



## **Tema de investigación 06**

# **CONFIGURACIÓN DE SERVIDORES WEB APACHE, TOMCAT EN FORMA SEGURA**

<b>Ángel Fabian Gómez Estupiñan</b> (Líder)	Cod: 2130535
Esteban Andrés Niño Mendez	Cod: 2142608
Javier Camilo Rueda Serrano	Cod: 2141380
Juan Felipe Silva Garcés	Cod: 2141362

## **SISTEMAS OPERATIVOS**

Grupo: D1  
Subgrupo: 06

Profesor  
Manuel Guillermo Flórez Becerra

Bucaramanga - Santander  
2020



## **VI CONFIGURACIÓN DE SERVIDORES WEB APACHE, TOMCAT EN FORMA SEGURA**



Nube, equipo y alojamiento [<https://pixabay.com/es/illustrations/nube-equipo-alojamiento-3406627/> ]



## INTRODUCCIÓN.

Los servidores web son ordenadores especializados que hacen posible el Web Hosting, es decir, cuando hablamos de servicio de hosting es el alquiler de un espacio en un servidor remoto para poder almacenar los archivos de un sitio web. En otras palabras, un servidor web es un computador u ordenador que almacena y envía datos vía el sistema de redes, conocido como la internet. Cuando un usuario accede o ingresa en una página de internet, el navegador hace una conexión con el servidor, enviando los datos y del mismo modo recibiendo los datos de respuesta, y de este modo lograr la interacción del usuario con la página web y poder ofrecer el servicio o funcionalidad de la plataforma accedida por el usuario. Los servidores Web existen para guardar y transmitir datos de un sitio web que le hace peticiones, cuando navegamos en una página Web estas se basan en protocolos HTTP (Hypertext Transfer Protocol) y HTTPS (Hypertext Transfer Protocol Secure).

Hay incontables variedades de servidores web, desde servidores propios de una plataforma en concreto y algunos otros multiplataforma, dependiendo también de la cantidad de clientes y tráfico a la que está enfocado el servidor.

Cuando nos referimos a Apache como un servidor web, no se trata de algo físico, se trata de un software multiplataforma que se ejecuta en un servidor, cuya función es conectar un servidor y los navegadores de los clientes web, bien sea Firefox, Google Chrome, Safari, etc. durante el intercambio de archivos cliente-servidor.

Apache Tomcat es un servidor HTTP producido específicamente para aplicaciones Java en lugar de sitios web estáticos, a pesar de que es posible usarlo en este tipo de sitios web, es menos eficiente, a diferencia de Apache, que viene siendo un servidor de propósito general, y es posible usarlo con otros lenguajes de programación.



## CONFIGURACIÓN DE SERVIDORES WEB APACHE - TOMCAT

### 1.1 OBJETIVOS

- Aprender cómo realizar la configuración de servidores web Apache de forma manual y como configurar el contenedor de servlets Tomcat para el desarrollo de páginas y aplicaciones web dinámicas, entendiendo cada una de las directivas disponibles.
  - ◆ Implementar un servidor web apache seguro en forma manual, es decir, configurando el servidor y explicando su significado de cada una de las directivas del apache.
  - ◆ Implementar el servidor Tomcat e interconectarlo con Apache.
  - ◆ Implementar sitios web seguros; es decir que se pueda utilizar el protocolo https, implementar SSH, clave pública y clave privada.
  - ◆ Realizar pruebas de funcionamiento con demos (aplicaciones) para probar apache y Tomcat funcionando correctamente.
  - ◆ Explicar con ejemplos en forma didáctica las opciones de configuración.

### 1.2 AUTOEVALUACION

- a) ¿Apache usa de alguna forma SSL?
- b) ¿Para qué se usa la línea de comando “a2enmod ssl”?
- c) ¿Cuales herramientas de Apache 2 se relacionan con la defensa de ataques de fuerza bruta hacia el servidor?
- d) Cuales son las características más importantes de Apache.
- e) ¿Qué son los hosts virtuales?
- f) El protocolo HTTP trabaja sobre el puerto 80, pero Apache2 dispone de la opción de utilizar las librerías SSL para aumentar la seguridad obteniendo el protocolo HTTPS, con ¿cuál puerto trabaja SSL?
- g) HTTP es un protocolo “sin estado”, ¿Qué significa esto?

### 1.3 PRACTICA

#### • **INSTALACIÓN DE UN SERVIDOR APACHE.**

*El proceso de instalación no es tan complejo en comparación con el de configuración, se limita únicamente a algunos sencillos pasos.*

- a) *Por medio de una terminal, actualizamos todas las librerías del S.O con la línea.*  
**`sudo apt-get update`**
- b) *Una vez actualizado el S.O instalamos los servicios de Apache con la siguiente línea.*  
**`sudo apt-get install apache2`**



c) Para ver la versión de Apache que tenemos instalada usaremos el comando.

***sudo apachectl -v***

```
angel@angel-X456UF ~ $ sudo apachectl -v
Server version: Apache/2.4.18 (Ubuntu)
Server built: 2019-10-08T13:31:25
angel@angel-X456UF ~ $
```

En nuestro caso es la versión 2.4.18.

d) Para comprobar que Apache se encuentre correctamente instalado buscamos la dirección IP anexa a nuestro pc, para ello usamos la línea de comando.

***hostname -i***

```
angel@angel-X456UF ~ $ hostname -i
127.0.1.1
angel@angel-X456UF ~ $
```

Una vez obtenida la nuestra dirección IP la ingresaremos como dirección URL en cualquier navegador web de nuestro pc.

#### • **CONFIGURACIÓN DE UN SERVIDOR APACHE.**

La configuración del servidor Apache es de suma importancia porque de esto depende su correcto funcionamiento, empezaremos el proceso de configuración estableciendo protocolos de seguridad para el servidor.

**¿Qué es SSL?** Secure Sockets Layer (capa de sockets seguros), es la tecnología estándar para mantener segura una conexión a internet. Hace que los datos transferidos entre dos sistemas sean imposibles de leer utilizando algoritmos de cifrado para codificar los datos que se transmiten.

Un socket es un método para la comunicación entre un programa del cliente y un programa del servidor en una red (sistemas distribuidos).

**¿Qué es HTTPS?** Hypertext Transfer Protocol Secure (protocolo seguro de transferencia de hipertexto) es un protocolo destinado a la transferencia segura de datos de hipertexto.

Una vez claros estos dos conceptos procederemos a habilitar los protocolos HTTPS y SSL en nuestro servidor Apache, para ello debemos dirigirnos a la carpeta donde se encuentra Apache, **etc/apache2**.

a) una vez allí activaremos el módulo de SSL en Apache, para ello ejecutaremos la línea de comando.

***sudo a2enmod ssl***

b) Siguiendo la recomendación que nos genera la consola, para activar las nuevas configuraciones reiniciamos el servicio de Apache con la siguiente línea de consola.



### **service apache2 restart**

```
angel@angel-X456UF / $ cd /etc/apache2
angel@angel-X456UF /etc/apache2 $ sudo a2enmod ssl
[sudo] password for angel:
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create s
elf-signed certificates.
To activate the new configuration, you need to run:
service apache2 restart
angel@angel-X456UF /etc/apache2 $ service apache2 restart
angel@angel-X456UF /etc/apache2 $
```

- c) Una vez activado el módulo de SSL en nuestro servidor, generaremos un cifrado de tipo RSA para los mensajes que se vayan a transmitir, para ello usaremos la siguiente línea de código usando permisos de usuario **sudo**.

**sudo openssl genrsa -des3 -out server.key 2048**

```
angel@angel-X456UF /etc/apache2 $ sudo openssl genrsa -des3 -out server.key 2048
[sudo] password for angel:
Generating RSA private key, 2048 bit long modulus
.....+++
....+++
e is 65537 (0x10001)
Enter pass phrase for server.key:
Verifying - Enter pass phrase for server.key:
```

En nuestro caso usamos como pass phrase para el servidor la palabra “**angel**”, la necesitaremos en nuestros casos posteriores.

- d) Ya activado el SSL y con su cifrado configurado tendremos que activar el certificado CSR (Certificate Signing Request), este certificado tiene información que será incluida finalmente en el certificado SSL, algunos de estos datos son nombre, empresa, dirección, país de residencia, common name (dominio para el que es generado el SSL) y una clave pública incluida en el certificado.

Para activar el certificado CSR usamos la siguiente línea de código.

