

# TEMA 3: Apache

ADMINISTRACIÓN DE SERVICIOS

ADMINISTRACIÓN DE SISTEMAS OPERATIVOS

2020/2021

Cristian Martín Fernández

# Contenido

- Introducción servidor Web Apache
- Instalación servidor web Apache
- Configuraciones ubuntu
- Servicio Apache2ctl

# SERVICIO WEB APACHE


- Implementación más popular de **servidor web**
- Apache es el servidor web más utilizado en sistemas Linux
- Arquitectura modular
- LAMP – Linux Apache MySQL PHP

# INSTALACIÓN SERVICIO WEB APACHE

- Instalación del servidor Apache:
  - `$ sudo apt install apache2`
- Una vez instalado podemos comprobar si está corriendo con el siguiente comando:
  - `$ sudo systemctl status apache2`
- Por defecto despliega el servidor en el puerto 80, y haciendo una redirección en la máquina virtual podemos acceder a la página por defecto

# INSTALACIÓN SERVICIO WEB APACHE

- Página de inicio de sesión por defecto de Apache (puerto 80 en máquina virtual):



## Apache2 Ubuntu Default Page

ubuntu

**It works!**

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

### Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in `/usr/share/doc/apache2/README.Debian.gz`**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

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

# Documentación Apache

- Para instalar la documentación de Apache usamos:
  - `$ sudo apt install apache2-doc`
- Este paquete instala la documentación en archivos HTML sobre apache en el directorio `/usr/share/doc/apache2-doc/manual`
- Se puede utilizar el navegador web instalado en nuestro escritorio para ver esta documentación.
- Supongamos que el servidor tiene la dirección la redirección al puerto 8888, entonces la documentación quedará accesible desde <http://localhost:8888/manual/en/index.html>

# CORTAFUEGOS

- Uncomplicated Firewall (ufw) es un cortafuegos diseñado y disponible en Ubuntu.
- Para activarlo utilizamos la siguiente opción:
  - `$ sudo ufw enable`
- Se puede desactivar con:
  - `$ sudo ufw disable`
- Apache se registra como perfil de aplicación en el cortafuegos. Se puede comprobar los perfiles de aplicación en el cortafuegos:
  - `$ sudo ufw app list`
- Después de activar el cortafuegos no podremos acceder al servidor apache.

# COMPROBACIÓN CORTAFUEGOS

- Para activar Apache y SSH utilizaremos:
  - `$ sudo ufw allow 'Apache'`
  - `$ sudo ufw allow 'OpenSSH'`
- También podemos activar un puerto específico o denegar:
  - `$ sudo ufw allow 9000`
  - `$ sudo ufw deny 9000`
- Para ver el listado de reglas del cortafuegos utilizaremos:
  - `$ sudo ufw status numbered`
- Para eliminar un número de regla (ej., 1):
  - `$ sudo ufw delete 1`



# CONVENCIONES UBUNTU

- Convenciones Apache en Ubuntu
- `/etc/apache2`
- En este directorio se encuentran todos los archivos de configuración de Apache
- `/etc/apache2/apache2.conf`
- Principal archivo de configuración de Apache usado para el ejecutable `apache2`.
- `/etc/apache2/envvars`
- Definición de las variables de entorno utilizados en los scripts de Apache
- `/etc/apache2/ports.conf`
- Este archivo se utiliza para definir los puertos que va a usar Apache. Por defecto está configurado para usar el puerto estándar 80 y en el caso de que el módulo SSL esté activo entonces también tendrá configurado en este archivo el puerto 443.

# CONVENCIONES UBUNTU

- `/etc/apache2/conf-available/`
- Contiene los ficheros de configuración disponibles
- En Ubuntu este directorio se suele utilizar para añadir opciones adicionales a Apache que un administrador o paquete incorpore.
- `/etc/apache2/conf-enabled/`
- Contiene enlaces simbólicos a los ficheros en `/etc/apache/conf-available`. Si existe un enlace simbólico, será habilitado la próxima vez que inicie Apache.

# CONVENCIONES UBUNTU

- `/etc/apache2/mods-available/`
- Este directorio contiene los archivos `.load` y `.conf` de todos los módulos disponibles en el sistema
- El archivo `.load` contiene toda la información necesaria para cargar el módulo en Apache.
- El archivo `.conf` contiene cualquier configuración extra que Apache pueda necesitar para usar el módulo una vez se ha cargado.
- Un listado a los archivos de este directorio nos facilita todos los módulos disponibles en Apache.

