

Servidores Web

Apache

Instalación servidor web Apache

- Instalamos el servidor apache:

```
sudo apt-get update
```

```
sudo apt-get install apache2
```

- Para ver la versión de apache:

```
apache2 -v
```

- Terminada la instalación debemos comprobar que Apache está en ejecución, y que el servicio está a la escucha en el puerto 80.

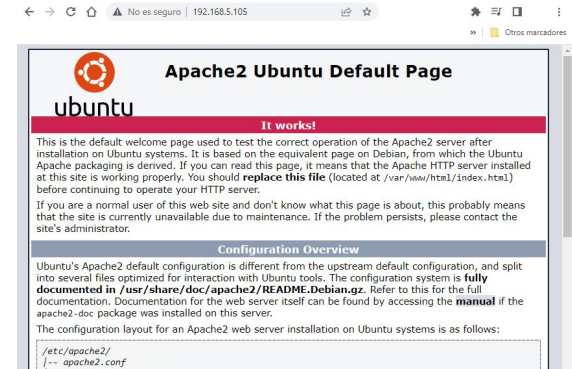
- `ps -ef | grep apache`

```
root@desktop:/var/www/ssl/htdocs# ps -ef | grep apache
www-data 3401      1    0  2022 ?        00:00:52 /usr/bin/htcacheclean -d 120 -p /var/cache/apache2/m
od_cache_disk -l 300M -n
root     121289     1    0  13:36 ?        00:00:00 /usr/sbin/apache2 -k start
www-data 121290  121289    0  13:36 ?        00:00:00 /usr/sbin/apache2 -k start
www-data 121291  121289    0  13:36 ?        00:00:00 /usr/sbin/apache2 -k start
root     122453  41365    0  14:24 pts/1    00:00:00 grep --color=auto apache
```

- `netstat -ltn`

```
root@desktop:/var/www/ssl/htdocs# netstat -ltn
Conexiones activas de Internet (solo servidores)
Proto Recib Envíad Dirección local      Dirección remota      Estado
tcp     0      0  127.0.0.53:53        0.0.0.0:*              ESCUCHAR
tcp     0      0  127.0.0.1:631        0.0.0.0:*              ESCUCHAR
tcp     0      0  0.0.0.0:10000        0.0.0.0:*              ESCUCHAR
tcp6    0      0  :::443              :::*                  ESCUCHAR
tcp6    0      0  :::80               :::*                  ESCUCHAR
tcp6    0      0  :::1:631            :::*                  ESCUCHAR
```

- Si accedemos desde un navegador `http://<dirección ip de la máquina virtual>` veremos la siguiente página



Instalación servidor web Apache

- Para detener, comenzar o recargar el servicio, se utiliza los siguientes comandos, dependiendo de la distribución empleada.

```
$ sudo /etc/init.d/apache2 {start | stop | restart | reload | status}
```

```
$ sudo service apache2 {start | stop | restart | reload | status}
```

```
$ sudo systemctl {start | stop | restart | reload | status} apache2
```

```
$ sudo systemctl {start | stop | restart | reload | status | fullstatus | configtest} apache2
```

- Al finalizar la instalación, se crean:
 - Los ficheros de configuración.
 - El usuario www-data y el grupo www-data
 - El directorio /var/www/html, cuyo propietario y grupo es root
- Los ficheros de configuración están en **/etc/apache2**
 - **/etc/apache2/apache2.conf**: Se determina el comportamiento general del servidor, el acceso a ciertos directorios. Aquí se incluyen otros ficheros de configuración.
 - **/etc/apache2/ports.conf**: Aquí se define los puertos o las interfaces de red junto a los puertos por los que escuchará el servidor.
 - **/etc/apache2/envvars**: Se definen variables de entorno para el comando apache2ctl

Instalación servidor web Apache

- Subdirectorios contenidos en /etc/apache2:
 - Directorios de **configuración de módulos**:
 - **mods_available**: Donde se encuentran los módulos disponibles, pero no activos o preparados para la carga. Aquí tenemos dos tipos de ficheros:
 - <nombre_modulo>.conf: Fichero de configuración
 - <nombre_modulo>.load: Fichero con las funcionalidades.
 - **mods_enabled**: Donde se encuentran los módulos preparados para la carga. En este directorio, tenemos enlaces a los ficheros *.conf y *.load de la carpeta mods_available.
 - Para activar un módulo se emplea el comando `a2enmod <nombre modulo>`, creando el enlace. `a2dismod <nombre modulo>` para desactivar.
 - Directorios de **configuración de servidores virtuales**:
 - **sites-available**: donde tenemos los ficheros de configuración de los host virtuales **disponibles**.
 - **sites-enabled**: donde tenemos los ficheros de configuración de los host virtuales que se están **utilizando actualmente**. Es un enlace simbólico al fichero de configuración de sites-available.
 - Para activarlos, se utiliza `a2ensite <nombre del directorio virtual>`, el cual creará un enlace en sites-enabled.

