero e intenta acceder al recurso **/tuNombre/** y comprueba que ahora sí se puede.

****

Desde la máquina del compañero:



**Toma una captura de los pasos 7 y 9.**

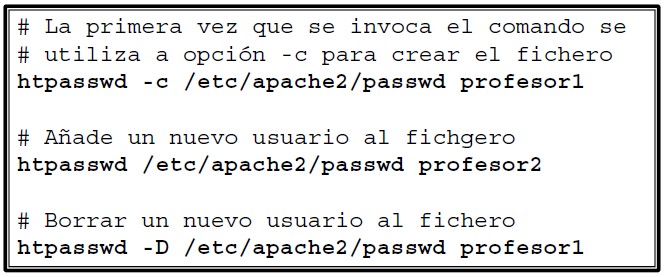
# C) Autenticación y autorización Basic y Digest

La autenticación es el proceso mediante el cual se puede verificar que alguien es quien dice ser. La autorización es el proceso mediante el cual se permite a acceder a un recurso solicitado.

En este punto vamos a usar las autenticaciones Basic y Digest.

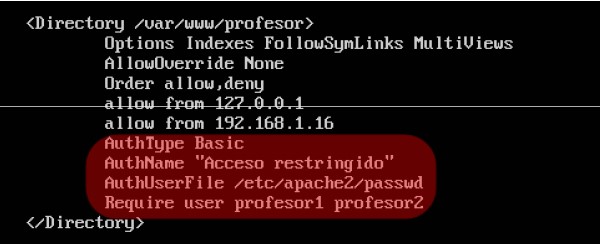
[(http://httpd.apache.org/docs/2.2/es/howto/auth.html)](http://httpd.apache.org/docs/2.2/es/howto/auth.html)

Autenticación Basic:

* La contraseña es enviada por el cliente en texto plano.
* Autenticación y autorización sobre fichero de texto (comando **htpasswd**).
* Usa los módulos **authn\_file y authz\_user**.

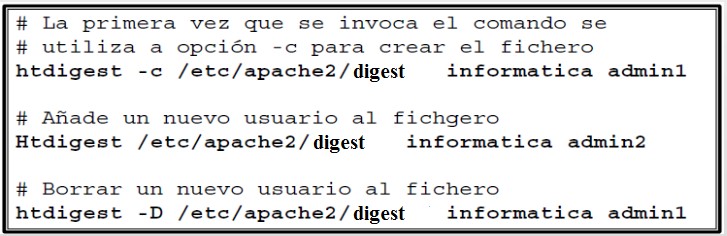
<http://httpd.apache.org/docs/2.2/es/programs/htpasswd.html>

* Definir directivas:
  + AuthType: tipo de autorización o AuthName: nombre de la autorización cuando el cliente reciba el mensaje o AuthUserFile: localización del fichero donde están los usuarios que pueden autenticarse
  + Require solo los usuarios o grupos de usuarios que aparecen en esta directiva pueden acceder al recurso.



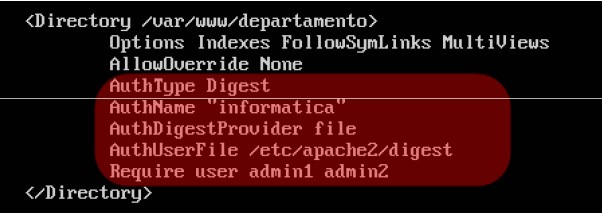
Autenticación digest:

* La contraseña se envía cifrada (cifrado débil) por el cliente.
* Autenticación y autorización sobre fichero de texto (comando htdigest)
* Módulos: **mod\_auth\_digest** y **mod\_auth\_user**



[*http://httpd.apache.org/docs/2.2/es/programs/htdigest.html*](http://httpd.apache.org/docs/2.2/es/programs/htdigest.html)

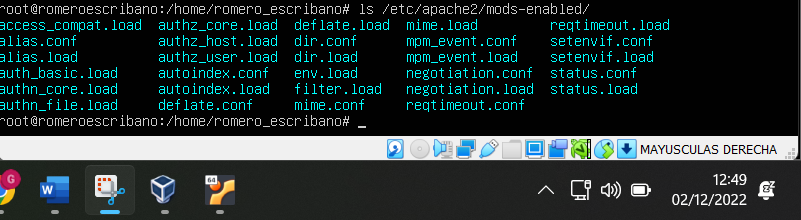
* Definir directivas:
  + AuthType: tipo de autorización
  + AuthName: nombre de la autorización cuando el cliente reciba el mensaje o AuthDigestProvider: establecen el método de almacenamiento de las contraseñas del servidor, en nuestro caso se almacenarán en un archivo y por tanto tendrán el valor file
  + AuthUserFile: localización del fichero donde están los usuarios que pueden autenticarse
  + Require solo los usuarios o grupos de usuarios que aparecen en esta directiva pueden acceder al recurso



En este punto vamos a configurar la autenticación Basic y Digest para recursos de Apache en nuestro servidor Linux.

## C.1) Autenticación Basic

**PASO 1)** Comprueba si el módulo **auth\_basic** está habilitado, si no lo está, habilítalo.

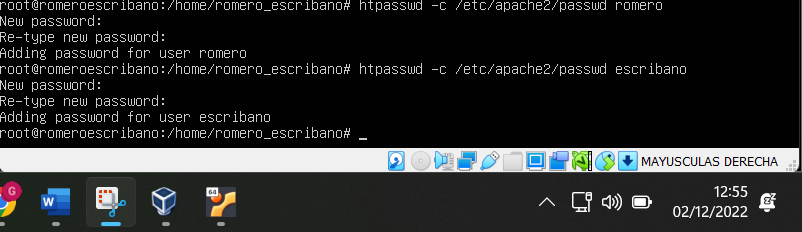


Si esta activado

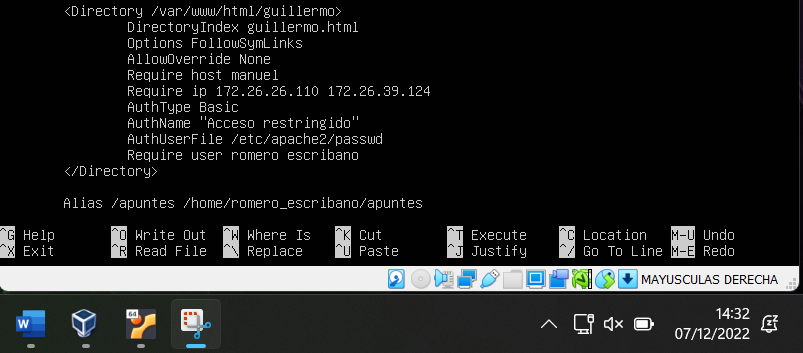
**PASO 2)** Vamos a crear el directorio **/nombreAlumno/** dentro de nuestro directorio raíz **/var/www/html/.** Dentro añadiremos un archivo **nombreAlumno.html** donde incluiremos el contenido que queramos.

Ya hicimos una carpeta parecida para el apartado anterior por lo tanto usaremos esta.

**PASO 3)** Para usar la autenticación Basic hay que crear un fichero accesible (el fichero que se creará será **/etc/apache2/passwd**) en el que se guardarán los usuarios y contraseñas. Para crear ese fichero se utilizará el comando httpaswd (ver cuadro arriba). Añade los usuarios **apellido1** y **apellido2**.

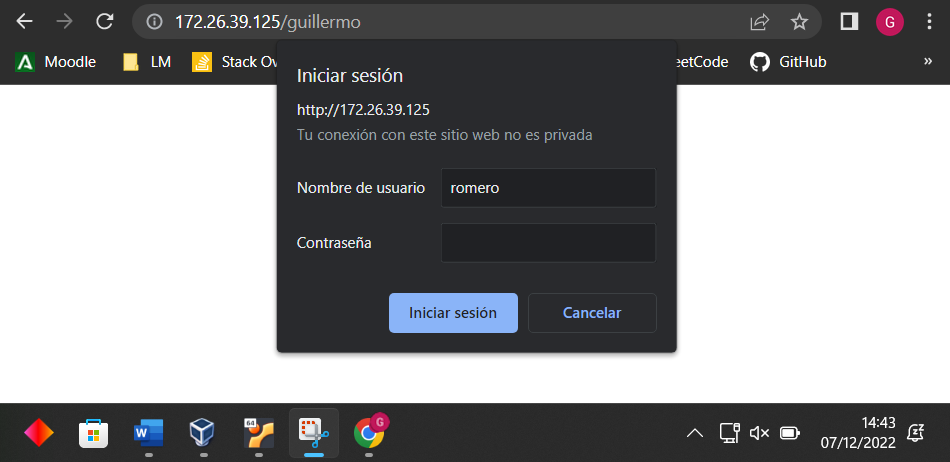


**PASO 4)** Edita el fichero de configuración **/etc/apache2/sites-available/000-default.conf** y permite el acceso al directorio **/var/www/html/nombreAlumno** a los usuarios **apellido1** y **apellido2** (ver cuadro ejemplo arriba).