# CONVENCIONES UBUNTU

- `/etc/apache2/mods-enabled/`
- Este directorio contiene enlaces simbólicos a los archivos `.load` y `.conf` en el directorio “`mods-available`”
- Cuando Apache inicia, revisa este directorio y carga todos los scripts de inicio referenciados en este directorio. En caso de inicio correcto, entonces el módulo queda activado.
- Ejemplo: Para activar el módulo CGI (es necesario también para el fichero `.conf`) :  

```
$ sudo ln -s /etc/apache2/mods-available/cgi.load  
/etc/apache2/mods-enabled/cgi.load
```

# CONVENCIONES UBUNTU

- Otra opción para activar o habilitar un módulo es usando la herramienta `a2enmod` seguido por el nombre del módulo a activar
- Ejemplo: para activar el módulo `cgi`:
  - `$ sudo a2enmod cgi`
- Del mismo modo, la herramienta `a2dismod` desactiva o deshabilita un módulo, seguido del nombre del módulo una vez activado:
- Ejemplo: para desactivar el módulo `cgi`:
  - `$ sudo a2dismod cgi`

# CONVENCIONES UBUNTU

- El comando de desactivación borra el enlace simbólico y para la ejecución del módulo.
- En cambio, el comando de activación sólo crea el enlace simbólico en el directorio “mods-enabled” pero no inicia el módulo.
- Para iniciar el módulo de nuevo es necesario recargar Apache:
  - `$ sudo service apache2 restart`

# CONVENCIONES UBUNTU

- `/etc/apache2/sites-available/`
- Apache fue uno de los primeros servidores Web que introdujo el concepto de **host virtuales** y que nos permite alojar varios sitios web en un solo servidor físico.
- Cada host virtual puede tener un dominio totalmente diferente del resto. Un sitio o host virtual que esté disponible en Apache tiene su propia configuración en este directorio.
- Por defecto, el único archivo que se mostrará si no hay ningún otro configurado será “default”. También puede aparecer “default-ssl”.

# CONVENCIONES UBUNTU

- Los hosts virtuales pueden estar **basados en IP**, lo que significa que cada host tiene una dirección IP diferente para cada sitio web.
- O **basados en nombre** (ej. *myweb1.com*, *myweb2.com*), lo que significa que tiene varios nombres que se ejecutan en una dirección IP. Esta es la opción recomendable para evitar depender de IPs.



# CONVENCIONES UBUNTU

- `/etc/apache2/sites-enabled/`
- De forma análoga al funcionamiento del directorio “mods-enabled”, este directorio contiene enlaces simbólicos a los archivos de configuración alojados en el directorio “sites-available”
- Para añadir un nuevo host virtual a Apache, hay que crear un archivo de configuración para dicho host que contenga un bloque completo `<virtualhost>` en el directorio “sites-available” y crear en este directorio un enlace simbólico que apunte a este archivo de configuración en ese directorio (“sites-available”)
- Ejemplo:
- `$ sudo ln -s /etc/apache2/sites-available/mysite`
- `/etc/apache2/sites-enabled/mysite`

# CONVENCIONES UBUNTU

- También es posible utilizar el script `a2ensite` y el nombre del sitio a activar
- Ejemplo:
  - `$ sudo a2ensite mysite`
- Igualmente el comando `a2dissite` desactiva el host virtual
- El sitio por defecto (default) es un caso especial en el cual se antepone “000” a su enlace simbólico para que cargue siempre el primero.