Instalación servidor web Apache

- Subdirectorios contenidos en /etc/apache2:
 - Directorios de **configuración genérica**:
 - **conf_available**: Archivos de configuraciones relacionadas con la internacionalización de los mensajes de error, el conjunto de caracteres. Además se incluyen los archivos de configuración de las aplicaciones web que se instalan para ejecutarse sobre Apache.
 - **conf-enabled**: Contiene los enlaces simbólicos a los ficheros del directorio conf-available.
 - Para activarlos, se utiliza `a2enconf <archivo_configuración>`, el cual creará un enlace en conf-enabled. Para desactivar `a2disconf <archivo_configuración>`.
 - Todos los cambios en los archivos de configuración de Apache, necesita REINICIAR EL SERVICIO.

```
root@servidor:/etc/apache2# ls
apache2.conf  conf-enabled  magic        mods-enabled  sites-available
conf-available  envvars      mods-available  ports.conf    sites-enabled
```

Servidor virtual por defecto

- Diferencia entre el *servidor principal* y los *servidores virtuales*. Si el servidor no tiene SERVIDORES VIRTUALES, será el servidor principal, quien se encargará de atender las peticiones.
- El servidor virtual por defecto se encuentra en `/etc/apache2/sites-available/000-default.conf`
- El archivo tiene las siguientes directivas
 - **<VirtualHost *:80>**. El servidor virtual viene definido por `<VirtualHost IP:Puerto>`. Al ser por defecto para todas las IP, establecemos *. El puerto 80 es para escuchar el protocolo HTTP.
 - **DocumentRoot**: Indica cuál es el directorio raíz del servidor. Por defecto es: `/var/www/html`
 - **ErrorLog**: Indica dónde se almacenarán los informes de error.
 - **CustomLog**: Indica dónde se almacenarán los acceso al servidor.

```
<VirtualHost *:80>
# The ServerName directive sets the request scheme
# the server uses to identify itself. This is used
# for redirection URLs. In the context of virtual hosts,
# it specifies what hostname must appear in the request
# to match this virtual host. For the default virtual host,
# the value is not decisive as it is used as a last resort
# fallback. However, you must set it for any further virtual
# hosts.
#ServerName www.example.com

ServerAdmin webmaster@localhost
DocumentRoot /var/www/html

# Available loglevels: trace8, ..., trace1, debug, info,
# 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, you should
# have a line like this in each virtual host to allow us to
# control whether we enable the option or not globally.
# include a line for only one particular virtual host to
# allow us to control whether we enable the option or not
# following line enables the CGI configuration for this host
# after it has been globally disabled with "a2disablecgis"
#Include conf-available/serve-cgi-bin.conf
</VirtualHost>
```

Directivas

- Las directivas permiten la configuración del servidor.
- Las directivas <VirtualHost>, <Directory> y <Files> actúan como contenedores de otras directivas permitiendo configurar ámbitos concretos.
- Las directivas se pueden clasificar en tres tipos.
 - **Directivas globales:** Son los aspectos globales del servidor.
 - Número máximo de clientes simultáneos
 - Tiempo de cancelación de conexión por falta de respuesta [Timeout]
 - Directorios de los archivos de configuración [ServerRoot]
 - Conexiones persistentes.
 - **Directivas del servidor principal:** Define el comportamiento del servidor principal. Este es el comportamiento por defecto de todos los servidores virtuales.
 - Directivas de los usuarios
 - Directivas de los grupos
 - Incluyendo los ficheros de configuración [IncludeOptional mods-enabled/*.load, IncludeOptional mods-enabled/*.conf, Include ports.conf]
 - **Directivas de configuración de los servidores virtuales:** Define el comportamiento de cada uno de los servidores virtuales.

