

UT5: El servicio http: **Apache** **2.4.**

Servicios en Red - 2º Curso CFGM SMR

Características generales.



▶ Apache:

- ▶ Aplicación “estrella” del mundo Linux.
- ▶ Es el servidor web más utilizado en el mundo en estos momentos.
- ▶ Algunas de sus características:
 - ▶ Es estable.
 - ▶ Multiplataforma: Windows, Linux, MacOS, ...
 - ▶ Altamente configurable.
 - ▶ Modular.
 - ▶ Actualmente se utiliza la versión 2.4 (Ubuntu 15.04).
 - ▶ La versión anterior era la 2.2 y algunas de las directivas (Require funcionaban de diferente manera).



Instalación y arranque del servidor.

- ▶ El nombre del paquete que debemos es **apache2**.
 - ▶ `sudo apt-get update`
 - ▶ `sudo apt-get install apache2`
- ▶ Para comprobar que se ha instalado nos conectaremos al host local <http://localhost>.
- ▶ Para arrancar, parar y comprobar el estado del servicio:
 - ▶ `sudo service apache2 start`
 - ▶ `sudo service apache2 stop`
 - ▶ `sudo service apache2 status`
 - ▶ `sudo service apache2 reload` → únicamente relee los archivos de configuración



Ficheros de configuración.

- ▶ Apache se configura poniendo directivas en archivos de configuración. El principal se llama `apache2.conf` o `httpd.conf`.
 - ▶ Además pueden existir otros archivos que han de indicarse en `apache2.conf` mediante la directiva `include`.
 - ▶ El directorio de configuración por defecto es:
 - ▶ **/etc/apache2**
 - ▶ Ficheros.
 - ▶ **Apache2.conf:** fichero de configuración genérico.
 - ▶ **Ports.conf:** puertos en los que escucha.
 - ▶ **Httpd.conf:** archivo vacío no usado en Ubuntu.
 - ▶ Directorios.
 - ▶ **Sites-available:** lista de archivos de configuración de los sitios preparados para ser usados.
 - ▶ **Sites-enabled:** enlaces simbólicos a los sitios (archivos de conf) de sites-available que están siendo usados..
 - ▶ **Mods-available:** lista de módulos disponibles.
-
- ▶ **Mods-enabled:** links simbólicos a módulos. Son los habilitados.
 - ▶ **Conf.d:** archivos de configuración de algunas aplicaciones.

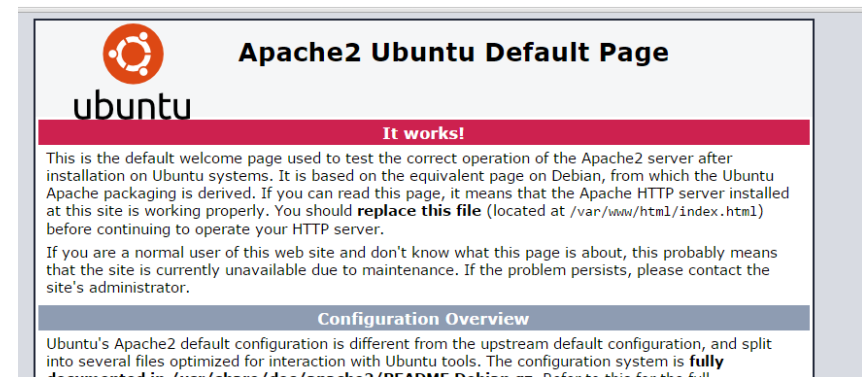
Despliegue web en Apache.

- ▶ Para poner en marcha un servidor web Apache, como todo los servidores web, hay que tener claros dos planos:
 - ▶ El contenido.
 - ▶ Se entiende por contenido, las páginas web que componen el sitio, es decir, los ficheros html, php, ...
 - ▶ La configuración del sitio.
 - ▶ Para que las páginas anteriores sean accedidas desde los clientes, debemos crear un fichero de configuración que especifique cuál será el comportamiento del sitio.

Nada más instalar apache, se crea y activa un sitio por defecto.

-El contenido está en `/var/www/html/index.html`.

-El fichero de configuración lo puedes ver en `/etc/apache2/sites-available/000-default.conf`



Despliegue web Apache: El contenido.

- ▶ **Antes de empezar a configurar el servidor debemos de tener el contenido. Para ello debemos saber y crear:**
 - ▶ **Cuál será el nombre a través del cuál será accesible el sitio.** Ejemplo: www.sri.com
 - ▶ **En qué puerto va a servir el sitio web anterior.** Ejemplo: puerto 80.
 - ▶ **Decidir y crear cuál será el directorio raíz en el cual se almacenará el sitio.** Ejemplo: /var/www/html/sri
 - ▶ **Crear la estructura de directorios.** Ejemplo: si va a tener subdirectorios o directorios virtuales.
 - ▶ **Crear y almacenar las páginas web que serán los documentos predeterminados.** Ejemplo: por ejemplo index.html.
 - ▶ **Tener claros qué permisos tendrán los clientes al acceder a las páginas en cada directorio.** Ejemplo: si se pedirá autenticación para algunas páginas del sitio.
 - ▶ **Tener claras que otras características de los directorios.** Ejemplo: mostrar contenidos de directorios, etc.
 - ▶ **Si algún subdirectorio podrá ser administrado por un usuario sin intervención del administrador.** Ejemplo: se dispone de ficheros .htaccess.
 - ▶ **Saber si se tratará de un sitio seguro.** Ejemplo: la web es accesible con <https://www.sri.com>.
-



Despliegue web Apache: Configuración.