**openssl req -new -key server.key -out server.csr**

```
angel@angel-X456UF /etc/apache2 $ sudo openssl req -new -key server.key -out server.csr
Enter pass phrase for server.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:CO
State or Province Name (full name) [Some-State]:Santander
Locality Name (eg, city) []:Bucaramanga
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UIS
Organizational Unit Name (eg, section) []:Ingenieria de Sistemas
Common Name (e.g. server FQDN or YOUR name) []:Servidor_angel
Email Address []:ango_1415@hotmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:1234
An optional company name []:UIIS
```



- e) En este punto ya tenemos nuestro certificado SSL casi terminado, solo falta establecer la duración de este, en nuestro caso podremos una duración de 365 días, para ello usaremos el siguiente código.
- `sudo openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt.`**

```
angel@angel-X456UF /etc/apache2 $ sudo openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt.  
Signature ok  
subject=C=C0/ST=Santander/L=Bucaramanga/O=UIS/OU=Ingenieria de Sistemas/CN=Servidor_angel/emailAddress=angelgo_1415@hotmail.com  
Getting Private key  
Enter pass phrase for server.key:  
angel@angel-X456UF /etc/apache2 $
```

- f) Haremos un respaldo de los siguientes archivos **server.key** y **server.crt.** que estan en la carpeta en la que hemos estado trabajando, **apache2**, cada uno corresponden al certificado CSR y la llave del servidor. Copiaremos estos archivos en los directorios **/etc/ssl/private** y **/etc/ssl/certs**, esto con los siguientes comandos.
- `sudo cp server.key /etc/ssl/private`**  
**`sudo cp server.crt. /etc/ssl/certs`**

```
angel@angel-X456UF /etc/apache2 $ sudo cp server.key /etc/ssl/private  
angel@angel-X456UF /etc/apache2 $ sudo cp server.crt. /etc/ssl/certs  
angel@angel-X456UF /etc/apache2 $
```

- g) Teniendo estos respaldos procederemos a configurar manualmente los sitios disponibles de nuestro servidor **apache2**, para ello editaremos el archivo **default-ssl.conf** que se encuentra en el directorio **sites-available** dentro de **apache2**, para ello usaremos un editor de texto con permisos de usuario **sudo**, en nuestro caso será **nano**, si no cuentas con él tendrás que instalarlo.
- `sudo apt-get install nano`**  
**`cd sites-available`**  
**`sudo nano default-sslconf`**

```
angel@angel-X456UF /etc/apache2 $ cd sites-available  
angel@angel-X456UF /etc/apache2/sites-available $ sudo nano default-ssl.conf
```

```
GNU nano 2.5.3 Archivo: default-ssl.conf  
#IfModule mod_ssl.c>  
<VirtualHost _default_:443>  
    ServerAdmin webmaster@localhost  
  
    DocumentRoot /var/www/html  
  
    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,  
    # error, crit, alert, emerg.  
    # It is also possible to configure the loglevel for particular  
    # modules, e.g.  
    #LogLevel info ssl:warn  
  
    ErrorLog ${APACHE_LOG_DIR}/error.log  
    CustomLog ${APACHE_LOG_DIR}/access.log combined  
  
    # For most configuration files from conf-available/, which are  
    # enabled or disabled at a global level, it is possible to  
  
^G Ver ayuda  ^O Guardar  ^W Buscar   ^K Cortar Texto  ^J Justificar  ^C Posición  
^X Salir      ^R Leer fich. ^E Reemplazar ^U Pegar txt    ^I Ortografía  ^_ Ir a línea
```





Ahora buscaremos la línea de texto que contenga la palabra “SSLOPTIONS”, para ello presionamos **Ctrl+w** y escribimos la palabra a buscar.

```
# For most configuration files from conf-available/, which are
# enabled or disabled at a global level, it is possible to
Buscar [#SSLOPTIONS]: #SSLOPTIONS
^G Ver ayuda  M-C Mayús/minú  M-B Ir atrás  M-J JustifTodo  ^Y Pri. línea  ^W Ini de pár.
^C Cancelar  M-R Exp. reg.  ^R Reemplazar  ^T Ir a línea  ^V Últ. línea  ^O Fin de pár.
```

```
GNU nano 2.5.3      Archivo: default-ssl.conf      Modificado

#       This exports the standard SSL/TLS related 'SSL_*' environment va$
#       Per default this exportation is switched off for performance rea$
#       because the extraction step is an expensive operation and is usu$
#       useless for serving static content. So one usually enables the
#       exportation for CGI and SSI requests only.
#       o OptRenegotiate:
#       This enables optimized SSL connection renegotiation handling whe$
#       directives are used in per-directory context.
#SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire
<FilesMatch "\.(cgi|shtml|phtml|php)$">
    SSLOptions +StdEnvVars
</FilesMatch>
<Directory /usr/lib/cgi-bin>
    SSLOptions +StdEnvVars
</Directory>

#       SSL Protocol Adjustments:
^G Ver ayuda  ^O Guardar  ^W Buscar  ^K Cortar Texto  ^J Justificar  ^C Posición
^X Salir      ^R Leer fich.  ^N Reemplazar  ^U Pegar txt  ^T Ortografía  ^I Ir a línea
```

```
#       This enables optimized SSL connection renegotiation handling whe$
#       directives are used in per-directory context.
SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire
<FilesMatch "\.(cgi|shtml|phtml|php)$">
    SSLOptions +StdEnvVars
</FilesMatch>
<Directory /usr/lib/cgi-bin>
    SSLOptions +StdEnvVars
</Directory>

#       SSL Protocol Adjustments:
Nombre del archivo a escribir: default-ssl.conf
^G Ver ayuda  M-D Format DOS  M-A Añadir  M-B Respalda fich
^C Cancelar  M-M Format Mac  M-P Anteponer  ^T A Ficheros
```

Elimina el # del principio de la línea y guarda el archivo, para ello usamos **Ctrl+O**, dejamos el mismo nombre del archivo para que se sobrescriba (si no entraste al archivo con **sudo** no permitirá guardarlo).  
Por último, salimos del editor nano con **Ctrl-X**, allí podrás guardar nuevamente el archivo.

- h) Con lo anterior conseguimos que el sitio por default para SSL quedara activado, para comprobar que es así usaremos la siguiente línea de código.  
**sudo a2ensite default-ssl.conf**



***sudo a2ensite default-ssl.***

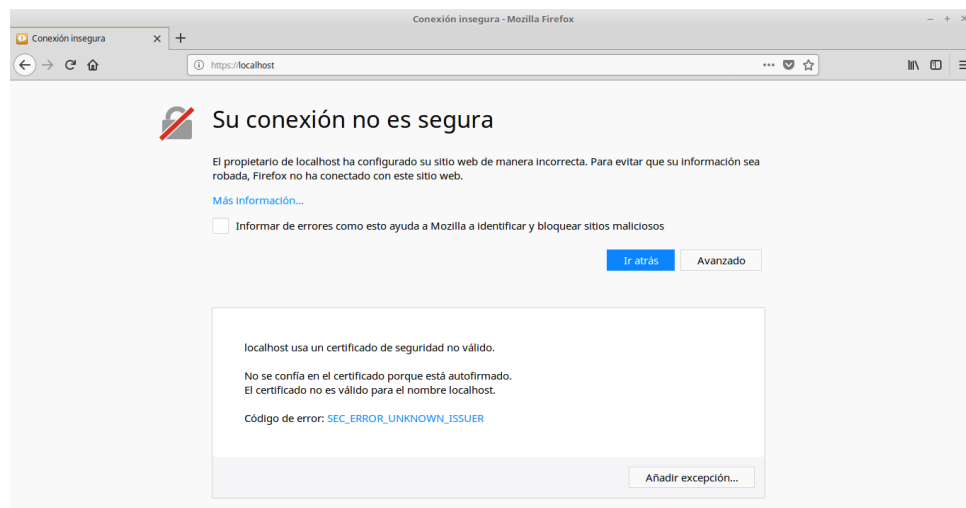
```
angel-X456UF sites-available # a2ensite default-ssl.conf
Enabling site default-ssl.
To activate the new configuration, you need to run:
  service apache2 reload
angel-X456UF sites-available # a2ensite default-ssl
Site default-ssl already enabled
angel-X456UF sites-available # exit
exit
angel@angel-X456UF /etc/apache2/sites-available $ a2ensite default-ssl
Site default-ssl already enabled
angel@angel-X456UF /etc/apache2/sites-available $
```

*i) Siguiendo las recomendaciones dadas por el sistema refrescaremos nuestro servidor apache2 con la siguiente línea de comando.*

***sudo service apache2 reload***

*j) Una vez recargado el servidor apache probaremos la conexión actual de nuestro localhost, para ver si ya tenemos acceso al mismo. Para ello vamos a un navegador web e ingresamos a la siguiente dirección url.*

***https://localhost***



- ***INSTALACIÓN DEL CONTENEDOR DE SERVELETS TOM-CAT***

*Un servlet es una clase en el lenguaje de programación Java, utilizada para ampliar las capacidades de un servidor y Tom-Cat es un tipo de contenedor para estos servlets.*

*a) Por tanto, para empezar la instalación de Java comenzaremos con la instalación del JDK (Java Development Kit), usaremos la siguiente línea de código.*



***sudo apt-get install default-jdk***

```
angel@angel-X456UF /etc/apache2 $ sudo apt-get install default-jdk
[sudo] password for angel:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
ca-certificates-java default-jdk-headless default-jre default-jre-headless
openjdk-8-jdk openjdk-8-jdk-headless openjdk-8-jre openjdk-8-jre-headless
```

- b) A continuación, es necesaria la creación de un usuario y grupo adicional en el directorio `/opt/tomcat` para el correcto funcionamiento de TomCat, para ello usaremos la siguiente línea de código.

