



DESPLIEGUE DE APLICACIONES WEB

# PROXY INVERSO Y BALANCEO DE CARGA CON SSL EN NGINX

# ÍNDICE

<b>INTRODUCCIÓN .....</b>	<b>3</b>
<b>A. PROXY INVERSO Y CIFRADO SSL .....</b>	<b>3</b>
<b>B. CERTIFICADOS .....</b>	<b>4</b>
<b>C. REQUISITOS PARA REALIZAR LA PRÁCTICA .....</b>	<b>5</b>
<b>CREACIÓN DEL CERTIFICADO AUTOFIRMADO.....</b>	<b>5</b>
<b>CONFIGURACIÓN SSL EN EL PROXY INVERSO.....</b>	<b>8</b>
<b>COMPROBACIÓN DE FUNCIONAMIENTO CORRECTO .....</b>	<b>9</b>
<b>REDIRECCIÓN FORZOZA A HTTPS .....</b>	<b>12</b>
<b>ELIMINACIÓN DEL LISTEN 80.....</b>	<b>13</b>
<b>CUESTIONES FINALES.....</b>	<b>14</b>

# INTRODUCCIÓN

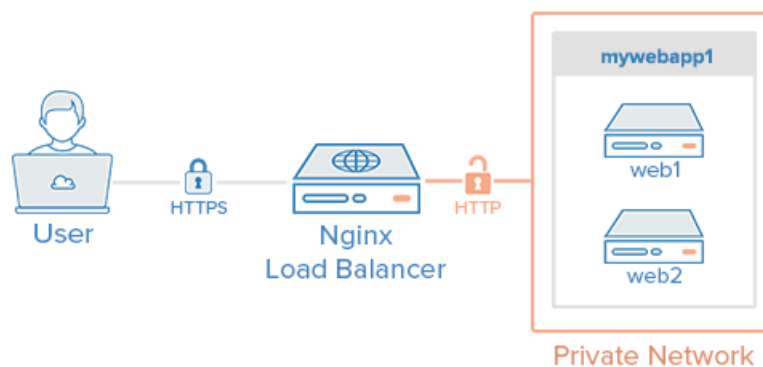
## A. PROXY INVERSO Y CIFRADO SSL

En anteriores prácticas hemos creado un proxy inverso para impedir que hubiera una conexión directa del cliente al servidor, sino que el cliente se tuviera que conectar al proxy y fuera este el que se comunicara con el servidor. Además, para que no todas las solicitudes fueran al mismo servidor, sino que se repartieran (balanceador de carga).

En esta práctica lo que vamos a hacer es darle una capa extra de seguridad a nuestro proxy inverso añadiéndole la capacidad de cifrar y descifrar por medio de SSL para, de esa manera, poder utilizar HTTPS en nuestras conexiones cliente-proxy. Al hacer que sea el proxy el que se encargue de esta tarea de cifrado/descifrado y no el servidor, se disminuye la carga de trabajo de este último.

Lo que vamos a realizar es lo que se ve en el siguiente gráfico:

### Nginx SSL Termination

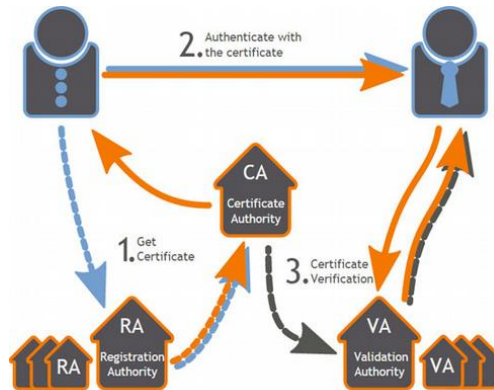


En primer lugar, tendríamos un usuario que a través de un cliente va a solicitar el acceso a nuestro sitio web y lo hará conectándose por medio de HTTPS a nuestro proxy inverso, posteriormente y a través de HTTP se producirá una conexión proxy-servidor para conseguir los recursos que solicita el cliente.