- ▶ Para aplicar las especificaciones anteriores hay que crear un fichero de configuración que se almacenará en `/etc/apache2/sites-available` y cuyo nombre deberá acabar en `.conf`.

Ejemplo: `wwwasir2.conf`.

- ▶ Los ficheros de configuración están formados por directivas.
- ▶ Contendrán una directiva por línea.
- ▶ En las directivas no se hace distinción entre mayúsculas y minúsculas.
- ▶ Para chequear la sintaxis se pueden usar los siguientes comandos:
 - ▶ `apache2ctl configtest`
- ▶ El log de errores del servicio están en archivo:
 - ▶ `/var/log/apache2/error.log`
 - ▶ `tail -f /var/log/apache2/error.log`
- ▶ También existe un log de accesos al servidor que se almacena en:
 - ▶ `/var/log/apache2/access.log`



Configuración básica del servidor.

► Directivas

- Determinan la manera en que debe comportarse el servidor.
- Se incluyen dentro de los archivos de configuración.
- Dependiendo del entorno en el que se escriban afectarán al servidor general, a un servidor virtual, a un directorio, fichero, etc.
- Hay directivas que actúan como contenedores de otras.
- Podemos dividir las en tres grupos:
 - **Directivas configuración general:** definen los parámetros del servidor principal y las opciones por defecto para los hosts virtuales.
 - **Directivas de secciones:** configuran partes concretas del funcionamiento de apache.
 - ~~► **Optimización:** configuran los recursos y la eficacia de apache.~~

Configuración básica del servidor.

► Directivas de configuración general.

► ServerAdmin

- Dirección de correo del administrador de la web.
- ServerAdmin root@asir.net

► ServerName

- Especifica el nombre del host que se indicará en la
- ServerName www.asir.net

► DocumentRoot

- Carpeta raíz donde se ubica el servidor. Ojo, si hay Alias no es esta.
- DocumentRoot "/var/www/asir"

► DirectoryIndex

- Fichero a buscar en caso de que no se
- DirectoryIndex index.html indice.html

►► Alias

OJO!!!! partir de Apache2.4 en el fichero de configuración /etc/apache2/apache2.conf, existe una directiva que impide el acceso al contenido bajo el directorio /. Por esa razón es necesario permitirlo explícitamente utilizando directivas Require explicadas más adelante.

Configuración básica del servidor.

► Directivas de configuración general.

- Permite colocar contenido web fuera de DocumentRoot.
- Alias “url-path” “recurso”
- Alias /wiki /home/alumno/wiki
- Se utiliza para crear directorios virtuales

OJO!!!! partir de Apache2.4 en el fichero de configuración /etc/apache2/apache2.conf, existe una directiva que impide el acceso al contenido bajo el directorio /. Por esa razón es necesario permitirlo explícitamente utilizando directivas Require explicadas más adelante.



Actividad.

- ▶ Instalar Apache.
 - ▶ Comprobar que funciona.
 - ▶ Comprobar los procesos y que el servidor se está ejecutando.
 - ▶ Acceder desde un cliente y comprobar en el archivo de accesos al servidor la IP de la máquina cliente.
 - ▶ ¿Cuáles son los sitios disponibles? ¿Cuál es el sitio que está activo en este momento?
 - ▶ Verificar los valores de las directivas en el archivo de configuración del sitio activo.
 - ▶ Verificar el funcionamiento de DirectoryIndex del sitio activo.
 - ▶ Para ello crear un documento html que muestre “Bienvenido al servidor web por defecto de TuNombre” y que sea ese el archivo que se muestre por defecto.
 - ▶ Prueba ahora a poner DirectoryIndex apuntando a un archivo que no existe e intenta explicar lo que ocurre.
 - ▶ Crear un directorio virtual apuntando al home del usuario que estás usando, almacena un fichero html y comprueba que puedes accederlo.
-

Configuración básica del servidor.

