

CONFIGURACIÓN SERVIDORES HTTP

Índice

1. INTRODUCCIÓN.....	2
2. IIS.....	2
A) Instalación.....	2
B) Fichero/s por defecto.....	4
C) Hosting virtual.....	4
D) Directorio virtual.....	5
E) Listado del contenido de un directorio.....	7
F) Logs del servidor.....	8
G) Autenticación.....	9
H) Instalación módulos.....	11
H.1) Instalación manual de módulos.....	15
3. APACHE.....	19
A) Instalación.....	19
B) Fichero/s por defecto.....	22
C) Hosting virtual.....	24
D) Directorio virtual.....	25
E) Listado del contenido de un directorio.....	28
F) Logs del servidor.....	29
G) Autenticación.....	32
H) Control de acceso.....	34
I) Instalación módulos.....	36
J) Ficheros .htaccess.....	37
K) El módulo <i>rewrite</i>	38
4. NGINX.....	40
A) Instalación y puesta en marcha.....	40
B) Configuración básica.....	41
B) Virtual hosting.....	41
C) Control de acceso.....	42
D) Listado del contenido de directorios.....	43
E) Autenticación.....	43
F) Logs del servidor.....	44
G) Ejecución de páginas dinámicas.....	45

1. INTRODUCCIÓN

Vamos a realizar alguna práctica de configuración del servicio HTTP, para ello vamos a tener virtualizado tanto Windows 2016 Server, como Ubuntu Server 18.04. Los servidores que vamos a estudiar son los 3 más utilizados:

- Apache
- Nginx
- Internet Information Services (IIS)

De cualquiera de los servidores existe abundante documentación en la web, además de las opciones que veremos aquí, existen muchas otras, las cuales puedes consultarlas en esos manuales.

2. IIS

Internet Information Services (www.iis.net) es una suite(conjunto) de servidores que se integran en sistemas Windows, entre otros incluye servicios como:

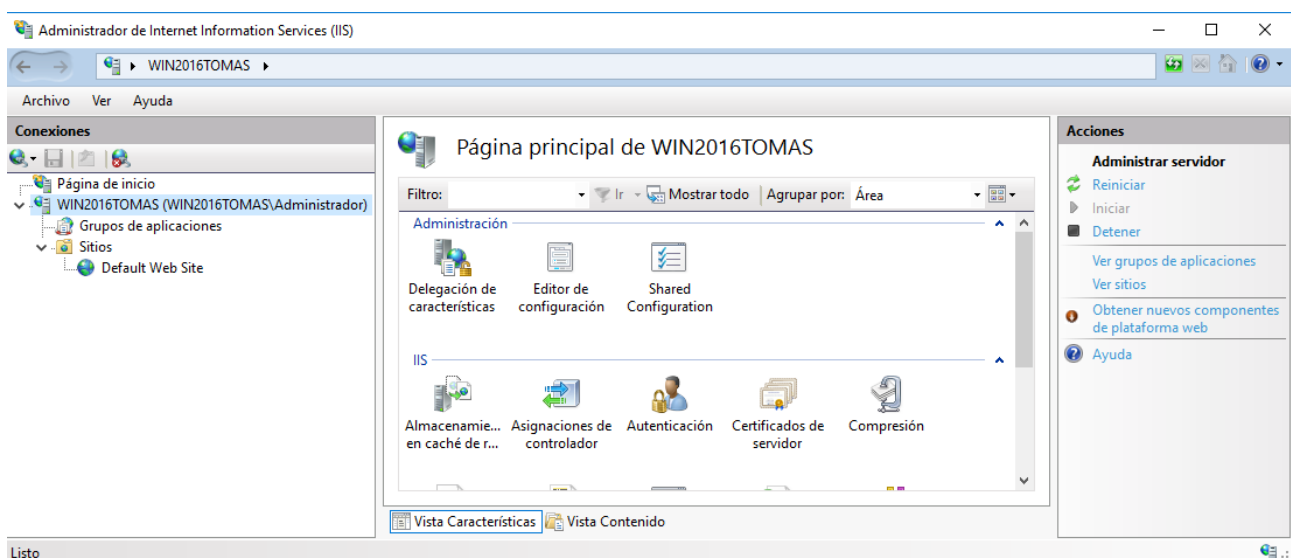
- web
- correo
- noticias
- ftp

Actualmente nos encontramos en la versión 10, la cual se integra con la instalación de Windows 2016 Server.

A) Instalación

El servicio se instala como un rol más en el servidor, pudiendo elegir posteriormente cuales funcionalidades queremos exactamente, Windows los denomina Servicios de rol, **en nuestro caso dejaremos los que vienen activos por defecto:**

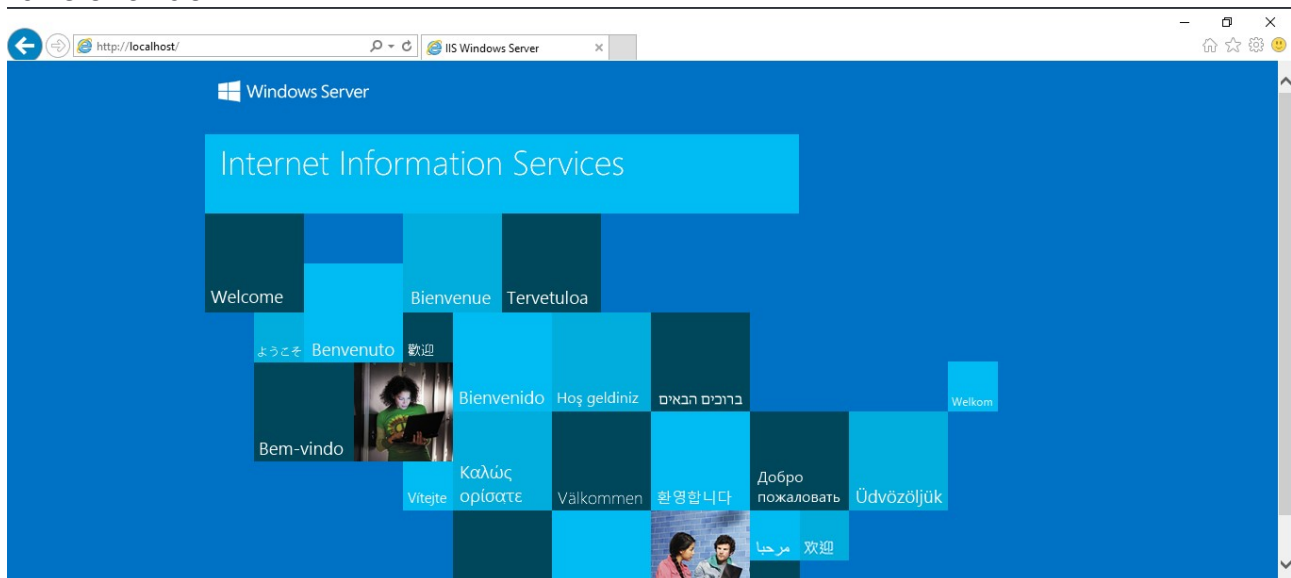
Una vez finalizado el proceso de instalación debemos ver la consola de administración de IIS:



Con respecto a la configuración de IIS a través de su consola, hay varios aspectos a tener en cuenta:

1. Hay 4 niveles de administración.
 - a) Servidor
 - b) Sitio (servidor virtual)
 - c) Directorio
 - d) Archivo
2. Las tareas básicas se pueden hacer con asistentes, pero para las configuraciones avanzadas hay que acudir a las ventanas de Propiedades del elemento que queremos configurar.
3. Por defecto las configuraciones se heredan de padres a hijos, si es aplicable. Aunque puede modificarse.

Para comprobar que nuestro servicio ya está funcionando, lo más sencillo es conectarnos a nuestro localhost, y comprobar que el sitio por defecto está funcionando.



Realiza en tu MV Windows 2016 Server la instalación de tu servicio IIS y comprueba que puedes acceder en local o a través de una conexión solo-anfitrión(host-only) de VBox

B) Fichero/s por defecto

Cuando cualquier dominio, uno de los parámetros más importantes es qué página va a buscar IIS en caso que no se especifique ninguna. Por ejemplo:

<http://localhost/formulario.html> → Pide la página formulario, escrita en html en cambio

<http://localhost> → NO pide ninguna página en concreto

¿Entonces como es posible que cuando pongo URL del 2º tipo los servidores contesten? Porque buscan en su listado de páginas por defecto

¿Donde está almacenado tu sitio web por defecto?

¿Qué página carga?

Crea el siguiente código html y llámalo index.html. Guárdalo en el directorio raíz de tu sitio web. ¿Qué ocurre?

<html>

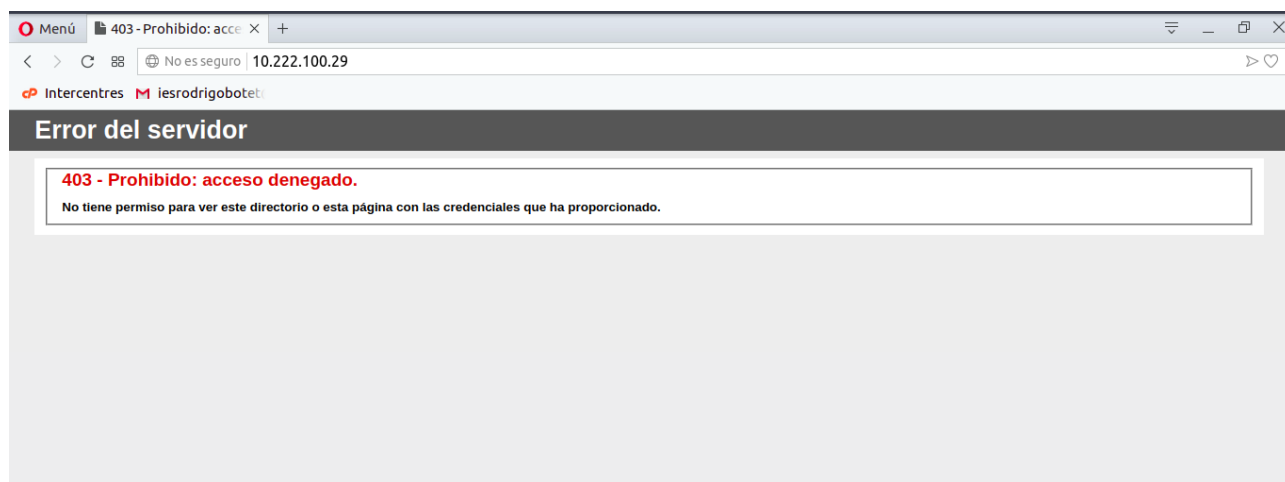
<body>

<h1>SERVIDOR WEB DE 2º DE ASIR</h1>

</body>

</html>

Si no encuentra ninguno de los ficheros que tenemos configurados Por Defecto, el navegador nos mostraría una página de error.



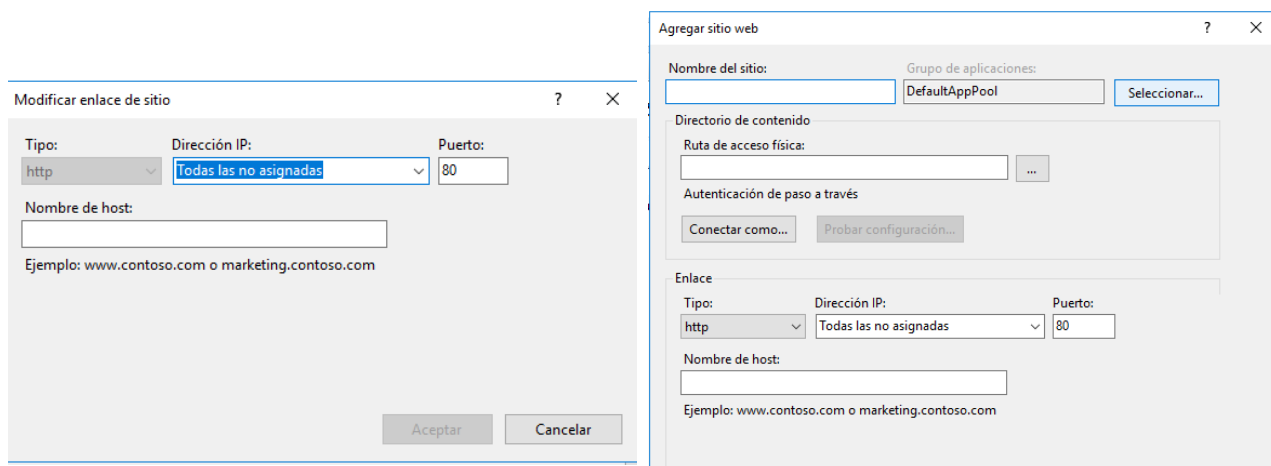
¿Páginas de error del servidor? ¿Dónde están?

