qwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnm

|  |
| --- |
| Despliegue Proyecto Web Java  Actividad 3. Grupal.  19/02/2021  Raúl Montero Peregrina Javier Pérez Vivar Patricia Villanueva Polo Jaime Juárez Fernández |

ÍNDICE

[1.- Enunciado 3](#_Toc64980045)

[2.- Introducción 6](#_Toc64980046)

[3.- PATRICIA-TOMCAT+APACHE+BBDD 6](#_Toc64980047)

[1. Instalar y configurar el servidor Apache 6](#_Toc64980048)

[2. Instalación y configuración de la BBDD 9](#_Toc64980049)

[3. Instalación y configuración de TomCat 11](#_Toc64980050)

[4. Instalar Tomcat para desplegar la aplicación web 12](#_Toc64980051)

[5. La BBDD 18](#_Toc64980052)

[4.- RAÚL-REPLICA ANTERIOR+CDN APACHE+CERTIFICADO DIGITAL 20](#_Toc64980053)

[1. Configuraciones y BBDD 20](#_Toc64980054)

[2. APACHE CDN 25](#_Toc64980055)

[3. CERTIFICADO DIGITAL EN APACHE 27](#_Toc64980056)

[Paso 1: Crear el certificado SSL 27](#_Toc64980057)

[Paso 2: Configurar Apache para usar SSL 28](#_Toc64980058)

[Paso 3: Ajustar el firewall 31](#_Toc64980059)

[Paso 4: Habilitar los cambios en Apache 32](#_Toc64980060)

[Paso 5: Probar el cifrado 32](#_Toc64980061)

[5.- JAIME. REPLICA ANTERIOR. INCLUYE FTP Y GESTIÓN DE USUARIOS……………………..……………](#_Toc64980053)35

Paso 1: Instalación del servidor FTP…………………………………………………………………………………….35

Paso 2: Modificar archivo de configuración…………………………………………………………………………35

[Paso 3: Crear nuevos usuarios y dar permisos…………………………………………………….………………](#_Toc64980051)37

[7.- APÉNDICES/BIBLIOGRAFÍA](#_Toc64980062)…………………………………………………………………………………………………39

[8.- OBSERVACIONES](#_Toc64980063)……………………………………………………………………………………………………………….40

# 1.- Enunciado

**Proyecto Web Java**

Este proyecto consiste en una aplicación web Java EE con servlets para acceder a

* Base de Datos
* FTP
* CDN (Apache)

El proyecto se encuentra en GitHub: https://github.com/ocenteno/sample-jndi

Configuración

**1. Base de Datos**

Se debe crear un recurso ***"jdbc/poolBaseDatos"*** que contendrá los parámetros de conexión. La Base de Datos deberá popularse con la siguiente estructura:

**CREATE TABLE PERSONAS (**

**DNI VARCHAR(8) DEFAULT '00000000' NOT NULL,**

**NOMBRE VARCHAR(20),**

**APELLIDOS VARCHAR(20),**

**PRIMARY KEY (dni)**

**);**

**2. Apache CDN**

CDN es una forma habitual de separar el contenido estático de una aplicación web Consiste en ubicarlo en un servidor virtual que se puede replicar para que cada usuario descargue ese contenido del servidor más cercano a su ubicación La aplicación espera que exista un recurso ***"jndi/CDN"*** que apunte a la dirección web donde se han subido los ficheros estáticos se deberá crear ese servidor virtual y el recurso correspondiente en el Servidor de Aplicaciones

**3. FTP**

Se deberá crear un recurso ***"jndi/FTP"*** que contendrá la URL del servidor FTP con el que trabajar El servidor utiliza 2 usuarios con sus respectivas contraseñas:

* registrado:registrado para poder leer y obtener ficheros del servidor, debe tener permisos de lectura
* administrador:administrador para poder subir ficheros al servidor, debe tener permisos de escritura La aplicación trabaja únicamente con el directorio raíz del servidor, no se implementará la descarga o la subida en otra carpeta

**4. Opcional: DNS**

Opcionalmente se podrá registrar nuestro servidor en un DNS (local o público) para poder hacer uso de URLs más representativas para cada una de las piezas del sistema: Apache, FTP y Tomcat

**5. Librerías y ficheros**

1. Base de Datos

Para la conexión a Base de Datos se requieren las librerías Java correspondientes a la Base de Datos en cuestión.