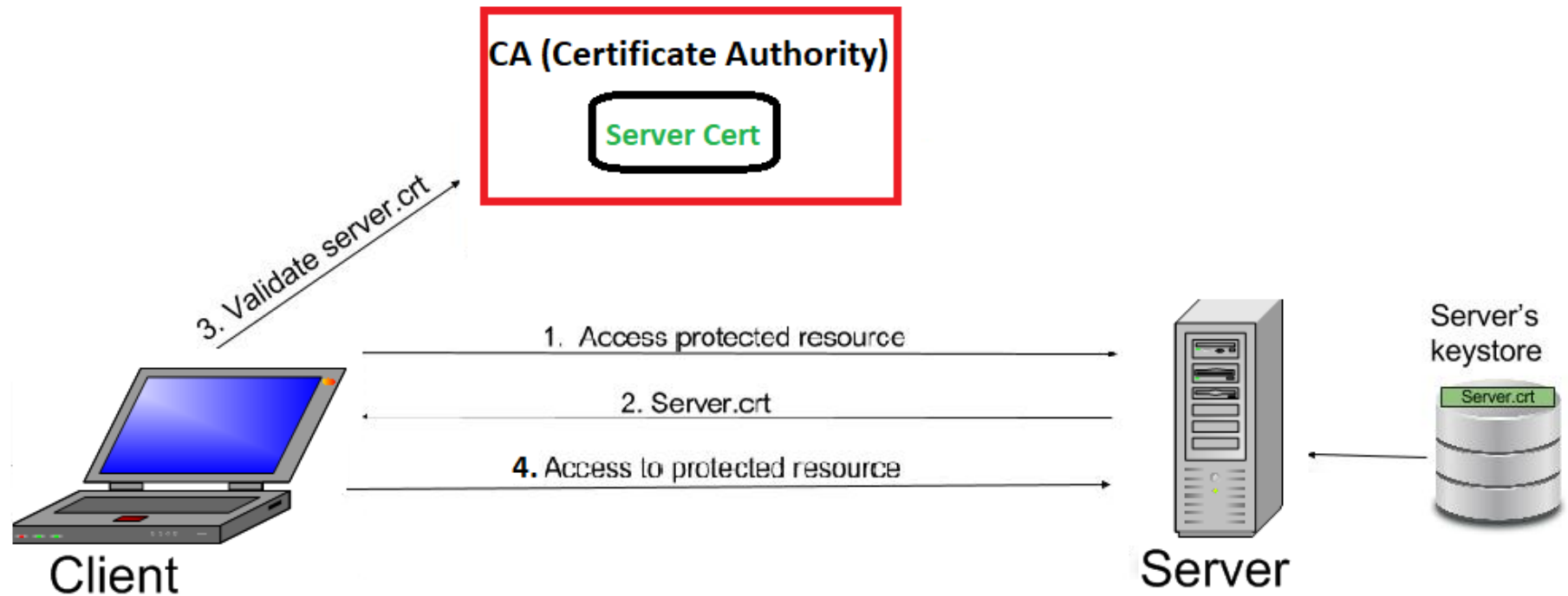
# ACTIVACIÓN MÓDULO SSL EN APACHE

- Mediante el tráfico TLS/SSL se cifra el tráfico entre servidores y clientes para mantener comunicaciones seguras y cifradas.
- Es necesario generar un certificado de clave pública-privada (X.509) para habilitar las comunicaciones seguras:
  - La clave SSL se mantiene secreta en el servidor.
  - El certificado SSL se comparte de forma pública con cualquiera que solicite el contenido.

# ACTIVACIÓN MÓDULO SSL EN APACHE

- Los certificados generados tienen que estar firmados por una entidad de certificación (CA) que verifica la identidad del servidor para que los clientes sepan a quién están enviando su información y prevenir ataques Man-in-the-Middle.
- Los navegadores están configurados para que confíen en determinadas autoridades de certificación y no hay que realizar ningún paso extra para usar estos certificados.
- Estos certificados suelen costar un precio y es necesaria su continua renovación.
- Opciones, Autoridades de certificación gratuitas : <https://letsencrypt.org/es/>

# ACTIVACIÓN MÓDULO SSL EN APACHE



# ACTIVACIÓN MÓDULO SSL EN APACHE

- Otra opción es utilizar certificados auto-firmados:
  - Estos certificados se pueden generar directamente en el servidor y no tienen un coste extra de una autoridad de certificación.
  - Convenientes para versiones de desarrollo y páginas webs con poco tráfico.
  - Saldrá un aviso en los navegadores de los usuarios solicitando una confirmación para confiar en el certificado.

# ACTIVACIÓN MÓDULO SSL EN APACHE

- OpenSSL es un kit de herramientas de criptografía que implementa estándares de criptografía relacionados requerido por SSL y TLS. Además, de permitir la generación de certificados X.509:
- ```
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/aso-apache-selfsigned.key -out /etc/ssl/certs/aso-apache-selfsigned.crt
```
- La opción `req -x509` es para un certificado X.509 auto-firmado.
- La opción `-nodes` evita la opción de pedir contraseña y no tengamos que introducirla cada vez en Apache.
- La opción `-days` es el tiempo de expiración del certificado.
- La opción `newkey` es para la generación de la en el mismo proceso usando el protocolo RSA de 2048 bits.