Directivas - DirectoryIndex

- DirectoryIndex se utiliza para indicar qué archivo se va a servir cuando no se especifica ninguno.
- Si no se establece ninguno, se recogerá el que haya configurado en el servidor principal.
- En el ejemplo, si se escribe <http://localhost> Mostrará el contenido que hay en daw.html

```
<VirtualHost *:80>
```

```
    ServerAdmin webmaster@localhost
```

```
    DocumentRoot /var/www/html
```

```
    DirectoryIndex daw.html
```

```
    ErrorLog ${APACHE_LOG_DIR}/error.log
```

```
    CustomLog ${APACHE_LOG_DIR}/access.log combined
```

```
</VirtualHost>
```

- Si en lugar de un fichero, queremos que busque varios, por ejemplo, un fichero html y un fichero php, se puede hacer, especificando ambos ficheros separados por un espacio en blanco. En el ejemplo, primero buscará el fichero index.php y si no se encuentra buscará el fichero index.html.

```
DirectoryIndex index.php index.html
```

- Si la directiva DirectoryIndex no aparece, y tampoco la directiva Option Index, al escribirse <http://localhost> nos devolverá un error 404.

Directivas - Directory y Options Indexes.

- La directiva <Directory> permite determinar cómo Apache sirve el contenido de un directorio.
- En la directiva <Directory> se establece la ruta del directorio sobre el que se quiere aplicar esta directiva, como por ejemplo <Directory /var/www/html>
- La directiva Options permite tomar varios parámetros [entre otros]:
 - **Indexes** presenta en el navegador un listado de los archivos y directorios contenidos en el directorio. Solo mostrará los archivos y directorios, cuando no haya una directiva DirectoryIndex o no se encuentre el fichero indicado en DirectoryIndex.
 - **FollowSymLinks** El servidor seguirá los enlaces simbólicos en este directorio, eso *no* cambia la ruta usada para encontrar equivalencias en las secciones
 - **All** Todas las opciones excepto MultiViews. Este es el valor por defecto.
- Aplicando un guión delante de la opción, deshabilitamos esta opción
 - Si queremos deshabilitar el listado de un directorio que actúa por herencia de otro superior podemos usar la sintaxis **Options -Indexes** en la configuración de dicho directorio.

```
<Directory /var/www/html/datos>  
    DirectoryIndex datos1.html  
    Options -Indexes +FollowSymLinks  
</Directory>
```

```
<Directory /var/www/html/datos>  
    DirectoryIndex datos1.html  
    Options Indexes  
</Directory>
```

Directivas - Logs

- En las directivas podemos configurar los ficheros de diario.
- Las directivas son:
 - **ErrorLog** permite indicar el archivo de logs de errores.
 - **LogLevel** indica el nivel de prioridad. (Solo los errores, solo las advertencias, etc.). Por defecto está definido todos los mensajes de advertencia y los de nivel superior, con el valor **warn**. Los valores que puede tomar son: debug, info, notice, warn, error, crit, alert y emerg.
 - **CustomLog** permite indicar cuál es el archivo que almacenará el registro con los logs de acceso.
 - **LogFormat** indica el formato del archivo de logs. Si no se especifica ninguno con LogFormat, se usará el formato especificado en apache2.conf para el servidor principal.

- Ejemplo:

ErrorLog \${APACHE_LOG_DIR}/error.log

CustomLog \${APACHE_LOG_DIR}/access.log combined

- APACHE_LOG_DIR es la variable establecida en /etc/apache2/envvars, su valor por defecto es:
/var/log/apache2

Directivas - Código de error

- **ErrorDocument** permite configurar el texto o el documento que se presentará cuando se produzca un error en el servidor.
- Ejemplo

ErrorDocument 404 "Página no encontrada en el servidor"

ErrorDocument 404

- El primero mostrará el mensaje "Página no encontrada en el servidor" si hay un error al no encontrar el recurso.
- El segundo mostrará el documento error404.html si hay un error al encontrar un recurso.



Directorios Virtuales

- Los directorios virtuales son directorios accesibles mediante el sitio web, pero que están fuera del directorio raíz (DocumentRoot) del servidor.
- Se pueden crear de dos formas:
 - Usando la directiva **alias**. En este caso se define un alias que se asocia a un directorio que será mostrado en el navegador.