C) Hosting virtual

Por cada sitio web no hay un servidor funcionando, en la mayoría de casos sería malgastar recursos. Los servidores web, IIS incluido, admiten lo que se denomina Virtual Hosting, que consiste en servir diferentes contenidos dependiendo de la petición(URL) que reciba. Esta distinción se puede hacer por 3 elementos:

- A) Por IP
- B) Por nombre de dominio
- C) Por núm de puerto

Para poder decir a IIS que un determinado dominio atiende peticiones para una IP, nombre de dominio o puerto en concreto, debemos rellenar la primera pantalla en el momento de Agregar sitio web o en la ventana Modificar enlaces de un sitio ya existente:



Modificar enlaces

Agregar sitio web

Realiza las siguientes tareas sobre tu servidor IIS:

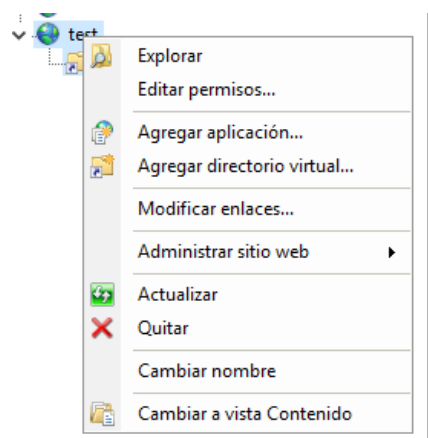
- 1)** Cambia el dominio por defecto para que escuche por el puerto 81. Comprueba su nuevo funcionamiento.
- 2)** Crea un nuevo dominio que escuche solo peticiones por una IP. Comprueba su funcionamiento.
- 3)** Crea otro dominio que escuche solo para un nombre concreto. Cambia el fichero hosts de tu cliente o tu servidor DNS para comprobar que efectivamente funciona.



D) Directorio virtual

Si te has dado cuenta, hasta ahora solo podemos crear dominios dentro del directorio raíz de nuestro servidor IIS (*por defecto C:\inetpub\wwwroot*). ¿Qué ocurre si queremos por lo que sea acceder a una ruta fuera de esa estructura de directorios?. Esto ocurre por ejemplo si tenemos alguna carpeta de imágenes compartida en nuestro sitio web o si algún contenido en concreto se encuentra en una unidad de red al estar compartida entre varios servidores replicados.

Añade un directorio virtual a alguno de tus sitios web que se encuentre fuera del directorio C:\inetpub\wwwroot



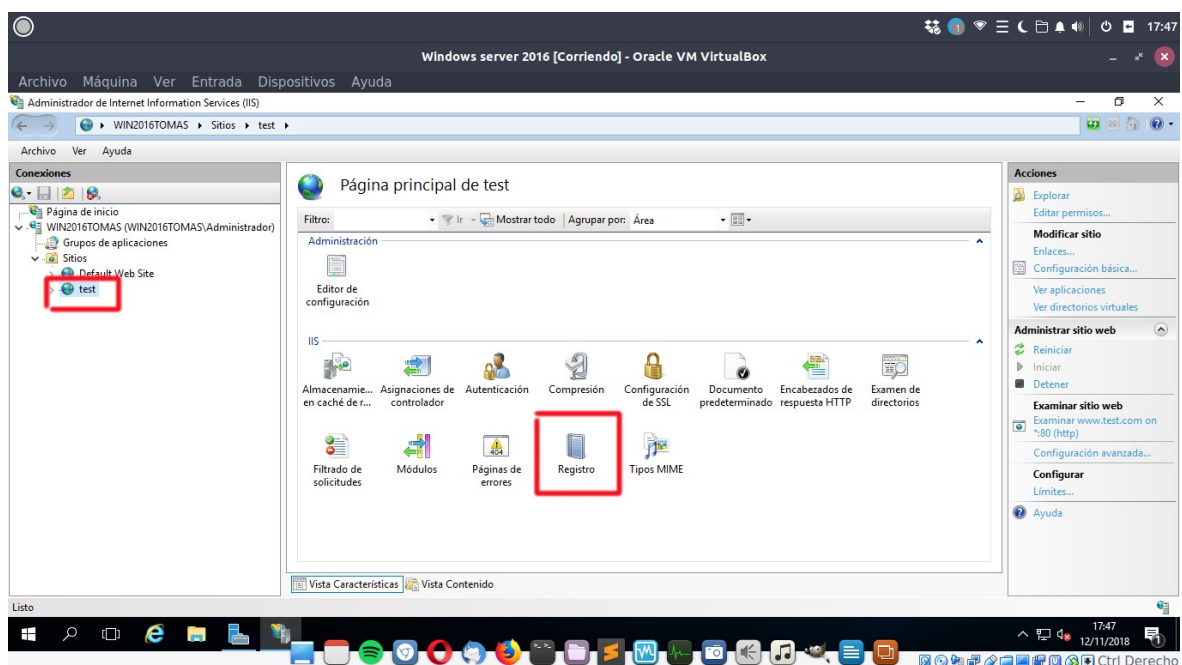
E) Listado del contenido de un directorio

Si cuando entramos en un dominio, no encuentra ninguno de sus páginas por defecto. El navegador devolverá un error http 403. Podemos configurar el servidor para que, en ese caso, muestre el contenido de la carpeta.

Usar esta configuración con prudencia, ya que puede ser inseguro listar contenidos de carpetas como norma.

[illegible]

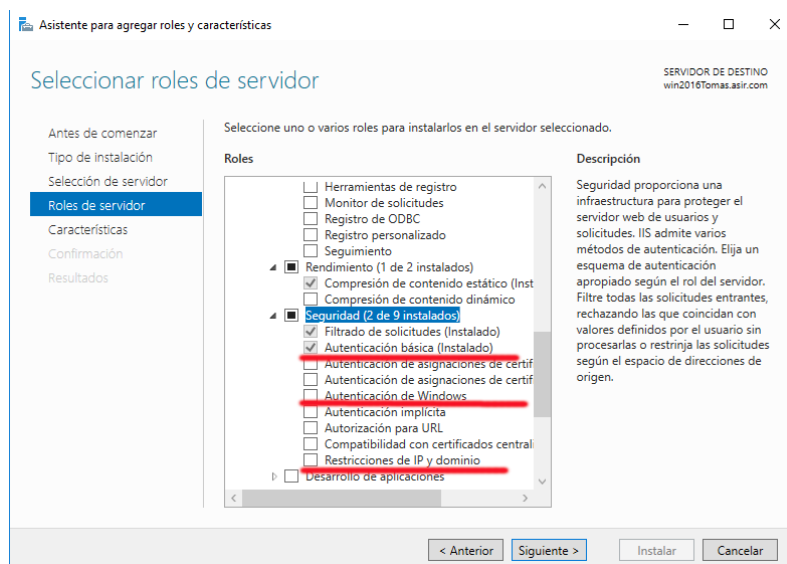
Como sysadmin(administrador/a de sistemas) nuestra función seguramente tendrá más que ver con la revisión del funcionamiento del servidor que con el comportamiento del contenido web propiamente dicho(esto ya sería cuestión de los programadores). Por lo tanto, una de las cosas que mejor hemos de controlar son los log del servidor. Estos archivos registran la actividad del IIS y en ellos podemos descubrir cuando algo ha dejado de funcionar correctamente.



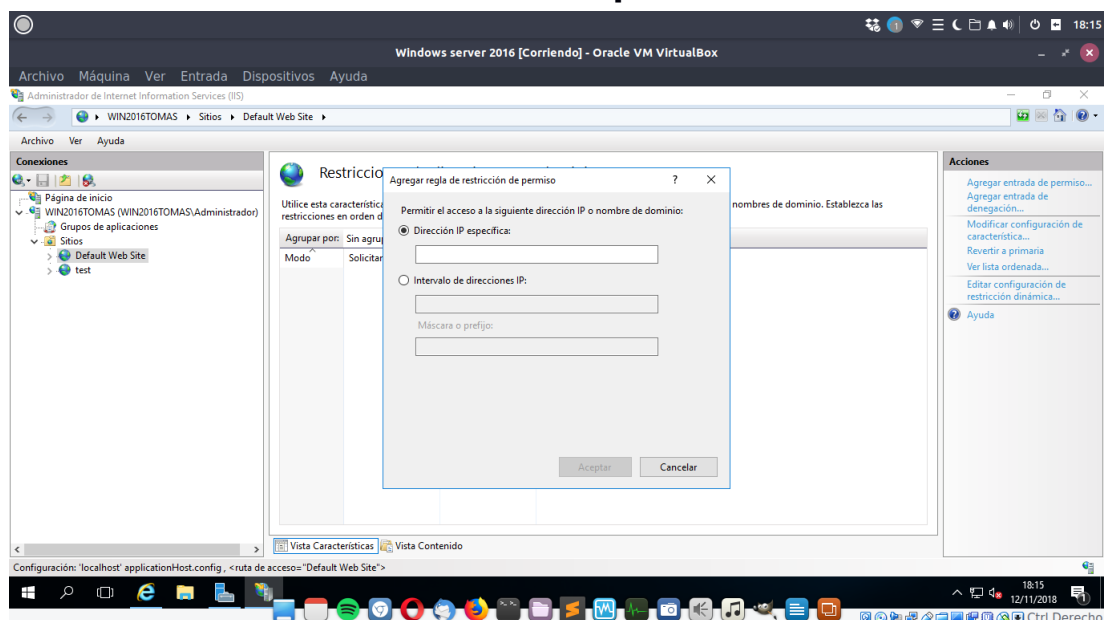
Revisa el log de alguno de tus sitios web y comprueba su contenido. ¿Sabrías decir qué significa la información que nos está proporcionando?. Recuerda revisar los códigos de respuesta HTTP.

G) Autenticación

Por defecto IIS no admite la autenticación, por lo que las páginas están disponibles a cualquier persona anónima. Si queremos proteger algún contenido concreto de nuestro sitio web debemos instalar algún servicio de validación de usuarios o de limitación de acceso por IP (para que puedan acceder solo desde la red local), debemos instalar alguna de las opciones existentes.



Podemos encontrar una descripción de los tipos de autenticación en el siguiente [enlace](#). Para configurar una limitación por IP, una de las opciones para un sitio es **Restricciones de acceso por IP**.



En esta pantalla podemos activar listado de IP o de direcciones de red, y elegirlos para darles acceso y para denegarlos.