Para MySQL y MariaDB se deberá utilizar: https://mvnrepository.com/artifact/mysql/mysql-connector-java

Para Oracle se deberá utilizar: https://mvnrepository.com/artifact/oracle/oracle-jdbc

Para otras Bases de Datos se deberá utilizar el JDBC correspondiente

2. Contenido

Éste fichero "contenido.zip" contiene

Una imagen que deberá desplegarse en el servidor Apache

Un video que deberá desplegarse en el servidor Apache

El propio fichero contenido.zip deberá desplegarse en el servidor Apache

3. Aplicación

Éste fichero "proyecto-java-ee.war" contiene la aplicación Java que deberá desplegarse en el servidor de aplicaciones

4. FTP

Deberá hacerse accesible en el servidor FTP el PDF que explica la resolución de la práctica y un fichero ZIP que contenga los ficheros de configuración que se han creado. Además, se deberán añadir estos ficheros de configuración al repositorio GitHub que se ha utilizado para las actividades anteriores

5. Despliegue

Se deberán publicar las librerías necesarias en la carpeta /lib del Servidor de Aplicaciones

Se deberá configurar el recurso "jdbc/poolBaseDatos" en el Contexto del Servidor de Aplicaciones

Se deberán configurar las variables de entorno "jndi/FTP" y "jndi/CDN" en el Contexto del Servidor de Aplicaciones

Se deberá desplegar el WAR del proyecto en el Servidor de Aplicaciones

Se deberán crear los usuarios "registrado" y "administrador" en el Servidor FTP

Se deberá configurar el Servidor Apache para poder acceder al directorio CDN y desplegar ahí los ficheros de prueba.

## 5.- PARTE JAVIER REPLICA ANTERIOR Y AÑADE CERTIFICADO DIGITAL EN APACHE

A continuación, vamos a crear un certificado SSL autofirmado para su uso como servidor web de Apache.

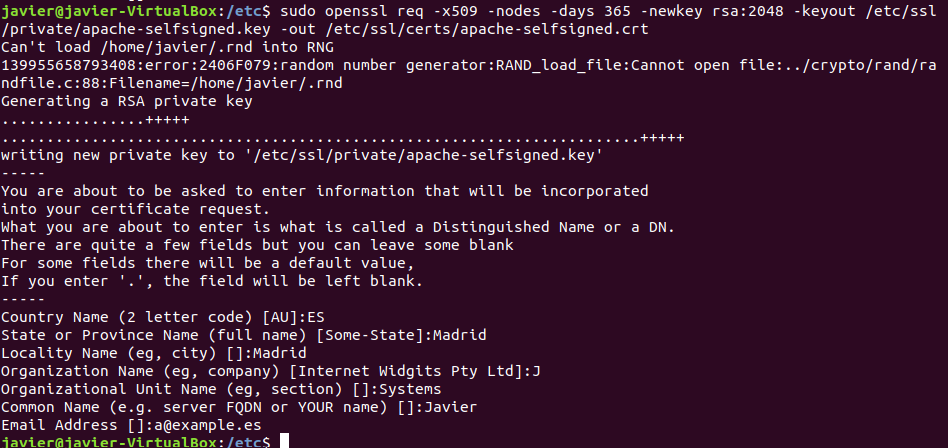
Un certificado autofirmado cifrará la comunicación entre nuestro servidor y cualquier cliente. Sin embargo, dado que no está firmado por ninguna de las autoridades certificadoras de confianza incluidas con los navegadores web, no podremos usar el certificado para validar la identidad de nuestro servidor de forma automática.

## Paso 1: Crear el certificado SSL

La TLS y la SSL funcionan utilizando una combinación de un certificado público y una clave privada. La clave SSL se mantiene secreta en el servidor. Se utiliza para cifrar contenido que se envía a los clientes. El certificado SSL se comparte de forma pública con cualquiera que solicite el contenido. Puede utilizarse para descifrar el contenido firmado por la clave SSL asociada.

Podemos crear un par de clave y certificado autofirmados con OpenSSL en un único comando:

* sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/apache-selfsigned.key -out /etc/ssl/certs/apache-selfsigned.crt



Se nos harán varias preguntas. Antes de abordar esto, observemos lo que sucede en el comando que emitimos:

* **openssl**: es la herramienta de línea de comandos básica para crear y administrar certificados, claves, y otros archivos de OpenSSL.
* **req**: este subcomando especifica que deseamos usar la administración de la solicitud de firma de certificados (CSR) X.509. El “X.509” es un estándar de infraestructura de claves públicas al que se adecuan SSL y TLS para la administración de claves y certificados a través de él. Queremos crear un nuevo certificado X.509, por lo que usaremos este subcomando.
* **-x509**: modifica aún más el subcomando anterior al indicar a la utilidad que deseamos crear un certificado autofirmado en lugar de generar una solicitud de firma de certificados, como normalmente sucede.
* **-nodes**: indica a OpenSSL que omita la opción para proteger nuestro certificado con una frase de contraseña. Necesitamos que Apache pueda leer el archivo, sin intervención del usuario, cuando se inicie el servidor. Una frase de contraseña evitaría que esto suceda porque tendríamos que ingresarla tras cada reinicio.
* **-days 365**: esta opción establece el tiempo durante el cual el certificado se considerará válido. En este caso, lo configuramos por un año.
* **-newkey rsa:2048**: especifica que deseamos generar un nuevo certificado y una nueva clave al mismo tiempo. No creamos la clave que se requiere para firmar el certificado en un paso anterior, por lo que debemos crearla junto con el certificado. La parte rsa:2048 le indica que cree una clave RSA de 2048 bits de extensión.
* **-keyout**: esta línea indica a OpenSSL dónde colocar el archivo de clave privada generado que estamos creando.
* **-out:** indica a OpenSSL dónde colocar el certificado que creamos.

Estas opciones crearán un archivo de clave y un certificado. El sistema hará algunas preguntas sobre nuestro servidor con el fin de insertar la información de forma correcta en el certificado.

**La línea más importante es aquella en la que se solicita Common Name (e.g. server FQDN or YOUR name). Debemos introducir el nombre de dominio asociado con nuestro servidor o la dirección IP pública de nuestro servidor.**

La totalidad de las solicitudes tendrán un aspecto similar a este:

Country Name (2 letter code) [AU]:US

State or Province Name (full name) [Some-State]:New York

Locality Name (eg, city) []:New York City

Organization Name (eg, company) [Internet Widgits Pty Ltd]:Bouncy Castles, Inc.

Organizational Unit Name (eg, section) []:Ministry of Water Slides

Common Name (e.g. server FQDN or YOUR name) []:server\_IP\_address

Email Address []:admin@your\_domain.com

Los dos archivos que hemos creado se ubicarán en los subdirectorios correspondientes en /etc/ssl.

## Paso 2: Configurar Apache para usar SSL

Hemos creado nuestros archivos de clave y certificado en el directorio /etc/ssl. Ahora solo debemos modificar nuestra configuración de Apache para aprovecharlos.

Aplicaremos algunos ajustes a nuestra configuración:

1. Crearemos un fragmento de configuración para especificar configuraciones SSL seguras predeterminadas.
2. Modificaremos el archivo de host virtual de Apache SSL incluido para apuntar a los certificados SSL que generamos.
3. (Recomendado) Modificaremos el archivo de host virtual no cifrado para redireccionar las solicitudes de forma automática al host virtual cifrado.

Al terminar, deberíamos contar con una configuración SSL segura.

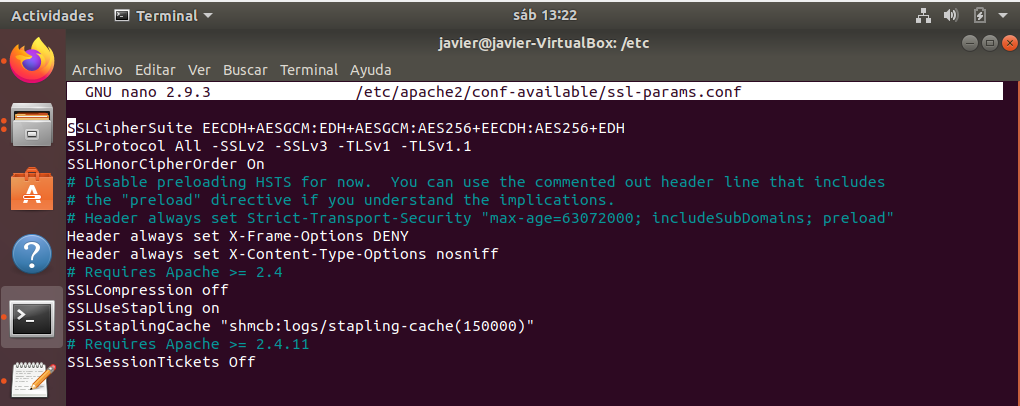
*Crearemos un fragmento de configuración de Apache con ajustes de cifrado seguro*

Primero, crearemos un fragmento de configuración de Apache para definir algunos ajustes de SSL. Con esto, se configurará Apache con un conjunto de cifrado SSL seguro y se habilitarán algunas características avanzadas que ayudarán a mantener protegido nuestro servidor. Los parámetros que configuraremos pueden utilizarse a través de cualquier hosting virtual que habilite SSL.

Crearemos un nuevo fragmento en el directorio /etc/apache2/conf-available. Daremos el nombre ssl-params.conf al archivo para que quede claro su propósito:

* sudo nano /etc/apache2/conf-available/ssl-params.conf





Para configurar Apache SSL de forma segura, usaremos las recomendaciones que Remy van Elst da en el sitio de [Cipherli.st](https://cipherli.st/). Este sitio está diseñado para proporcionar configuraciones de cifrado fáciles de utilizar para software popular.