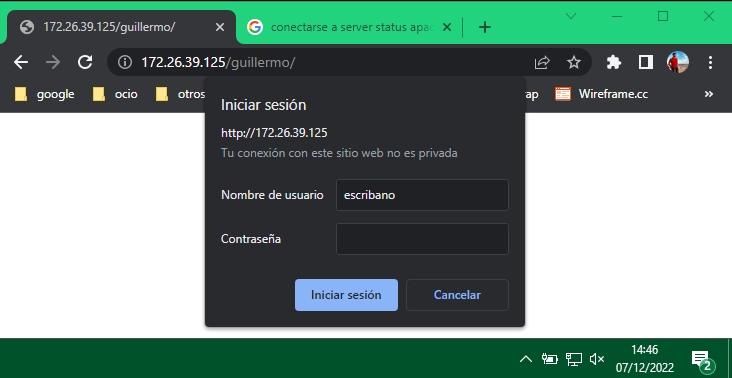


**PASO 5)** Reinicia el servidor para que los cambios tengan efecto.

**PASO 6)** Abre un navegador desde tu máquina física y accede al recurso **/nombreAlumno** como usuario **apellido1**.



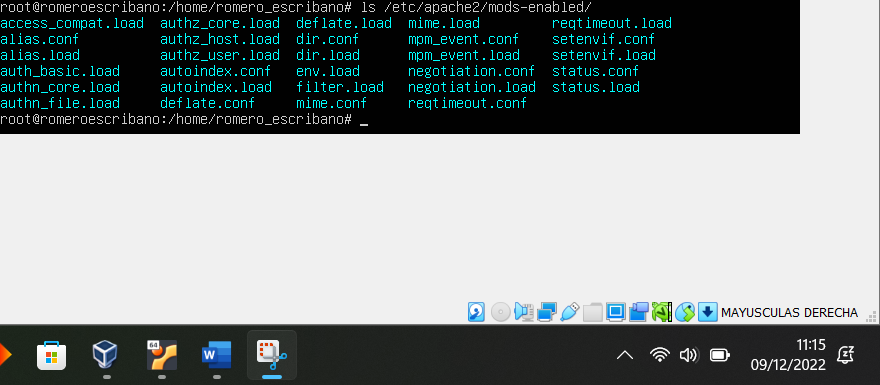
**PASO 7)** Abre un navegador desde la máquina de un compañero y accede al recurso **/nombreAlumno** como usuario **apellido2**.



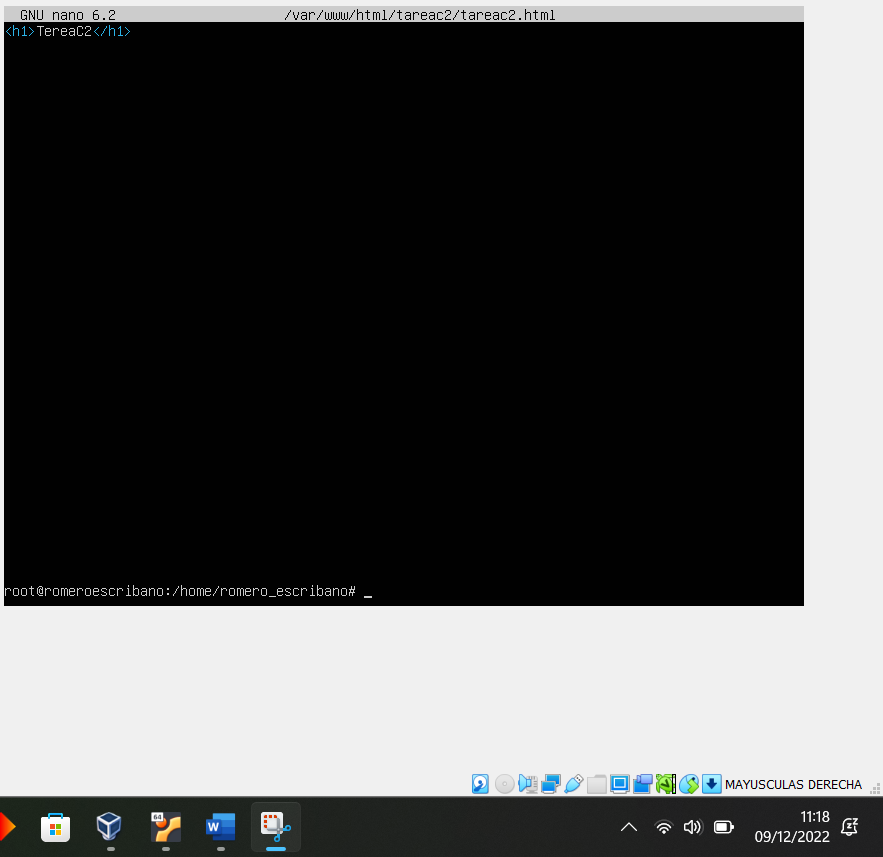
**Toma capturas de los pasos 3,4, 6 y 7 (de estas últimos una captura cuando sale el cuadro para autenticarte y luego una vez dentro del recurso /amigo).**

## C.2) Autenticación Digest

**PASO 1)** Comprueba si el módulo **auth\_digest** está habilitado, si no lo está, habilítalo.



**PASO 2)** Vamos a crear el directorio **/tareac2/** dentro de nuestro directorio raíz **/var/www/html/.** Dentro añadiremos un archivo **tareac2.html** donde incluiremos el contenido que queramos.

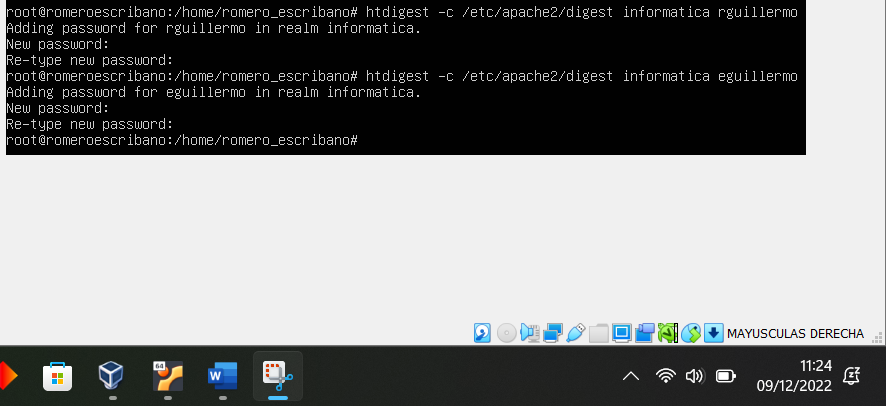


**PASO 3)** Para usar la autenticación Digest también hay que crear un fichero accesible (el fichero que se creará será también **/etc/apache2/passwd** pero para digest) en el que se guardarán los usuarios y contraseñas, pero esta vez asociados a un dominio (en el cuadro ejemplo de arriba el dominio o “realm” es informática). Para crear ese fichero se utilizará el comando httdigest (ver cuadro arriba). Añade los usuarios **inicialPrimerApellidoNombre** y **inicialSegundoApellidoNombre**.

Ejemplo: Amapola Gutierrez de la Vega:

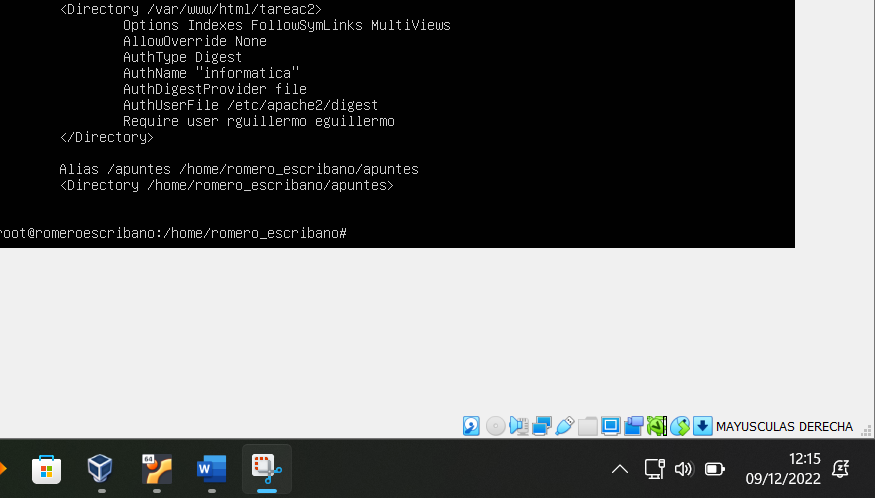
gamapola

vamapola



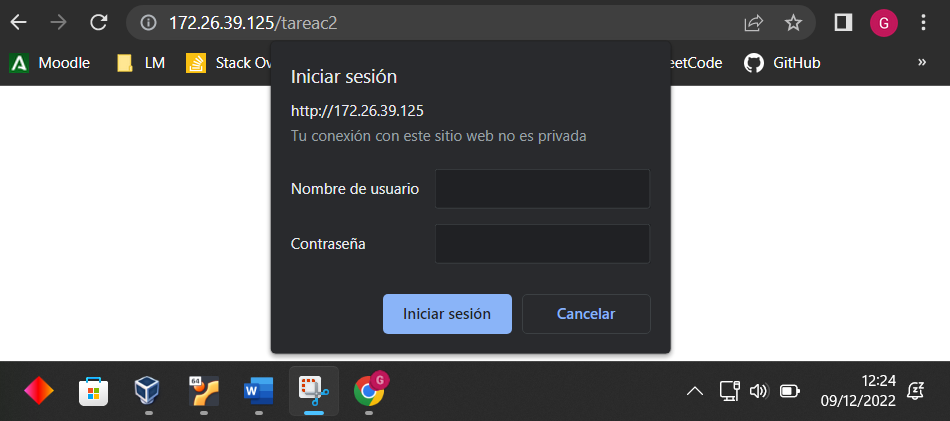
CONTRASEÑA : alumno

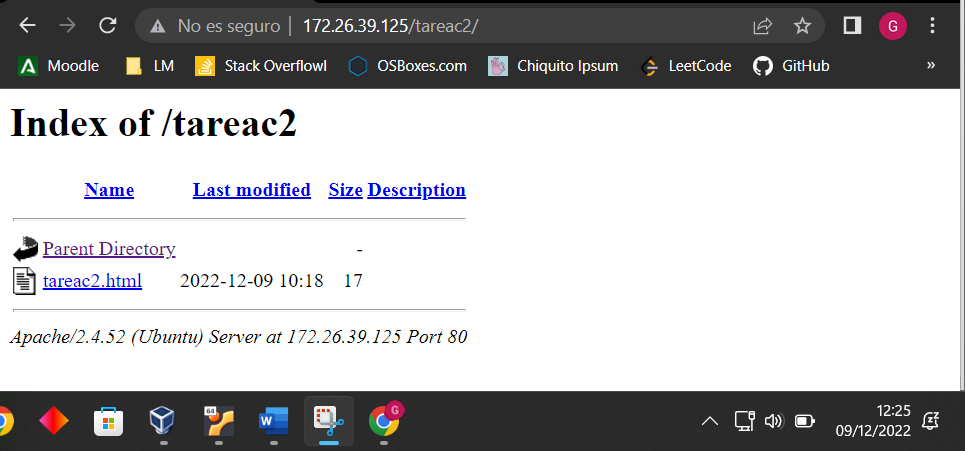
**PASO 4)** Edita el fichero de configuración **/etc/apache2/sites-available/000-default.conf** y permite el acceso al directorio **/var/www/html/tareac2** a los usuarios **inicialPrimerApellidoNombre** y **inicialSegundoApellidoNombre** (ver cuadro ejemplo arriba). Ten en cuenta que en la directiva AuthName tienes que poner lo mismo que pusiste en el dominio o “realm”.



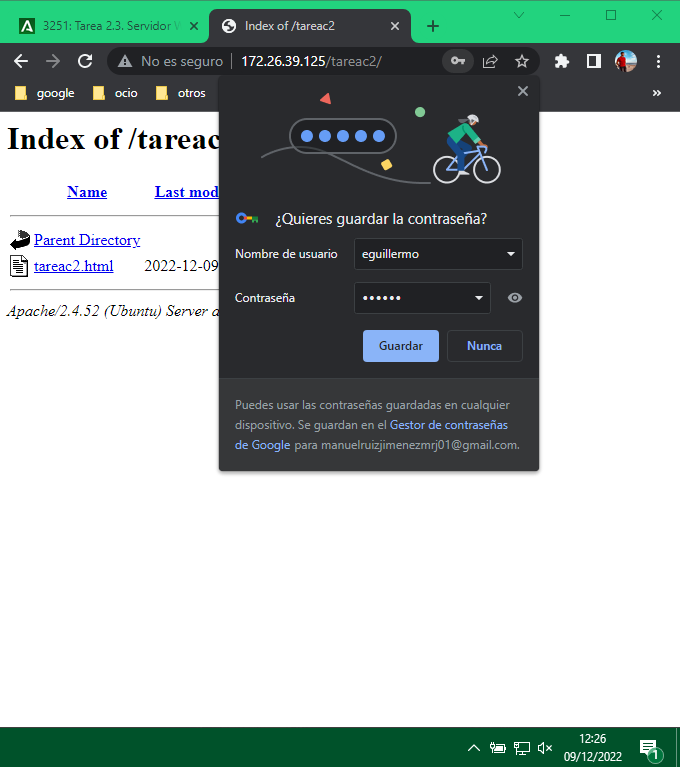
**PASO 5)** Reinicia el servidor para que los cambios tengan efecto.

**PASO 6)** Abre un navegador desde tu máquina física y accede al recurso **/tareac2** como usuario **inicialPrimerApellidoNombre**.





**PASO 7)** Abre un navegador desde la máquina de un compañero y accede al recurso **/tareac2** como usuario **inicialSegundoApellidoNombre**.



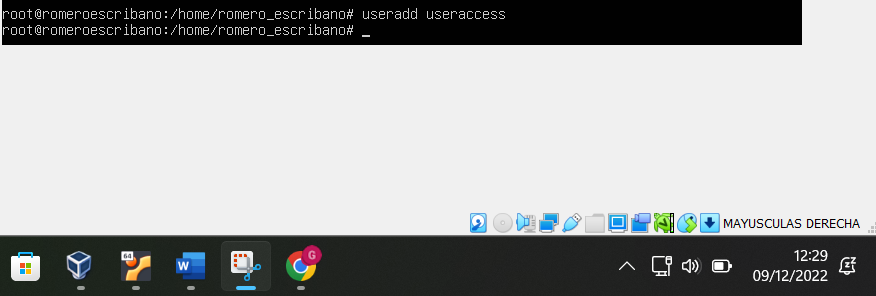
**Toma una captura de los pasos 3, 4, 6 y 7 (de estas últimos una captura cuando sale el cuadro para autenticarte y luego una vez dentro del recurso /primo).**

# D) Ficheros .htaccess (si no sale poner pantallazo de haberlo intentado)

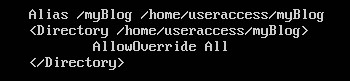
Los archivos .htaccess permiten configurar de manera personalizada directorios concretos que se quieran servir desde el Servidor Apache, pero sin que estos cambios afecten a la configuración general del servidor Apache. Básicamente permite “personalizar” el cómo se sirven unos contenidos que pertenecen a un directorio concreto.

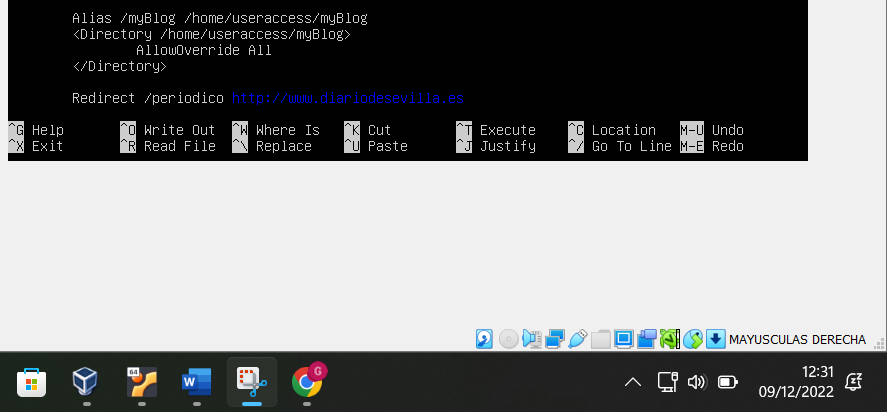
Para poder hacer uso de los ficheros .htaccess tenemos que permitir en el archivo de configuración de apache (httpd.conf) su uso mediante la directiva “AllowOverride”.

**PASO 1)** Crea el usuario **useraccess**.



**PASO 2)** Abre el fichero de configuración **000-default** y crea el **alias myBlog** dentro de la carpeta personal del nuevo usuario **useraccess**. Deja como única directiva **AllowOverride All**.