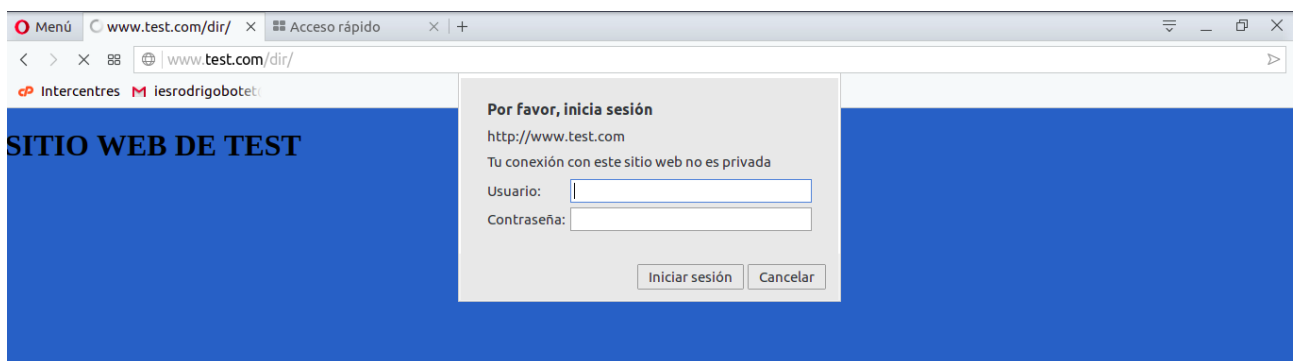
Para activar el acceso por usuario, también sobre la carpeta en concreto de nuestro dominio, también tenemos la opción de **Autenticación**.

Para poder hacer pruebas puedes crear usuarios directamente en la línea de comandos de Windows:

```
net user usuario1 pa$S$W0rd /add
```

```
net user usuario2 pa$S$W0rd /add
```

Si activamos la autenticación básica [nuestr@s](#) [usuari@s](#) tendrán acceso a carpetas privadas en función de los permisos NTFS que tengan asignadas. **Recordad el desactivar la autenticación anónima.**



En mi ejemplo si pongo los datos de mi usuario de windows, accederé al contenido web. **No he configurado mi servidor como HTTPS, por lo que los datos viajan en texto plano.**

H) Instalación módulos

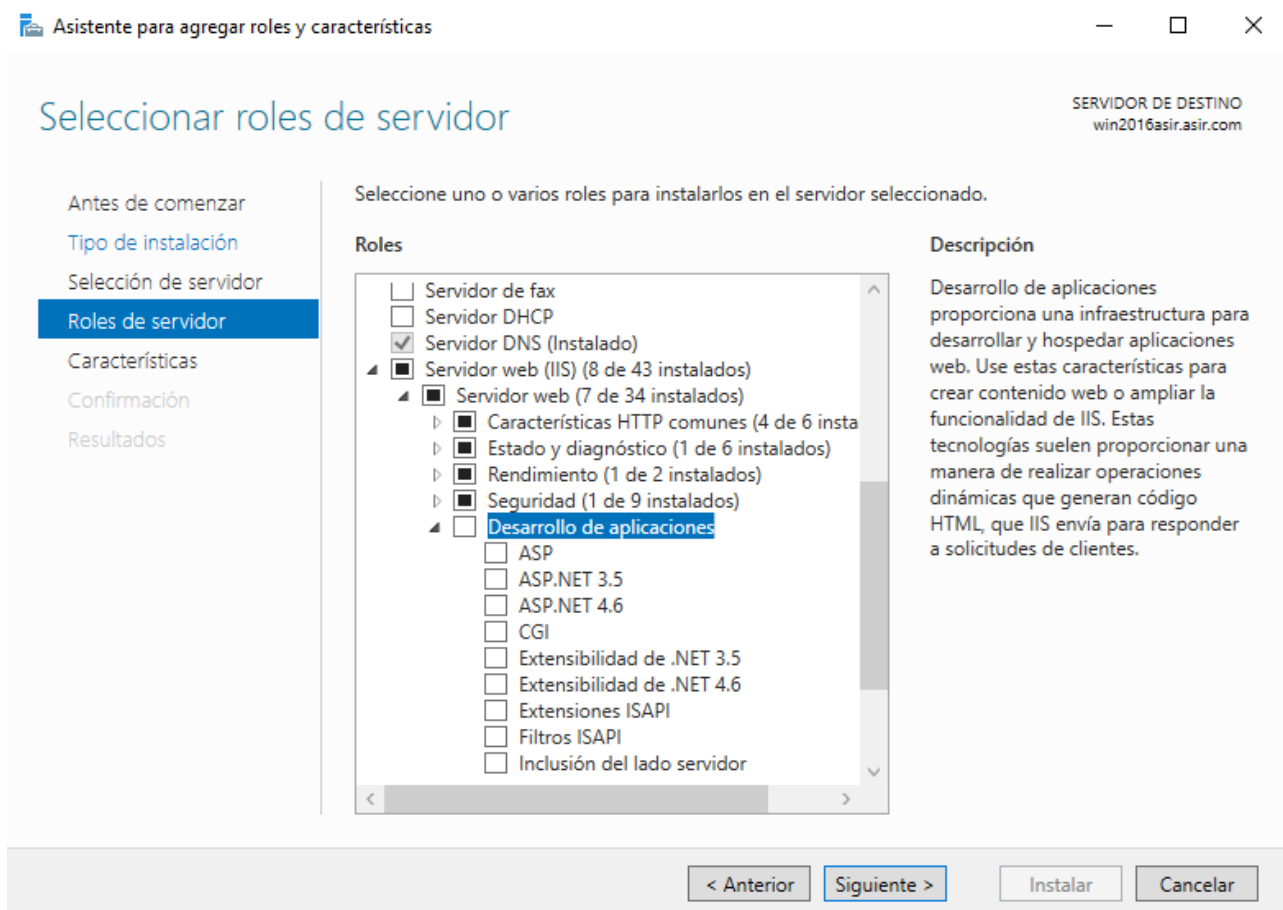
Cuando tengamos la instalación de nuestro servidor finalizada, debemos comprobar que el lenguaje de programación de servidor(**php, asp, jsp...**) que vayamos a utilizar en nuestros sitios web, se ejecute correctamente, o sea, esté instalado el **módulo** que lo ejecuta. Lo mismo ocurriría con el Sistema Gestor de BD que vayamos a usar (**Mysql, PostGreSQL, Oracle..**).

En el ejemplo que nos ocupa, vamos a comprobar que viene instalado ASP, el lenguaje de programación de servidores web de Microsoft. Para ello vamos a crear en nuestro directorio raíz un fichero que llamaremos **default.asp** con el siguiente contenido:

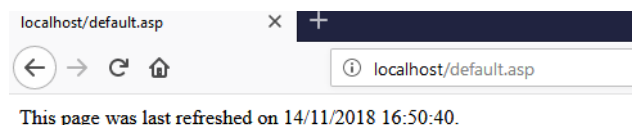
```
<HTML>
  <BODY>
    This page was last refreshed on <%= Now() %>.
  </BODY>
</HTML>
```

Lo llamamos así porque uno de los ficheros por defecto que busca iis es default.asp.

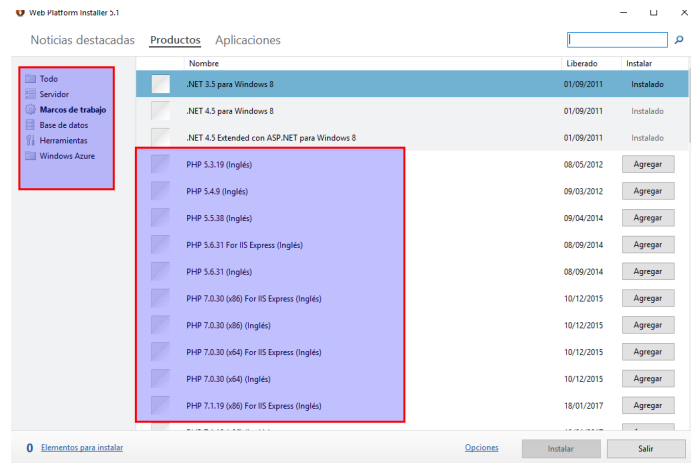
Su ejecución nos daría un error, lo que nos indica que NO viene instalado por defecto en IIS el módulo de ASP. ¿Como lo instalamos?



De esta manera hemos instalado ASP, con lo que ya veremos el resultado de su ejecución:



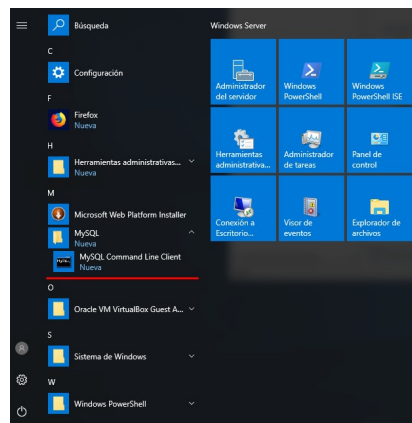
Seguramente nos interesará instalar el lenguaje más utilizado hoy en día, junto con una BD de uso muy frecuente (PHP+MySQL). Para ello, Microsoft nos ofrece el **Web Plattform Installer**, herramienta que nos facilita la instalación de multitud de herramientas sobre nuestro servidor. Una vez instalado, debemos ver algo como lo siguiente (**es probable que haya que reiniciar el servidor para que sea accesible**):



Aquí ya podemos instalar las extensiones que queramos. En nuestro ejemplo **(Cuidado con las versiones, dependiendo de nuestra versión de SO, puede fallar o no):**

- A) PHP 5.3**
- B) MySQL 5.1**

Para probar A podríamos hacer un **phpinfo();** y cambiar nuestras páginas por defecto y para la B podríamos conectarnos por consola a nuestra BD, con los datos de administrador que le hemos dado en el proceso de instalación.



En una consola:

```
C:\Program Files\MySQL\MySQL Server 5.1\bin>mysql.exe

Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.1.73-community MySQL Community Server (GPL)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| test |
+-----+
3 rows in set (0.00 sec)

mysql>
```

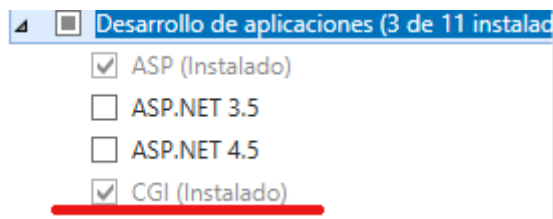
Prueba tu servidor, tanto con una conexión a la BD, como escribiendo un fichero PHPINFO y comprobando que se ejecuta correctamente, mostrando toda la información de PHP en el navegador.

H.1) Instalación manual de módulos

En caso de querer un mayor control en el proceso de instalación, podemos optar por no utilizar Web Plattform Installer, y realizar paso a paso todas las acciones que este realiza, este proceso es más parecido a lo que tendremos que realizar en Apache o NginX:

1. Descargar el paquete de PHP que vayamos a utilizar. **Atent@s a la compatibilidad de versiones.** Debes tener en cuenta:

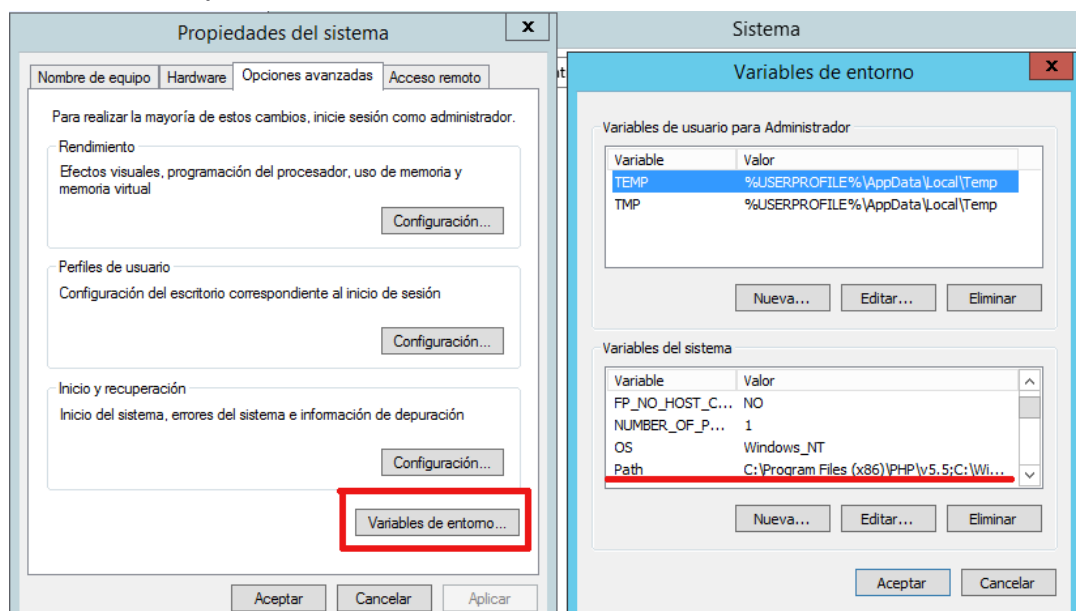
- [Descarga la versión de PHP que necesites](#). Dependerá de las aplicaciones que vayas a ejecutar en tu servidor.
- Tal y como dice el manual de instalación en Windows, de la web oficial de php: *“PHP requires the Visual C runtime(CRT). Many applications require that so it may already be installed.”*
- Se va a llamar a PHP a través del módulo FastCGI, por lo que debe estar instalado en nuestro IIS.