***Sudo useradd -m -U -d /opt/tomcat -s /bin/false tomcat***

```
angel@angel-X456UF /etc/apache2 $ sudo useradd -m -U -d /opt/tomcat -s /bin/false tomcat
angel@angel-X456UF /etc/apache2 $
```

- c) A continuación, instalaremos otras dos herramientas muy importantes para el uso de nuestro servidor web con Tom Cat y es **wget** y **unzip**, la primera, wget, es una herramienta informática creada por el Proyecto GNU. Puedes usarlo para recuperar contenido y archivos de varios servidores web. Admite descargas a través de FTP, SFTP, HTTP y HTTPS. La segunda, unzip, es un programa para comprimir y descomprimir archivos.

Para instalar wget usamos la siguiente línea de comandos.

***sudo apt install unzip wget***

```
angel@angel-X456UF /etc/apache2 $ sudo apt install unzip wget
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
unzip ya está en su versión más reciente (6.0-20ubuntu1).
Se actualizarán los siguientes paquetes:
  wget
1 actualizados, 0 nuevos se instalarán, 0 para eliminar y 611 no actualizados.
Se necesita descargar 299 kB de archivos.
Se utilizarán 0 B de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s
Des:1 http://archive.ubuntu.com/ubuntu xenial-updates/main amd64 wget amd64 1.17.1-lubunt
u1.5 [299 kB]
Descargados 299 kB en 2s (128 kB/s)
(Leyendo la base de datos ... 211420 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../wget 1.17.1-lubuntul.5 amd64.deb ...
Desempaquetando wget (1.17.1-lubuntul.5) sobre (1.17.1-lubuntul.3) ...
Procesando disparadores para install-info (6.1.0.dfsg.1-5) ...
Procesando disparadores para man-db (2.7.5-1) ...
Configurando wget (1.17.1-lubuntul.5) ...
```

Extraeremos desde la página oficial de apache los datos de Tom Cat, en esta ocasión instalaremos la versión 9.0.31, a pesar de que no es la más reciente si es la más estable.

- d) La extracción la haremos en el directorio `/tmp` puesto que allí podemos trabajar con archivos de una forma temporal, para llegar a él usaremos el siguiente comando.

***cd /tmp***



- e) Una vez allí descargaremos el archivo **apache-tomcat-9.0.31.zip** de TomCat que se encuentra en la siguiente dirección web:  
**<https://downloads.apache.org/tomcat/tomcat-9/v9.0.31/bin/>**, esto lo haremos con la siguiente línea de código.  
**wget https://downloads.apache.org/tomcat/tomcat-9/v9.0.31/bin/apache-tomcat-9.0.31.zip**

```
angel@angel-X456UF /tmp $ wget https://downloads.apache.org/tomcat/tomcat-9/v9.0.31/bin/apache-tomcat-9.0.31.zip
--2020-03-12 19:05:25-- https://downloads.apache.org/tomcat/tomcat-9/v9.0.31/bin/apache-tomcat-9.0.31.zip
Resolviendo downloads.apache.org (downloads.apache.org)... 88.99.95.219, 2a01:4f8:10a:201a::2
Conectando con downloads.apache.org (downloads.apache.org)[88.99.95.219]:443... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 11651113 (11M) [application/zip]
Grabando a: "apache-tomcat-9.0.31.zip"

apache-tomcat-9.0.31 100%[=====>] 11,11M 2,24MB/s in 7,4s
2020-03-12 19:05:33 (1,50 MB/s) - "apache-tomcat-9.0.31.zip" guardado [11651113/11651113]
```

- f) Una vez descargado lo extraeremos con la siguiente línea de comando.  
**unzip apache-tomcat-9.0.31.zip**

```
angel@angel-X456UF /tmp $ unzip apache-tomcat-9.0.31.zip
Archive: apache-tomcat-9.0.31.zip
  creating: apache-tomcat-9.0.31/
  creating: apache-tomcat-9.0.31/bin/
  creating: apache-tomcat-9.0.31/conf/
  creating: apache-tomcat-9.0.31/lib/
  creating: apache-tomcat-9.0.31/logs/
  creating: apache-tomcat-9.0.31/temp/
  creating: apache-tomcat-9.0.31/webapps/
  creating: apache-tomcat-9.0.31/webapps/ROOT/
  creating: apache-tomcat-9.0.31/webapps/ROOT/WEB-INF/
  creating: apache-tomcat-9.0.31/webapps/docs/
  creating: apache-tomcat-9.0.31/webapps/docs/WEB-INF/
```

Una vez que termine la extracción, buscaremos el archivo **apache-tomcat-9.0.31** y lo moveremos a la carpeta en la que se encuentra el usuario que creamos para TomCat, es decir **/opt/tomcat**, esto para que salga de la carpeta de los archivos temporales y no se pierda.

- g) Para mover el archivo a **/opt/tomcat** usaremos la siguiente línea de comandos.  
**sudo mv apache-tomcat-9.0.31 /opt/tomcat/**

- h) Ya tenemos nuestro archivo TomCat en la carpeta del usuario que creamos específicamente para él, necesitamos darle permisos al usuario creado para que pueda acceder al directorio de TomCat, para ella usaremos el comando.  
**sudo chown -R tomcat: /opt/tomcat**

```
angel@angel-X456UF /tmp $ sudo chown -R tomcat: /opt/tomcat
angel@angel-X456UF /tmp $
```

- i) De esta forma ya podremos acceder al directorio TomCat, ahora mediante el comando.



**`sudo chmod +x /opt/tomcat/apache-tomcat-9.0.31/bin/*.sh`**

```
angel@angel-X456UF /tmp $ sudo chmod +x /opt/tomcat/apache-tomcat-9.0.31/bin/*.sh
[sudo] password for angel:
angel@angel-X456UF /tmp $
```

Permitiremos que todos los scripts dentro del directorio **`opt/tomcat/apache-9.0.31/bin`** se puedan ejecutar.

- j) Hasta este punto ya podremos usar Tom Cat como servicio, iremos al directorio **`/etc/systemd/system`** y mediante el editor **nano** abriremos el archivo **tomcat.service**.

**`cd /etc/systemd/system`**  
**`sudo nano tomcat.service`**

```
angel@angel-X456UF /etc/systemd/system $ sudo nano tomcat.service
```

- k) Allí encontraremos el archivo vacío, en el ingresaremos la siguiente información.

[Unit]

Description=Tomcat 9 servlet container

After=network.target

[Service]

Type=forking

User=tomcat

Group=tomcat

Environment="JAVA\_HOME=/usr/lib/jvm/default-java"

Environment="JAVA\_OPTS=-Djava.security.egd=file:///dev/urandom"

Environment="CATALINA\_BASE=/opt/tomcat/apache-tomcat-9-0-31"

Environment="CATALINA\_HOME=/opt/tomcat/apache-tomcat-9-0-31"

Environment="CATALINA\_PID=/opt/tomcat/latest/temp/tomcat.pid"

Environment="CATALINA\_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC"

ExecStart=/opt/tomcat/apache-tomcat-9-0-31/bin/startup.sh

ExecStop=/opt/tomcat/apache-tomcat-9-0-31/bin/shutdown.sh

[Install]

WantedBy=multi-user.target

Tenga muy en cuenta las direcciones aquí establecidas, preferiblemente vaya hasta dichos directorios para corroborar que existan y evitar errores.