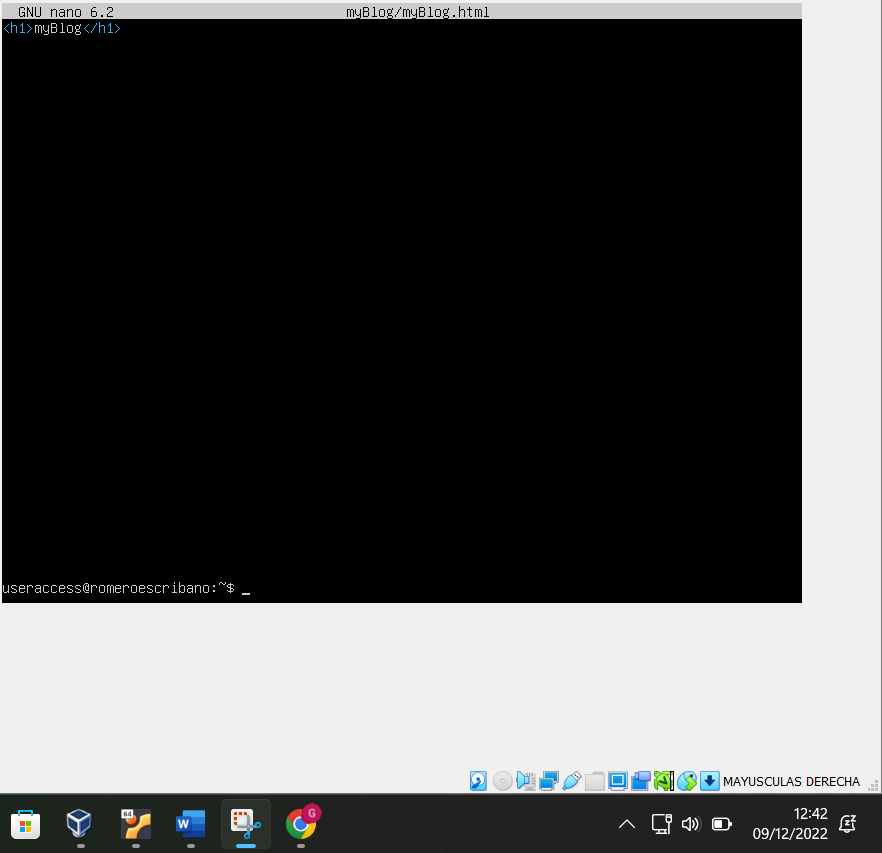




**PASO 3)** Reinicia el servidor para que los cambios tengan efecto.

**PASO 4)** Inicia sesión con el nuevo usuario **useraccess**.

**PASO 5)** Crea dentro del directorio home de este usuario **el directorio myBlog**. Crea dentro el archivo **myBlog.html** con el contenido que quieras.



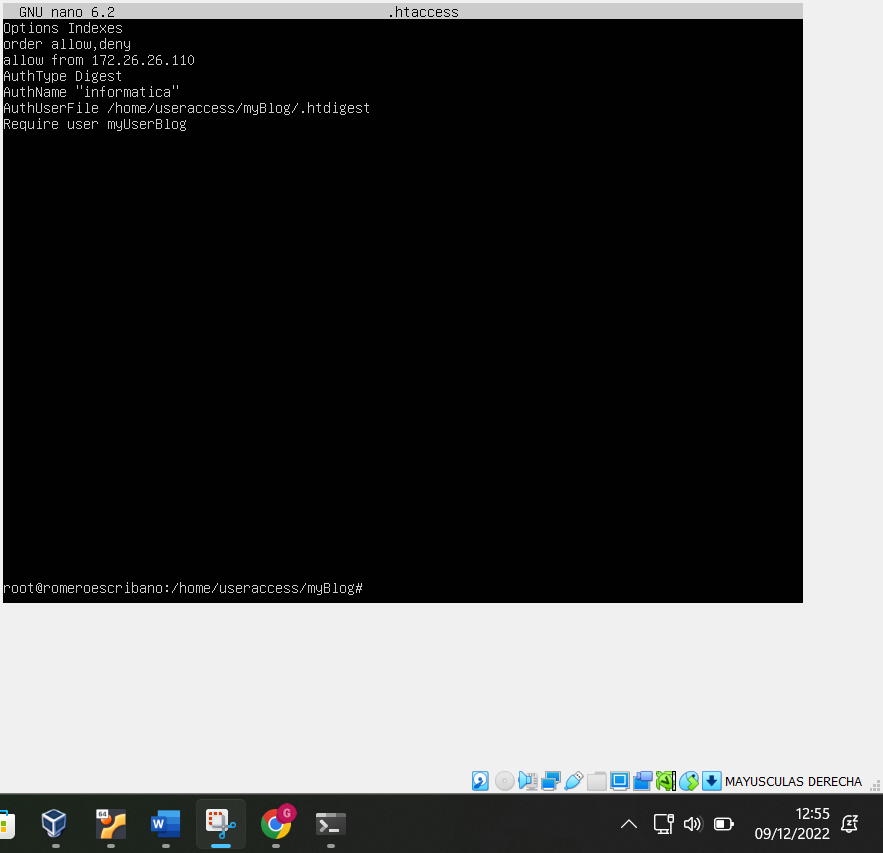
**PASO 6)** Para el acceso a los recursos de myBlog vamos a usar un tipo de autenticación Digest, por lo que dentro de este directorio vamos a crear el fichero **.htdigest** para el servidor informática y para el usuario myUserBlog (ver punto anterior acceso mediante Digest).



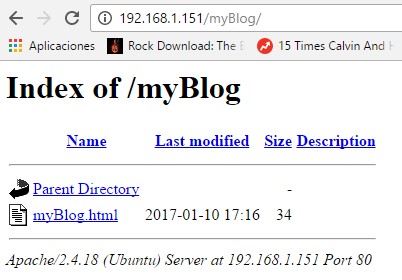
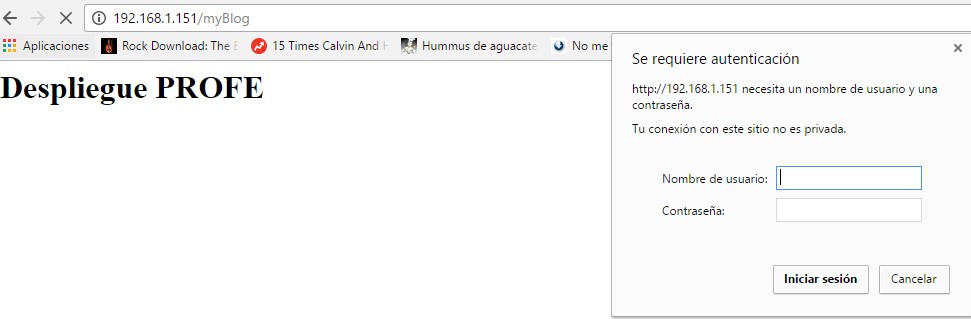
**PASO 7)** Ahora tendremos que crear el fichero **.htaccess** (también dentro de myBlog).

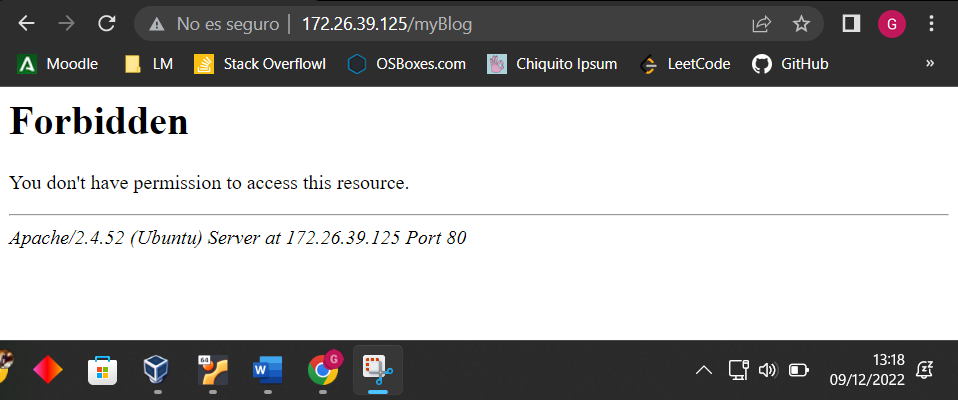
Dentro añadiremos las directivas necesarias para que se acceda solo desde nuestra máquina física (no es necesario poner las directivas Directory pues ya las incluimos en nuestro Alias para este directorio dentro de 000-default).





**PASO 8)** Vamos a acceder desde nuestra máquina física al recurso **myBlog** para ver que nos pide la autenticación y que podemos acceder al recurso.