Las configuraciones propuestas en el sitio del vínculo anterior ofrecen una seguridad confiable. A veces, esto se hace a costa de una mayor compatibilidad con el cliente. Si necesita ofrecer compatibilidad con clientes anteriores, existe una lista alternativa a la que se puede acceder haciendo clic en el enlace de la página con la etiqueta “Sí, quiero un conjunto de cifrado que funcione con el software anterior o heredado”. Esa lista puede sustituirse por los elementos copiados a continuación.

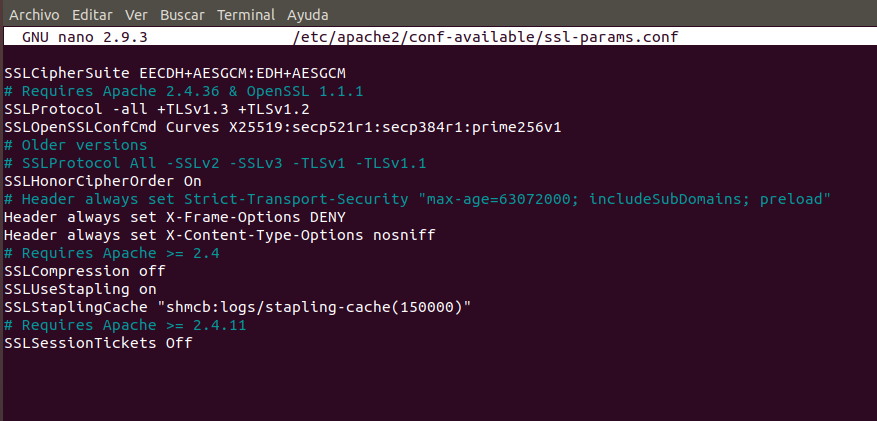
La elección de la configuración que utilizará dependerá en gran medida de aquello para lo que deba ofrecer compatibilidad. Ambas opciones proporcionarán una gran seguridad.

Para nuestros propósitos, podemos copiar las configuraciones proporcionadas en su totalidad. Sólo haremos un pequeño cambio. Deshabilitaremos el encabezado Strict-Transport-Security (HSTS).

La precarga del HSTS proporciona una mayor seguridad, pero puede tener consecuencias importantes si se habilita accidentalmente o de forma incorrecta. En esta guía, no habilitaremos las configuraciones, pero puede aplicar una modificación si está seguro de comprender las implicaciones.

Pegaremos la configuración en el archivo ssl-params.conf que abrimos:

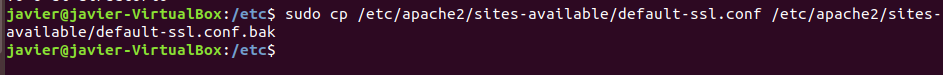
/etc/apache2/conf-available/ssl-params.conf



*Modificar el archivo de host virtual de Apache SSL predeterminado*

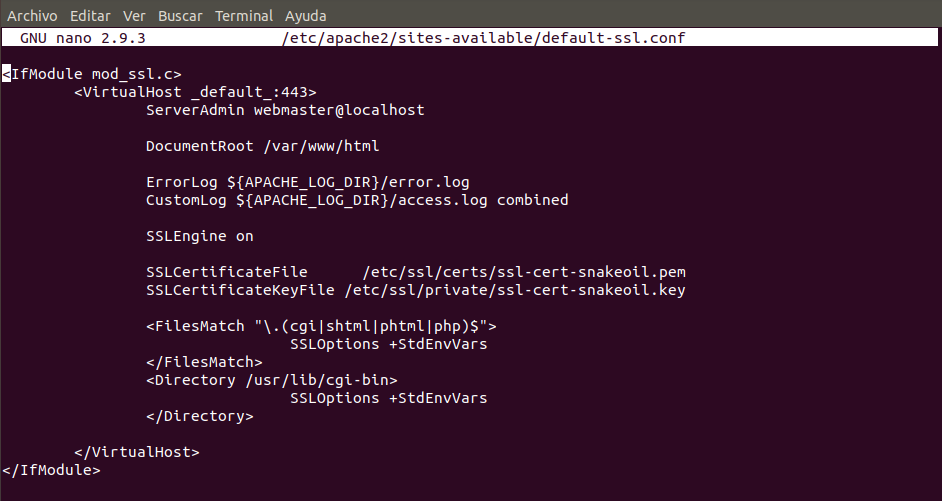
A continuación, modificaremos /etc/apache2/sites-available/default-ssl.conf, el archivo de host virtual de SSL predeterminado. Antes de continuar, realizaremos una copia de seguridad del archivo original de host virtual de SSL:

* sudo cp /etc/apache2/sites-available/default-ssl.conf /etc/apache2/sites-available/default-ssl.conf.bak



Ahora, abriremos el archivo de host virtual de SSL para realizar ajustes:

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



 En su interior, con la mayoría de los comentarios eliminados, el archivo de host virtual debería tener un aspecto similar al siguiente por defecto:

/etc/apache2/sites-available/default-ssl.conf

<IfModule mod\_ssl.c>

<VirtualHost \_default\_:44**3**>

ServerAdmin webmaster@localhost

DocumentRoot /var/www/html

ErrorLog ${APACHE\_LOG\_DIR}/error.log

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

SSLEngine on

SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem

SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key

<FilesMatch "\.(cgi|shtml|phtml|php)$">

SSLOptions +StdEnvVars

</FilesMatch>

<Directory /usr/lib/cgi-bin>

SSLOptions +StdEnvVars

</Directory>

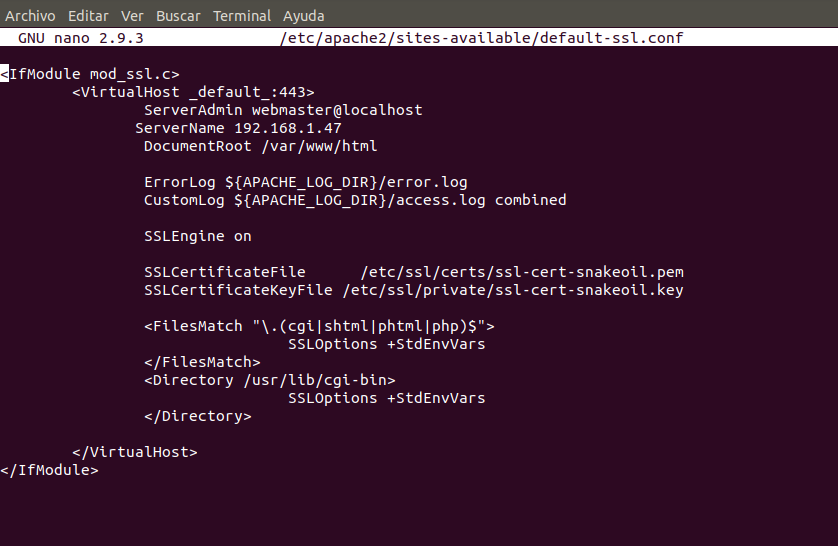
</VirtualHost>

</IfModule>

Haremos algunos ajustes menores en el archivo. Configuraremos la dirección de correo electrónico de ServerAdmin, ServerName, etc.), y ajustaremos las directivas SSL para que apunten a nuestros archivos de certificados y claves.

Tras realizar estos cambios, nuestro bloque de servidor tiene este aspecto:

/etc/apache2/sites-available/default-ssl.conf

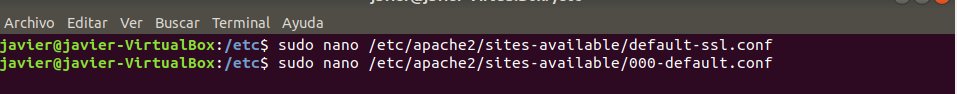


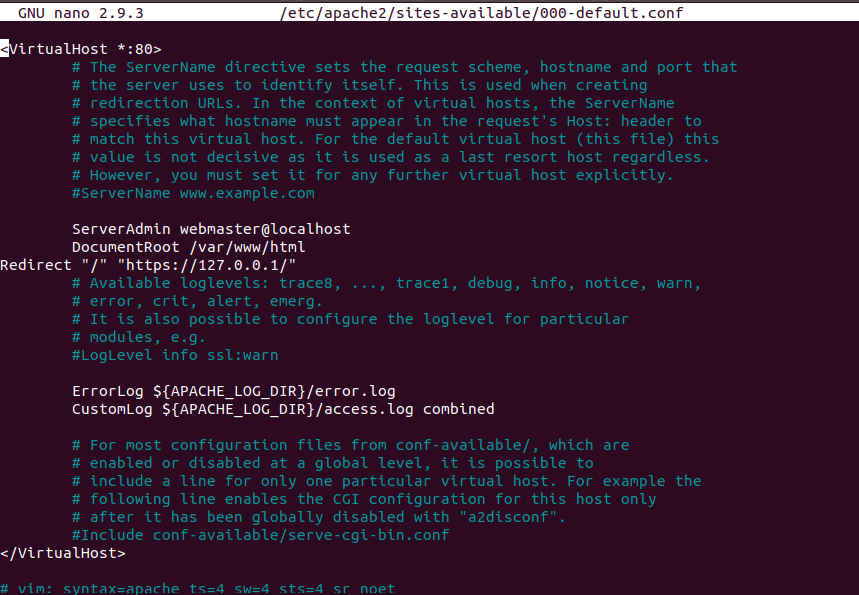
*(Recomendado) Modificar el archivo de host HTTP para el redireccionamiento a HTTPS*