- Al utilizarse FastCGI, la versión de PHP que ha de utilizarse es **Non Thread Safe (NTS)**
- **En el área de Utilidades del Aula virtual se ha dejado una versión correcta, tanto de PHP, como del Visual C++**

¿Qué es todo eso de..?
CGI/FastCGI
Thread Safe/ Non Thread Safe

2. Modificar la variable de entorno PATH, para incluir la ruta donde hemos descomprimido PHP.

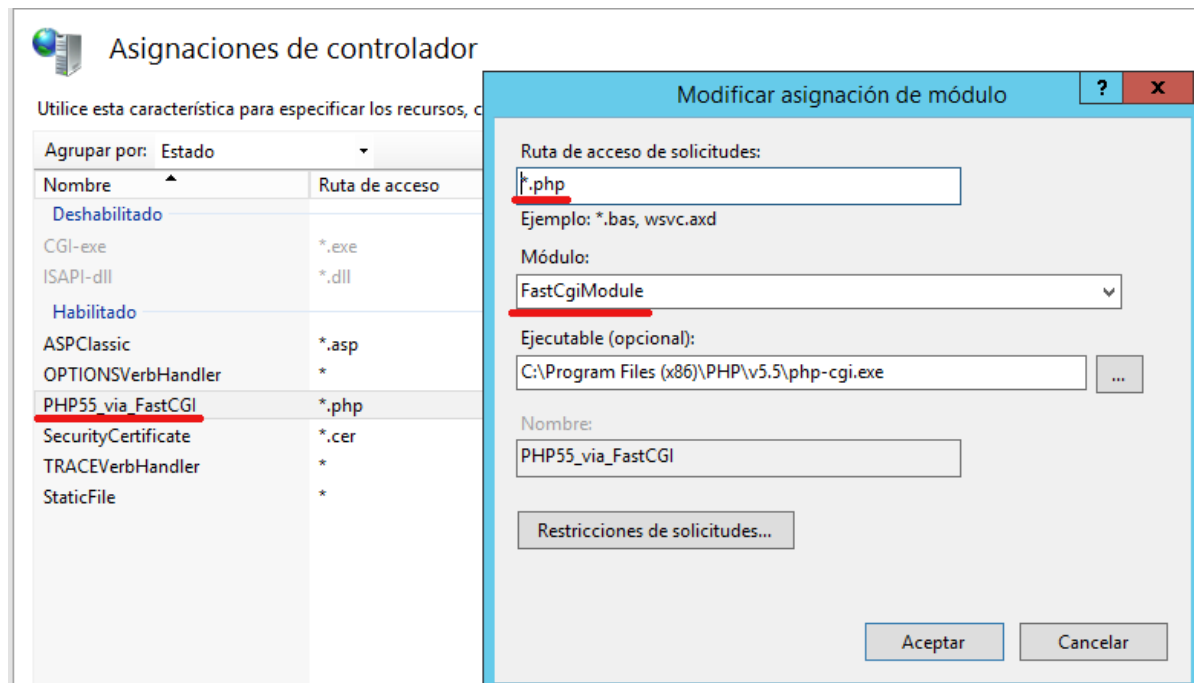


3. Crear la asignación de controlador (handler mapping) correspondiente.



Asignaciones de controlador

Con esto le decimos que todas las peticiones que lleguen con la extensión php, las atienda nuestro módulo de PHP.



Puedes encontrar multitud de manuales en la web de como realizar este proceso, por ejemplo en:

<https://techexpert.tips/es/windows-es/instalar-php-en-windows-server-iis/>

3. APACHE

Este es el servidor de páginas web por excelencia. Una de sus principales ventajas es que existe en prácticamente cualquier Sistema Operativo. Podemos encontrar instalaciones de Apache ya realizadas y configuradas, lo que nos facilitaría la puesta en marcha de un servicio web:

- Instalaciones de Apache+módulos útiles: **XAMP, LAMP, VirtualServer..**
- Máquinas virtuales con el servicio instalado y configurado: **Bitnami, Vagrant..**
- Contenedores con el sistema instalado: **Docker, LXC..**

En nuestro caso no vamos a hacer uso de estas opciones, ya que lo que nos interesa es conocer el funcionamiento de este servicio para poderlo configurar correctamente.

A) Instalación

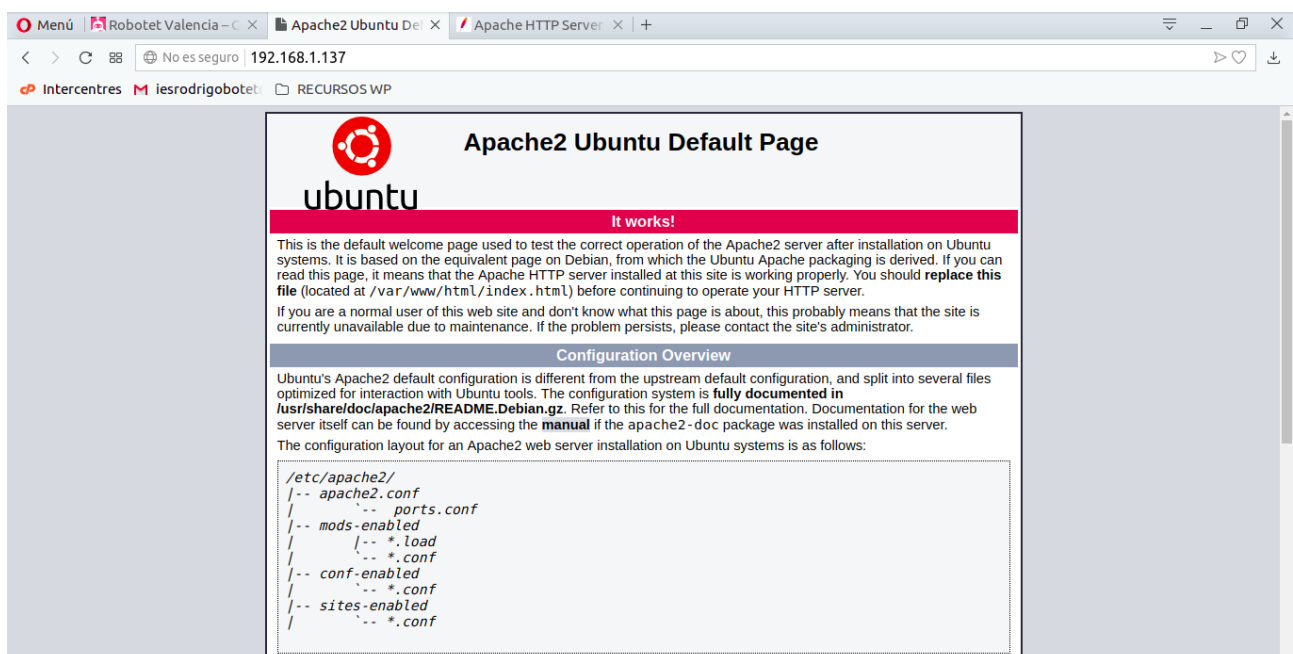
La instalación de este servicio es similar a los servicios que ya conocemos, al encontrarse el paquete en todos los repositorios oficiales de las distribuciones habituales (en nuestro caso vamos a usar una Ubuntu Server 18.04). Con la ejecución del comando instalaremos el servicio:

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

Para gestionar el servicio podemos utilizar los comandos ya conocidos:

```
administrador@srvub1804tomas:~$ sudo service apache2  
Usage: apache2 {start|stop|graceful-stop|restart|reload|force-reload}
```

Una vez comprobemos que el servicio se ha instalado y está funcionando, ya podríamos acceder al sitio web por defecto que el instalador de Apache incluye.



En esta página de inicio, ya se nos está ofreciendo información bastante útil, por ejemplo que si hemos instalado también el paquete **apache2-doc**, tendremos un acceso directo a la completa documentación oficial de Apache, por otra parte nos muestra un resumen de la ESTRUCTURA DE FICHEROS Y DIRECTORIOS en la que se organiza nuestro servidor. Conocer la estructura de ficheros de configuración de Apache es el primer paso para poderlo administrar y configurar.

```
/etc/apache2/  
|-- apache2.conf  
|   |-- ports.conf  
|-- mods-enabled  
|   |-- *.load  
|   |-- *.conf  
|-- conf-enabled  
|   |-- *.conf  
|-- sites-enabled  
|   |-- *.conf  
|
```

- apache2.conf is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- ports.conf is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the mods-enabled/, conf-enabled/ and sites-enabled/ directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective *-available/ counterparts. These should be managed by using our helpers `a2enmod`, `a2dismod`, `a2ensite`, `a2dissite`, and `a2enconf`, `a2disconf`. See their respective man pages for detailed information.
- The binary is called `apache2`. Due to the use of environment variables, in the default configuration, `apache2` needs to be started/stopped with `/etc/init.d/apache2` or `apache2ctl`. **Calling `/usr/bin/apache2` directly will not work** with the default configuration.

Document Roots

By default, Ubuntu does not allow access through the web browser to *any* file apart of those located in `/var/www`, **public_html** directories (when enabled) and `/usr/share` (for web applications). If your site is using a web document root located elsewhere (such as in `/srv`) you may need to whitelist your document root directory in `/etc/apache2/apache2.conf`.

The default Ubuntu document root is `/var/www/html`. You can make your own virtual hosts under `/var/www`. This is different to previous releases which provides better security out of the box.

A modo de resumen, la configuración de Apache se basa en la siguiente:

- A) apache2.conf:** Contiene los parámetros generales. En otras distribuciones el fichero se denomina `httpd.conf`. Con directivas incluye hace referencia al resto de ficheros de configuración.
- B) ports.conf:** IPs y puertos donde escucha el servidor.
- C) Envars:** Variables de entorno que usa Apache.
- D) MÓDULOS:** encargados de añadir funcionalidades a Apache
 - **mods-available:** módulos disponibles, con ficheros `LOAD` y `CONF`.
 - **mods-enabled:** módulos activos, se tratan de enlaces simbólicos a `mods-available`.
- E) SITIOS:** por evitar ficheros de configuración muy extensos, se escribe un fichero de configuración por cada sitio web alojado.(virtual hosting)
 - **sites-available:** ficheros de configuración de sitios virtuales disponibles
 - **sites-enabled:** sitios activos, se tratan de enlaces simbólicos a `sites-available`.

Instala el servidor y accede a los distintos ficheros y carpetas que se han instalado. Comprueba qué tipo de configuraciones podrías realizar en cada uno de ellos.

B) Fichero/s por defecto

Para configurar las páginas que el servidor busca, cuando no se ha indicado ninguna. Recordando el punto B del apartado anterior:

<http://localhost/formulario.html> → Pide la página formulario, escrita en html

<http://localhost> → NO pide ninguna página en concreto

¿Cómo indico los ficheros a buscar por defecto?

Sabiendo que la directiva es **DirectoryIndex**

Vamos a intentar localizarlo nosotros mismos y, a través del manual, comprobad en qué **contextos** lo podemos utilizar y cómo.

¿Donde está almacenado tu sitio web por defecto?

¿Qué página carga?

Cambia el nombre del fichero a `index_aux.html`. ¿Que ocurre?