**Toma una captura de los pasos 2,6,7 y 8.**

# E) Ficheros de registros (logs)

|  |
| --- |
| Los ficheros de registros nos ofrecen información de errores y accesos del servidor Apache.    En linux los ficheros de registro son:  Errores **/var/log/apache2/error.log**  Accesos **/var/log/apache2/access.log**    En windows:  Error **C:\Program Files\Apache Software Foundation\Apache2.2\log\error.log**  Accesos **C:\Program Files\Apache Software Foundation\Apache2.2\log\access.log**    Algunas de las directivas que tienen que ver con estos ficheros de registros son:  ErrorLog: Especifica los archivos donde se guardan los errores del servidor  LogLevel: Establece el nivel de detalle de los registros de mensajes de error  CustomLog: Identifica el archivo de registro de accesos y su formato (por defecto, combined)  LogFormat: Configura el formato para los archivos de registros del servidor Web (realmente depende de la configuración dada en CustomLog). |

**PASO 1)** En tu servidor Linux, consulta el fichero 000-default y responde a las siguientes preguntas:

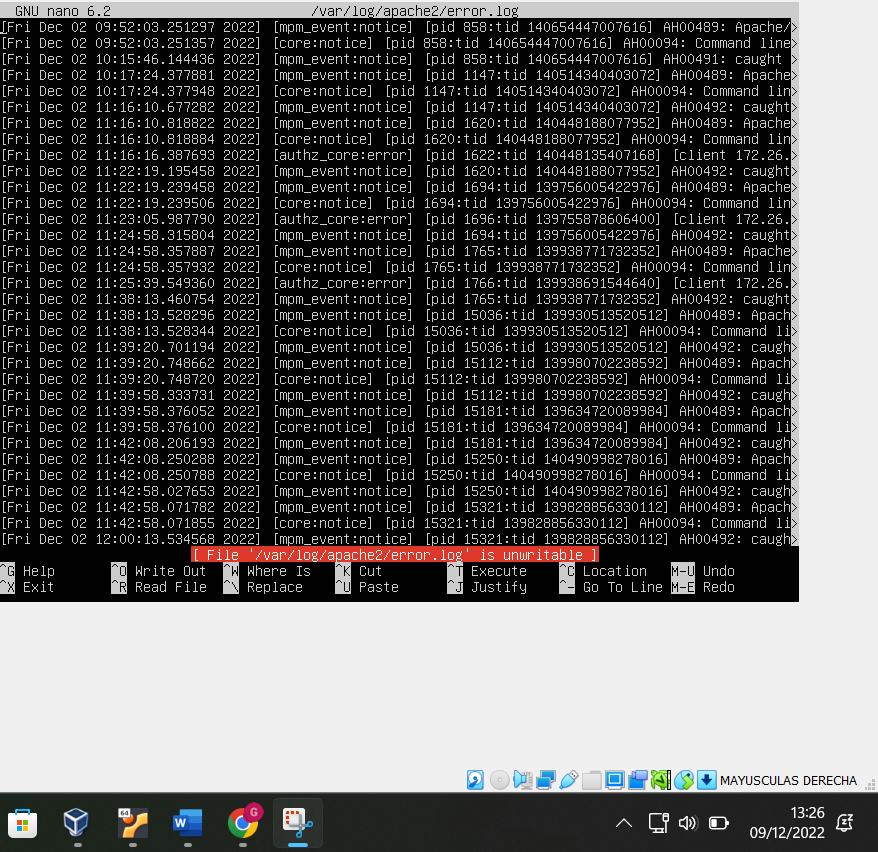
¿Qué directiva marca la ruta del archivo de los errores? ¿Cuál es el fichero de logs de errores? ¿Qué nivel de prioridad tiene?

La directiva que marca la ruta del archivo de errores es ErrorLog, cuyo archivo es error.log que tiene la prioridad más alta.

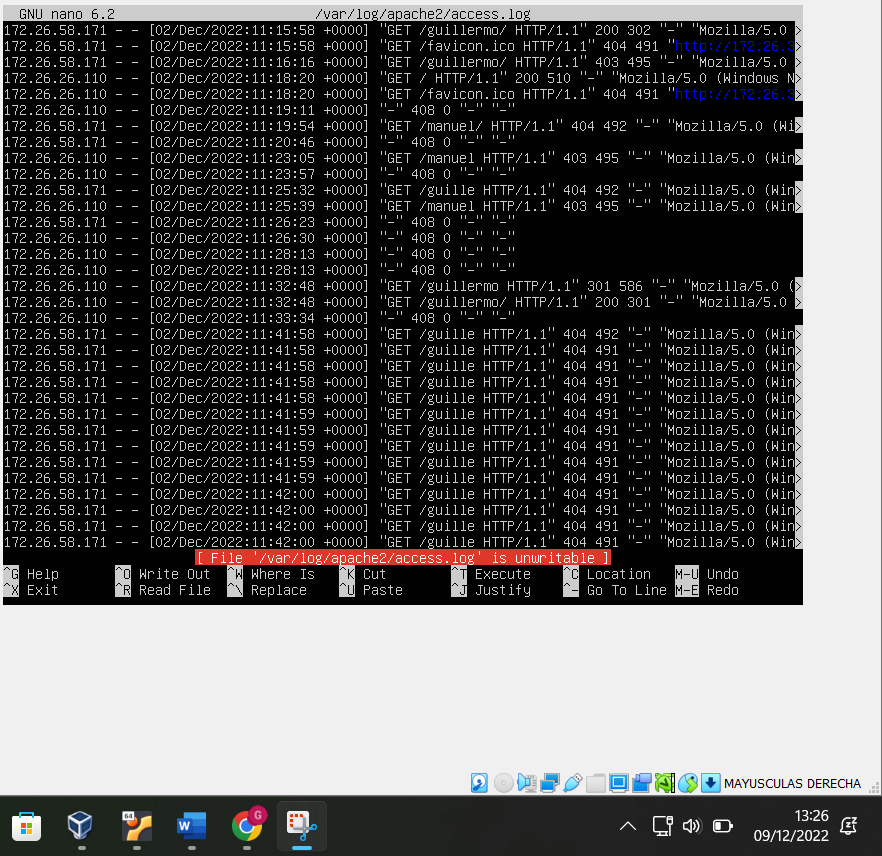
¿Qué directiva marca la ruta del archivo de los accesos? ¿Cuál es el fichero de logs de accesos?

La directiva que marca la ruta del archivo de accesos es CustomLog, cuyo archivo es Access.log

**PASO 2)** Consulta el log de errores



**PASO 3)** Consulta el log de accesos

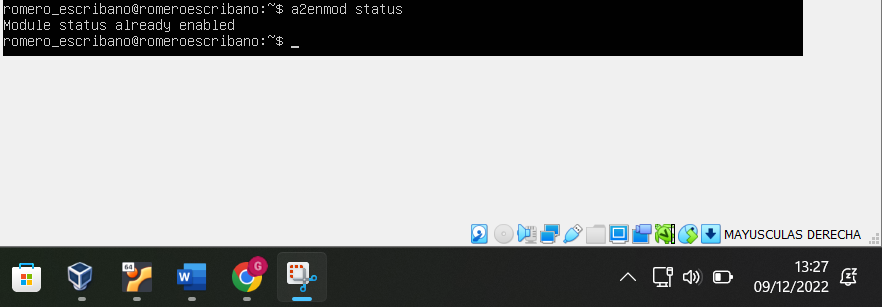


***Toma una captura de los pasos 2 y 3 (del final de cada fichero).***

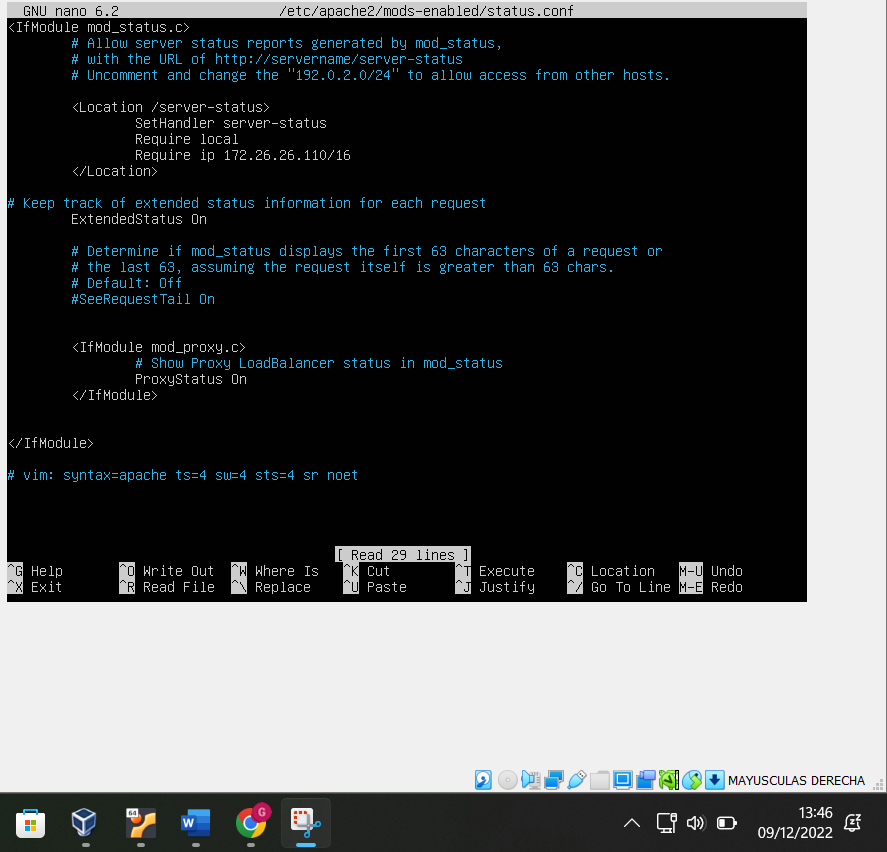
# F) Módulos status e info

|  |
| --- |
| status e info son módulos de monitorización. En concreto:    status permite monitorizar el rendimiento del servidor Apache (generando un HTML).  info proporciona una vista resumida de la configuración del servidor. |