Luego guardaremos el archivo con **Ctrl+O** con el mismo nombre y saldremos del editor nano con **Ctrl+X**, si no entraste al editor nano con permisos de usuario sudo no te permitirá guardar los cambios.

```
GNU nano 2.5.3 Archivo: tomcat.service

[Unit]
Description=Tomcat 9 servlet container
After=network.target

[Service]
Type=forking

User=tomcat
Group=tomcat

Environment="JAVA_HOME=/usr/lib/jvm/default-java"
Environment="JAVA_OPTS=-Djava.security.egd=file:///dev/urandom"

Environment="CATALINA_BASE=/opt/tomcat/apache-tomcat-9.0.31"
Environment="CATALINA_HOME=/opt/tomcat/apache-tomcat-9.0.31"
Environment="CATALINA_PID=/opt/tomcat/apache-tomcat-9.0.31/temp/tomcat.pid"
Environment="CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC"

ExecStart=/opt/tomcat/apache-tomcat-9.0.31/bin/startup.sh
ExecStop=/opt/tomcat/apache-tomcat-9.0.31/bin/shutdown.sh

[Install]
WantedBy=multi-user.target

^G Ver ayuda  ^O Guardar    ^W Buscar     ^K Cortar Texto ^J Justificar
^X Salir      ^R Leer fich. ^E Reemplazar ^U Pegar txt    ^T Ortografía
```

l) Ahora, mediante los comandos.

**sudo systemctl daemon-reload** // refresca antes de ejecutar

**sudo systemctl start tomcat** // inicia tomcat

**sudo systemctl status tomcat.service** // muestra el estado actual del servicio

podremos comprobar el estado del servicio, si está activo o no y si se configuró correctamente.

```
angel@angel-X456UF /etc/systemd/system $ sudo systemctl daemon-reload
angel@angel-X456UF /etc/systemd/system $ sudo systemctl start tomcat
angel@angel-X456UF /etc/systemd/system $ sudo systemctl status tomcat.service
● tomcat.service - Tomcat 9 servlet container
   Loaded: loaded (/etc/systemd/system/tomcat.service; disabled; vendor preset
   Active: active (running) since jue 2020-03-12 21:39:58 COT; 11min ago
   Main PID: 4327 (java)
   CGroup: /system.slice/tomcat.service
           └─4327 /usr/lib/jvm/default-java/bin/java -Djava.util.logging.confi

mar 12 21:39:58 angel-X456UF systemd[1]: Starting Tomcat 9 servlet container..
mar 12 21:39:58 angel-X456UF systemd[1]: Started Tomcat 9 servlet container.
mar 12 21:51:50 angel-X456UF systemd[1]: Started Tomcat 9 servlet container.
lines 1-10/10 (END)
```

m) Una vez funcionando Apache-TomCat activaremos el firewall de seguridad para el puerto 8080, con el siguiente comando.

**sudo ufw allow 8080/tcp**

```
angel@angel-X456UF /etc/systemd/system $ sudo ufw allow 8080/tcp
Reglas actualizadas
Reglas actualizadas (v6)
angel@angel-X456UF /etc/systemd/system $
```

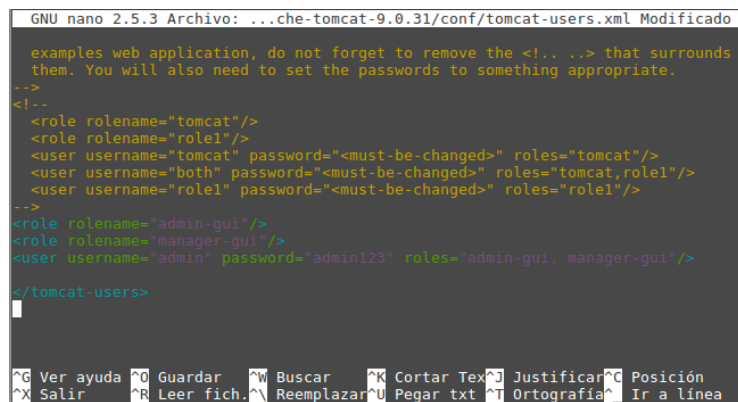


- n) Ahora debemos configurar una interfaz para el administrador web, para ello usaremos el editor **nano** para configurar el archivo “tomcat-users.xml” con el siguiente comando.

**sudo nano /opt/tomcat/apache-tomcat-9.0.31/conf/tomcat-users.xml**

Una vez en el editor **nano** agregamos nombres de rol y usuario, esto lo haremos con las siguientes líneas en la parte final del archivo.

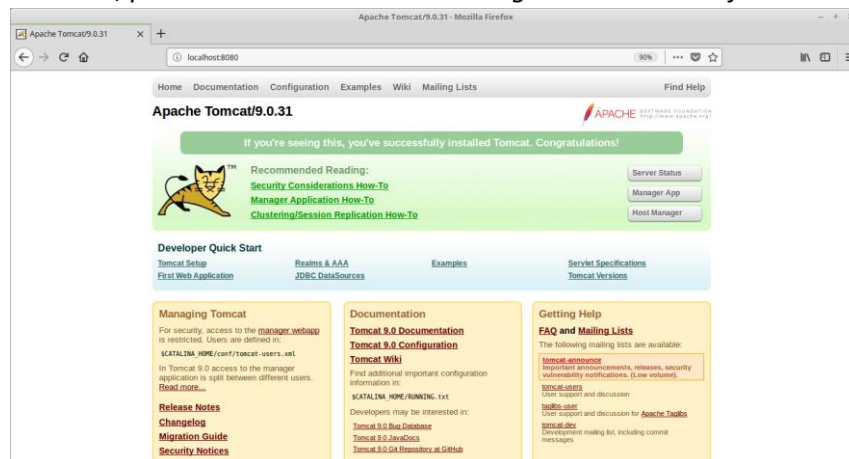
```
<role rolename="admin-gui"/>
<role rolename="manager-gui"/>
<user username="admin" password="admin123" roles="admin-gui, manager-gui"/>
```



```
GNU nano 2.5.3 Archivo: ...che-tomcat-9.0.31/conf/tomcat-users.xml Modificado
examples web application, do not forget to remove the <!-- ... --> that surrounds
them. You will also need to set the passwords to something appropriate.
-->
<!--
<role rolename="tomcat"/>
<role rolename="role1"/>
<user username="tomcat" password="<must-be-changed>" roles="tomcat"/>
<user username="both" password="<must-be-changed>" roles="tomcat,role1"/>
<user username="role1" password="<must-be-changed>" roles="role1"/>
-->
<role rolename="admin-gui"/>
<role rolename="manager-gui"/>
<user username="admin" password="admin123" roles="admin-gui, manager-gui"/>
</tomcat-users>

^G Ver ayuda ^O Guardar ^W Buscar ^K Cortar Text ^J Justificar ^C Posición
^X Salir ^R Leer fich. ^N Reemplazar ^U Pegar txt ^T Ortografía ^_ Ir a línea
```

- o) Por último, probaremos desde nuestro navegador web la interfaz localhost:8080 .



Hasta este punto tanto Apache como TomCat se encuentran instalados y con la posibilidad de funcionar. A continuación, configuraremos los sitios virtuales que serán alojados en nuestro servidor.

- p) Crearemos un directorio para nuestro primer sitio web “operativos.com” con la siguiente línea de comandos.





**`sudo mkdir -p /var/www/operativos.com/html`**  
con **-p** crea directorios padres para el directorio según sean necesario.

```
angel@angel-X456UF ~ $ sudo mkdir -p /var/www/operativos.com/html
[sudo] password for angel:
angel@angel-X456UF ~ $
```

- q) Luego de crear el directorio para nuestro sitio web le asignaremos un usuario propietario con la siguiente línea de código.

**`sudo chown -R $USER:$USER /var/www/operativos.com/html`**

```
angel@angel-X456UF ~ $ sudo chown -R $USER:$USER /var/www/operativos.com/html
angel@angel-X456UF ~ $
```

- r) Mediante el siguiente comando podremos asegurar que únicamente el usuario tenga permisos de escritura, los pertenecientes al mismo grupo del usuario y otros podrán únicamente leer y ejecutar lo que se encuentre en nuestro directorio.

**`sudo chmod -R 755 /var/www/operativos.com/html`**

```
angel@angel-X456UF ~ $ sudo chmod -R 755 /var/www/operativos.com/html
angel@angel-X456UF ~ $
```

- s) Con el directorio que almacenará nuestro sitio web ya creado, pondremos dentro de él el archivo **"index.html"** que será la primera página que cualquier otro usuario verá al intentar ingresar a nuestro sitio web.