# ACTIVACIÓN MÓDULO SSL EN APACHE

- Modificamos el archivo de configuración de la página web default para enlazarla el certificado creado:

- `$ sudo nano /etc/apache2/sites-available/default-ssl.conf`

- Y enlazamos nuestro certificado añadiendo lo siguiente:

```
SSLCertificateFile /etc/ssl/certs/aso-apache-selfsigned.crt
```

```
SSLCertificateKeyFile /etc/ssl/private/aso-apache-selfsigned.key
```



# ACTIVACIÓN MÓDULO SSL EN APACHE

- Activar el módulo SSL:
  - `$ sudo a2enmod ssl`
- Activar el sitio de SSL:
  - `$ sudo a2ensite default-ssl`
- Activar el tráfico SSL en el firewall:
  - `sudo ufw allow 'Apache Secure'`
- Añadir nueva redirección de puertos para el puerto 443.
- Reiniciar servicio Apache para coger los cambios:
  - `$ sudo service apache2 restart`

# Host Virtuales

- Para crear un nuevo virtual host hay que crear un nuevo fichero de configuración en `/etc/apache2/sites-available/`
- Ejemplo para el dominio `example.com` y el alias [www.example.com](http://www.example.com)

```
<VirtualHost *:80> # Escucha en todas las IP y el puerto 80
    ServerName example.com
    ServerAlias www.example.com
    DocumentRoot "/var/www/domain"
</VirtualHost>
```

- Hay que crear una carpeta en `/var/www/domain` para alojar el sitio web.

# Creación de Host Virtuales

- Vamos a crear dos host virtuales y dos páginas webs de ejemplo para los host <http://primera.com> y <http://segunda.com>

# Creación de Host Virtuales

- Editamos el primer host y copiamos el contenido:
  - `sudo nano /etc/apache2/sites-available/primer.conf`

`<VirtualHost *:80>`

`ServerName primera.com`

`ServerAlias www.primera.com`

`DocumentRoot /var/www/primer.com/public_html`

`</VirtualHost>`

# Creación de Host Virtuales

- Editamos el segundo host y copiamos el contenido:
  - `sudo nano /etc/apache2/sites-available/segunda.conf`

`<VirtualHost *:80>`

`ServerName segunda.com`

`ServerAlias www.segunda.com`

`DocumentRoot /var/www/segunda.com/public_html`

`</VirtualHost>`

# Creación de Host Virtuales

- Creamos la carpeta para alojar el primer host:
  - `$ sudo mkdir -p /var/www/primera.com/public_html`
- Editamos el contenido de la primera página y copiamos el texto del HTML de ejemplo:
  - `$ sudo nano /var/www/primera.com/public_html/index.html`
- Damos permisos de escritura y lectura a todos los usuarios:
  - `sudo chmod -R 755 /var/www`
- Texto HTML de ejemplo para index.html:

```
<html>
```

```
<head>
```

```
<title>ASO</title>
```

```
</head>
```

```
<body> <h1>Primera Pagina</h1> </body>
```

```
</html>
```

# Creación de Host Virtuales

- Creamos la carpeta para alojar el segundo host:
  - `$ sudo mkdir -p /var/www/segunda.com/public_html`
- Editamos el contenido de la primera página y copiamos el texto del HTML de ejemplo:
  - `$ sudo nano /var/www/segunda.com/public_html/index.html`
- Damos permisos de escritura y lectura a todos los usuarios:
  - `sudo chmod -R 755 /var/www`
- Texto HTML de ejemplo para index.html:

```
<html>
```

```
<head>
```

```
<title>ASO</title>
```

```
</head>
```

```
<body> <h1>Segunda Pagina</h1> </body>
```

```
</html>
```

# Creación de Host Virtuales

- Una vez creado la estructura de ficheros y los host virtuales, los activamos los dos hosts virtuales.
- Activación de la primera página:
  - `sudo a2ensite primera`
- Activación de la segunda:
  - `sudo a2ensite segunda`
- Y hacemos un restart del servicio de Apache para coger los cambios:
  - `sudo systemctl reload apache2`



# CONVENCIONES UBUNTU

## `/var/www`

- Esta es la raíz de documentos por defecto para Apache. Cualquier archivo HTML que sea legible por Apache y se coloque en este directorio, se encontrará disponible una vez que lo visitemos con nuestro navegador (está publicado en el servidor web). Existe por defecto un fichero “html/index.html” que muestra un mensaje confirmando que el sitio web funciona y que está en construcción.