► Directivas de secciones:

► <Directory /directorio> ...</Directory>

Engloba una o más directivas de configuración que sólo se aplican al directorio y subdirectorios especificados. Dentro, en el lugar de los ... se usan las directivas siguientes:

- **DirectoryIndex:** Página de inicio
- **Options:** controla que características estarán disponibles para un directorio particular
 - **[+/-]indexes:** muestra el contenido del directorio si no encuentra un archivo índice.
 - **[+/-] FollowSymLinks:** permite seguir enlaces simbólicos dentro del directorio.
 - **[+/-] Multiviews:** sirve para no tener que especificar en la URL la extensión de un archivo situado en este directorio. Por ejemplo: si un servidor tiene en el directorio /var/www/misitio/web.html cuando un cliente escribe en la URL <http://www.loquesea.com/misitio/web>, si la opción multiviews está puesta para el directorio /var/www/misitio/ buscará algún archivo llamado web.* y mostrará el que encuentre. También serviría si pusiéramos DirectoryIndex indice y no especificáramos extensión.
 - Si ponemos **Options All** activará todas las opciones disponibles salvo Multiviews que hay que especificarla.
 - Por defecto se heredan los parámetros de los directorios superiores. Para eliminar una opción, deberemos poner un signo "-" delante. Ejemplo: -Indexes.



Configuración básica del servidor.

► Códigos de error.

► ErrorDocument

- Permite mostrar mensajes y páginas de error personalizadas.
- Si queremos que se muestre un texto concreto para un error concreto la sintaxis es la siguiente:

ErrorDocument XXX "Texto" donde XXX es el código de error correspondiente.

- Ejemplo: ErrorDocument 404 "Página no encontrada en el sitio"
- Si queremos que muestre un página:

ErrorDocument XXX Fichero, donde en fichero debes especificar el fichero incluyendo la ruta para accederlo.

- Ejemplo: ErrorDocument 404 /404.html, donde 404.html está en la raíz del sitio (si es /var/www, ese será el directorio donde debes colocar la página 404.html).



Configuración básica del servidor.

► Ficheros de log.

► Loglevel

- Nivel de “gravedad” de los mensajes de error que se guardarán en el fichero de log.
- Por defecto, warn (avisos).

► ErrorLog

- Especifica donde se va a encontrar el fichero de log de errores.
- Por defecto es `/var/log/apache2/error.log`.

► CustomLog

- En esta directiva especificaremos en qué fichero queremos registrar los accesos al sitio web que estamos configurando y el formato que se utilizará para guardar dichos registros.
- Por defecto, se guardan en `/var/log/apache2/access.log`, y el formato será combined (mirar la documentación para entender qué campos son los que se registran con este formato).



Módulos: instalación, configuración y uso.

► Módulos en apache.

- Un módulo en apache es una manera de agrupar funcionalidades para el servidor.
- Los módulos podrán incluirse o excluirse.
- Razón: no todos los servidores requieran las mismas funcionalidades.
 - Ejemplo: una instalación que utiliza php puede no requerir Java o puede que no todas requieran hosting virtual.
 - Por lo tanto, si fueran incluidas todas las funcionalidades posibles en una versión única de Apache, esto lo haría sumamente pesado en cuanto a requerimientos de Memoria RAM y espacio en Disco Duro, por esto se opta por modularizar e incluir solo lo necesario



Módulos: instalación, configuración y uso.

- ▶ Tipos de módulos según la forma y el momento en qué se cargan.
 - ▶ **Módulos estáticos** (static):
 - Son aquellos que se compilan con apache2 cuando se realiza la instalación.
 - Para comprobar cuáles se han instalado: `apache2ctl -l`
 - ▶ **Módulos dinámicos** o compartidos DSO, Dynamic Shared Object (shared):
 - Se almacenan separados y pueden ser cargados opcionalmente.
 - Son los que aparecen en `/etc/apache2/mods-enabled`.
 - Estos son enlazados mientras que los estáticos son compilados.
 - ▶ `apache2ctl -t -D DUMP_MODULES`
- ▶ Tipos de módulos según funcionalidad. (clasificación en la documentación de apache2)
 - ▶ Funcionalidad básica
 - ▶ Funcionalidad de multiprocesamiento
 - ▶ Otras funcionalidades



Módulos: instalación, configuración y uso.

```
root@ubuntu:/etc/apache2# apache2ctl -t -D DUMP_MODULES
```

Loaded Modules:

```
core_module (static)
log_config_module (static)
logio_module (static)
mpm_worker_module (static)
http_module (static)
so_module (static)
alias_module (shared)
```

```
auth_basic_module (shared)
authn_file_module (shared)
authz_default_module (shared)
authz_groupfile_module (shared)
authz_host_module (shared)
authz_user_module (shared)
autoindex_module (shared)
cgid_module (shared)
deflate_module (shared)
dir_module (shared)
env_module (shared)
mime_module (shared)
negotiation_module (shared)
reqtimeout_module (shared)
setenvif_module (shared)
status_module (shared)
```

```
angelica@ubuntu:/etc/apache2/mods-available$ sudo apache2ctl -l
```

Compiled in modules:

```
core.c
mod_so.c
mod_watchdog.c
http_core.c
mod_log_config.c
mod_logio.c
mod_version.c
mod_unixd.c
```

```
root@ubuntu:/etc/apache2# ls mods-enabled/
```

alias.conf	authz_user.load	dir.conf	reqtimeout.conf
alias.load	autoindex.conf	dir.load	reqtimeout.load
auth_basic.load	autoindex.load	env.load	setenvif.conf
authn_file.load	cgid.conf	mime.conf	setenvif.load
authz_default.load	cgid.load	mime.load	status.conf
authz_groupfile.load	deflate.conf	negotiation.conf	status.load
authz_host.load	deflate.load	negotiation.load	

Syntax OK



Módulos: instalación, configuración y uso.