Alias /wiki /home/pepe/wiki

```
<Directory /home/pepe/wiki>
```

```
    DirectoryIndex index.html
```

```
    Require all granted
```

```
</Directory>
```

Al escribir en el navegador <http://localhost/wiki> se mostrará el contenido de /home/pepe/wiki

- Con la directiva **Options FollowSymLinks** se habilita la posibilidad de que si en el directorio de un sitio web existen enlaces simbólicos (soft links) a otros directorios. Para ello hay que crear los enlaces previamente con **ln -s**

```
$ sudo ln -s /home/pepe/blog /var/www/html/blog
```

```
<Directory /var/www/html/blog>
```

```
    DirectoryIndex index.html
```

```
    Option FollowSymLinks
```

```
</Directory>
```

Al escribir en el navegador <http://localhost/blog> se mostrará el contenido de /home/pepe/blog

Directorios Virtuales

- Los directorios virtuales son directorios accesibles mediante el sitio web, pero que están fuera del directorio raíz (DocumentRoot) del servidor.
- Se pueden crear de dos formas:
 - Usando la directiva **alias**. En este caso se define un alias que se asocia a un directorio que será mostrado en el navegador.

Alias /wiki /home/pepe/wiki

```
<Directory /home/pepe/wiki>
```

```
    DirectoryIndex index.html
```

```
    Require all granted
```

```
</Directory>
```

Al escribir en el navegador <http://localhost/wiki> se mostrará el contenido de /home/pepe/wiki

- Con la directiva **Options FollowSymLinks** se habilita la posibilidad de que si en el directorio de un sitio web existen enlaces simbólicos (soft links) a otros directorios. Para ello hay que crear los enlaces previamente con **ln -s**

```
$ sudo ln -s /home/pepe/blog /var/www/html/blog
```

```
<Directory /var/www/html/blog>
```

```
    DirectoryIndex index.html
```

```
    Option FollowSymLinks
```

```
</Directory>
```

Al escribir en el navegador <http://localhost/blog> se mostrará el contenido de /home/pepe/blog

Módulos

- Los **módulos** son una manera de **agrupar y modularizar ciertos funcionamientos** para el servidor. De esta forma no necesitamos que estén todos los elementos activos.
- Para conocer los módulos cargado estáticamente se utiliza `$ apache2ctl -l`
- Los módulos de apache permiten **carga dinámica**
- La extensión de los módulos de Apache es .so
- En el **directorio /etc/apache2** tenemos dos directorios:
 - **mod_available**: Donde se encuentran los módulos disponibles, pero no activos o preparados para la carga. Aquí tenemos dos tipos de ficheros:
 - `<nombre_modulo>.conf`: Fichero de configuración
 - `<nombre_modulo>.load`: Fichero con las funcionalidades.
 - **mod_enabled**: Donde se encuentran los módulos preparados para la carga. En este directorio, tenemos enlaces a los ficheros *.conf y *.load de la carpeta mod_available.
- Si queremos activar la utilización de un módulo en Apache2, se utiliza el comando **a2enmod**, este comando creará un enlace en mod_enabled.
`sudo a2enmod proxy_html`
- Para utilizarlo solo hay que reiniciar el servidor: `systemctl restart apache2`
- Para dejar de utilizar el módulo solo hay que eliminar el enlace en mods-enabled

Instalación de un Módulo en Apache

- El módulo **userdir** permite que cualquier usuario cree su propio sitio web en un directorio dentro de su home.
- En el fichero **/etc/apache2/mods-available/userdir.conf** observamos que se habilitan el uso de directorios personales en la carpeta `public_html` [Root no está incluido aquí]

Editando archivo de configuración: `/etc/apache2/mods-available/userdir.conf`

```
1 <IfModule mod_userdir.c>
2   UserDir public_html
3   UserDir disabled root
4
5   <Directory /home/*/public_html>
6       AllowOverride FileInfo AuthConfig Limit Indexes
7       Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
8       Require method GET POST OPTIONS
9   </Directory>
10 </IfModule>
11
12 # vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

- **public_html** indica donde estarán los archivos HTML de los usuarios es *public_html*
- La sección **<Directory>** está indicando que de los home de todos los usuarios, podrán ser listados sus contenidos [Options Indexes]



Salvar



Guardar y cerrar

Instalación de un Módulo en Apache

- Para comprobar el módulo, primero tenemos que activarlo.