En su estado actual, el servidor proporcionará tanto tráfico HTTP no cifrado como tráfico HTTPS cifrado. Para una mayor seguridad, en la mayoría de los casos se recomienda redireccionar HTTP a HTTPS de forma automática. Si no deseamos esta funcionalidad, podemos omitir esta sección.

Para ajustar el archivo de host virtual no cifrado de modo que se redireccione todo el tráfico y cuente con cifrado SSL, abrimos el archivo /etc/apache2/sites-available/000-default.conf:

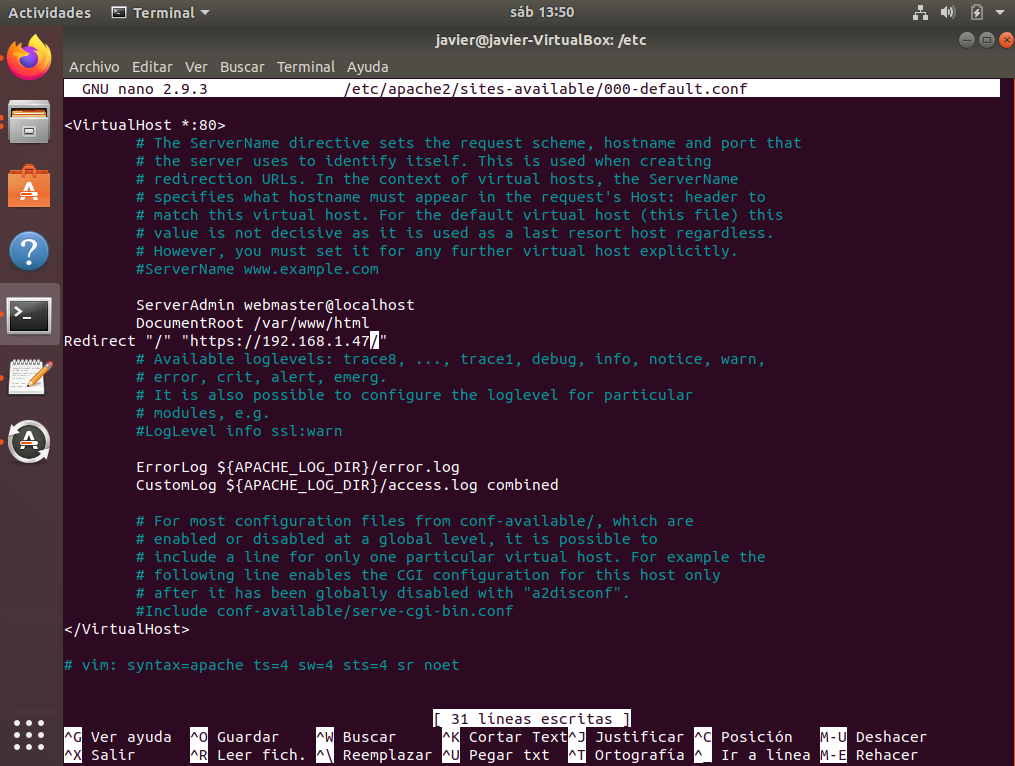
* sudo nano /etc/apache2/sites-available/000-default.conf





Dentro de los bloques de configuración de VirtualHost, debemos añadir una directiva Redirect, que dirija todo el tráfico a la versión SSL del sitio:

/etc/apache2/sites-available/000-default.conf

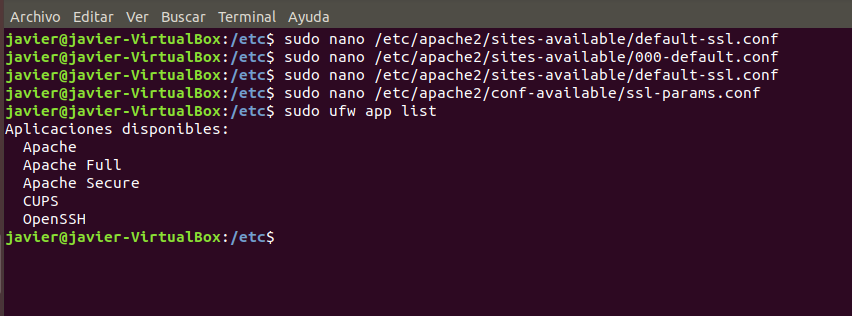


## Paso 3: Ajustar el firewall

Ajustamos la configuración para permitir el tráfico de SSL. Afortunadamente, Apache registra algunos perfiles con ufw después de la instalación.