- ▶ Para activar y desactivar uno módulo dinámico se utilizan los comandos:
 - ▶ `a2enmod nombre_modulo` *//activa un módulo*
 - ▶ `a2dismod nombre_modulo` *//desactiva un módulo*
 - ▶ Tras activar o desactivar un módulo, siempre es necesario reiniciar el servicio apache2:
`sudo service apache2 restart`



Módulos: instalación, configuración y uso. Monitorización del servicio web.

Uso de módulos: Monitorización del servicio web.

► mod_status:

► **Descripción del módulo:**

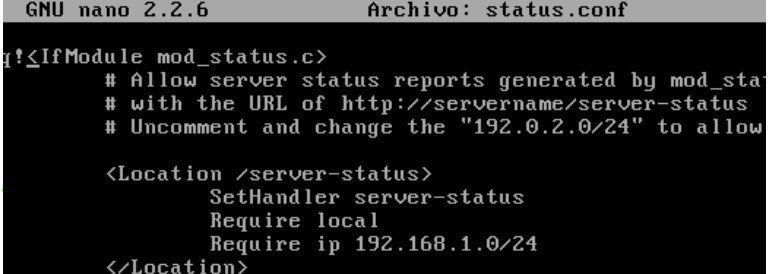
- Proporciona información sobre la actividad y el rendimiento del servidor.
- Se presenta una página html que muestra al administrado estadísticas del servidor.

► **Instalación del módulo:**

- Para habilitarlo: `sudo a2enmod status` *//recuerda reiniciar el servicio*

► **Configuración del módulo:**

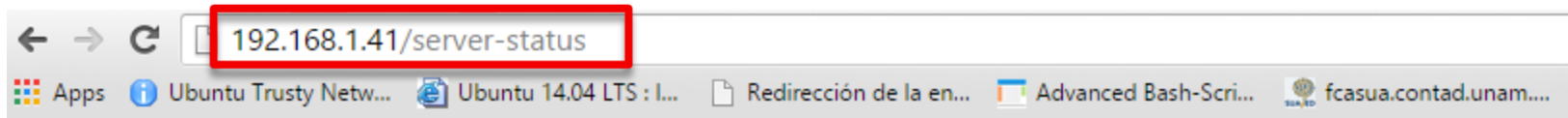
- Editaremos el fichero de configuración en `/etc/apache2/mods-enabled/status.conf` y permitiremos el acceso desde las IP, redes, hosts, o dominios que consideremos oportuno. En la imagen se permite el acceso desde cualquier equipo en la red `192.168.1.0/24`.
- Reiniciar el servicio tras el cambio.



```
GNU nano 2.2.6 Archivo: status.conf
# IfModule mod_status.c>
# Allow server status reports generated by mod_status
# with the URL of http://servername/server-status
# Uncomment and change the "192.0.2.0/24" to allow
# access from the desired hosts or host ranges.

<Location /server-status>
    SetHandler server-status
    Require local
    Require ip 192.168.1.0/24
</Location>
```

Módulos: instalación, configuración y uso. Monitorización del servicio web.



Apache Server Status for 192.168.1.41 (via 192.168.1.41)

Server Version: Apache/2.4.10 (Ubuntu)
Server MPM: event
Server Built: Jul 24 2015 17:25:17

Current Time: Wednesday, 02-Dec-2015 19:42:59 CET
Restart Time: Wednesday, 02-Dec-2015 19:42:55 CET
Parent Server Config. Generation: 1
Parent Server MPM Generation: 0
Server uptime: 4 seconds
Server load: 0.00 0.01 0.05
Total accesses: 0 - Total Traffic: 0 kB
CPU Usage: u0 s0 cu0 cs0
0 requests/sec - 0 B/second -
1 requests currently being processed, 49 idle workers

PID	Connections		Threads		Async connections		
	total	accepting	busy	idle	writing	keep-alive	closing
4148	0	yes	0	25	0	0	0
4149	0	yes	1	24	0	0	0
Sum	0		1	49	0	0	0

Aquí vemos lo que se muestra al acceder.
El equipo 192.168.1.41 tiene apache2
instalado con y el módulo status activado
y configurado según lo explicado en la
diapositiva anterior.

Módulos: instalación, configuración y uso. Monitorización del servicio web.

Uso de módulos: Monitorización del servicio web.

► mod_info:

► *Descripción del módulo:*

- Proporciona información resumida de la configuración del servidor.