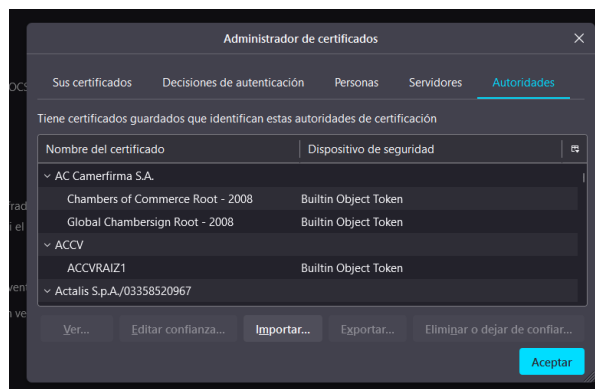
Aunque parezca poco seguro el que la conexión proxy-servidor se haga por medio de HTTP y no de HTTPS la realidad es que no lo es. Si pensamos en un caso real, lo normal es que proxy y servidores estén siendo administrados por las mismas personas, por lo que no existe peligro de que el tráfico generado en la conexión entre estos vaya sin cifrar. Además, en caso de decidir que también fuera cifrado, perdería todo el sentido que utilizáramos el proxy como cifrador/descifrador SSL ya que estaríamos realizando la misma tarea dos veces.

## B. CERTIFICADOS

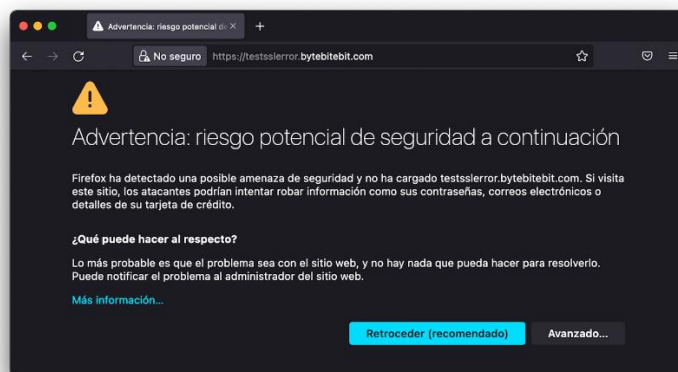
Las conexiones HTTPS se basan en el uso de certificados digitales que son expedidos por Autoridades de Certificación (CA).



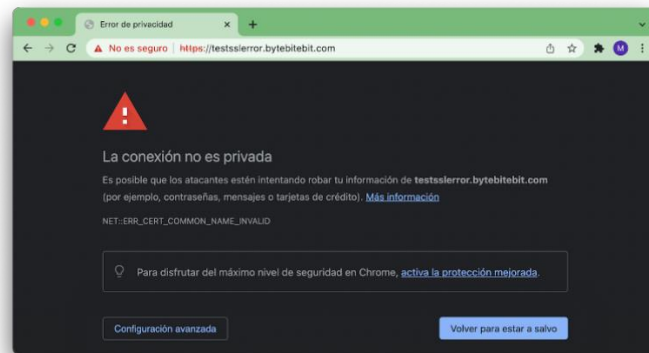
Los navegadores tienen ya precargadas varias CA en las que confían por defecto a la hora de navegar por webs con conexiones HTTPS, se pueden consultar en los ajustes del navegador en la sección de privacidad.