```
$ sudo a2enmod userdir
```

[NOTA: Debido a que este módulo necesita acceso como ejecución al directorio del usuario ejecuta:

```
$ sudo cd /home
```

```
$ sudo chmod o+x /home/<usuario>
```

- En nuestro home (~) creamos una carpeta llamada public_html, y creamos un html

```
<html>
```

```
<head></head>
```

```
<body>
```

```
<center>
```

```
<h1> Bien
```



No es seguro

192.168.5.103/~rafa/

```
<h2> Ser
```

```
</center>
```

```
</body>
```

```
</html>
```

Bienvenido al servidor USERDIR Apache

Servicios en Red

Control de acceso

- El acceso a los recursos [por dirección IP o nombre de dominio] es controlado mediante la declaración de distintas entradas que se pueden aplicar a 3 niveles diferentes:
 - **<Directory “directorio”>** implica que las órdenes especificadas en el control de acceso se aplican al directorio especificado y sus subdirectorios.

```
<Directory "/var/www">
```

```
...
```

```
</Directory>
```

- **<Location “url”>** implica que las órdenes especificadas en el control de acceso se aplican a la url especificada

```
<Location "URL">
```

```
...
```

```
</Location>
```

- **<File “fichero”>** implica que las órdenes especificadas en el control de acceso se aplican exclusivamente al fichero especificado.

```
<Files "topami.html">
```

```
...
```

```
</Files>
```

- Todas estas entradas [Directory, Location, File] tienen equivalentes con Match. [DirectoryMatch, LocationMatch, FileMatch], donde utilizamos expresiones regulares en el nombre, de forma que se pueda hacer referencia a múltiples directorios, url, o ficheros. En el ejemplo se aplica a todos los ficheros que terminan en .ht, en cualquier ubicación de Apache

```
<FilesMatch "^\.ht">
```

Control de acceso

- Las directivas se ejecutan dependiendo de su funcionamiento, así primero se aplica Directory, a continuación DirectoryMatch, después File, FileMatch, Location, LocationMatch.
- En el siguiente ejemplo, Location permite el acceso a todos, y Directory lo prohíbe, luego al evaluarse la entrada <Location> en último lugar, la restricción de Directory queda sin efecto.

```
<Location "/">  
    Require all granted  
</Location>  
<Directory "/var/www/html">  
    Require all denied  
</Directory>
```

- Esta directiva se utiliza para controlar el acceso por dirección IP o nombre de dominio
 - Require all [denied | granted] Permite acceder o denegar acceso a todos
 - Require [not] host {nombre | dominio } Permite o no acceso al host por nombre o dominio
 - Require [not] ip {ip | subred} Permite o no acceso si la dirección IP del cliente o la red/subred a la que pertenece cumple el requisito. [Require ip 192.168.210.0/24]
 - <RequireAll>, <RequireAny>, <RequireNone> son directivas contenedoras, lo que hace que todo lo que esté dentro, se aplique a todo.
 - RequireAll son aplicadas todas las directivas
 - RequireAny indica que al menos una debe de cumplirse
 - RequireNone indica que ninguna debe de cumplirse