► *Instalación del módulo:*

- Para habilitarlo: `sudo a2enmod info` //recuerda reiniciar el servicio

► *Configuración del módulo:*

- Editaremos el fichero de configuración en `/etc/apache2/mods-enabled/info.conf` y permitiremos el acceso desde las IP, redes, hosts, o dominios que consideremos oportuno. En la imagen se permite el acceso desde cualquier equipo en la red `192.168.1.0/24`.
- Reiniciar el servicio tras el cambio.

```
GNU nano 2.2.6 Archivo: info.conf

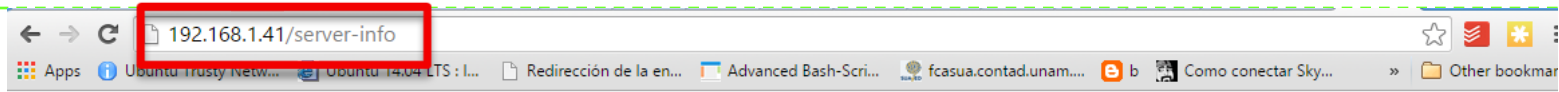
<IfModule mod_info.c>

    # Allow remote server configuration reports, with the URL
    # http://servername/server-info (requires that mod_info.c
    # Uncomment and change the "192.0.2.0/24" to allow access
    #
    <Location /server-info>
        SetHandler server-info
        Require local
        Require ip 192.168.1_0/24
    </Location>

</IfModule>

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

Módulos: instalación, configuración y uso. Monitorización del servicio web.



Apache Server Information

Subpages:

[Configuration Files](#), [Server Settings](#), [Module List](#), [Active Hooks](#), [Available Providers](#)

Sections:

[Loaded Modules](#), [Server Settings](#), [Startup Hooks](#), [Request Hooks](#), [Other Hooks](#), [Providers](#)

Loaded Modules

[core.c](#), [event.c](#), [http_core.c](#), [mod_access_compat.c](#), [mod_alias.c](#), [mod_auth_basic.c](#), [mod_authn_core.c](#), [mod_authn_file.c](#), [mod_authz_core.c](#), [mod_authz_host.c](#), [mod_authz_user.c](#), [mod_autoindex.c](#), [mod_deflate.c](#), [mod_dir.c](#), [mod_env.c](#), [mod_filter.c](#), [mod_info.c](#), [mod_log_config.c](#), [mod_logio.c](#), [mod_mime.c](#), [mod_negotiation.c](#), [mod_setenvif.c](#), [mod_soc.c](#), [mod_status.c](#), [mod_unixd.c](#), [mod_version.c](#), [mod_watchdog.c](#),

Server Settings

Server Version: Apache/2.4.10 (Ubuntu)
Server Built: Jul 24 2015 17:25:17
Server loaded APR Version: 1.5.1
Compiled with APR Version: 1.5.1
Server loaded APU Version: 1.5.4
Compiled with APU Version: 1.5.4
Module Magic Number: 20120211:37
Hostname/port: 192.168.1.41:80
Timeouts: connection: 300 keep-alive: 5
MPM Name: event
MPM Information: Max Daemons: 3 Threaded: yes Forked: yes
Server Architecture: 32-bit
Server Root: /etc/apache2
Config File: /etc/apache2/apache2.conf
Server Built With:
-D APR_HAS_SENDFILE
-D APR_HAS_MMAP
-D APR_HAVE_IPV6 (IPv4-mapped addresses enabled)
-D APR_USE_SYSVSEM_SERIALIZE
-D SINGLE_LISTEN_UNSERIALIZED_ACCEPT
-D APR_HAS_OTHER_CHILD
-D AP_HAVE_RELIABLE_PIPED_LOGS

Aquí vemos lo que se muestra al acceder.
El equipo 192.168.1.41 tiene apache2
instalado con y el módulo status activado
y configurado según lo explicado en la
diapositiva anterior.

Host virtuales. Creación, configuración y utilización.

- ▶ Configurando hosts virtuales podemos alojar varios sitios web (varios dominios) en el mismo servidor (en la misma máquina), siendo uno totalmente independiente del otro.
- ▶ El sitio por defecto que viene configurado (000-default) en apache es un host virtual más.
- ▶ Existen varios tipos de host virtuales:
 - ▶ **Basados en nombre:** permite alojar varios nombres de host o varios dominios en una misma máquina. Todos los hosts que compartan la misma IP deben declararse mediante la directiva NameVirtualHost.
 - ▶ **Basados en puerto:** se responde de diferente manera si se accede a un puerto u otro.
 - ▶ **Basados en IP:** una máquina tiene varias direcciones IP y a cada una de ellas servirá un sitio web diferente.
- ▶ Por ser los más utilizados, nos centramos en los basados en nombre.



Host virtuales. Creación, configuración y utilización.

- ▶ Se trata de servir desde la misma máquina, en la misma IP y a través del mismo puerto, dos sitios web, por ejemplo: `www.web1.com` y `www.web2.com`
 - ▶ Cada sitio deberá de tener su propio DocumentRoot, es decir, el directorio donde se almacenará el sitio.
 - ▶ Los pasos a seguir serían de manera esquemática los siguientes:
 - ▶ Crear para cada sitio el esquema de directorios donde se almacenará el contenido.
 - ▶ Crear las páginas web (el contenido) y almacenarlo en los directorios correctos.
 - ▶ Crear un fichero de configuración por cada sitio (en `sites-available`).
 - ▶ MUY IMPORTANTE: Hay que especificar las directivas `ServerName` y `DocumentRoot` pues será lo que diferencie a ambos sitios.
 - ▶ Activar ambos sitios (`a2ensite nombresitio`).
 - ▶ Añadir en el servidor DNS los dos dominios y los registros `www` correspondientes (tendrán la misma IP).
-

Actividad

- ▶ Crear dos hosts virtuales basados en nombre:
 - ▶ El sitio www.cocinillas.net, se almacena en /var/www/cocinillasnet y la página de inicio será index.html.
 - ▶ El sitio www.depesca.com, se almacena en /var/www/depesca.com y la página de inicio se llamará depesca.html.
 - ▶ Comprueba que puedes acceder correctamente desde un cliente diferente a la propia máquina de ubuntu.



Autenticación y control de acceso.

- ▶ Se trata de directivas que **permitirán o denegarán el acceso a partes del sitio web** que se esté configurando.
- ▶ Como hablamos de “PARTES” todas las directivas que se mencionan a continuación estarán siempre incluidas **dentro de una directiva <Directory>**.
- ▶ También podrán encontrarse en ficheros tipo **.htaccess**.
- ▶ Directiva Require: **Require [not] entity-name [entity-name.]**
 - Esta directiva comprueba si la entidad especificada en entity-name se autoriza o no se autoriza.
 - Require ip ip-address. *//Autoriza a la ip-address a acceder.*
 - Require host host-name. *//Permite el acceso a host-name.*
 - Require all granted *// Acceso permitido a todos incondicionalmente.*
 - Require all denied *// Acceso denegado a todos incondicionalmente.*
 - (***) Require not ip ip-address. *//Deniega el acceso a ip-address.*
 - (***) Require not host host-name. *//Deniega el acceso a host-name*

(*) CUANDO QUERAMOS USAR SENTENCIAS CON NOT SE DEBEN INCLUIR DENTRO DE UN BLOQUE EN EL QUE APAREZCA UN RANGO MAYOR. VER EJEMPLOS.**

Autenticación y control de acceso.

- ▶ Se pueden agrupar varias directivas Require en un bloque para exigir que se cumplan varias condiciones, o que se cumpla cualquiera o ninguna (lógica similar a las ACLs de los routers).

- ▶ Nosotros vamos a trabajar con:

`<RequireAll> ... </RequireAll>`

Encierra un grupo de directivas de autorización de las cuáles ninguna puede fallar y al menos una se debe cumplir para que se dé por satisfactorio la comprobación.

`<RequireAny> ...</RequireAny>`

Encierra un grupo de directivas de autorización de las cuáles se debe cumplir al menos una para que se dé por satisfactorio la comprobación.

- ▶ Podéis encontrar más detalles en la documentación oficial:

https://httpd.apache.org/docs/2.4/mod/mod_authz_core.html#requireall



Autenticación y control de acceso.

- **Ejemplo1:** En la siguiente imagen las páginas almacenadas bajo el directorio /var/www/angelica sólo serán accesibles por la IP 192.168.1.35