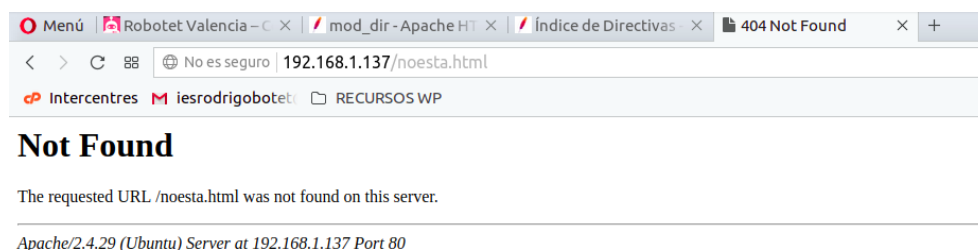
Crea el siguiente código html y llámalo `index.htm`. Guárdalo en el directorio raíz de tu sitio web. ¿Qué ocurre?

```
<html>
<body>
    <h1>SERVIDOR APACHE WEB DE 2º DE ASIR</h1>
</body>
</html>
```

Si no encuentra ninguno de los ficheros que tenemos configurados Por Defecto, el navegador nos mostraría un listado del directorio. A diferencia de Windows, el listado del contenido del directorio viene activado. Lo trataremos en un punto posterior.



Apache devuelve unas páginas predefinidas dependiendo del código de error HTTP que devuelva la llamada. Por ejemplo (para un error 404):



Las páginas de error por defecto también se pueden personalizar, con la directiva **ErrorDocument**. Lo primero sería consultar el manual para conocer su sintaxis y dónde podemos utilizar esta directiva.

Modifica la configuración de tu sitio web por defecto para que la página de error que muestre en caso de NO ENCONTRAR el recurso solicitado, sea una definida por ti. (llamada por ejemplo 404.html). Reinicia el servidor para que los cambios tengan efecto. Fuerza un error de este tipo en tu servidor.

C) Hosting virtual

Para alojar varios sitios web en un mismo servidor, tenemos la opción del Virtual Hosting, se puede configurar por **IP, por nombre o por puerto**.

Como puedes imaginar, ésta es una de las características más importantes del servidor, por lo que tiene una **sección específica en el manual**. La configuración se realizaría a nivel de sitio, por lo que debe ir indicado en el fichero .conf correspondiente a ese sitio web.

A continuación el contenido típico, unificado en un fichero de cómo se realizaría un virtual hosting por nombre:

Running several name-based web sites on a single IP address.

Your server has multiple hostnames that resolve to a single address, and you want to respon

Note

Creating virtual host configurations on your Apache server does not magically cause DN DNS, resolving to your IP address, or nobody else will be able to see your web site. You from the machine with those hosts entries.

```
# Ensure that Apache listens on port 80
Listen 80
<VirtualHost *:80>
    DocumentRoot "/www/example1"
    ServerName www.example.com

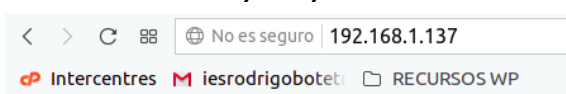
    # Other directives here
</VirtualHost>

<VirtualHost *:80>
    DocumentRoot "/www/example2"
    ServerName www.example.org

    # Other directives here
</VirtualHost>
```

Hay que recordar que en nuestro caso, para que todo funcione deberíamos hacerlo de la siguiente manera:

1. Escribir la configuración de cada servidor virtual en un fichero .conf con el mismo nombre que el dominio y guardarlo en **sites-available**
2. Activar el site con **a2ensite** (se desactiva con **a2dissite**). Esta acción crea o borra los enlaces simbólicos en **sites-enabled**.
3. Se puede ampliar la funcionalidad usando la directiva **ServerAlias**
4. **Recordar o configurar nuestro servidor DNS del tema anterior, o el fichero /etc/hosts**



Index of /

Name	Last modified	Size	Description
 indexd.html	2018-11-18 21:31	11K	

Apache/2.4.29 (Ubuntu) Server at 192.168.1.137 Port 80



Realiza las siguientes tareas sobre tu servidor Apache:

- 1) Configura el servidor para que escuche solo peticiones por una IP. Comprueba su funcionamiento.
- 2) Crea varios hosts virtuales con distintas páginas web de inicio, añádelos a tu fichero hosts y comprueba que dependiendo del nombre que uses, muestra un contenido u otro.
- 3) Usa la directiva `ServerAlias` para modificar el comportamiento de tu servidor.

D) Directorio virtual

Para añadir directorios del sistema de ficheros del servidor al propio servidor web (se usa el término *mapear*), se utiliza la directiva **Alias**. En la documentación oficial de Apache:

Alias Directive

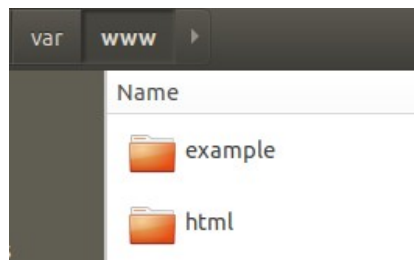
Description: Maps URLs to filesystem locations
Syntax: `Alias [URL-path] file-path|directory-path`
Context: server config, virtual host, directory
Status: Base
Module: `mod_alias`

The **Alias** directive allows documents to be stored in the local filesystem other than under the `DocumentRoot`. URLs with a (%-decoded) path beginning with `URL-path` will be mapped to local files beginning with `directory-path`. The `URL-path` is case-sensitive, even on case-insensitive file systems.

```
Alias "/image" "/ftp/pub/image"
```

Al igual que en IIS, lo importante es el alias que le demos a la ruta. **Si no existen problemas de permisos de acceso**, a partir de ese momento, ya podremos tener acceso vía web a esa parte del sistema de ficheros del servidor.

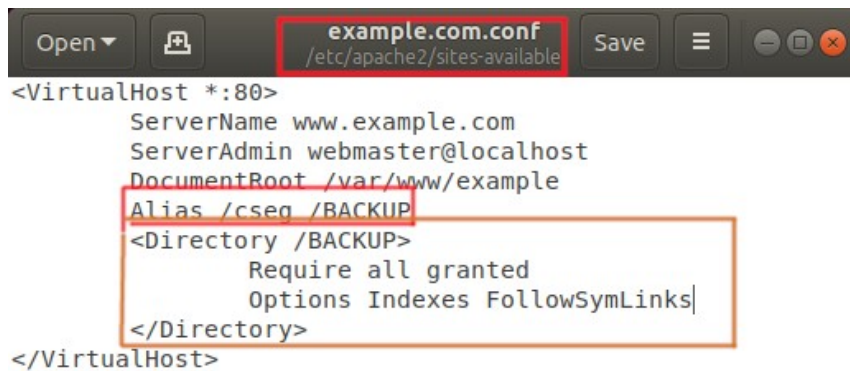
En mi servidor de ejemplo, actualmente tengo dos sitios:



los tengo accesibles de diferente manera, tal y como le indico en los siguientes ficheros

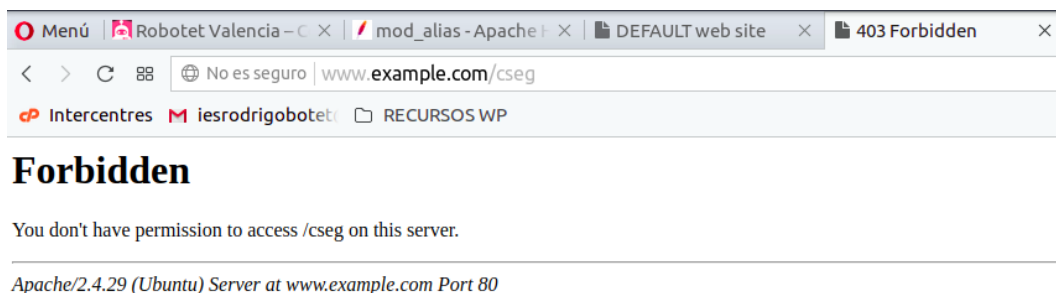


Ahora voy a añadir un alias a *example.com* (atentos-as a la barra / que ponemos), todo lo que está enmarcado en marrón lo explicaremos en puntos siguientes, pero debe ser incluido para que funcione:



```
Open example.com.conf /etc/apache2/sites-available Save
<VirtualHost *:80>
    ServerName www.example.com
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/example
    Alias /cseg /BACKUP
    <Directory /BACKUP>
        Require all granted
        Options Indexes FollowSymLinks
    </Directory>
</VirtualHost>
```

Si probamos el resultado de nuestra nueva configuración ya debería funcionar, si por el contrario, nos devuelve un error de permisos, con el código de error HTTP 403:



Debemos repasar los permisos de la carpeta donde hemos apuntado nuestra alias, si comprobamos los permisos de la carpeta:

```
drwxr-x--- 2 root root 4096 nov 21 09:45 BACKUP
```

Vemos que el propietario y el grupo propietario de esa carpeta es root. Además también vemos que para todo aquel que no sea root o que no pertenezca al grupo root, no tiene ningún permiso sobre esa la carpeta. Para poder acceder vía navegador a ese contenido necesitamos, al menos, los permisos de escritura. Tenemos dos opciones:

1. Cambiar permisos para que el resto de usuarios pueda LEER el contenido. (CHMOD)
2. Cambiar el propietario de esa carpeta (CHOWN)

Si optamos por la segunda opción, quedaría algo como lo siguiente:

```
administrador@srvub1804tomas:/$ sudo chown -R www-data:www-data BACKUP/
administrador@srvub1804tomas:/$ ls -l
total 2051180
drwxr-x--- 2 www-data www-data 4096 nov 21 09:45 BACKUP
```

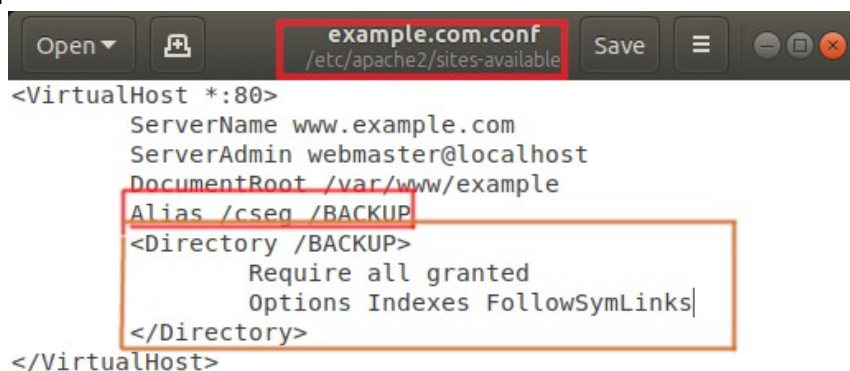
tras este paso el contenido ya es visible, siendo una carpeta que no esta dentro del directorio raíz del servidor:



Añade un directorio virtual a alguno de tus sitios web que se encuentre fuera del directorio `C:\var\www`

E) Listado del contenido de un directorio

Al contrario que en IIS, el listado de los directorios en el servidor por defecto que incorpora la instalación de Apache, SI permite el listado de directorio. Si queremos activar esta opción (recordamos que debe hacerse con precaución) lo que debemos hacer es lo indicado en la imagen anterior, enmarcado con una línea marrón.



Con la opción ([enlace a las opciones existentes en apache](#)) **OPTIONS INDEXES** estaríamos autorizando el listado del contenido del directorio.

Directiva Options	
Descripción:	Configures what features are available in a particular directory
Sintaxis:	Options [+ -]option [[+ -]option] ...
Valor por defecto:	Options All
Contexto:	server config, virtual host, directory, .htaccess
Anula:	Options
Estado:	Core
Módulo:	core

The `Options` directive controls which server features are available in a particular directory.

F) Logs del servidor

En un servicio de uso tan intenso como el web, es fundamental realizar una gestión del log apropiada. Lo interesante es controlar:

1. **Qué se guarda:** nivel de error, warning...
2. **Cómo se guarda:** formato con fecha hora...
3. **Dónde se guarda:** ubicación, rotación de los log..