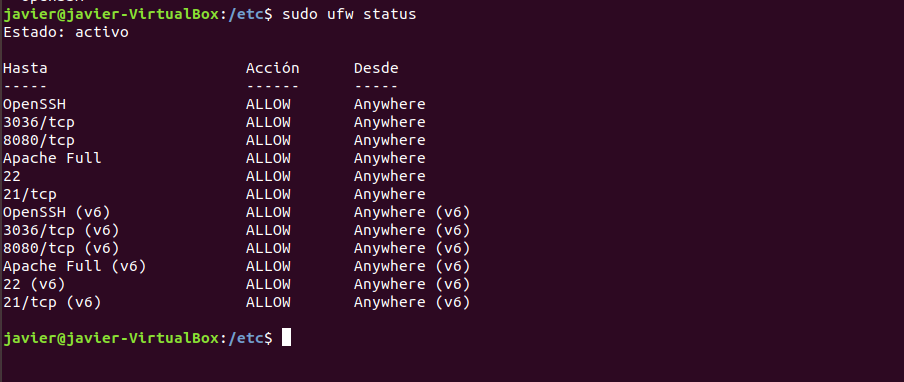
Podemos ver los perfiles disponibles escribiendo lo siguiente:

* sudo ufw app list



Para ver la configuración actual escribimos lo siguiente:

* sudo ufw status

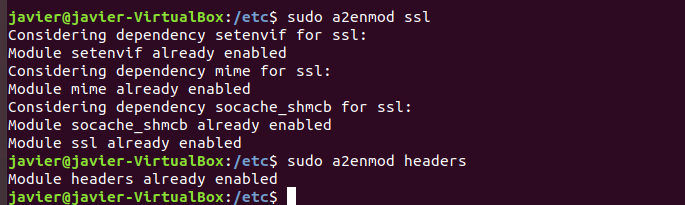


## Paso 4: Habilitar los cambios en Apache

Una vez realizados los cambios y ajustamos el firewall, podemos habilitar los módulos y encabezados SSL de Apache, y también nuestro host virtual listo para SSL, y reiniciar Apache.

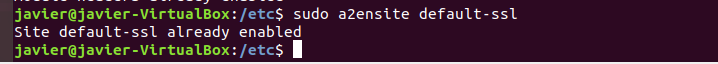
Podemos habilitar mod\_ssl, el módulo SSL de Apache y mod\_headers, que necesitan algunas de las configuraciones de nuestro fragmento SSL, con el comando a2enmod:

* sudo a2enmod ssl
* sudo a2enmod headers



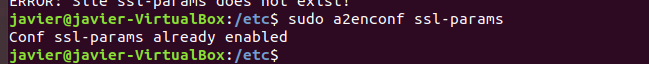
A continuación, podemos habilitar nuestro host virtual SSL con el comando a2ensite:

* sudo a2ensite default-ssl



También debemos habilitar nuestro archivo ssl-params.conf, para leer los valores que configuramos:

* sudo a2enconf ssl-params



En este punto, nuestro sitio y los módulos necesarios quedarán habilitados. Deberíamos comprobar que no haya errores de sintaxis en nuestros archivos. Podemos hacerlo escribiendo lo siguiente:

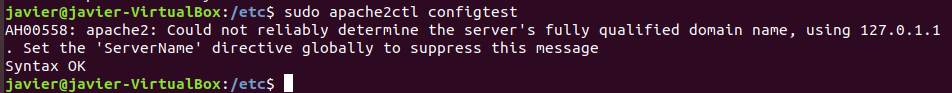
* sudo apache2ctl configtest

Si la operación se completa de forma correcta, obtendremos un resultado similar a este:

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

Vamos a comprobarlo:



La primera línea es solo un mensaje que indica que la directiva ServerName no está configurada a nivel global. Si queremos deshacernos de ese mensaje, podemos establecer ServerName en el nombre de dominio o la dirección IP de su servidor en /etc/apache2/apache2.conf. Esto es opcional, ya que el mensaje no causará problemas.