```
<Directory /var/www/angelica>  
    Require all denied  
    Require ip 192.168.1.35  
</Directory>
```

- **Ejemplo2:** En la siguiente situación, todas las IP tendrán acceso salvo el equipo 192.168.1.35. Aquí vemos que aparece “not ip” pero dentro de un bloque RequireAll.

```
<Directory /var/www/angelica>  
<RequireAll>  
    Require all granted  
    Require not ip 192.168.1.35  
</RequireAll>  
</Directory>
```

- **Ejemplo3:** En el siguiente ejemplo, sólo 192.168.1.35 tendrá acceso.

```
<Directory /var/www/angelica>  
<RequireAll>  
    Require all granted  
    Require ip 192.168.1.35  
</RequireAll>  
</Directory>
```



Autenticación y control de acceso.

- **Ejemplo 4:** En la siguiente imagen todos los equipos tendrán acceso.

```
<Directory /var/www/angelica>
<RequireAll>
    Require all granted
    Require ip 192.168.1.35
</RequireAll>
</Directory>
```

- **Ejemplo5:** Mismo efecto que Ejemplo 1. Sólo 192.168.1.35 tiene acceso.

```
<Directory /var/www/angelica>
<RequireAny>
    Require all denied
    Require ip 192.168.1.35
</RequireAny>
</Directory>
```

- **Ejemplo6:** En el siguiente ejemplo nadie tiene acceso.