Por defecto, Apache guarda los archivos de registro en la ruta `/var/log/apache2`. Hay que tener en cuenta que guarda por un lado los accesos al contenido de nuestro/s sitio/s web (**acces.log**) y por otro los errores del servidor Apache (**error.log**), en los que se incluyen los errores de configuración o los fallos. De los virtual host crea un fichero aparte (**other_vhost..**), además genera una gran cantidad de ficheros 1,2,3... por la rotación que realiza.

*Realiza una breve comprobación de los ficheros de log existentes en tu servidor, y de su contenido. Comprueba que los accesos a los distintos host virtuales que puedas tener se almacenan en un sitio distinto. **Comprueba que un error HTTP (por ejemplo 404), se almacena en access no en error.***

Para poder modificar el comportamiento de nuestro servidor tenemos directivas como:

- **ErrorLog:** Establece el nombre del archivo en el que el servidor registrará los **errores** que encuentre. *Si la ruta del archivo no es absoluta, entonces se asume que es relativa al ServerRoot.*
- **CustomLog:** Se utiliza para el registro de las solicitudes en el servidor(acces.log). Si la ubicación es relativa, también toma como referencia la directiva ServerRoot. También se le pasa el formato del log, con la misma estructura que LogFormat.
- **LogLevel:** Ajusta en nivel de los mensajes registrados en los registros de **errores**. Los siguientes niveles están disponibles, en orden decreciente de importancia:

Level	Description	Example
emerg	Emergencies - system is unusable.	"Child cannot open lock file. Exiting"
alert	Action must be taken immediately.	"getpwuid: couldn't determine user name from uid"
crit	Critical Conditions.	"socket: Failed to get a socket, exiting child"
error	Error conditions.	"Premature end of script headers"
warn	Warning conditions.	"child process 1234 did not exit, sending another SIGHUP"
notice	Normal but significant condition.	"httpd: caught SIGBUS, attempting to dump core in ..."
info	Informational.	"Server seems busy, (you may need to increase StartServers, or Min/MaxSpareServers)..."
debug	Debug-level messages	"Opening config file ..."

- **LogFormat:** Se utiliza con ErrorLog (en CustomLog se incluye en la propia directiva). El argumento es una cadena. Esta cadena se utiliza para registrar cada solicitud en el archivo de registro. Puedes encontrar una explicación más concreta en la documentación.

Custom Log Formats	
The format argument to the <code>LogFormat</code> and <code>CustomLog</code> directives is a string. This string is used to log each request to the log file. It can contain literal characters copied into the log files and the C-style control characters <code>"\n"</code> and <code>"\t"</code> to represent new-lines and tabs. Literal quotes and backslashes should be escaped with backslashes.	
The characteristics of the request itself are logged by placing <code>"%"</code> directives in the format string, which are replaced in the log file by the values as follows:	
Format String	Description
<code>%%</code>	The percent sign.
<code>%a</code>	Client IP address of the request (see the <code>mod_remoteip</code> module).
<code>%(c)a</code>	Underlying peer IP address of the connection (see the <code>mod_remoteip</code> module).
<code>%A</code>	Local IP-address.
<code>%B</code>	Size of response in bytes, excluding HTTP headers.
<code>%b</code>	Size of response in bytes, excluding HTTP headers. In CLF format, i.e. a <code>.-.</code> rather than a 0 when no bytes are sent.
<code>%(VARNAME)c</code>	The contents of cookie <code>VARNAME</code> in the request sent to the server. Only version 0 cookies are fully supported.
<code>%D</code>	The time taken to serve the request, in microseconds.
<code>%(VARNAME)e</code>	The contents of the environment variable <code>VARNAME</code> .
<code>%f</code>	Filename.
<code>%h</code>	Remote hostname. Will log the IP address if <code>HostnameLookups</code> is set to <code>Off</code> , which is the default. If it logs the hostname for only a few hosts, you probably have access control directives mentioning them by name. See the Require host documentation .
<code>%H</code>	The request protocol.
<code>%(VARNAME)I</code>	The contents of <code>VARNAME</code> : header line(s) in the request sent to the server. Changes made by other modules (e.g. <code>mod_headers</code>) affect this. If you're interested in what the request header was prior to when most modules would have modified it, use <code>mod_setenvif</code> to copy the header into an internal environment variable and log that value with the <code>%(VARNAME)e</code> described above.

Un ejemplo de utilización de estas directivas los podemos ver a continuación:

```
<VirtualHost *:80>
    ServerName www.example.com
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/example
    Alias /cseg /BACKUP
    <Directory /BACKUP>
        Require all granted
        Options Indexes FollowSymLinks
    </Directory>
    LogLevel info
    LogFormat "%h %l %u %t \"%r\" %>s %0 \"%{Referer}i\" \"%{User-Agent}i\"" combined
    ErrorLog /var/www/example/logs/error.log
    CustomLog /var/www/example/logs/access.log combined
</VirtualHost>
```

¡Atención! Incluir los logs DENTRO del espacio de un sitio web(DocumentRoot) puede ser peligroso si el tamaño de los logs crece demasiado.

G) Autenticación.

Apache ofrece varias herramientas para gestionar la autenticación para el acceso a las CARPETAS de nuestros sitios web. Podemos encontrar una [sección del manual dedicada a la autenticación](#), bastante útil. Tenemos desde una configuración básica hasta, por ejemplo, el uso de un directorio LDAP. Vamos a utilizar la primera de ellas, a modo de ejemplo, siendo conscientes de sus limitaciones.

En este punto, todavía no hemos configurado un servidor seguro, por lo que los datos de acceso que utilizaríamos viajarían en texto plano a través de la red.

Los módulos que deben estar cargados para que la autenticación básica funciones son:

- auth_basic
- authn_file
- authz_user

Para comprobar los módulos cargados:

```
$apachectl -M
```

```

administrador@srvub1804tomas:~$ apachectl -M
AH00558: apache2: Could not reliably determine the server's
fully to suppress this message
Loaded Modules:
core_module (static)
so_module (static)
watchdog_module (static)
http_module (static)
log_config_module (static)
logio_module (static)
version_module (static)
unixd_module (static)
access_compat_module (shared)
alias_module (shared)
auth_basic_module (shared)
authn_core_module (shared)
authn_file_module (shared)
authz_core_module (shared)
authz_host_module (shared)
authz_user_module (shared)
autoindex_module (shared)
deflate_module (shared)
dir_module (shared)
env_module (shared)
filter_module (shared)
mime_module (shared)
mpm_prefork_module (shared)
negotiation_module (shared)

```

En el apartado I, de instalación de módulos, se ve con detalle la gestión de los distintos módulos que puede gestionar Apache.

Para proteger el acceso a algún directorio de tu servidor, primero, necesitarás crear un fichero de contraseñas (el formato del fichero depende del método de autenticación elegido). Para la autenticación básica, usaremos un fichero de contraseña de tipo texto. Este fichero deberá estar en un sitio que no se pueda tener acceso desde la web. Esto también implica que nadie pueda descargarse el fichero de contraseñas. Por ejemplo, si tus documentos están guardados fuera de /var/www, NO PONDRÁS tu archivo de contraseñas en /var/www/passwd.

El fichero de contraseñas se crea con la utilidad [htpasswd](#) que viene con Apache. Para crear el fichero, escribiremos, te preguntará por una contraseña, y después te pedirá que la vuelvas a escribir para confirmarla:

```

$htpasswd -c /usr/local/apache/passwd/passwords
alumno

```

Con esto ya hemos añadido al usuario alumno a nuestro fichero de contraseñas de Apache.

Para conseguir que varí@s users tengan acceso, se deben utilizar las características de acceso a Grupos, en el [manual de apache está perfectamente explicado](#).

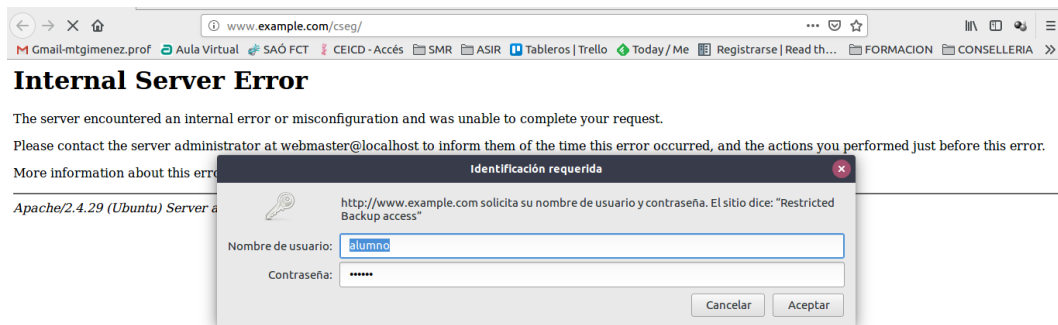
Lo siguiente, después de la generación de el/los fichero/s de usuarios/grupos será configurar el servidor para que pida los datos de acceso al acceder a un determinado **directorio (directiva Directory)**. Tal y como se muestra a continuación:


```
example.com.conf
/etc/apache2/sites-available

000-default.conf x example.com.conf x access

<VirtualHost *:80>
  ServerName www.example.com
  ServerAdmin webmaster@localhost
  DocumentRoot /var/www/example
  Alias /cseg /BACKUP
  <Directory /BACKUP>
    AuthType Basic
    AuthName "Restricted Backup access"
    AuthUserFile "/usr/local/apache/passwd/passwords"
    Require user alumno
    Options Indexes FollowSymLinks
  </Directory>
  LogLevel info
  LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" combined
  ErrorLog /var/www/example/logs/error.log
  CustomLog /var/www/example/logs/access.log combined
</VirtualHost>
```

Si en ese momento intentamos el acceso a la carpeta cseg:



Hemos obtenido un error, pero vamos a comprobar la utilidad de los ficheros de log vistos en el apartado anterior.

```
*error.log
/var/www/example/logs

000-default.conf x example.com.conf x access.log x *error.log x

[[Sun Dec 16 09:07:24.368087 2018] [authn_file:error] [pid 24572] (2)No such file or directory: [client 192.168.56.1:45272] AH01620: Could not open password file: /usr/local/apache/passwd/passwords
```

El log de error nos dice que el fichero de passwords al que estoy referenciando no existe, lo he indicado de manera incorrecta. Si corrijo la línea correspondiente en el fichero de configuración del virtual host, todo funcionara correctamente.

```
<Directory /BACKUP>
  AuthType Basic
  AuthName "Restricted Backup access"
  AuthUserFile "/etc/apache2/auth/htusers"
  Require user alumno
  Options Indexes FollowSymLinks
</Directory>
```

Otra utilidad relacionada con la autenticación, es la [activación de los directorios web de usuario](#).



H) Control de acceso

El control de acceso es otro de los elementos de seguridad que ofrece Apache, existe en la documentación oficial, un [how-to de control de acceso](#) bastante útil. También en castellano. Con él, podemos limitar el acceso a nuestra web, o a secciones de nuestra web, bien por IP, por direcciones de red o por otros métodos.

La directiva con la que se aplican las limitaciones de acceso es [REQUIRE](#), y junto con la directiva [ORDER](#), nos va a permitir conseguir la seguridad deseada.

En nuestro ejemplo, podríamos configurar el acceso a nuestra carpeta virtual cseg, con una configuración como la siguiente:

```
<Directory /BACKUP>
    Order Deny,Allow
    AuthType Basic
    AuthName "Restricted Backup access"
    AuthUserFile "/etc/apache2/auth/htusers"
    <RequireAll>
        Require user alumno
        Require ip 192.168.56
    </RequireAll>
    Options Indexes FollowSymLinks
</Directory>
```

Con esa configuración, además de pedir un usuario concreto, impedimos el acceso desde fuera de nuestra red local.