En el caso de que intentemos acceder a una web cuyo certificado no haya sido emitido por alguna de estas CA, el navegador nos informará de que estamos intentando acceder a un sitio potencialmente peligroso. Que, en función del navegador podemos ver de una u otra forma, por ejemplo, en Firefox veremos esta ventana:



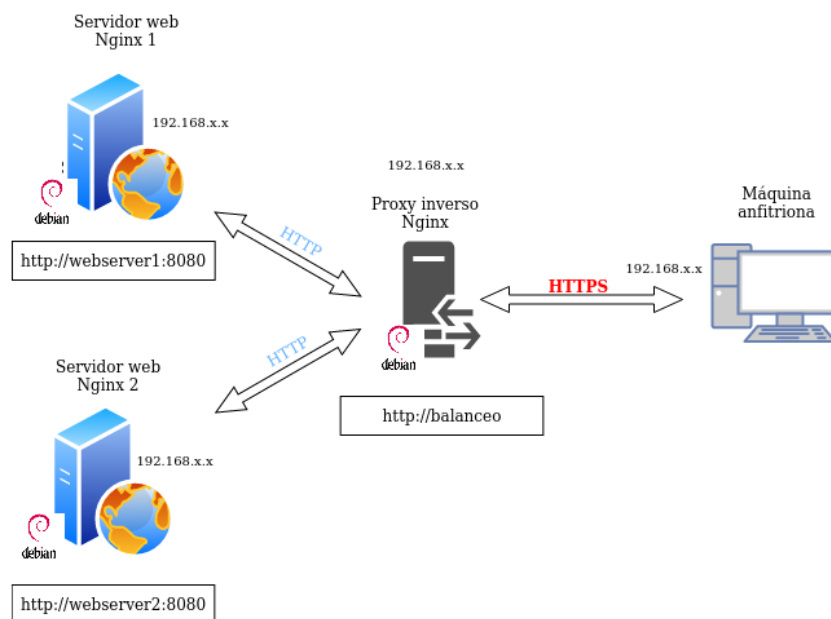
Y en Google Chrome esta otra:



### C. REQUISITOS PARA REALIZAR LA PRÁCTICA

En esta práctica vamos a usar los mismos recursos que en la práctica anterior, es decir, nuestras dos máquinas servidoras Debian 12 y nuestra máquina que ejerce de proxy-balanceador de carga para comprobar el funcionamiento de todo el sistema utilizaremos la máquina virtual de Ubuntu que utilizamos como cliente en anteriores prácticas.

Al completar la práctica, nos debe quedar un gráfico de este estilo:



### **CREACIÓN DEL CERTIFICADO AUTOFIRMADO**

Lo primero que vamos a hacer es crear un certificado autofirmado propio (en la realidad habría que utilizar un CA de confianza y pagarlo) esto va a hacer que cuando accedamos por HTTPS a nuestro sitio web el navegador nos lanzará el aviso que vimos en la introducción.

Vamos a trabajar sobre la máquina virtual que ejerce de proxy y balanceador de carga.

Tenemos que crear un directorio `ssl/` dentro del directorio `/etc/nginx/` por lo que usaremos el comando siguiente:

```
sudo mkdir /etc/nginx/ssl
```

```
albertom-servidor@debian-deaw:~$ sudo mkdir /etc/nginx/ssl
albertom-servidor@debian-deaw:~$ ls -la /etc/nginx/
total 88
drwxr-xr-x  9 root root  4096 nov  1 08:52 .
drwxr-xr-x 135 root root 12288 nov  1 08:43 ..
drwxr-xr-x  2 root root  4096 mar 14 2023 conf.d
-rw-r--r--  1 root root  1125 mar 14 2023 fastcgi.conf
-rw-r--r--  1 root root  1055 mar 14 2023 fastcgi_params
-rw-r--r--  1 root root  2837 mar 14 2023 koi-utf
-rw-r--r--  1 root root  2223 mar 14 2023 koi-win
-rw-r--r--  1 root root  4338 mar 14 2023 mime.types
drwxr-xr-x  2 root root  4096 mar 14 2023 modules-available
drwxr-xr-x  2 root root  4096 mar 14 2023 modules-enabled
-rw-r--r--  1 root root  1446 mar 14 2023 nginx.conf
-rw-r--r--  1 root root   180 mar 14 2023 proxy_params
-rw-r--r--  1 root root   636 mar 14 2023 scgi_params
drwxr-xr-x  2 root root  4096 nov  1 08:36 sites-available
drwxr-xr-x  2 root root  4096 nov  1 08:30 sites-enabled
drwxr-xr-x  2 root root  4096 nov  1 08:02 snippets
drwxr-xr-x  2 root root  4096 nov  1 08:52 ssl
-rw-r--r--  1 root root   664 mar 14 2023 uwsgi_params
-rw-r--r--  1 root root  3071 mar 14 2023 win-utf
```