**`sudo nano /var/www/operativos.com/html/index.html`**

```
angel@angel-X456UF ~ $ sudo nano /var/www/operativos.com/html/index.html
```

Una vez aquí puedes crear la página índice como tu prefieras, en nuestro caso haremos un saludo e informaremos que se ha accedido satisfactoriamente al sitio web.

```
GNU nano 2.5.3 Archivo: /var/www/operativos.com/html/index.html
<html>
<head>
<title> Directorio Operativos </title>
</head>
<body>
<h1> !!Hola!! Saludos desde /var/www/operativos.com
<br>
<h2> Has logrado acceder a nuestro directorio, significa que todo funciona$
</body>
</html>
[ 10 líneas leídas ]
^G Ver ayuda  ^O Guardar  ^W Buscar  ^K Cortar Texto  ^J Justificar
^X Salir      ^R Leer fich. ^_ Reemplazar ^U Pegar txt  ^T Ortografía
```

- t) Para el correcto funcionamiento de nuestro directorio es necesaria la creación de un archivo de alojamiento virtual que contenga las directivas para la ejecución de Apache.

Para ello usaremos la siguiente línea de código.

**`sudo nano /etc/apache2/sites-available/operativos.com.conf`**

Dentro de este mismo archivo ingresamos las siguientes directivas.





```
<VirtualHost *:8085>
  ServerAdmin admin@operativos.com
  ServerName operativos.com
  ServerAlias www.operativos.com
  DocumentRoot /var/www/operativos.com/html
  ErrorLog ${APACHE_LOG_DIR}/error.log
  CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

- u) Note que tomamos el puerto 8085 del localhost debido a que el 8080 ya lo usa TomCat, además especificamos la ruta que debe seguir para encontrar el índice, que en nuestro caso es **`/var/www/operativos.com/html`**

```
angel@angel-X456UF ~ $ sudo nano /etc/apache2/sites-available/operativos.com.conf
```

```
GNU nano 2.5.3 Archivo: ...sites-available/operativos.com.conf
VirtualHost *:8085>
  ServerAdmin admin@operativos.com
  ServerName operativos.com
  ServerAlias www.operativos.com
  DocumentRoot /var/www/operativos.com/html
  ErrorLog ${APACHE_LOG_DIR}/error.log
  CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>

^G Ver ayuda  ^O Guardar   ^W Buscar    ^K Cortar Texto ^J Justificar
^X Salir      ^R Leer fich. ^\ Reemplazar ^U Pegar txt   ^T Ortografía
```

- v) Ahora tendremos que activar el puerto 8085 en Apache para que nuestro directorio lo pueda usar, para ello usaremos el siguiente comando.

**`sudo nano /etc/apache2/ports.conf`**

```
angel@angel-X456UF ~ $ sudo nano /etc/apache2/ports.conf
```

- w) Una vez dentro agregaremos la línea **Listen 8085** justo debajo de Listen 80, de esta forma quedará habilitado el puerto 8085 en Apache.

x)



```
GNU nano 2.5.3 Archivo: /etc/apache2/ports.conf

# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 80
Listen 8085

<IfModule ssl_module>
    Listen 443

[ 16 líneas leídas ]
^G Ver ayuda ^O Guardar ^W Buscar ^K Cortar Texto ^J Justificar
^X Salir ^R Leer fich. ^\ Reemplazar ^U Pegar txt ^T Ortografía
```

- y) Teniendo las directivas para la ejecución de nuestro directorio en el servidor y el puerto 8085 escuchando, activamos y validamos las directivas con el siguiente comando.
- sudo a2ensite operativos.com.conf**

```
angel@angel-X456UF ~ $ sudo a2ensite operativos.com.conf
Enabling site operativos.com.
To activate the new configuration, you need to run:
    service apache2 reload
angel@angel-X456UF ~ $
```

- z) A pesar de las recomendaciones del sistema aun no recargaremos el servidor Apache puesto que debemos deshabilitar el sitio que tiene configurado por defecto, esta es simplemente una plantilla que podemos usar para crear las directivas para nuestros directorios, para ello usaremos la siguiente línea de comando.
- sudo a2dissite 000-default.conf**

```
angel@angel-X456UF ~ $ sudo a2dissite 000-default.conf
Site 000-default disabled.
To activate the new configuration, you need to run:
    service apache2 reload
angel@angel-X456UF ~ $
```

Por último y antes de recargar el servidor, correremos una prueba útil para encontrar errores de configuración del servidor con el siguiente comando.

**sudo apache2ctl configtest**

```
angel@angel-X456UF ~ $ sudo apache2ctl configtest
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message
Syntax OK
```

Ahora para terminar completamente la configuración recargamos el servidor con el siguiente comando

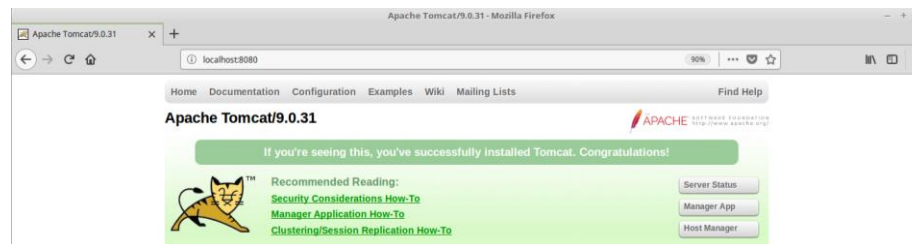
**sudo service apache2 reload**

```
angel@angel-X456UF ~ $ sudo service apache2 reload
angel@angel-X456UF ~ $
```

Ahora probaremos que obtengamos respuesta del puerto 8085 de nuestro navegador, para ello basta con ingresar como URL: **localhost:8085**



*Y si probamos el **localhost:8080** debería seguir funcionando nuestro Tomcat.*





## ANEXO 1

### 1.4 EJERCICIOS

#### EJERCICIOS

- Hospedar una página web en el servidor web y búsquelo por la interfaz localhost.

Si el directorio raíz para el puerto 8080 es **/var/www/html**, entonces crearemos allí un **.html** y lo abriremos desde un navegador.

Sea el archivo **ejercicio1.html**

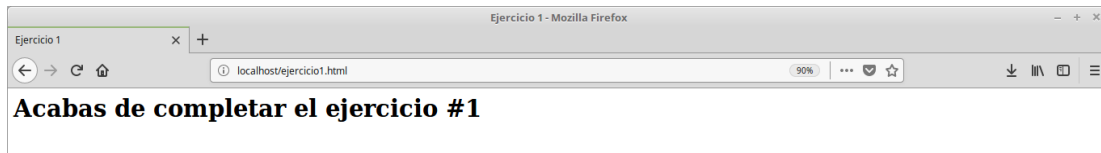
```
angel@angel-X456UF ~ $ sudo nano /var/www/html/ejercicio1.html
angel@angel-X456UF ~ $
```

```
angel@angel-X456UF ~
Archivo Editar Ver Buscar Terminal Ayuda
GNU nano 2.5.3 Archivo: /var/www/html/ejercicio1.html

<html>
<head>
  <title> Ejercicio 1 </title>
</head>
<body>
  <h1> Acabas de completar el ejercicio #1
</body>
</html>

^G Ver ayuda ^O Guardar ^W Buscar ^K Cortar Tex ^J Justificar ^C Posición
^X Salir ^R Leer fich. ^N Reemplazar ^U Pegar txt ^T Ortografía ^_ Ir a línea
```

Ahora trataremos de acceder a él a través de la dirección URL <http://localhost/ejercicio1.html>

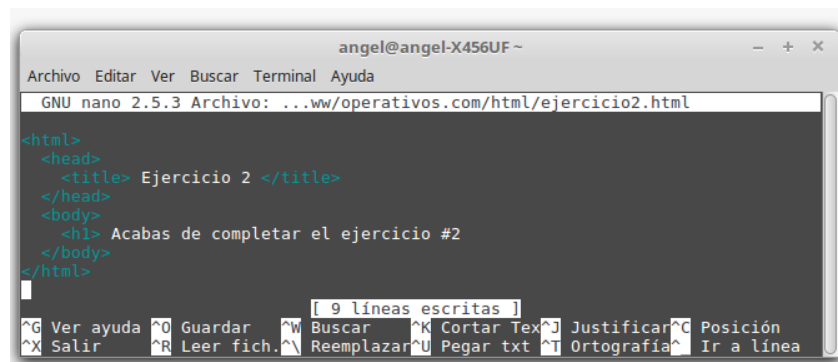


- Hospedar una página web en el servidor web y búsquelo por la interfaz localhost:8085.

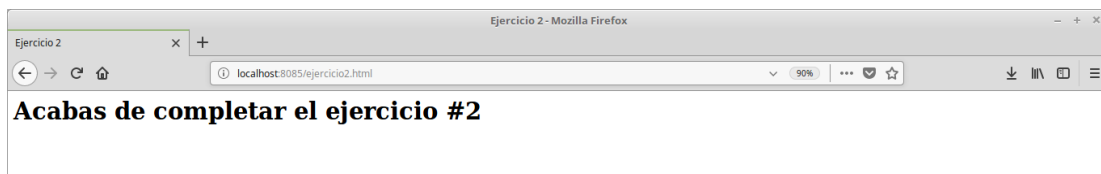
Si el directorio raíz para el puerto 8080 es `/var/www/operativos.com/html`, entonces crearemos allí un `.html` y lo abriremos desde un navegador.

Sea el archivo ***ejercicio2.html***