I) Instalación módulos

Como ya has imaginado, todas las funcionalidades de Apache, están incluidas en distintos módulos, que pueden ser instalados y desinstalados según nos interese. En la carpeta de configuración de Apache, al igual que ocurre con los host virtuales, existen unas carpetas modules-available y modules-enabled, para la gestión de los módulos.

Hasta ahora hemos visto funcionalidades de apache que vienen incluidas en los módulos activos por defecto, por lo que no hemos tenido que instalar ninguno, pero si queremos poder alojar sitios web que, por ejemplo, ejecuten código php, o accedan a BD Mysql o PostGreSQL, por poner algún ejemplo, es necesario que sepamos como gestionar los módulos.

En nuestro caso vamos a instalar los módulos necesarios para poder ejecutar PHP y trabajar con bases de datos MYSQL. Aunque viene una instalación por defecto en la versión de Apache que se obtiene desde los repositorios de

Ubuntu (**ver la salida de un phpinfo para comprobar**), vamos a instalar la última versión de cada módulo.

- Para activar/desactivar módulos tenemos los comandos `a2enmod/a2dismod`, una vez desactivados, con borrar los ficheros correspondientes en `mods-available`, ya tendríamos realizada la desinstalación.
- Para instalar módulos, normalmente con un sencillo `apt-get` realizaríamos la instalación y configuración inicial del módulo correspondiente (teniendo controlados los repositorios activos en nuestro sistema operativo).

En nuestro ejemplo, antes de instalar cualquier nuevo módulo:

```
administrador@ubuntu_server_18_04:~$ apachectl -M
Loaded Modules:
  core_module (static)
  so_module (static)
  watchdog_module (static)
  http_module (static)
  log_config_module (static)
  logio_module (static)
  version_module (static)
  unixd_module (static)
  access_compat_module (shared)
  alias_module (shared)
  auth_basic_module (shared)
  authn_core_module (shared)
  authn_file_module (shared)
  authz_core_module (shared)
  authz_host_module (shared)
  authz_user_module (shared)
  autoindex_module (shared)
  deflate_module (shared)
  dir_module (shared)
  env_module (shared)
  filter_module (shared)
  mime_module (shared)
  mpm_event_module (shared)
  negotiation_module (shared)
  reqtimeout_module (shared)
  setenvif_module (shared)
  status_module (shared)
```

Realizamos la instalación para la versión 7.2 de PHP, la librería para mysql y para postgres. (solo librerías, ya que no necesariamente el servidor de BD tiene que estar alojado con Apache ni la instalación completa de PHP tampoco)

```
administrador@ubuntu_server_18_04:~$ sudo apt-get install libapache2-mod-php php7.2-mysql php7.2-pgsql
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
```

Después deberíamos activar los módulos correspondientes:

```
administrador@ubuntu_server_18_04:~$ sudo phpenmod mysql
administrador@ubuntu_server_18_04:~$ sudo phpenmod mysqlnd
administrador@ubuntu_server_18_04:~$ sudo phpenmod pdo_mysql
administrador@ubuntu_server_18_04:~$ sudo phpenmod pdo_pgsql
administrador@ubuntu_server_18_04:~$ sudo phpenmod pgsql
administrador@ubuntu_server_18_04:~$ apache2ctl -M
apache2      apache2ctl  apachectl
administrador@ubuntu_server_18_04:~$ apache2ctl -M
apache2      apache2ctl
administrador@ubuntu_server_18_04:~$ apache2ctl restart
```

Con esto ya tendríamos la instalación mínima realizada para poder ejecutar PHP, mysql y postgres en nuestro servidor (comprobar phpinfo). En este caso suponemos que el servidor de BD está alojado en otro equipo. Si quisiéramos instalarlo TODO en nuestro equipo el proceso sería distinto, existen [multitud de tutoriales en la web](#).

J) Ficheros .htaccess

Apache nos ofrece todavía más posibilidades de personalización, a través de los ficheros .htaccess ([acceso a la documentación](#)). Con estos ficheros permitimos a los usuarios de nuestro servidor web el modificar algunas configuraciones, siempre que lo autoricemos y siempre dentro de sus directorios.

Deberías evitar usar ficheros .htaccess completamente si se tiene acceso al fichero de configuración principal de httpd (o sea, si somos administradores de Apache). Usar ficheros .htaccess ralentiza su servidor Apache http. Cualquier directiva que pueda incluir en un fichero .htaccess estará mejor configurada dentro de una sección Directory, tendrá el mismo efecto y mejor rendimiento. Contienen una o más directivas, se coloca en un documento específico de un directorio, y estas directivas aplican a ese directorio y todos sus subdirectorios.

Si quiere llamar a su fichero .htaccess de otra manera, puede cambiar el nombre del fichero usando la directiva AccessFileName.

Los ficheros .htaccess usan la misma sintaxis que los ficheros de la configuración principal. Lo que puede utilizar en estos ficheros lo determina la directiva [AllowOverride](#). Esta directiva especifica, en categorías, qué directivas tendrán efecto si se encuentran en un fichero .htaccess.

Por ejemplo, modificamos en nuestro servidor la configuración de la carpeta logs que hemos creado en el apdo de logs de manera que añadimos el AllowOverride, y quitamos el permiso de Indexes.

```
<VirtualHost *:80>
    ServerName www.example.com
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/example
    Alias /cseeg /BACKUP
    <Directory /BACKUP>
        Order Deny,Allow
        AuthType Basic
        AuthName "Restricted Backup access"
        AuthUserFile "/etc/apache2/auth/htusers"
        <RequireAll>
            Require user alumno
            Require ip 192.168.56
        </RequireAll>
        Options Indexes FollowSymLinks
    </Directory>
    <Directory /var/www/example>
        Options -Indexes
        AllowOverride All
    </Directory>
    LogLevel info
    LogFormat "%h %l %u %t \"%r\" %> %0 \"%{Referer}i\" \"%{User-Agent}i\"" combined
    ErrorLog /var/www/example/logs/error.log
    CustomLog /var/www/example/logs/access.log combined
</VirtualHost>
```

No mostramos el contenido del directorio, pero permitimos que, mediante un fichero .htaccess el gestor de esa web pueda modificar esto. En este caso, en la carpeta `/var/www/example` crearíamos un fichero llamado .htaccess con el siguiente contenido.

```
Open ▾ ⓘ .htaccess
/var/www/example/logs
Options +Indexes
```

Con esto permitimos el indizado del directorio logs

K) El módulo *rewrite*

Este módulo nos va a permitir ‘engañar’ a l@s usuari@s de nuestros sitios web al acceder a una URL diferente en realidad a la que se esté introduciendo en la barra de direcciones del navegador. Es una herramienta muy útil de Apache, tanto que tiene su propio [tutorial en la documentación oficial](#). Entre sus utilidades puedes encontrar:

1. Crear URL amigables más fáciles de recordar y mejor posicionadas por los motores de búsqueda.
2. ‘Ocultar’ la estructura real de nuestro sitio web, escondiendo la ruta de directorios en la que realmente se encuentra la página accedida.
3. Redirigir todos los accesos por HTTP a HTTPS, evitando accesos inseguros a nuestro sitio.
4. Redirigir de un dominio a otro, si hemos cambiado de dominio pero tenemos usuari@s que acceden por la dirección antigua (la dirección antigua debe estar todavía en los servidores DNS también)
5.

Puedes encontrar varios ejemplos en [esta web](#), con la que comenzar a practicar con esta potente herramienta.

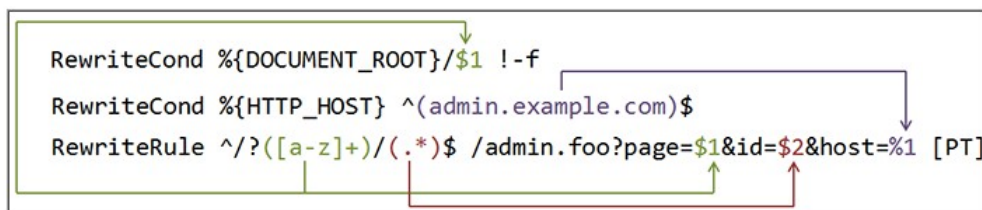


Figure 1: The back-reference flow through a rule.

In this example, a request for `/test/1234` would be transformed into `/admin.foo?page=test&id=1234&host=admin.example.com`.

Ejemplo de uso en la documentación oficial de Apache

4. NGINX

Este programa (se pronuncia EngineX) está logrando cada vez mayor relevancia en la web, ya que puede utilizarse para varias cosas:

- A) Servidor web
- B) Proxy inverso → balanceador de carga
- C) Mail proxy

Como servidor web, NginX nació sobre la idea de mejorar el rendimiento de Apache, ya que con el aumento de las velocidades de conexión y de número de usuarios de muchos servicios web, la sobrecarga de los servidores empieza a ser el cuello de botella en algunos casos.

La diferencia principal entre Apache y Nginx es la gestión que realizan del multi-procesamiento.

1) APACHE: A través de los módulos mpm de multiprocesamiento, se basan en la creación de diferentes procesos por cada cliente, y diferentes hilos en cada proceso. Esto conlleva la sobrecarga propia de recursos tanto en memoria como en CPU en la gestión de los mismos, en los intercambios de estado entre procesos y en los bloqueos hasta finalización de tareas.

2) NGINX: Usa una arquitectura basada en eventos, realiza la concurrencia sin un proceso o hilo adicional para cada conexión nueva, y es que un único procedimiento nginx es capaz de procesar miles de conexiones HTTP simultáneamente. Esto se lleva a cabo mediante un mecanismo de bucles conocido como event loop o bucle de eventos, que permite procesar las solicitudes de los clientes de manera asíncrona en un hilo.

Lamentablemente, NGINX no tiene una documentación tan buena como Apache. Aunque están haciendo esfuerzos por conseguirla. La [wiki de NGINX](#) puede ser un buen punto de partida para conocer el programa. **También puede ser recomendable el siguiente enlace.**

A) Instalación y puesta en marcha

NginX ya se encuentra instalado en la mayoría de repositorios de las distribuciones de Linux, entre ellas las de Ubuntu, por lo que con un sencillo comando ya lo tenemos instalado en nuestro servidor.:

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

Una vez finalizada la instalación en nuestra máquina de pruebas, podemos ver que no arranca el servicio por que el puerto 80 ya está en uso. Recuerda que tienes una instalación de Apache funcionando por ese puerto ya, y que dos aplicaciones no pueden compartir puerto.

```
Not attempting to start NGINX, port 80 is already in use.  
Configurando nginx (1.14.0-0ubuntu1.2) ...  
Procesando disparadores para ureadahead (0.100.0-20) ...  
Procesando disparadores para ufw (0.35-5) ...
```

Es un fallo muy típico tener algún servicio corriendo sobre algún puerto y querer arrancar otro sobre él. En este caso NGINX nos avisa, pero no siempre es así. Para evitar esto, o desactivamos Apache, o cambiamos el número de puerto sobre el que funciona el servidor.