```
<RequireAny>  
    Require local  
    Require ip 192.168.108.0/24  
</RequireAny>
```

Si es local o está dentro de la subred, se permitirá el acceso.

Autenticación

- Cuando el servidor Apache recibe la petición de una página web, antes de ofrecerla, verifica que dicha petición está autorizada. Realiza las siguientes comprobaciones:
 - **1. Autenticación:** Primero comprueba la identidad del usuario. El usuario ha de proporcionar su nombre y contraseña [estos datos se llaman **credenciales**]. Si estos datos coinciden con los almacenados en su base de datos, se considera la autenticación completada. Utilizada con la directiva **AuthType**.
 - **Básico:** El usuario se introduce en el navegador web y se envía sin cifrar al servidor
 - **Digest:** El usuario se introduce en el navegador web y se mandan cifrados al servidor.
 - **2. Autorización:** El servidor debe comprobar si el usuario validado tiene acceso a la información que solicita. El servidor dispone una lista de usuarios que pueden acceder a los recursos que ofrece. En Apache, se gestiona mediante directivas **<Directory>** o en **.htaccess**
 - **3. Control de acceso:** Establece y controla las máquinas que tienen acceso a un recurso. Se gestiona mediante **<Directory>**, **<Files>** y **<Location>** o a través de **.htaccess**

Autenticación

- Para utilizar `auth_basic` o `auth_digest` debemos de activar los módulos: `auth_basic` o `auth_digest` con `a2enmod`.
- Las directivas que se utilizan son las siguientes:
 - **AuthName:** Es una cadena de texto que se mostrará en la página de acceso para que el usuario sepa que ha de identificarse
 - **AuthUserFile:** Es el fichero que contiene los nombres de usuarios, sus contraseñas, solo accesible por Apache. Se utiliza el comando `/usr/sbin/htpasswd`
 - `htpasswd -c [-p | -d | -m | -s] <fichero> <usuario>` ; crea el archivo y le añade los credenciales del usuario
 - `htpasswd [-p | -d | -m | -s] <fichero> <usuario>` ; añade al archivo los credenciales del usuario
 - `htpasswd -D <fichero> <usuario>` ; elimina del archivo los credenciales del usuario
 - **AuthGroupFile:** Es el fichero que contiene los nombres de los grupos de usuarios y los usuarios que conforman el grupo. Tiene la siguiente sintaxis.
 - `<nombre del grupo>: <usuario1> <usuario2> ... <usuarioN>`

Autenticación

- Ejemplo: Creamos dos usuarios, jose y pepe y permitimos que solo pueda acceder pepe.
 - Creamos jose y pepe con htpasswd, almacenamos el contenido en /etc/apache2/passwd
 - `htpasswd -c /etc/apache2/passwd jose`
 - `htpasswd /etc/apache2/passwd pepe`
 - En el fichero /etc/apache2/sites-available/000-default.conf permitimos el acceso privado al directorio /var/www/html/basic a solamente pepe

```
<Directory /var/www/html/basic>
    Options Indexes FollowSymLinks
    AuthType Basic
    AuthName "Acceso privado"
    AuthUserFile "/etc/apache2/passwd"
    <RequireALL>
        Require user pepe
    </RequireALL>
</Directory>
```

Autenticación Con LDAP

- **LDAP** (Lightweight Directory Access Protocol) es un protocolo de acceso a directorios utilizado para gestionar y autenticar información de usuarios y recursos en una red.
- Un "**directorio**" en este contexto es una **base de datos optimizada** para lectura y diseñada para almacenar información jerárquica como nombres, direcciones, datos de contacto, credenciales de usuarios y otros recursos, en una estructura tipo árbol.
- Características:
 - **Protocolo ligero:** pensado específicamente para facilitar búsquedas y accesos a grandes volúmenes de datos.
 - **Estructura jerárquica:** la información en LDAP se organiza en una estructura tipo árbol que facilita la categorización y búsqueda de información.
 - **Autenticación centralizada:** usado para autenticación centralizada en entornos de red, donde un solo servidor LDAP puede gestionar las credenciales y permisos de acceso de múltiples aplicaciones y sistemas, permitiendo un inicio de sesión único.
 - **Escalabilidad y flexibilidad:** lo que lo hace adecuado para entornos de pequeñas empresas hasta grandes corporaciones
- En una empresa, un servidor LDAP puede almacenar información sobre los empleados (nombres, departamentos, emails, roles), y al integrarlo con aplicaciones, se podría gestionar el acceso de cada empleado a sistemas y recursos de la empresa usando sus credenciales LDAP.

LDPA integrado con Apache2

- Para utilizarlo con apache2 hay que instalar los módulos: **ldap authnz_ldap**
sudo a2enmod ldap authnz_ldap
- Hay que configurar el sitio, en el directorio por defecto con las siguientes directivas.

```
<Directory /var/www/html/protegido>
```

```
AuthType Basic
```

```
AuthName "Acceso privado"
```

```
AuthBasicProvider ldap
```

```
AuthLDAPURL "ldap://servidor_ldap:389/ou=users,dc=example,dc=com?uid?sub"
```