**PASO 1)** En tu servidor Linux, habilita el módulo **status**.

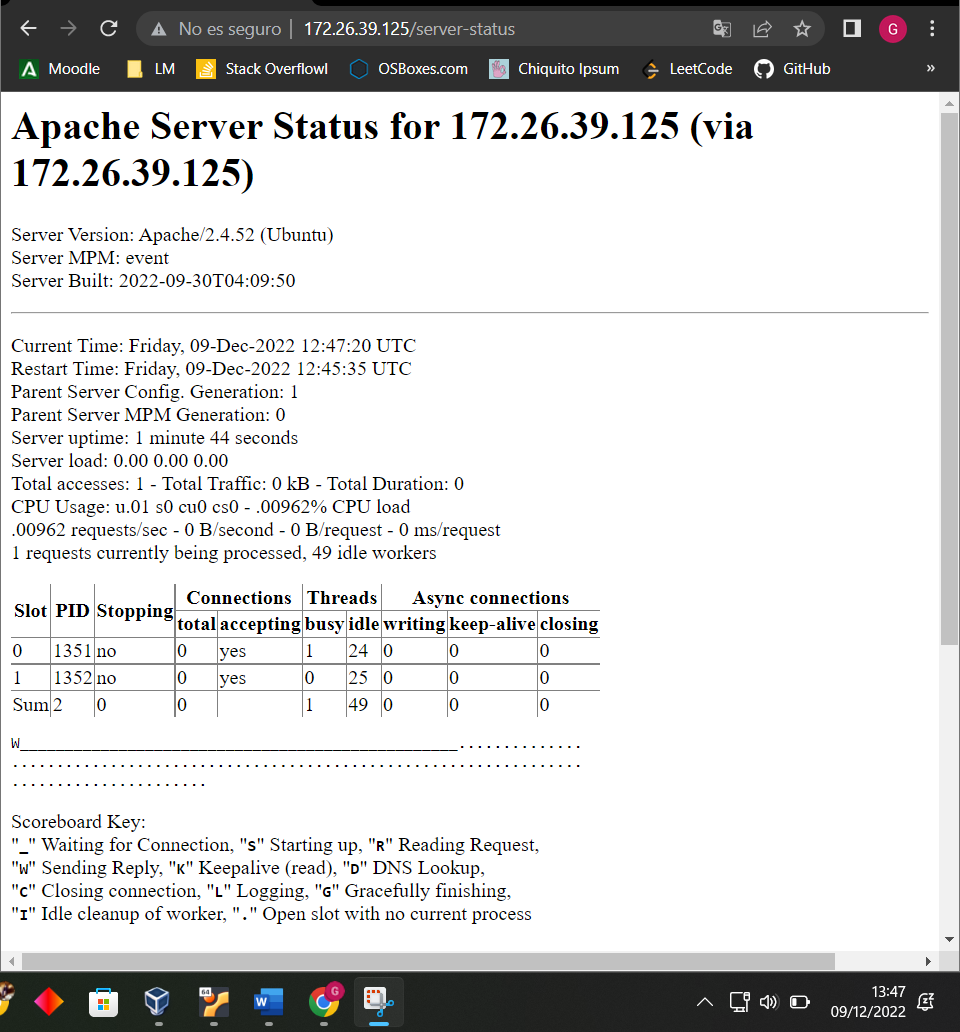


**PASO 2)** El fichero de configuración del módulo es **status.conf**, edita el fichero y habilita el acceso desde tu máquina física.



**PASO 3)** Reinicia el servidor para aplicar los cambios.

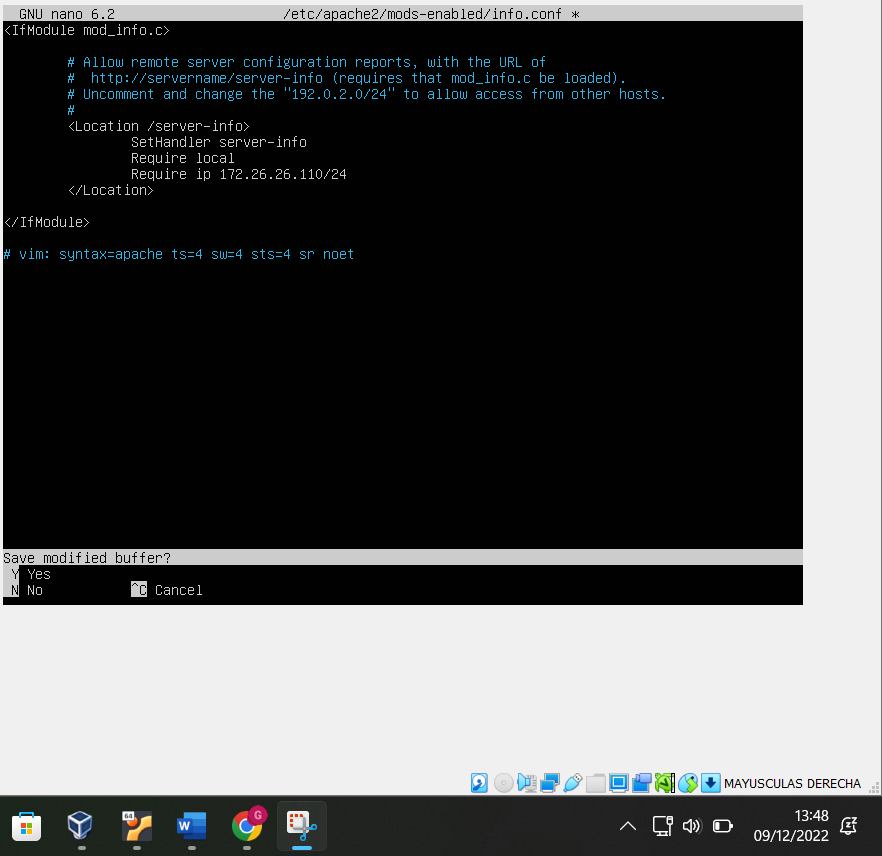
**PASO 4)** Desde tu máquina física conéctate al recurso server-status



**Toma una captura de los pasos 2 y 4.**

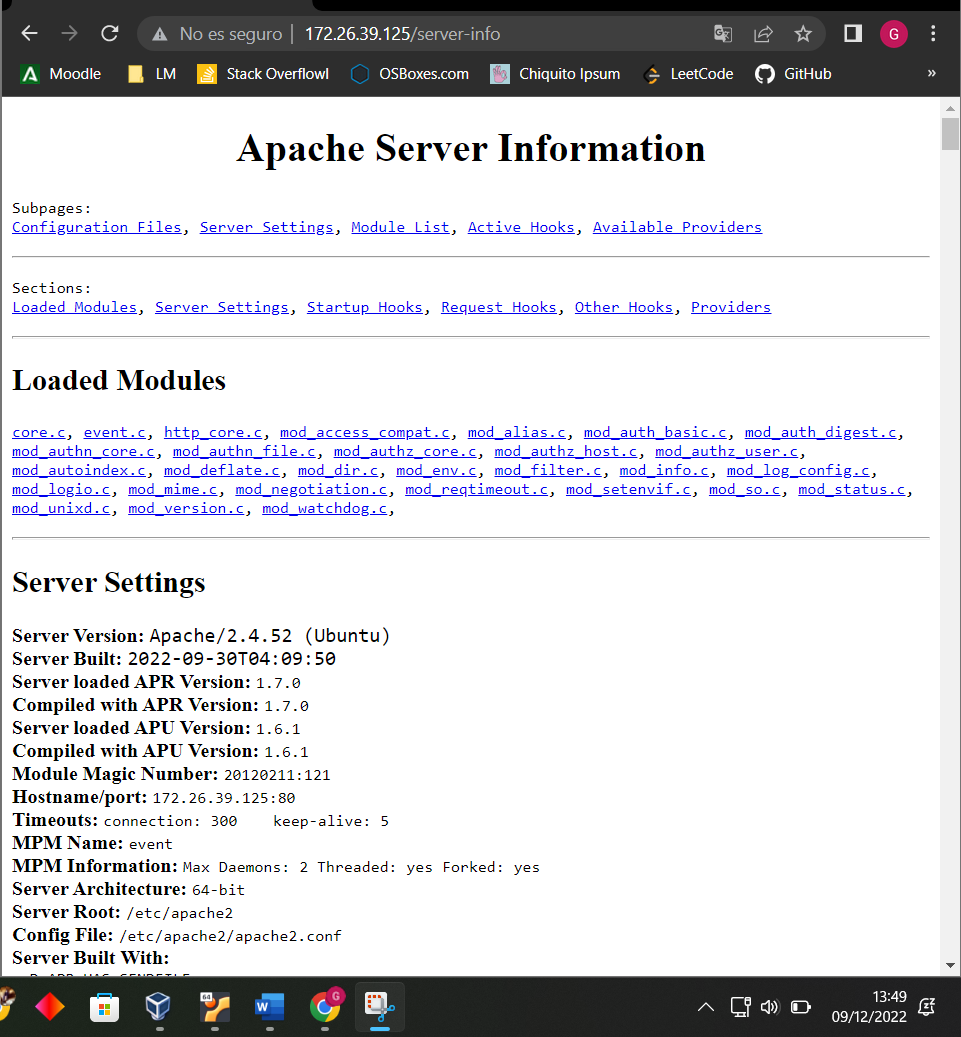
**PASO 5)** En tu servidor Linux, habilita el módulo **info**.