Para desactivar en el arranque los servicios..systemctl enable/disable

Una vez instalado podemos gestionarlo de varias maneras:
Con la llamada que ya conocemos para otros servicios:

```
$sudo service nginx start/stop/reload/status
```

Llamando al comando nginx:

```
$sudo nginx  
$sudo nginx -s start/stop/..  
$sudo nginx -t (probar la configuración)
```

(atención: si el server ya está funcionando devuelve un error como el de la imagen siguiente)

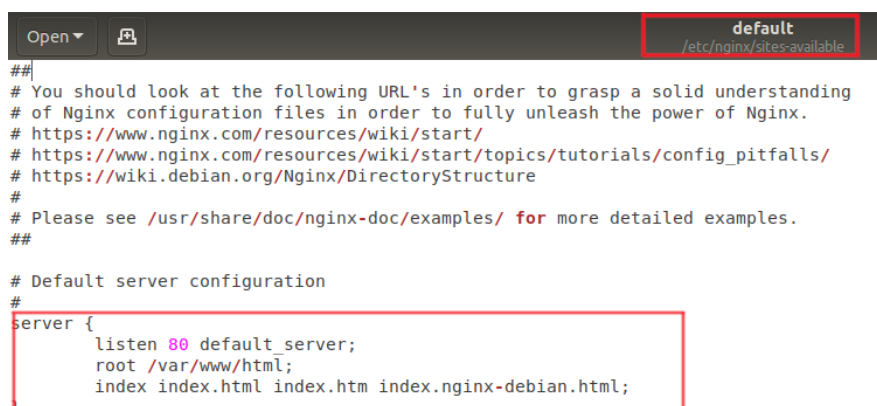
```
admin@administrador@srvub1804tomas:~$ sudo nginx  
[sudo] password for administrador:  
nginx: [emerg] bind() to 0.0.0.0:8080 failed (98: Address already in use)  
nginx: [emerg] bind() to [::]:8080 failed (98: Address already in use)  
nginx: [emerg] bind() to 0.0.0.0:8080 failed (98: Address already in use)  
nginx: [emerg] bind() to [::]:8080 failed (98: Address already in use)  
nginx: [emerg] bind() to 0.0.0.0:8080 failed (98: Address already in use)  
nginx: [emerg] bind() to [::]:8080 failed (98: Address already in use)  
nginx: [emerg] bind() to 0.0.0.0:8080 failed (98: Address already in use)  
nginx: [emerg] bind() to [::]:8080 failed (98: Address already in use)  
nginx: [emerg] bind() to 0.0.0.0:8080 failed (98: Address already in use)  
nginx: [emerg] bind() to [::]:8080 failed (98: Address already in use)  
nginx: [emerg] still could not bind()
```

B) Configuración básica

La configuración básica se encuentra en el fichero `/etc/nginx/nginx.conf`, y su sintaxis varía un poco respecto a Apache. Las directivas se agrupan dentro de llaves.

Existen unas carpetas `sites-available` y `sites-enabled`, similares a las de Apache. En ellas creamos un fichero por cada virtual host que queremos alojar. Un ejemplo sencillo sería:

Fichero por defecto (Sin virtual hosting):



```
Open ▾ [icon] default /etc/nginx/sites-available  
##  
# You should look at the following URL's in order to grasp a solid understanding  
# of Nginx configuration files in order to fully unleash the power of Nginx.  
# https://www.nginx.com/resources/wiki/start/  
# https://www.nginx.com/resources/wiki/start/topics/tutorials/config_pitfalls/  
# https://wiki.debian.org/Nginx/DirectoryStructure  
#  
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.  
##  
  
# Default server configuration  
#  
server {  
    listen 80 default_server;  
    root /var/www/html;  
    index index.html index.htm index.nginx-debian.html;  
}
```

B) Virtual hosting

Aquí sí que nos encontramos con una filosofía calcada a Apache, y que se resume en las siguientes tareas:

1) Crear el fichero para el virtual host en `/etc/nginx/sites-available`

- 2) Definir la ruta al contenido web, el orden de los ficheros por defecto que buscará y finalmente el nombre que espera para ese host virtual.
- 3) Crear el enlace simbólico correspondiente en `/etc/nginx/sites-available` . (A mano, aquí no tenemos el comando `a2ensite`)

Un fichero de ejemplo sería:

```
Open ▾  example.com
/etc/nginx/sites-available

# Virtual Host configuration for example.com
#
# You can move that to a different file under sites-available/ and symlink that
# to sites-enabled/ to enable it.

server {
    listen 80;
    server_name example.com;
    root /var/www/example;
    index index.html;
}
```

Para crear el enlace simbólico correspondiente, habría que ejecutar algo como lo siguiente:

```
administrador@srvubi804tomas:/etc/nginx/sites-enabled$ sudo ln -s ../sites-available/example.com example.com
administrador@srvubi804tomas:/etc/nginx/sites-enabled$ ls -l
total 0
lrwxrwxrwx 1 root root 34 dic 16 12:10 default -> /etc/nginx/sites-available/default
lrwxrwxrwx 1 root root 30 dic 16 16:49 example.com -> ../sites-available/example.com
administrador@srvubi804tomas:/etc/nginx/sites-enabled$ sudo nginx -s reload
```

C) Control de acceso

Podemos limitar los accesos a nuestra web, o a parte de ella, mediante la directiva `ALLOW|DENY`, la cual podemos aplicar a varios niveles. Para que funcione esto, el módulo [ngx_http_access_module](#) debe estar funcionando, en la instalación que hemos hecho (`nginx-full`), debería estar incluido:

A) A nivel de servidor: Incluyendo el `allow` en el fichero `nginx.conf`, dentro de `http`.

B) A nivel de virtual host, dentro del fichero correspondiente.

Por ejemplo en nuestro servidor de ejemplo:

```
# Default server configuration
#
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    allow 192.168.56.0/24;
    deny all;
```

Para comprobar los módulos que están cargados, podemos ejecutar `nginx -V`, si tenemos la necesidad de cargar alguno, deberíamos recompilar el programa.

D) Listado del contenido de directorios

Al igual que en Apache tenemos la directiva INDEXES para permitir listar el contenido de un directorio, si no se encuentra el correspondiente fichero index, en NGINX existe la opción de realizar lo mismo con la opción autoindex. Puedes encontrar un tutorial en la siguiente [página web](#). Esta configuración recuerda que se puede realizar a dos niveles.

A) A nivel de servidor: Para todas las carpetas de un sitio web. Incluyendo el autoindex on dentro de la clausura server{}. (ATENCIÓN ESTO PUEDE SER PELIGROSO)

B) Para una carpeta en concreto: Poniendo el autoindex on en una clausura location {}. Puedes ver como funciona el location en el [siguiente enlace](#).

Un ejemplo de donde pondríamos la directiva en cada caso sería:

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name _;
    allow 192.168.56.0/24;
    deny all;
    root /var/www/html;
    index index.php index.html index.htm index.nginx-debian.html;
    A autoindex on;
    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ =404;
    }
    B location /test {
        autoindex on;
    }
}
```

E) Autenticación.

Esta sea, probablemente, la parte más similar a Apache. Podemos definir el fichero de usuarios con el comando htpasswd, para luego utilizarlo de manera muy similar que lo hicimos anteriormente. [Esta parte está muy bien explicada en la documentación oficial](#).

Recordar que htpasswd es una utilidad de apache, por lo que si hemos desinstalado ese servidor, debemos reinstalar apache2-utils (éste paquete no incluye el servidor).

```
administrador@ubuntu_server_18_04:~$ sudo htpasswd -c /etc/nginx/.htpasswd user1
[sudo] password for administrador:
New password:
Re-type new password:
Adding password for user user1
administrador@ubuntu_server_18_04:~$ sudo htpasswd /etc/nginx/.htpasswd user2
New password:
Re-type new password:
Adding password for user user2
administrador@ubuntu_server_18_04:~$ cat /etc/nginx/.htpasswd
user1:$apr1$MqqXn6az$gGDPccG961iIo2UxNEmUF0
user2:$apr1$JJQ5s9Iq$3mPeG1JGjZeECefUTCCZN1
```

Una vez ya tenemos el fichero creado, podemos aplicar la directiva [auth_basic](#) a diferentes niveles:

- **http** → **todo el servidor**
- **server** → **un virtualhost**
- **location** → **una carpeta, tipo de fichero...**
- **limit_except**

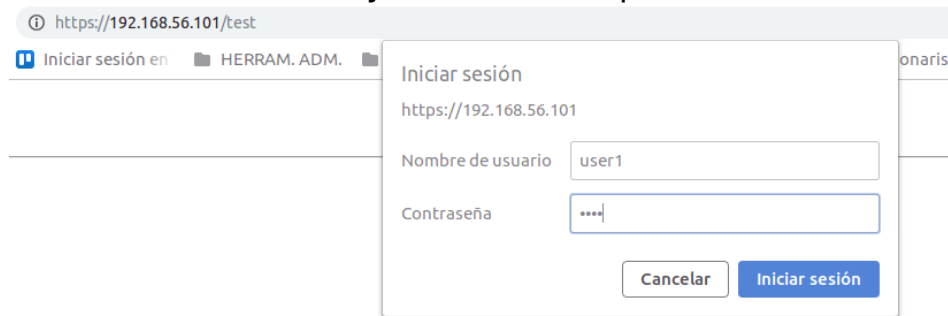
Veamos un ejemplo completo:

```
# Default server configuration
#
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name _;
    #allow 192.168.56.0/24;
    #deny all;
    root /var/www/html;
    index index.php index.html index.htm index.nginx-debian.html;

    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ =404;
    }

    location /test {
        autoindex on;
        auth_basic "Administrator's area";
        auth_basic user file /etc/nginx/.htpasswd;
    }
}
```

Si todo funciona correctamente, ya sabemos lo que debemos obtener:



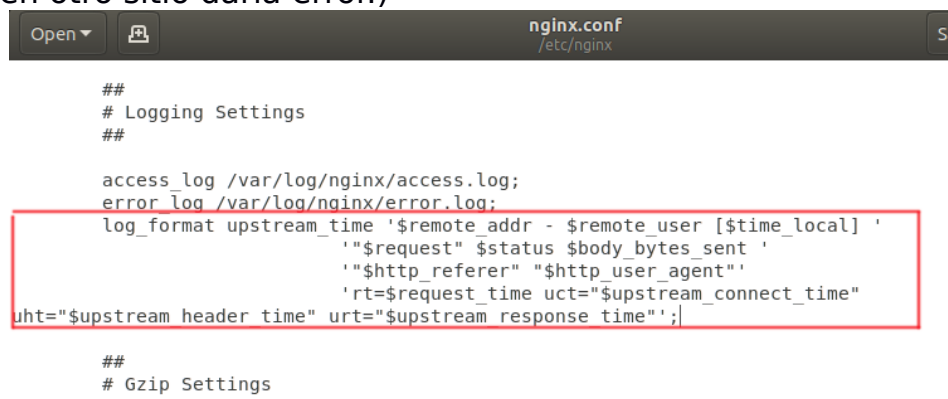
F) Logs del servidor.

NGINX también permite el control sobre el log de nuestros servidores, [tal y como explican en la documentación oficial](#). Las directivas principales son [error_log](#), y [acces_log](#) las cuales podemos aplicar a nivel de server, location...

Por defecto, NGINX guarda sus logs en `/var/log/nginx`, y con la misma estructura que Apache, un fichero `acces.log` y otro `error.log`

Para poder cambiar esto, un ejemplo sería:

En el fichero `nginx.conf` ponemos el formato que queremos para el log (si lo ponemos en otro sitio daría error.)



En el fichero del virtual host, por ejemplo, redirigimos el log para un directorio en concreto:

```
Open ▾  default  
/etc/nginx/sites-available  
Open a file  
location /test {  
    autoindex on;  
    auth_basic "Administrator's area";  
    auth_basic_user_file /etc/nginx/.httpasswd;  
    access_log /var/log/nginx/test_access.log upstream_time;  
}
```

G) Ejecución de páginas dinámicas

Para ejecutar páginas dinámicas, con lenguajes del lado del servidor, tipo PHP, debemos configurar QUIEN atiende esas peticiones, y como siempre se trata de un módulo concreto. Pero antes de nada debemos comprender que, por ejemplo PHP, puede ser atendido de varias maneras. [Puedes encontrar una explicación en el siguiente enlace.](#)

En apache sí existe un módulo propio de php, además de poderlo ejecutar con las otros dos opciones. Sin embargo en nginx debemos atender las peticiones php vía CGI/fastCGI. Para conseguirlo tenemos que realizar lo siguiente:

- 1.- Instalar el módulo correspondiente. En nuestro ejemplo el de fastCGI.

```
$sudo apt-get install php-fpm
```

- 2.- Añadir lo siguiente al fichero de configuración:

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name _;
    allow 192.168.56.0/24;
    deny all;
    root /var/www/html;
    index index.php index.html index.htm index.nginx-debian.html;
    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php7.0-fpm.sock;
    }
    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ =404;
    }
}
```

Con esto ya tendríamos la posibilidad de ejecutar código php en nuestro servidor NGINX.