```
<Directory /var/www/angelica>
<RequireAll>
    Require all denied
    Require ip 192.168.1.35
</RequireAll>
</Directory>
```



Autenticación y control de acceso.

- Para que al solicitar usuario y contraseña (autenticación http) para acceder a partes del sitio web que estemos configurando se deben añadir dentro de la directiva `<Directory>` o de `.htaccess` las siguientes directivas:

- **AuthName:** Con esta directiva especificamos la frase que aparecerá en la pantalla de autenticación.

Ejemplo: `AuthName "Este directorio es privado introduce tus datos."`

- **AuthType:** nos permite especificar el tipo de autenticación (basic/digest igual que en IIS).

Para usar la autenticación básica se podría requerir activar el módulo correspondiente.

Para ello: `sudo a2enmod auth_basic`

- **AuthUserFile:** archivo que contiene user/password a usar para autenticación

Ejemplo: `AuthUserFile /etc/apache2/.htpasswd`

- **Require user** usuario: selecciona qué usuario(s) del fichero son los que van a tener acceso al recurso.

Ejemplo: `Require user juan`

- **Requiere valid-user:** todos los usuarios del fichero tienen permiso de acceso.



Autenticación y control de acceso.

(continuación)

- **Requiere valid-user:** todos los usuarios del fichero tienen permiso de acceso.
- **Require group asir**

▪ Para poder usar la autenticación de usuarios hay que tener un fichero de usuarios y contraseñas. El nombre y la ruta ha de coincidir con la de AuthUserFile

sudo htpasswd -c /etc/apache2/.htpasswd usuario

// crea el archivo y añade usuario. El flag -c sólo se utiliza al crear el fichero.

sudo htpasswd /etc/apache2/.htpasswd juan

// para añadir el usuario juan al fichero que ya teníamos creado, no ponemos -c.



Autenticación y control de acceso.

► Ejemplo:

- Si quisiéramos que el acceso a una página web en /var/www/sri mediante usuario y contraseña, deberíamos de añadir al fichero de configuración del sitio las siguientes línea:

```
<VirtualHost *:80>
    ServerName www.asir.net
    DocumentRoot /var/www
    DirectoryIndex index.html
    <Directory /var/www/sri>
        DirectoryIndex sri.html
        Options Indexes, FollowSymLinks, Multiviews
        AuthName "Acceso sólo para alumnos matriculados."
        AuthType Basic
        AuthUserFile /web/users
        Require valid-users
    </Directory>
</VirtualHost>
```

- Asumimos que hemos creado el fichero /web/users ejecutando los siguientes comandos:

```
sudo htpasswd -c /web/users alumno1
sudo htpasswd /web/users alumno2, alumno3, alumno4
```



Autenticación y control de acceso.

► Los archivos .htaccess.

- Son archivos que permiten una **configuración personalizada de cada directorio** sin que intervenga el administrador.
- Cuando un cliente realiza una petición al servidor, este busca desde “/” hasta el subdirectorio **donde se encuentre el archivo solicitado**, el archivo **.htaccess** y tiene en cuenta las directivas especificadas en él para atender dicha petición.
- Estos archivos se leen en cada petición, y por tanto, no será necesario reiniciar el servicio.
- Para poder usar los archivos .htaccess debemos:
 - En el archivo de configuración de la web, añadir una directiva de tipo <Directory> para el directorio del usuario y añadir la opción AllowOverride All.
 - El usuario deberá crear en su directorio el archivo .htaccess en el que controlará la configuración de su parte del servidor.



Autenticación y control de acceso.

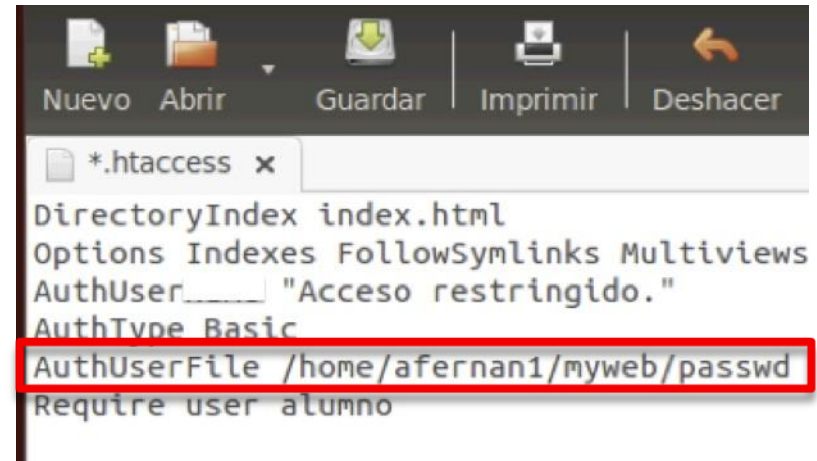
► Los archivos .htaccess. (continuación)

► Por ejemplo, si tengo una web en /home/afernani/myweb, puedo controlar su configuración creando un archivo .htaccess en ese directorio.

► Ese archivo podría ser:

- Podemos configurar un index.
- Las opciones para este directorio.
- Autenticación básica.

En este último caso, el user afernan1, debe crear su archivo passwd.