**PASO 6)** El fichero de configuración del módulo es **info.conf**, edita el fichero y habilita el acceso desde tu máquina física.



**PASO 7)** Reinicia el servidor para aplicar los cambios.

**PASO 8)** Desde tu máquina física conéctate al recurso server-info



Consulta el fichero server-info, ¿tienes cargado el módulo mod\_mime? ¿en caso que lo tuvieras, tiene el módulo cargada la configuración de caracteres UTF-32?



**Toma una captura de los pasos 6 y 8.**

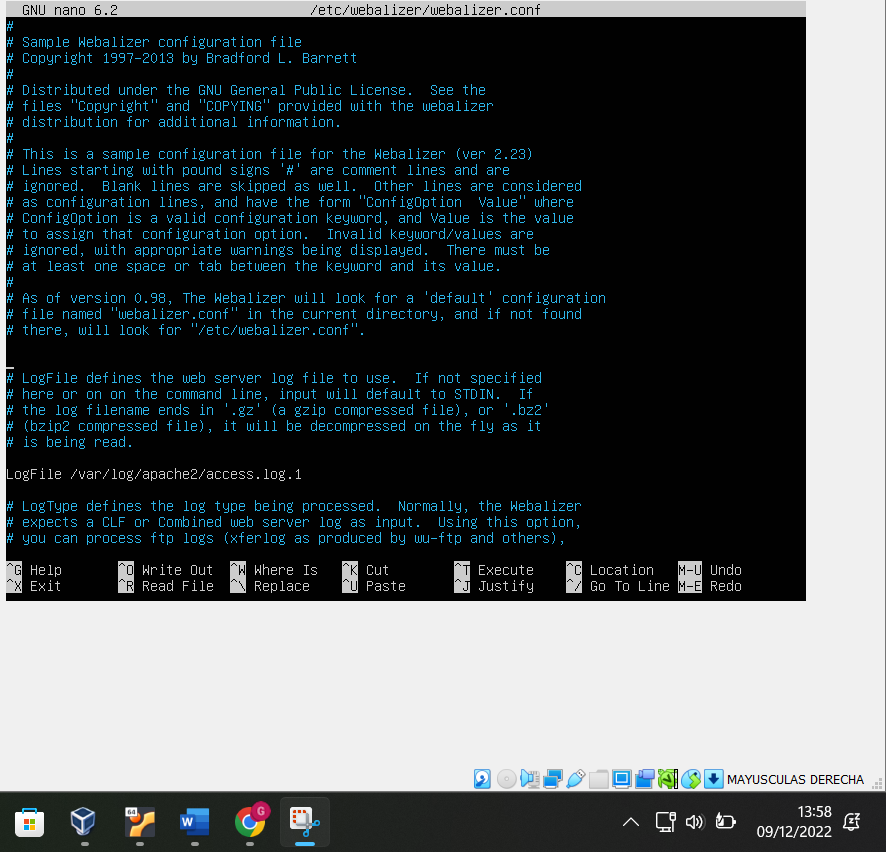
# G) Webalizer

Otra forma de monitorizar nuestro servidor apache es mediante aplicaciones analizadoras de logs, como es el caso de Webalizer. Esta aplicación se puede instalar en nuestro servidor y a partir de los archivos logs te crea unas estadíticas que puedes consultar en formato html.

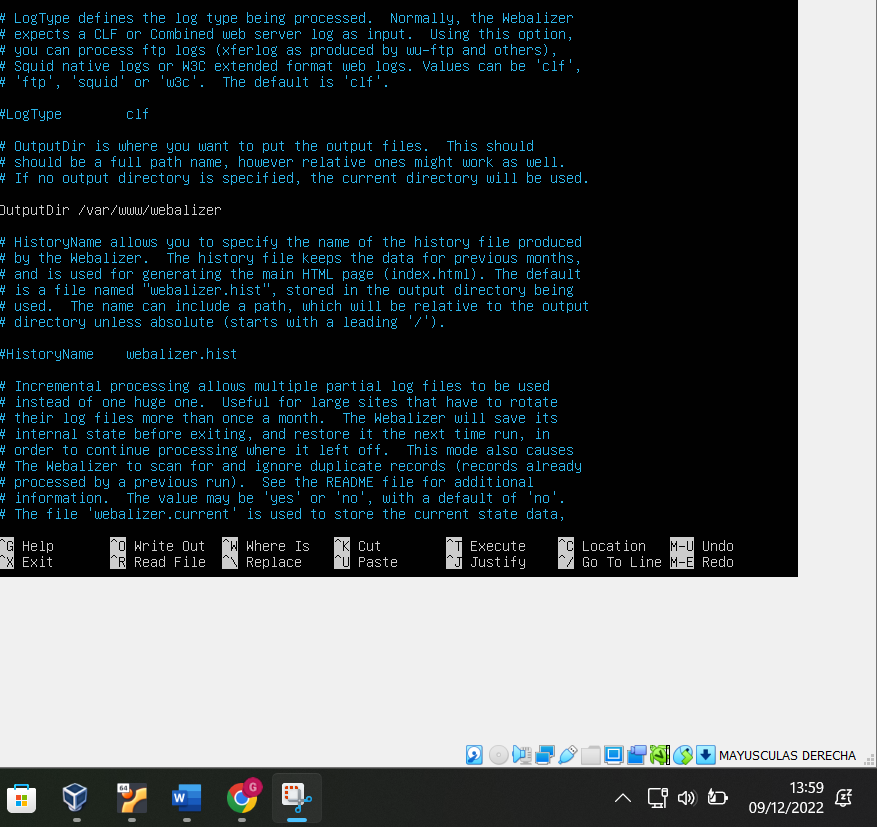
**PASO 1)** En tu servidor Linux, instala la aplicación Webalizer (usa apt-get install, pero antes actualiza el servidor Linux).

**PASO 2)** Una vez instalado se habrá creado un directorio para la aplicación en el **directorio /etc/.** Abre el fichero de configuración de webalizer, ¿de qué fichero log coge los datos para hacer las estadísticas? ¿es correcta la ruta y el nombre del fichero? Si no es así, modifícala.

Coge los datos de Access.log.1



**PASO 3)** La instalación también implica la creación del recurso que se servirá desde el navegador, ¿Dónde está este fichero? ¿Es correcta la ubicación para servirlo? Si no es así, muévelo a la ubicación correcta.