```
angel@angel-X456UF ~ $ sudo nano /var/www/operativos.com/html/ejercicio2.html
angel@angel-X456UF ~ $
```



Ahora trataremos de acceder a él a través de la dirección URL **<http://localhost:8085/ejercicio2.html>**





- Cambiar el puerto 8085 de **operativos.com** por 8086 (deje abierto el 8085).

Para poder hacer el cambio de puerto tendremos que acceder a su configuración de directivas, si recuerdas para acceder a ellas usamos el comando:

**`sudo nano /etc/apache2/sites-available/operativos.com.conf`**

Luego cambiamos en VirtualHost el puerto, de 8085 a 8086

```
angel@angel-X456UF ~
Archivo Editar Ver Buscar Terminal Ayuda
GNU nano 2.5.3 Archivo: ...e2/sites-available/operativos.com.conf

<VirtualHost *:8086>
    ServerAdmin admin@operativos.com
    ServerName operativos.com
    ServerAlias www.operativos.com
    DocumentRoot /var/www/operativos.com/html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
[ 8 líneas escritas ]
^G Ver ayuda ^O Guardar ^W Buscar ^K Cortar Tex ^J Justificar ^C Posición
^X Salir ^R Leer fich. ^_ Reemplazar ^U Pegar txt ^T Ortografía ^_ Ir a línea
```

Luego accedemos a la configuración de los puertos de Apache2 para que activar el puerto 8086, dejando el puerto 8085 abierto para ello usamos la siguiente línea de comando:

**`sudo nano /etc/apache2/ports.conf`**

```
angel@angel-X456UF ~
Archivo Editar Ver Buscar Terminal Ayuda
GNU nano 2.5.3 Archivo: /etc/apache2/ports.conf

# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 80
Listen 8085
Listen 8086

<IfModule ssl_module>
[ 17 líneas escritas ]
^G Ver ayuda ^O Guardar ^W Buscar ^K Cortar Tex ^J Justificar ^C Posición
^X Salir ^R Leer fich. ^_ Reemplazar ^U Pegar txt ^T Ortografía ^_ Ir a línea
```

Ahora recargaremos el servicio de Apache con el siguiente comando:

**`service apache2 reload`**



Ahora trataremos de acceder al puerto 8086 a través de la dirección URL  
**<http://localhost:8086>**



- Cree un nuevo directorio llamado `ejercicio4.com` y dele permisos de lectura, escritura y ejecución para usuario, lectura y ejecución para grupo y de solo lectura para otros.

Con el siguiente comando, en el directorio de Apache para los sitios web creamos nuestro nuevo directorio con el siguiente comando:

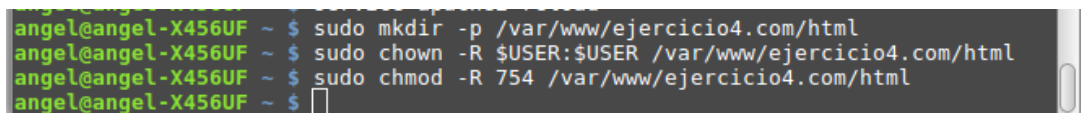
```
sudo mkdir -p /var/www/ejercicio4.com/html
```

Asignamos un usuario propietario con la siguiente línea de código:

```
sudo chown -R $USER:$USER /var/www/ejercicio4.com/html
```

Mediante el siguiente comando configuramos el acceso de usuarios, grupo y otros:

```
sudo chmod -R 754 /var/www/ejercicio4.com/html
```





- Establezca las directivas de conexión para nuestro nuevo directorio para conectarlo al puerto 8085.

Para ello usaremos la siguiente línea de comando:

**`sudo nano /etc/apache2/sites-available/ejercicio4.com.conf`**

Dentro de este mismo archivo ingresamos las siguientes directivas:

Activamos y validamos las directivas con el siguiente comando:

**`sudo a2ensite ejercicio4.com.conf`**

por último, recargamos apache con:

**`service apache2 reload`**





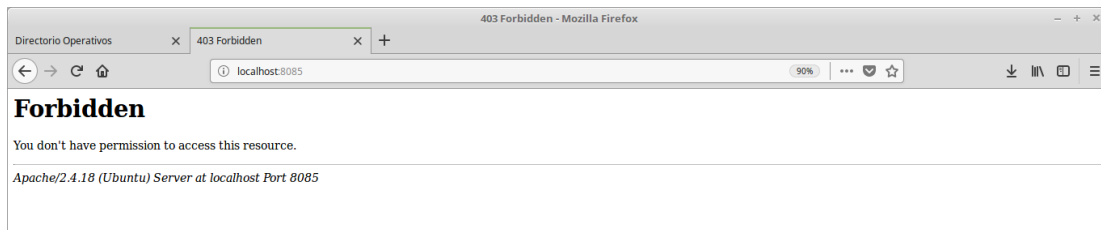
- Cree un archivo `index.html` para `ejercicio4.com` y pruebelo haciendo conexión al puerto 8085:

*Creamos el archivo `index.html` dentro del directorio `ejercicio4.com`*

**`sudo nano /var/www/ejercicio4.com/html/index.html`**

*Ahora probaremos el funcionamiento de nuestro sitio web `ejercicio4.com` accediendo a la dirección URL:*

**`http://localhost:8085`**



*Si recuerdas, en el ejercicio 4, para los otros usuarios solo admitimos permisos de lectura, pero no de ejecución, por tanto, está bien el ejercicio.*



- *Cambie los permisos de tal forma que se pueda ver el contenido del index.html de ejercicio4.com.*

*Para cambiar los permisos usamos la siguiente línea de comandos:*

**`sudo chmod -R 755 /var/www/ejercicio4.com/html`**

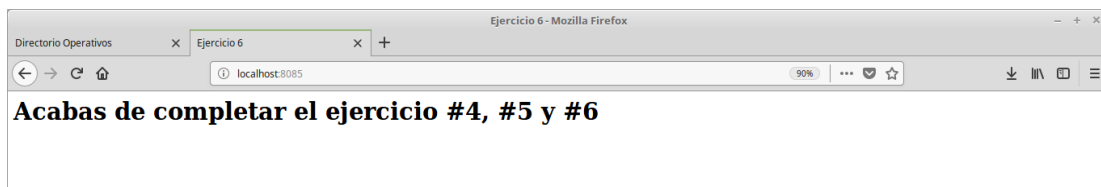
*por último, recargaremos apache 2:*

**`service apache2 reload`**

```
angel@angel-X456UF ~ $ sudo chmod -R 755 /var/www/ejercicio4.com/html
angel@angel-X456UF ~ $ service apache2 reload
angel@angel-X456UF ~ $
```

*Ahora probaremos el funcionamiento de nuestro sitio web ejercicio4.com accediendo a la dirección URL:*

**`http://localhost:8085`**





- Configurar cortafuegos

```
Actividades Terminal
Archivo Editar Ver Buscar Terminal Pestañas Ayuda
guarumo@debian: ~/Documentos/campaign_api x guarumo@debian:
guarumo@debian:~$ sudo ufw app list
[sudo] password for guarumo:
Available applications:
  AIM
  Bonjour
  CIFS
  CUPS
  DNS
  Deluge
  TMAP
```

Listar los perfiles de aplicación. **sudo ufw app list**

Ahora habilite el apache con : **sudo ufw enable**

```
Actividades Terminal
Archivo Editar Ver Buscar Terminal Pestañas Ayuda
guarumo@debian: ~/Documentos/campaign_api x guarumo@debian:
guarumo@debian:~$ sudo ufw enable
Firewall is active and enabled on system startup
guarumo@debian:~$
```