## `/usr/lib/cgi-bin/`

- Localización por defecto para los scripts CGI (Common Gateway Interface – Interfaz de entrada común) para intercambio de objetos MIME (Multipurpose Internet Mail extensions – Extensiones multipropósito de correo de Internet). Incluye archivos de tipo texto, imagen, audio, video. Etc

# CONVENCIONES UBUNTU

- `/var/log/apache2/`
- Este es el directorio estándar donde se almacenan los registros de estado de Apache.
- El archivo `access.log` contiene información acerca de qué archivos se han accedido a través del servidor web, y el archivo `error.log` almacena cualquier error de Apache.
- Si hay problemas en el inicio de Apache, en este fichero podemos encontrar información detallada de los posibles errores que hayan podido ocurrir.

# SERVICIO Apache2ctl

- Apache2ctl
- El programa `/usr/bin/apache2ctl` es el programa de línea de comandos principal de apache en Ubuntu.
- `$ sudo apache2ctl "comando"`
- donde "comando" puede ser: "start", "stop" y "restart" para las funciones de iniciar, parar y reiniciar respectivamente.
- `$ sudo apache2ctl restart`
- `$ sudo /etc/init.d/apache2 restart`
- `$ sudo service apache2 restart`
- Estos tres comando son equivalentes y hacen lo mismo

# SERVICIO Apache2ctl

- Detener Apache correctamente
- Existe un riesgo potencial asociado con los comandos restart y stop.
- Cuando reiniciamos o detenemos el servicio, se terminan todos los procesos de apache que en ese momento se encuentran en ejecución, incluso si se encuentran en mitad de servir archivos a un usuario.
- Si se diera este caso, entonces el usuario cargará tanta información como haya obtenido hasta la interrupción del servidor web y después fallará, obligando al usuario a recargar la página.
- Para evitar esta situación, apache2ctl provee los comandos “graceful” y “graceful-stop”. Estos comandos reinician y detienen el servidor web Apache respectivamente, pero cuando lo hacen, esperan a que cada proceso termine cualquier petición en trámite.

# SERVICIO Apache2ctl

- En un sitio activo, un reinicio con graceful podría no ser advertido por ningún usuario.
- En general, a menos que no nos importe que se puedan cerrar las conexiones abiertas, deberíamos usar exclusivamente las opciones “graceful” y “graceful-stop”
- La única excepción es cuando añadimos certificados SSL al sitio o cualquier otro cambio que necesite un reinicio completo de Apache para que los cambios se habiliten. En este caso tenemos que hacer una recarga completa del servicio.

# SERVICIO Apache2ctl

- Comandos de diagnóstico de apache2ctl
- Los otros comandos principales para apache2ctl proveen más características de diagnóstico.
- `configtest` comprobará nuestra configuración actual de Apache en busca de errores. Esta función puede resultar muy útil para automatizar operaciones sobre Apache mediante scripts, como por ejemplo la operación de añadir un nuevo host virtual en el servidor web.
- Si se produce cualquier error al añadir el host virtual y falla el reinicio de Apache, entonces nos podríamos encontrar con la situación de que dejasen de funcionar el resto de servidores virtuales en Apache, en incluso el servicio completo.
- Podemos establecer una lógica en nuestros scripts que reinicien Apache sólo una vez que el servidor hay superado el “configtest” (test de configuración)

# SERVICIO Apache2ctl

- Otra opción interesante y muy útil es la de monitorización. A medida que un servidor Apache aumenta su tráfico, normalmente es útil analizar información sobre servicio. Algunas parámetros o estadísticas útiles puede ser: cuantos procesos Apache están activos, qué están haciendo esos procesos y cuantas conexiones tenemos activas o disponibles.
- Los comandos de `apache2ctl status` y `fullstatus` proporcionan un diagnóstico con gran cantidad de datos. El comando “status” devuelve una visión general del estado de nuestro servidor Apache, incluyendo cuando tiempo ha estado el servidor, cuantas peticiones están activas, cuantos procesos están inactivos y también nos proporciona un mapa ASCII con todos los procesos disponibles y su estado.
- Para usar estos comandos es necesario instalar un navegador:
  - `$ sudo apt install lynx`

# Bibliografía

- Web server Apache: <https://ubuntu.com/server/docs/web-servers-apache>
- Apache Server on Ubuntu 20.04:  
[https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-20-04#step-5-%E2%80%94-setting-up-virtual-hosts-\(recommended\)](https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-20-04#step-5-%E2%80%94-setting-up-virtual-hosts-(recommended))
- Let's encrypt en Apache Server:  
<https://www.digitalocean.com/community/tutorials/how-to-secure-apache-with-let-s-encrypt-on-ubuntu-20-04-es>
- Configurar SSL en Apache:  
<https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-apache-in-ubuntu-18-04-es>