```
AuthLDAPBindDN "cn=usuario_ldap,dc=example,dc=com"
```

```
AuthLDAPBindPassword "contraseña_admin"
```

```
Require valid-user
```

```
</Directory>
```

- Donde:
 - **AuthBasicProvider:** Selecciona ldap como proveedor de autenticación.
 - **AuthLDAPURL:** URL del servidor LDAP. Incluye la IP o el nombre de dominio del servidor, el puerto (389 o 636 para SSL), y la base de búsqueda en el directorio LDAP.
 - **AuthLDAPBindDN** y **AuthLDAPBindPassword:** Credenciales para enlazar el servidor LDAP y hacer consultas (puede ser opcional si el servidor permite búsquedas anónimas).
 - **Require valid-user:** Permite el acceso solo a usuarios válidos en el servidor LDAP.

Archivos .htaccess

- Los archivos .htaccess permiten a los usuarios que no tienen permisos, modificar la configuración, y así poder ejercer algún control sobre el comportamiento de su parte en el servidor Apache2
- En el archivo /etc/apache2/apache2.conf podemos definir el nombre del archivo, y podríamos cambiarlo para que en lugar de .htaccess sea por ejemplo .htconfiguracion

AccessFileName .htconfiguracion

- El archivo .htaccess se sitúa en el directorio al que tiene que afectar, y las modificaciones que se realicen, no requieren reiniciar el servidor web, ya que en cada petición se lee el archivo .htaccess.
- El servidor tiene que tener la **configuración AllowOverride que permite la sobreescritura**.

```
<Directory />
```


```
****
```

```
AllowOverride All
```

```
***
```

```
</Directory>
```

Al activarse AllowOverride, Apache2
buscará los ficheros .htaccess en cada
directorio de acceso



Archivos .htaccess

- Los valores que puede tomar la directiva AllowOverride son:
 - **All**: Permite todas las directivas. Con esta opción podemos utilizar todas las directivas.
 - **None**: No permite ninguna directiva.
 - **AuthConfig**: Permite directivas de autenticación de usuarios.
 - **FileInfo**: Permite directivas de control del tipo de documentos.
 - **Indexes**: Permite directivas de indexado de directorios.
 - **Limit**: Permite directivas que controlan el acceso por dirección IP o nombre de hosts del cliente.
 - **Options**: Permite directivas que controlan funcionalidades de los directorios.
- Hay que tener muy en cuenta que las directivas son aplicadas en el orden en que son encontradas por Apache.
- Un fichero .htaccess de un nivel inferior puede sobrescribir las directivas de un nivel superior.
- Se utiliza normalmente:
 - Especificar restricciones de seguridad para un determinado directorio.
 - Reescribir URLs largas y complejas en otras más sencilla
 - Bloquear a usuarios por su dirección IP
- Los ficheros .htaccess están protegidos por una directiva en el fichero general de Apache.

```
<Files ~ "^\.ht">
```

```
    Require all denied
```

```
</Files>
```

Host virtuales por nombre en Apache

- Gracias al uso de hosts virtuales, Apache2 permite la posibilidad de **alojar varios dominios en una sola máquina**.
- Esto quiere decir, que podemos alojar dos sitios web: **servidor1.auladaw.com** y **servidor2.auladaw.com en la misma máquina**, respondiendo ambos en la misma IP.
- Apache soporta dos tipos de hosts virtuales: Host virtuales basados en nombres y Host virtuales basados en IP
- En el servidor, los ficheros de configuración de cada host virtual lo encontramos en **/etc/apache2** y tenemos dos carpetas
 - **sites-available**: donde tenemos los ficheros de configuración de los host virtuales **disponibles**
 - **sites-enabled**: donde tenemos los ficheros de configuración de los host virtuales que se están **utilizando actualmente**. Es un enlace simbólico al fichero de configuración de sites-available.
- Para activarlos, se utiliza `a2ensite <nombre del directorio virtual>`, el cual creará un enlace en sites-enabled.
- Los ficheros de configuración de los host virtuales, empezarán con `<VirtualHost ...>`
 - Si queremos tener un host virtual **basado en nombres**:
`<VirtualHost *:80>`
`ServerName www.aula.com`
 - Si queremos que sea **mediante ip**: `<VirtualHost 192.168.5.103:80>`

Host virtuales por nombre en Apache

- Los ficheros de configuración de los host virtuales, empezarán con <VirtualHost ...>
 - Si queremos tener un host virtual **basado en nombres**:

```
<VirtualHost *:80>  
    ServerName www.aula.com  
    DocumentRoot /var/www/aula  
    <Directory /var/www/aula>  
        ...  
    </Directory>  
</VirtualHost>
```
 - Si queremos que sea **mediante ip** [Tenemos que tener varias interfaces de red en el servidor para ello]