```
Actividades Terminal
Archivo Editar Ver Buscar Terminal Pestañas Ayuda
guarumo@debian: ~/Documentos/campaign_api x guarumo@debian:
guarumo@debian:~$ sudo ufw default deny
Default incoming policy changed to 'deny'
(be sure to update your rules accordingly)
guarumo@debian:~$
```

Termine la activación con el comando **sudo ufw status verbose**



Se debería mostrar que el tráfico HTTP se encuentra permitido:

```
Actividades Terminal dom, 15 de mar, 14:11
guarumo@debian: ~
Archivo Editar Ver Buscar Terminal Pestañas Ayuda
guarumo@debian: ~/Documentos/campaign_api x guarumo@debian: ~/Documentos/campaign_api/ser... x
guarumo@debian:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), deny (routed)
New profiles: skip

To Action From
--
80 ALLOW IN Anywhere
443 ALLOW IN Anywhere
80/tcp ALLOW IN Anywhere
443/tcp ALLOW IN Anywhere
8080/tcp ALLOW IN Anywhere
80 (v6) ALLOW IN Anywhere (v6)
443 (v6) ALLOW IN Anywhere (v6)
80/tcp (v6) ALLOW IN Anywhere (v6)
443/tcp (v6) ALLOW IN Anywhere (v6)
8080/tcp (v6) ALLOW IN Anywhere (v6)
guarumo@debian:~$
```



- Activar el modo seguro de SSH

Para activarlo tenemos que introducir el siguiente comando: **sudo service ssh status**

```
Actividades Terminal ▾ dom, 15 de mar, 14:11
guarumo@debian: ~

Archivo Editar Ver Buscar Terminal Pestañas Ayuda
guarumo@debian: ~/Documentos/campaign_api x guarumo@debian: ~/Documentos/campaign_api/ser... x

guarumo@debian:~$ sudo service ssh status
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2020-03-09 08:22:29 -05; 6 days ago
     Docs: man:sshd(8)
           man:sshd_config(5)
   Main PID: 764 (sshd)
     Tasks: 1 (limit: 4915)
    Memory: 1.6M
   CGroup: /system.slice/ssh.service
           └─764 /usr/sbin/sshd -D

mar 09 08:22:26 debian systemd[1]: Starting OpenBSD Secure Shell server...
mar 09 08:22:29 debian sshd[764]: Server listening on 0.0.0.0 port 22.
mar 09 08:22:29 debian sshd[764]: Server listening on :: port 22.
mar 09 08:22:29 debian systemd[1]: Started OpenBSD Secure Shell server.
guarumo@debian:~$
```

Ahora digitamos el siguiente comando **hostname -I** se te retornará algunas direcciones separadas por espacios. Pruébalas todas en tu navegador web para asegurar su funcionamiento.

```
Actividades Terminal ▾
Archivo Editar Ver Buscar Terminal Pestañas Ayuda
guarumo@debian: ~/Documentos/campaign_api x g

guarumo@debian:~$ hostname -I
192.168.1.27 172.21.0.1 172.17.0.1 172.16.237.1
guarumo@debian:~$
```



- Administrando el proceso de Apache

Ahora que ya cuentas con un servidor web activo y en ejecución, podemos familiarizarnos con algunos comandos básicos de administración con el siguiente comando **sudo systemctl stop apache2** y ponemos la contraseña superusuario.

Para iniciar el servidor web digitamos **sudo systemctl start apache2**

Para detener y reiniciar el servicio en un solo paso, ingresamos **sudo systemctl restart apache2**

si estamos realizando cambios en la configuración, podemos recargar Apache sin necesidad de perder las conexiones que pudieran estar activas con **sudo systemctl reload apache2**

Para evitar que apache inicie siempre automáticamente se ingresa el siguiente comando ya que por defecto está configurado para que al iniciar la máquina este se encienda **sudo systemctl disable apache2**

Para rehabilitar el servicio durante el arranque digitamos **sudo cd /systemctl enable apache2**

```
Actividades Terminal dom, 15 de mar, 14:19
guarumo@debian: ~

Archivo Editar Ver Buscar Terminal Pestañas Ayuda
guarumo@debian: ~/Documentos/campaign_api x guarumo@debian: ~/Documentos/campaign_api/ser... x guarumo@deb

guarumo@debian:~$ sudo systemctl stop apache2
guarumo@debian:~$ sudo systemctl start apache2
guarumo@debian:~$ sudo systemctl restart apache2
guarumo@debian:~$ sudo systemctl reload apache2
guarumo@debian:~$ sudo systemctl disable apache2
Synchronizing state of apache2.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable apache2
Removed /etc/systemd/system/multi-user.target.wants/apache2.service.
guarumo@debian:~$ sudo systemctl enable apache2
Synchronizing state of apache2.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable apache2
Created symlink /etc/systemd/system/multi-user.target.wants/apache2.service → /lib/systemd/system/apache2.service.
guarumo@debian:~$
```



- Configurar Host-Virtuales por IP.

se pueden usar para encapsular detalles de configuración y alojar más de un dominio fuera de un único servidor.

**sudo nano /etc/apache2/sites-available/bloques.conf**

```
#Fichero de configuracion para hosts virtuales por ip
ServerRoot "/etc/apache2/"
DocumentRoot "/var/www"
PidFile /var/run/apache2.pid
User www-data
Group www-data
ErrorLog /var/log/apache2/error.log
Listen 80
TypesConfig /etc/mime.types
DefaultType text/plain
DirectoryIndex index.html index.htm
```

```
#Establecimos la config para cada Host virtual
<VirtualHost 192.168.2.166>
```

```
#Cada uno tiene su nombre
ServerName "ejemplo.com"
DocumentRoot "/var/www/ejemplo.com"
```

```
#Dividimos los ficheros de log
ErrorLog /tmp/aym_ERROR.log
TransferLog /tmp/aym_ERROR.log
</VirtualHost>
```

```
<VirtualHost 192.168.2.200>
ServerName "itd.juntaex.es"
DocumentRoot "/var/www/itd"
Error Log /tmp/itd_ACCESS.log
</VirtualHost>
```



- Configurar Host-Virtuales por puerto.

**sudo nano /etc/apache2/sites-available/bloques.conf**

```
Actividades Terminal
Archivo Editar Ver Buscar Terminal Pestañas Ayuda
guarumo@debian: ~/Documentos/campaign_api x guarumo@
GNU nano 3.2 /et

#Fichero de configuración para hosts virtuales por ip
ServerRoot "/etc/apache2/"
DocumentRoot "/var/www"
PidFile /var/run/apache2.pid
User www-data
Group www-data
ErrorLog /var/log/apache2/error.log
Listen 80
TypesConfig /etc/mime.types
DefaultType text/plain
DirectoryIndex index.html index.htm
```

```
#Establecimos la config para cada Host virtual
<VirtualHost 192.168.2.166:80>
ServerName "aym.juntaex.es"
DocumentRoot "/var/www/aym"
ErrorLog /tmp/aym_ERROR.log
TransferLog /tmp/aym_ERROR.log
</VirtualHost>
<VirtualHost 192.168.2.166:8080>
ServerName "intranet.aym.juntaex.es"
DocumentRoot "/var/www/intranet"
ErrorLog /tmp/aym_intranet_ERROR.log
TransferLog /tmp/aym_intranet_ERROR.log
</VirtualHost>
<VirtualHost 192.168.2.200>
ServerName "idt.juntaex.es"
DocumentRoot "/var/www/idt"
Error Log /tmp/idt_ACCESS.log
</VirtualHost>

^G Ver ayuda      ^O Guardar      ^W Buscar      ^K Cortar
^X Salir          ^R Leer fich.   ^\ Reemplazar   ^U Pegar t
```





- Configurar Host-Virtuales por Nombre.