```
afernani@ubuntu:~/myweb$ htpasswd -c /home/afernani1/myweb/passwd alumno
New password:
Re-type new password:
Adding password for user alumno
```

Certificados. Servidores de certificados.

Servidor de Certificados en Ubuntu: openssl.

- ▶ OpenSSL es paquete de administración y bibliotecas relacionadas con criptografía que ayudan a implementar SSL en servicios de red. Entre otras cosas, permite generar certificados digitales.
- ▶ Instalaremos OpenSSL en Ubuntu para generar un certificado digital autofirmado y así poder servir sitios web a través del protocolo https.
- ▶ Los pasos a seguir serán los siguientes:

1. Instalar OpenSSL.

```
angelica@ubuntu:~$ sudo apt-get install openssl
```

2. Crear una clave privada: fichero.key. En la imagen, la clave que se guarda en servidor.key.

```
angelica@ubuntu:~$ sudo openssl genrsa -out servidor.key 2048
Generating RSA private key, 2048 bit long modulus
.....+++
..+++
e is 65537 (0x10001)
```

3. Crear un CSR – Certificate Signing Request, algo así como una solicitud de firma de certificado. Se tratará de un fichero.csr. Tras ejecutar el comando, pedirá una serie de datos de la empresa/dominio para el cual se solicita el certificado. Finalmente se deberá generar el fichero servidor.csr.

```
angelica@ubuntu:~$ sudo openssl req -new -key servidor.key -out servidor.csr
```

```
angelica@ubuntu:~$ ls
servidor.csr  servidor.key
```

4. Generación del certificado, fichero.crt. Este certificado se generará a partir de la clave generada y del CSR anteriores. En el ejemplo se generará un certificado digital en formato x509 válido durante 365 y que se almacenará en el fichero servidor.crt.

```
angelica@ubuntu:~$ sudo openssl x509 -req -days 365 -in servidor.csr -signkey servidor.key -out servidor.crt
Signature ok
subject=/C=SP/ST=Cantabria/L=Torrelavega/O=IES Miguel Herrero/OU=IESMHP/CN=www.iesmhp.local./emailAddress=test@iesmhp.local
Getting Private key
```

Certificados. Servidores de certificados.

Configurar SSL en Apache.

1. Lo primero será **activar el módulo ssl** correspondiente:
 - ▶ `sudo a2enmod ssl`
 - ▶ `service apache2 restart`
2. Nos aseguramos de que en `/etc/apache2/ports.conf` se encuentra la directiva **Listen 443**.
3. A continuación crearemos un nuevo directorio donde se almacenará la clave pública del servidor y el certificado autofirmado que posteriormente generaremos.
 - ▶ `mkdir /etc/apache2/ssl`
4. **Copiamos la clave** (`servidor.key`) **y el certificado** (`servidor.crt`) en el directorio `/etc/apache2/ssl` creado en el paso anterior.
5. Configuramos **el archivo de configuración del sitio seguro**.

RECOMENDACIÓN: Os recomiendo mantener el fichero de configuración no seguro y crear el fichero de configuración con seguridad a partir del anterior.

Por ejemplo, si tengo `miweb.conf`, hacer una copia y renombrarlo a `miweb-ssl.conf`.

Sobre `miweb-ssl.conf` hacer los pasos que se indican en la siguiente diapositiva.



Certificados. Servidores de certificados.

Sobre el fichero de configuración SSL deberéis hacer estas modificaciones:

1. Lo primero será hacer que el sitio se sirva a través del puerto https, es decir el 443: `<VirtualHost *:443>`
 1. `<VirtualHost *:443>`
 2. Añadiremos las líneas siguientes y nos aseguramos de que los directorios y los nombres de clave y certificado coinciden con los que hemos creado en los pasos anteriores:
 1. `SSLEngine on`
 2. `SSLCertificateFile /etc/apache2/ssl/apache2.crt`
 3. `SSLCertificateKeyFile /etc/apache2/ssl/apache2.key`
 3. Activaremos el sitio seguro que estemos creando si no está ya añadido:
 1. `sudo a2ensite miweb-ssl.conf`
 4. Y reiniciamos el servicio de apache:
 1. `sudo service apache2 restart`
 5. Para probar que funciona nos conectaremos a <https://www.nombredelaweb.com> y comprobamos que se descarga el certificado que hemos creado.
-



Más información ...

- ▶ En la web oficial de apache tenéis mucha más información sobre otras directivas, módulos, etc.
- ▶ Se encuentra publicada en español en la web:

<http://httpd.apache.org/docs/2.4/es/>