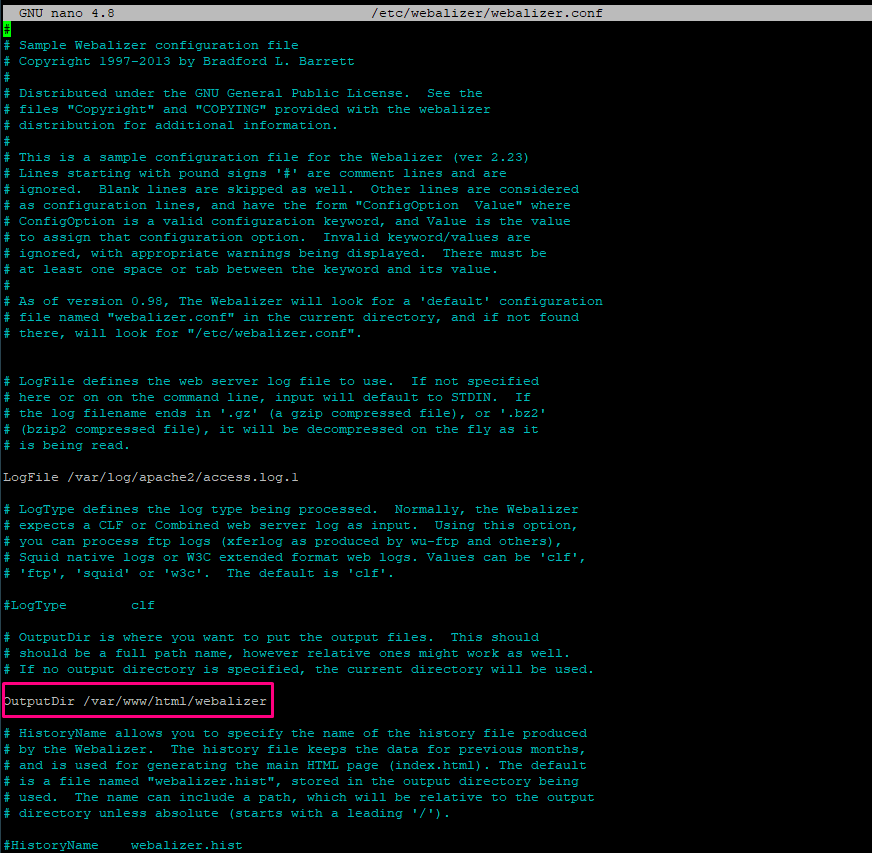


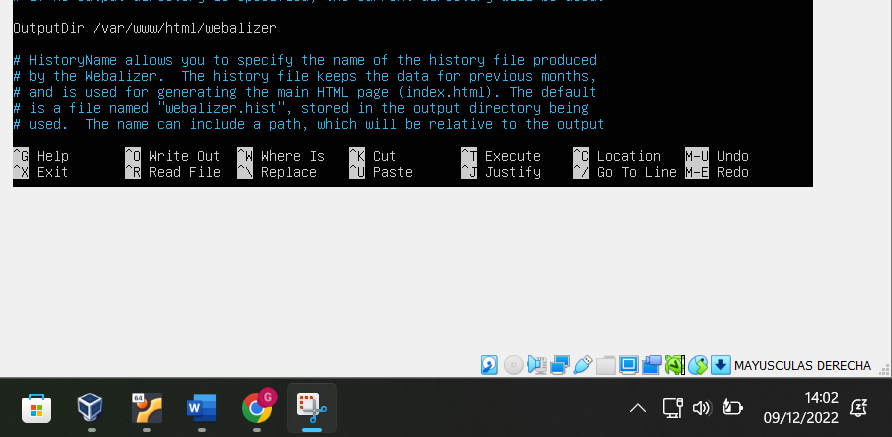
Podemos notar que una vez se descargó Webalizer **la ruta por defecto donde queda almacenado es** **/var/www/webalizer** y este parámetro **debemos moverlo a la ruta /var/www/html** para que la sincronización entre Apache y Webalizer sea correcta. Para realizar este proceso simplemente ejecutamos lo siguiente:

sudo mv /var/www/webalizer /var/www/html/

A continuación, vamos a **editar el archivo de configuración de Webalizer** introduce la siguiente instrucción:

sudo nano /etc/webalizer/webalizer.conf

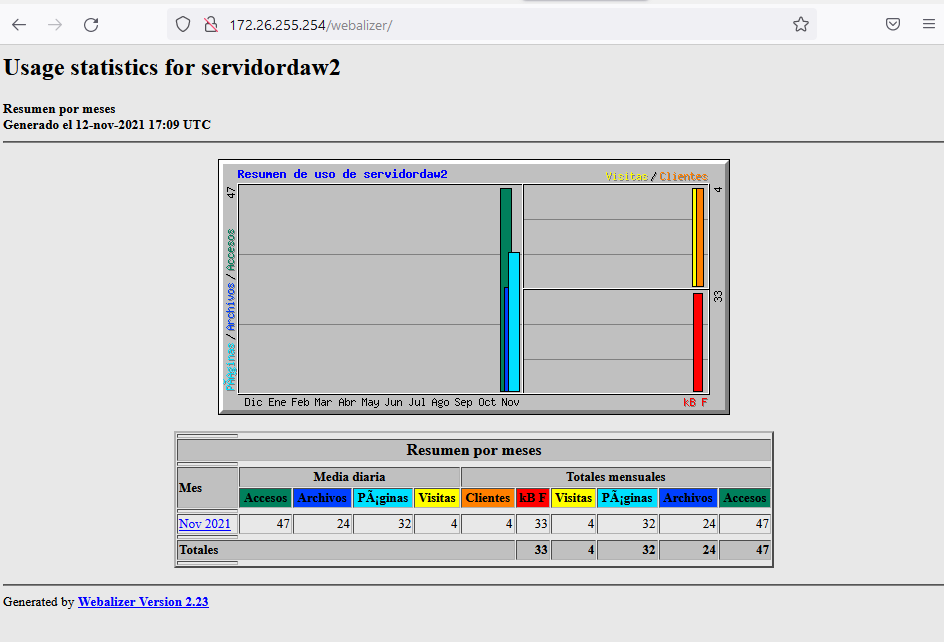


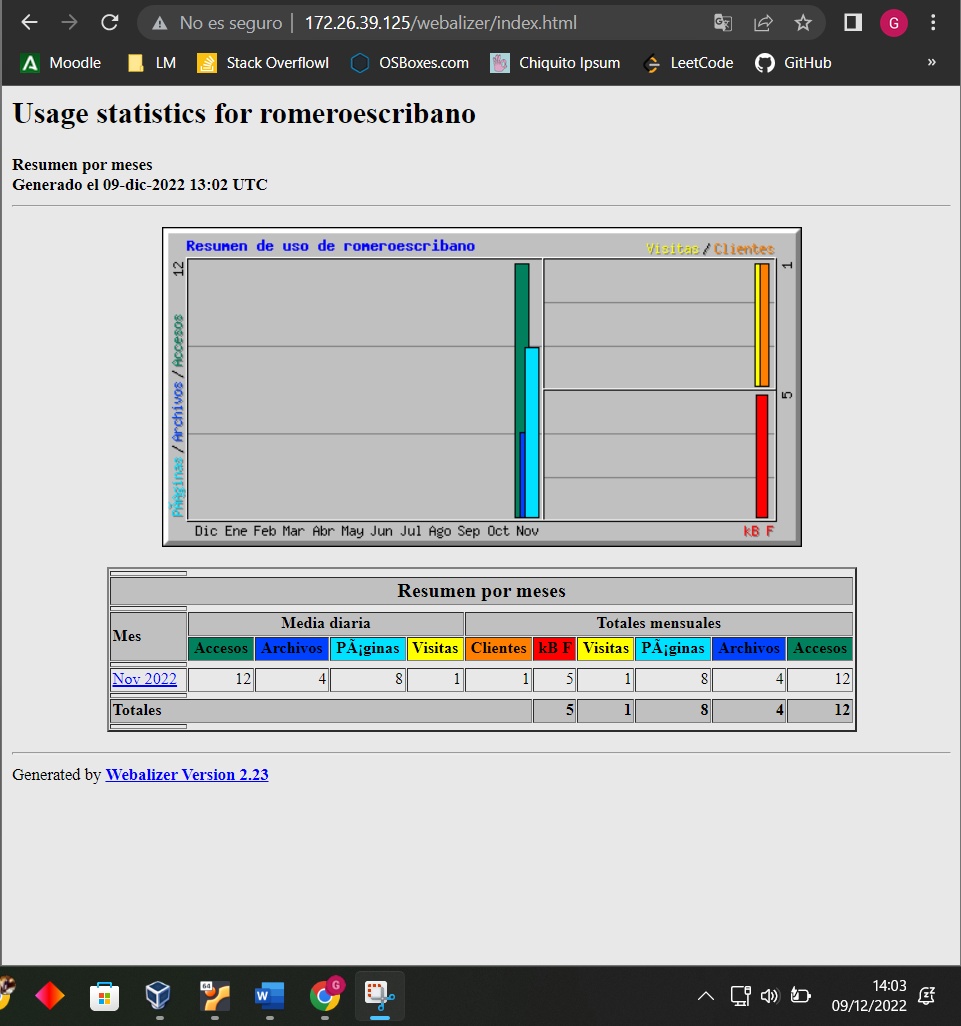


**PASO 4)** Lanza el programa (con permisos de administrador) para que lea el fichero de log correspondiente y genere el documento html con las estadísticas.

**sudo webalizer**

**PASO 5)** Accede al recurso /webalizer/ desde tu máquina física.



**Toma una captura de los pasos 2 y 5.**

# F) GitHub

Sube el documento al repositorio llamado Despliegue a la carpeta correspondiente.

**Toma capturas de pantalla de los comandos utilizados y del repositorio de la página Web.**