```
guardar@debian: ~/Documentos/campaign_apr -> guardar
GNU nano 3.2

#Fichero de configuracion para hosts virtuales por ip
ServerRoot "/etc/apache2/"
DocumentRoot "/var/www"
PidFile /var/run/apache2.pid
User www-data
Group www-data
ErrorLog /var/log/apache2/error.log
Listen 80
TypesConfig /etc/mime.types
DefaultType text/plain
DirectoryIndex index.html index.htm
```

```
#La configuracion de hosts virtuales por nombre se aplica
#a las peticiones recibidas en esta IP
NameVirtualHost 192.168.2.166
```

```
#Establecimos la config para cada Host virtual
<VirtualHost ejemplo.com>
ServerName "ejemplo.com"
DocumentRoot "/var/www/ejemplo.com"
ErrorLog /tmp/aym_ERROR.log
TransferLog /tmp/aym_ERROR.log
</VirtualHost>
```

```
<VirtualHost bs.juntaex.es>
ServerName "bs.juntaex.es"
DocumentRoot "/var/www/bs"
Error Log /tmp/bs_ACCESS.log
TransferLog /tmp/bs_ERROR.log
</VirtualHost>
```

```
^G Ver ayuda      ^O Guardar      ^\
^X Salir          ^R Leer fich.    ^_
```

Los siguientes ejercicios son tomados puntualmente del artículo web ***“Como instalar el servidor web Apache en Ubuntu 18.04”*** y ***“Instalar, configurar y solucionar problemas en el servidor web de Linux (Apache)”***



- **Administrando el proceso de Apache**

Ahora que ya cuentas con un servidor web activo y en ejecución, podemos familiarizarnos con algunos comandos básicos de administración.

Para detener tu servidor web, digita:

```
sudo systemctl stop apache2
```

Para iniciar tu servidor web, digita:

```
sudo systemctl start apache2
```

Para detener y reiniciar el servicio en un solo paso, puedes ingresar:

```
sudo systemctl restart apache2
```

Si únicamente estás realizando cambios en la configuración, puedes recargar Apache sin necesidad de perder las conexiones que pudieran estar activas. Para ello, usa el comando:

```
sudo systemctl reload apache2
```

Por defecto, Apache se configura para iniciarse automáticamente cuando el servidor arranca. Si no se quiere esto, se puede deshabilitar este comportamiento, ingresando:

```
sudo systemctl disable apache2
```

Para rehabilitar el servicio durante el arranque, digita:

```
sudo systemctl enable apache2
```

Después de ingresar este comando, Apache debería iniciarse automáticamente durante el arranque del servidor.



- **Autenticación Apache2**

Es posible que desee restringir algunas partes a visitantes específicos. Es como un directorio con contraseña.

En Apache, puedes almacenar un archivo de información de autenticación llamado `.htpasswd`.

Puedes utilizar el comando `htpasswd` para hacer eso.

Primero, crea el archivo `htpasswd` utilizando el comando `htpasswd`:

```
$ htpasswd -c /home/adam/.htpasswd myuser
```

La opción `-c` es necesaria la primera vez que ejecuta `htpasswd`, pero cuando agregues más usuarios no debes usar `-c` porque sobrescribirás el archivo.

Luego crea un archivo `.htaccess` en la carpeta `public_html` y escribe lo siguiente:

```
<Location /vip>

AuthName "test"

AuthType Basic

AuthUserFile /home/adam/.htpasswd

Order deny,allow

require valid-user

</Location>
```

Se requiere `AuthName`, puede usar cualquier cadena que desees.

`AuthType Basic` dice que está utilizando el archivo de usuario `htpasswd`.

`AuthUserFile` apunta al archivo que contiene la contraseña generada desde el comando `htpasswd`.



La línea de Order indica que Apache debe denegar el acceso de forma predeterminada y solo permite el acceso a los usuarios especificados en el archivo htpasswd.

La directiva require significa que cualquier usuario en el archivo htpasswd está permitido.

- **Soporte suEXEC2**

Un método popular es usar suEXEC, un programa que se ejecuta con permisos de root y hace que los programas CGI se ejecuten como las ID de usuarios y grupos de un usuario específico, no del usuario del servidor Apache.

Puedes especificar el usuario en cada host virtual de esta manera:

```
<VirtualHost www.example.com>  
  
SuExecUserGroup adam adamGroup  
  
</VirtualHost>
```

Así de simple.

- **Opción Include**

Esta opción te permite incluir otros archivos de configuración.

Puedes almacenar toda la configuración para diferentes dominios virtuales, y Apache los incluirá en tiempo de ejecución.

```
Include filePath
```

- **Opción UserDir**

Esta opción especifica el directorio que contendrá los archivos a los que se podrá acceder a través del servidor web. Este directorio generalmente se llama public\_html y está ubicado en el directorio de inicio del usuario.

Por ejemplo, si tienes un usuario Adam que quiere que su contenido web esté disponible a través del servidor web Apache.



Primero, hacemos una carpeta public\_html en su directorio de inicio.

A continuación, establece el permiso para la carpeta public\_html:

```
$ chmod 644 public_html
```

Ahora, si ponemos un archivo index.html, podrás acceder a través del navegador de esta manera:

<http://YOURHOSTNAME/~adam>

```
UserDir public_html
```

- **Opción Alias**

Esta opción especifica la ubicación de los archivos que están fuera de la ubicación de DocumentRoot y que necesitan ser servidos por el servidor web Apache.

Como por ejemplos archivos fuera del DocumentRoot que deseas que estén disponibles para los visitantes.

```
Alias URL_Path Actual_Path
```

- **Opción ErrorLog**

Esta opción especifica el archivo de registro de errores para el servidor web Apache..

```
ErrorLog /var/log/httpd/error_log
```



## ANEXO 2

### Respuestas a la Autoevaluación

**a)** ¿Apache usa de alguna forma SSL?

Los Secure Socket Layer, normalmente llamados SSL, son un método para pasar informaciones reservadas a otras máquinas utilizando distintos códigos de ocultamiento. Aunque en el paquete principal de Apache no está presente un soporte SSL, este puede ser descargado y configurado posteriormente.

**b)** ¿Para qué se usa la línea de comando “a2enmod ssl”?

Habilita el módulo ssl en apache.

**c)** ¿Cuales herramientas de Apache 2 se relacionan con la defensa de ataques de fuerza bruta hacia el servidor?

Mod\_Security y Mod\_Evasive

**d)** ¿Cuáles son las características más importantes de Apache?

Apache es un servidor web flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos HTTP. Sus principales características son multiplataforma, modular, extensible.

**e)** ¿Qué son los hosts virtuales?

El concepto de “virtual host” se refiere a que en un mismo servidor web se pueden hospedar múltiples proyectos cada uno con su propio dominio, aunque todos pertenezcan a la misma dirección IP pública.

**f)** El protocolo HTTP trabaja sobre el puerto 80, pero Apache2 dispone de la opción de utilizar las librerías SSL para aumentar la seguridad obteniendo el protocolo HTTPS, con ¿cuál puerto trabaja SSL?

SSL trabaja con el puerto 443

**g)** HTTP es un protocolo “sin estado”, ¿Qué significa esto?

Que HTTP sea un protocolo sin estado significa que no guarda información sobre conexiones anteriores



## ANEXO 3

### BIBLIOGRAFÍA

1. Como instalar el servidor web Apache en Ubuntu 18.04.  
Recuperado de: <https://www.digitalocean.com/community/tutorials/como-instalar-el-servidor-web-apache-en-ubuntu-18-04-es>  
*En la sección Community de la página Digital Ocean Justin Ellingwood nos presenta una breve introducción a servidores HTTP Apache y nos guía cómo instalar un servidor web en Ubuntu 18.04.*
2. Instalar, configurar y solucionar problemas en el servidor web de Linux (Apache).  
Recuperado de: <https://likegeeks.com/es/servidor-web-de-linux-apache/>  
*Mokhtar Ebrahim, de la página Like Geeks nos habla sobre servidores web de Linux, como instalarlos y configurarlos.*
3. Qué es SSL.  
Recuperado de: <https://www.websecurity.digicert.com/es/es/security-topics/what-is-ssl-tls-https>  
*La página de Digicert, una empresa que presta servicios de Website Security nos explica ampliamente que es SSL, TLS y HTTPS.*
4. Qué es HTTPS.  
Recuperado de:  
[https://es.wikipedia.org/wiki/Protocolo\\_seguro\\_de\\_transferencia\\_de\\_hipertexto](https://es.wikipedia.org/wiki/Protocolo_seguro_de_transferencia_de_hipertexto)  
*En la enciclopedia libre Wikipedia nos explican ampliamente que es el Protocolo Seguro de Transferencia de Hipertexto*
5. Certificado CSR.  
Recuperado de: <https://www.certificadosdigitales.net/que-es-un-csr/>  
*La página Certificados Digitales nos explica que es el certificado CSR y por qué es tan importante para establecer el certificado SSL*
6. Configurar Virtual Hosts en Apache y Ubuntu.  
Recuperado de: <https://blog.ahierro.es/como-configurar-virtual-hosts-en-apache-y-ubuntu/>  
*AHierro.es es el blog personal de Andrés Hierro Hernández, este contiene bastantes publicaciones relacionadas con tecnología, entre ellas un tutorial muy didáctico de cómo configurar adecuadamente un Virtual Host para un servidor web Apache instalado en Ubuntu*