```
<VirtualHost 192.168.5.103:80>  
    ...  
</VirtualHost>  
<VirtualHost 192.168.5.104:80>  
    ...  
</VirtualHost>
```
 - Si queremos que sea mediante puerto [Tenemos que tener todos los puertos habilitados]

```
Listen 80  
Listen 81  
<VirtualHost IP:80>  
    ...  
</VirtualHost>  
<VirtualHost IP:81>  
    ...  
</VirtualHost>
```

Activar la seguridad en Apache

SERVIDOR HTTPS POR DEFECTO

- En primer lugar hay que activar el módulo ssl.
`$ sudo a2enmod ssl`
- En el archivo que almacena los puertos de escucha `/etc/Apache2/ports.conf`, comprueba que existe una directiva `<IfModule>` que incluya la escucha en el puerto 443 si el módulo SSL está activado.
[Con `netstat -ntn` puedes comprobar si se escucha el puerto]

Editando archivo de configuración: `/etc/apache2/ports.conf`

```
1 # If you just change the port or add more ports
2 # have to change the VirtualHost statement in
3 # /etc/apache2/sites-enabled/000-default.conf
4
5 Listen 80
6
7 <IfModule ssl_module>
8     Listen 443
9 </IfModule>
10
11 <IfModule mod_gnutls.c>
12     Listen 443
13 </IfModule>
14
```

Activar la seguridad en Apache

- Comprobamos que en el directorio `/etc/apache2/sites-available` existe el archivo `default-ssl.conf` que contiene la configuración por defecto de un servidor HTTPS.
- Habilitamos el servidor virtual `ssl` por defecto de Apache [reinicia el servidor a continuación]:
`$ sudo a2ensite default-ssl.conf`
- Editamos el fichero **default-ssl.conf** las directivas para activar el SSL son las siguientes:
 - **SSLEngine**. Activa [on]/desactiva [off] el protocolo SSL/TLS en un servidor virtual. (Por defecto está deshabilitado tanto para el servidor principal como para todos los servidores virtuales.)
 - **SSLCertificateFile**. Indica el nombre del archivo que contiene el certificado del servidor
 - **SSLCertificateKeyFile**. Indica el nombre del archivo que contiene la clave privada del servidor.
- El servidor `default-ssl.conf` tiene un certificado autofirmado por defecto que ya está definido, así que podemos utilizarlo tal cual.
- Para la práctica vamos a utilizar OpenSSL, que nos permite generar certificados autofirmados, pero en un entorno de servidor de producción real, estos certificados se deberían solicitar a una autoridad de certificación.
- Para generar un certificado autofirmado, utilizamos la siguiente línea:
- `$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/default-selfsigned.key -out /etc/ssl/certs/default-selfsigned.crt`

Crear un host virtual seguro.

- Para generar un certificado autofirmado, utilizamos la siguiente línea
 - Cómo dominio utilizaremos `www.aula.com`
 - `$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/aulacom-selfsigned.key -out /etc/ssl/certs/aulacom-selfsigned.crt`
- Creamos la carpeta `/var/www/html/websegura`
- Creamos el fichero `websegura.conf` en el `/etc/apache2/sites-available`

Archivo `/etc/apache2/sites-available/websegura`

`<IfModule mod_ssl.c>`

`<VirtualHost *:443>`

`ServerName www.aula.com`

`DocumentRoot /var/www/html/websegura`

`<Directory /var/www/html/websegura>`

`SSLRequireSSL`

`DirectoryIndex index.html`

`Options Indexes FollowSymLinks`

`AllowOverride None`

`Require all granted`

`</Directory>`

`ErrorLog ${APACHE_LOG_DIR}/websegura.error.log`

`LogLevel warn`

`CustomLog ${APACHE_LOG_DIR}/websegura.access.log combined`

`SSLEngine On`

`SSLCertificateFile /etc/ssl/certs/aulacom-selfsigned.crt`

`SSLCertificateKeyFile /etc/ssl/private/aulacom-selfsigned.key`

`</VirtualHost>`

`</IfModule>`

Activar la seguridad en Apache

- Ahora ya tenemos el servidor con https [Recuerda que al no ser un certificado firmado por una CA, nos saldrá una alerta en el navegador]