Si el resultado contiene Syntax OK, en el archivo de configuración no habrá errores de sintaxis. Podemos reiniciar Apache de forma segura para implementar nuestros cambios:

* sudo systemctl restart apache2

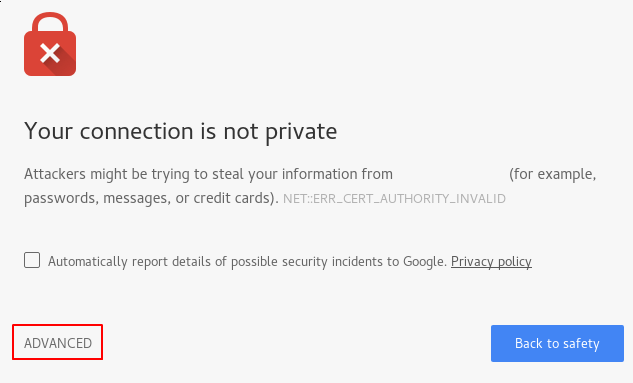
## Paso 5: Probar el cifrado

Ahora, estamos listos para probar nuestro servidor SSL.

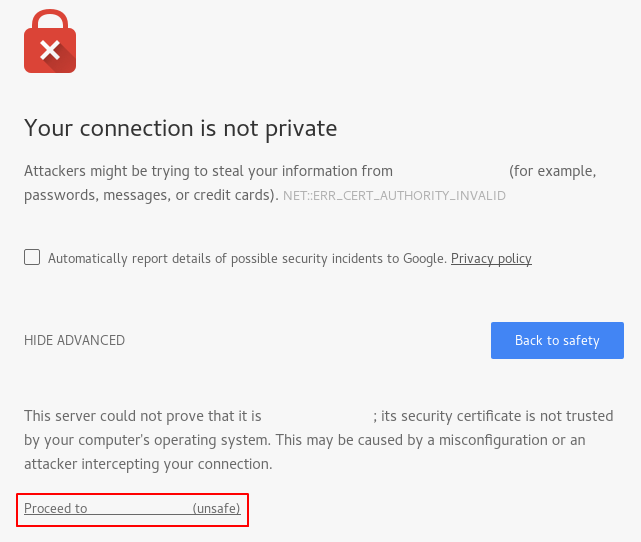
Abrimos el navegador web y escribimos https:// seguido del nombre de dominio o la IP del servidor en la barra de direcciones:

https://server\_domain\_or\_IP

Debido a que el certificado que creamos no está firmado por una de las autoridades de certificados de confianza de nuestro navegador, es probable que veamos la siguiente advertencia:



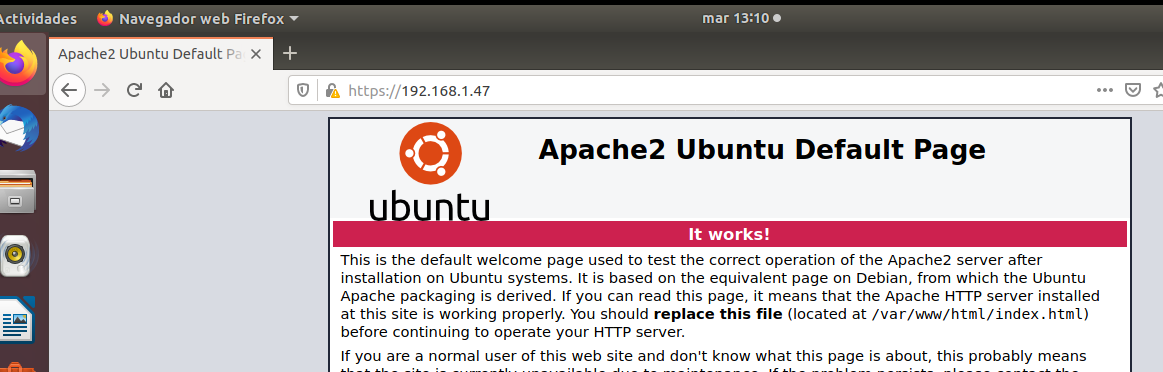
Esto está previsto y es normal. Sólo nos interesa el aspecto de cifrado de nuestro certificado. No nos importa la validación de terceros de la autenticidad de nuestro host. Hacemos clic en “ADVANCED” y luego en el enlace proporcionado para acceder a su host de cualquier manera:



Vemos que en la de direcciones del navegador un candado con una “x” encima. En este caso, esto solo significa que no es posible validar el certificado. Todavía se está cifrando su conexión.

Comprobamos que el redireccionamiento funciona de manera correcta, consignamos

<http://192.168.1.47> y directamente se nos pasa a <https://192.168.1.47> y aparece la página de Ubuntu con lo que se ha instalado de forma correcta el certificado y nuestro servidor Apache es seguro:



Dentro del equipo algunos de los compañeros han configurado localhost, como <https://127.0.0.1>. Repetimos el proceso anterior y comprobamos que también funciona correctamente