Vamos a crear el certificado y las claves de forma simultánea con un único comando:

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/nginx/ssl/server.key -out /etc/nginx/ssl/server.crt
```

Cada una de las partes del comando significan lo siguiente:

- `openssl`: Es la herramienta básica para crear y administrar certificados, claves y otros archivos de tipo OpenSSL.
- `req`: Es un subcomando y se utiliza para generar una solicitud de certificados y solicitudes de firma de certificados.
- `-x509`: Modificación del subcomando anterior. Con esto le decimos que queremos crear un certificado autofirmado en lugar de generar una solicitud de firma del certificado (la opción por defecto).
- `-nodes`: Con esta opción se omite el asegurar nuestro certificado con contraseña. Esto es necesario porque Nginx debe poder leer el archivo sin la intervención del usuario cuando se inicia el servidor, si pusiéramos una contraseña se evitaría este inicio automático y habría que introducir la contraseña cada vez que reiniciamos el sistema.
- `-days 365`: Establece el tiempo durante el cual el certificado será válido, en este caso durante 1 año.
- `-newkey rsa:2048`: Especifica que queremos generar un nuevo certificado y clave al mismo tiempo. Utilizaremos una clave de cifrado de tipo RSA de 2048 bits de longitud.

- -keyout: Especifica el lugar donde OpenSSL debe colocar el archivo de la clave privada que estamos generando.
- -out: Especifica el lugar donde OpenSSL debe colocar el certificado que estamos generando.

```
albertom-servidor@debian-deaw:~$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/nginx/ssl/server.key -out /etc/nginx/ssl/server.crt
.....
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.
```

Si hemos introducido el comando correcto ahora nos pedirá que insertemos una serie de parámetros como el país, la provincia, el nombre...

```
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Comunidad de Madrid
Locality Name (eg, city) []:Coslada
Organization Name (eg, company) [Internet Widgits Pty Ltd]:IES Luis Braille
Organizational Unit Name (eg, section) []:2DAW - DEAW - ALBERTO
Common Name (e.g. server FQDN or YOUR name) []:balanceo
Email Address []:miCorreo@email.com
```

Con esto estará creado tanto las claves como el certificado, podemos certificarlo haciendo un listado del directorio /etc/nginx/ssl/:

```
albertom-servidor@debian-deaw:~$ ls -la /etc/nginx/ssl/
total 16
drwxr-xr-x 2 root root 4096 nov  1 08:55 .
drwxr-xr-x 9 root root 4096 nov  1 08:52 ..
-rw-r--r-- 1 root root 1533 nov  1 08:55 server.crt
-rw----- 1 root root 1704 nov  1 08:53 server.key
```

Y estos son el aspecto que tienen la clave:

```
albertom-servidor@debian-deaw:~$ sudo cat /etc/nginx/ssl/server.key
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQFAASCBKcwggSjAgEAAoIBAQC5Jomt5M0W0fMt
JfGuvKxKJhb9XwDgEzsa9nAXi5tF9o8IAAoKBAWidCDjm1lpGpGWi3DkppZhvzdQ
OYCznugtKiRjIqY275Mzq0+erQ9MQklVwV/bIEK/LuGjt+cEQbcio2Nx1iv1MfF0
5HSMfYQ6TyeJX3CID2FHypE6VK43P+36b58F+S1XHe7ZTVGGyTmZhwUMfXrHA5bZ
s772i5HhCof/Btz5Rp0kXkYnbpCmDAIJXDOWda6/MecOH0uqFF6WkYquCZ/5GVvB
2NGx2N23Fju0YQDHncmIAH+bv6khJlPjTXzoXlI8pThefVwmN9nt0/Jqm1iucdRM
x+ER4yypAgMBAAECggEAB40Fj0i5n3guUuWsmth5e02p3tvuxHSsUqltwVwRclyz
ERu1TRrEYw9y3u1xPErX2Kinqj+Mpgxfff+/+rNWmZLJak4zTUugd2nH5bv/LXfX
l057WwuqBTODgsKHgqIudpRsxBH8uMxsX7AL+8XLrHgZb+4LQ8VGfZ/Kg/M7pqec
SMccdg6Q3oDiVpd4Tg0EhJtUfVQs44/xL4JsViYjcrPCiuyk8QrwU4pZyXqt1INxW
1xzYpH/JAXvYZi9rIgKXHRV0F6UoBYgerE7Bo7gthDTPaVqoQP/65ZL3PjQgF4S
Rvqs1gbvYu0HzffrS8HEUM2ga0lyzaJYj+vVwGZMbQKBGQDfQ9jCIsVwa0BKkpuP
pMFYOV2J1Kny3HqsM1B2n3MmjSaZNYaQXWrs8ybzud1MYsk82CCMhdG+DCBHpoz2
DQFP0yUjQ5pHujysRBI3ubNKEsXa7cv4VCuUsMxLraCn7gBsothYucSmBQCwroP
NuKtWuBPYbZyJiWLEfY+Nkn77QKBGQDUTBstrvYLEIKqmfvttQxzc18q93S3L3x
t7Afu1EBiDcBwDfInEm4yM5yADULMA0247pWasx+QcMztswdEYrUEODKLR7pItl
ch169150AEIr/LM5YTl0CXh0yN+JF01KsNKTgK0Yzu8/keASrANG6EAqlopiUnRa
D0yE7Xt0LQKBGAZ0SUH6b25oWeYafhto9RVEnJBImesmH5ikWk4Sa10EUGeIzEI
wDxfhbsehSRiB0B0Zic2Xavqv27E4IjjeTuRr8/Oo1KEkqfRZBzWz/9TD1vrG3a
xEEYKNEry5XW4wCRRWSvIz6gsvf8/fp0B4rz5R/7x35j222ZHUZwJANLaoGAYOHn
fueedSYvr3XWNoCtEfJF/EG5jfsWU6MUSSberOhQge6bV6FdkNLBDsL/Bh127N
mr39Fm0i/oFYHiCx9QpIBQ8KbnKE+uK8+WR9XPNZP2GvGLx5z665vhj9ij6ME03m
i1HprKyjGkukG/7fQhHm8D/AH2apmoAjQXZvJHECGYEAyrPm87p6kiNApeTR2CHZ
mEPEZQKE1L69PXDWkLNo06u3exrb0bovgZc8j0K5e2Q7P5j+30ZdN9KbRgetcRb3
Hvb3Qp4km6SY3/LsPSu4px5jv3mpx2hHoxLQMYV4bI48fK9NziFTWaoN8Tp0HtdY
vY2jcZAVUSUKHhN4FhiWKMq=
-----END PRIVATE KEY-----
```

Y el certificado:

```
albertom-servidor@debian-deaw:~$ sudo cat /etc/nginx/ssl/server.crt
-----BEGIN CERTIFICATE-----
MIIEPzCCAyegAwIBAgIUa60scvJ7s5D9DQEq0QzEvYr+IT8wDQYJKoZIhvcNAQEL
BQAwg4xCzAJBgNVBAYTAkVTRWwGgYDVQQIDBNDb211bm1kYWNQZGUGTWFkcm1k
MRAwDgYDVQQHDAdDb3NsYWRhMRkwFwYDVQQKDBBJRVVpYBpGx1MR4w
HAYDVQQLDUByREFXIC0gREVBYyAtIEFMQkV5VE8xETAPBgNVBAMMCGJhbGZyY2Vv
MSEwHwYJKoZIhvcNAQkBFhJtaUNvcnJlb0B1bWZpbC5jb20wHhcNMjMxMTAxMDc1
NTAzWhcNMjMxMDc1NTAzWjCBjzELMAkGA1UEBhMCVVMxHDAaBgNVBAGME0Nv
bXVuaWRhZCBkZSBYWRyaWQxEDA0BgNVBACMB0Nvc2xhZGExGTAXBgNVBAoMEEFf
UyBMdWl3IEJyYXV1bWZyYyAtIEFMQkV5VE8xETAPBgNVBAsMTJEQVcGLSBERUF
XIC0gQUxCRVJUTzERMA8GA1UEAwwIYmF5ZW5jZW8xITAFBgkqhkiG9w0BCQEWEmIp
Q29ycmVvQGVTYVW1sLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAL
kmaia3kzRY5+a018a68iEomFv1fA0AT0xr2cBeLm1/2jwgACgoEBAJ0IO0bWwKakZ
aLcOSmpmG/N1A5gl0e6C2IqsmJDLbvKxmT56tD0xCSVXB9sgQr8u4a035wRBtyKjY
3HWK+Ux8U7kdIx9hdpPJ4lfcIgpYUfKkTpUjrc/7fptLwX5KVcd7t1NUYbJOZmFZ
Qx9escD1tmzvvaLkeKh/8G3NKunSReRidu1wx0Ag1cOhZ1r8x5w4c66oUXpRiq4Jn/
kZXAHY0bHY3bcW045hAMedyYgAf5u/qQe0U+NNfoHeUjy10F59VaY32e3T8mqawK
5x1EzH4RHjLKKCAwEAAANTMFEEHQYDVR0BBYEFHQXQF935gchwoeevDEua9EoEzP
MB8GA1UdIwQYMBaAFHQXQF935gchwoeevDEua9EoEzPMA8GA1UdEwEB/wQFMAMB
Af8wDQYJKoZIhvcNAQELBQADggEBAJeZ0aINhfQTTvN3RAB/Azitn+Nwsz18Zczkg
CZW hQcKz5TjDKpR0njz5j2cig/ooFw/b1qAaga9sx+KCuVZc6EryFTVe7DgTeR
NJ5ydXVf47EwupBN2InTnw8k61fZyKfocG+3ejfb15x7OKHyJL58oct6pB6CaR
aNF+89wqeQVa0jwqHIoqiCCWwifZTt/rT7jMnEqa8eV50j4G+D38XW4uyC9M07J
1pwNqFXZTmAviFRgJj1zUzaqMd3QX1SP6V1xzdzjRA4d00z7H2p9xwoIxx8QsE
71JX8vWAgig8nAiQ5ZmCfTqm2g3gkphCmU0GQWJzDenPPmXEJYd1p7sMI=
-----END CERTIFICATE-----
```

## CONFIGURACIÓN SSL EN EL PROXY INVERSO

Una vez que tenemos creado el certificado autofirmado debemos implementarlo en nuestro archivo de configuración del proxy, es decir, en el archivo que creamos en la práctica anterior en el directorio /etc/nginx/sites-available/

`sudo nano /etc/nginx/sites-available/balanceo`

```
albertom-servidor@debian-deaw:~$ sudo nano /etc/nginx/sites-available/balanceo
```

Y el bloque server debe quedar así:

```
server {
    listen 443 ssl;
    ssl_certificate /etc/nginx/ssl/server.crt;
    ssl_certificate_key /etc/nginx/ssl/server.key;
    ssl_protocols TLSv1.3;
    ssl_ciphers ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES128:ECDH+3DES:DH+3DES:RSA+AESGCM:RSA+AES:RSA+3DES:!aNULL:!MD5:!DSS;
    server_name www.balanceo.es;
    access_log /var/log/nginx/https_access.log;
    location / {
        proxy_pass http://backend_hosts;
    }
}
```

- listen: Pasamos de usar el puerto 80 a usar el 443 que es el puerto por defecto para HTTPS.
- ssl\_certificate: Aquí especificamos el directorio donde está el certificado. En este caso /etc/nginx/ssl/server.crt.
- ssl\_certificate\_key: Aquí especificamos el directorio donde está la clave. En este caso /etc/nginx/ssl/server.key.
- ssl\_protocols: Aquí se colocan las versiones de los protocolos que consideramos seguros hoy en día. En este caso hemos elegido TLSv1.3.
- ssl\_ciphers: En este apartado se colocan los cifrados que consideramos seguros hoy en día. En este caso hemos colocado los siguientes:

ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES128:ECDH+3DES:DH+3DES:RSA+AESGCM:RSA+AES:RSA+3DES:!aNULL:!MD5:!DSS



- `server_name`: Al igual que en otras prácticas será el nombre de dominio que tendrá la IP del proxy. En este caso seguimos usando el `www.balanceo.es` de la práctica anterior.
- `access_log`: Vamos a guardar los logs de acceso al archivo `https_access.log`.

Cerramos el archivo de configuración guardando cambio y reiniciamos el servicio de Nginx.

```
albertom-servidor@debian-deaw:~$ sudo systemctl restart nginx.service
albertom-servidor@debian-deaw:~$ sudo systemctl status nginx.service
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; preset: enabled)
   Active: active (running) since Wed 2023-11-01 09:00:31 CET; 5s ago
     Docs: man:nginx(8)
   Process: 2991 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Process: 2992 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Main PID: 2994 (nginx)
    Tasks: 3 (limit: 2284)
   Memory: 2.6M
      CPU: 21ms
   CGroup: /system.slice/nginx.service
           └─2994 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             └─2995 "nginx: worker process"
               └─2996 "nginx: worker process"

nov 01 09:00:31 debian-deaw systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server...
nov 01 09:00:31 debian-deaw systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server.
```

## COMPROBACIÓN DE FUNCIONAMIENTO CORRECTO

Completado el paso previo de configuración del sitio debemos comprobar que el funcionamiento de nuestro sitio web es correcto, para ello vamos a arrancar ambas máquinas servidoras y modificamos el archivo `/etc/hosts` del proxy con las nuevas IPs de los servidores (vamos a realizar las pruebas en Red NAT):

```
GNU nano 7.2
127.0.0.1      localhost
127.0.1.1      debian-deaw
10.0.2.15      webserver1
10.0.2.19      webserver2
10.0.2.18      www.balanceo.es
```

Y en la máquina cliente también modificamos este archivo añadiendo la IP del proxy con el correspondiente nombre de dominio:

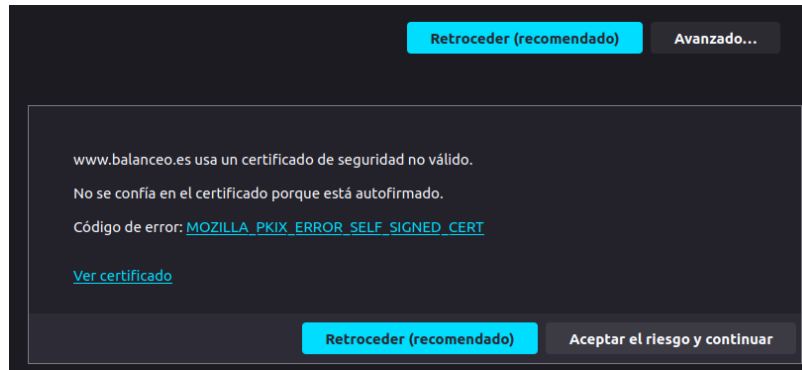
```
GNU nano 6.2
127.0.0.1      localhost
127.0.1.1      albertomcliente-VirtualBox
10.0.2.18      www.balanceo.es
```

Si ahora intentamos acceder a `https://www.balanceo.es` nos debería salir una pantalla como la siguiente:



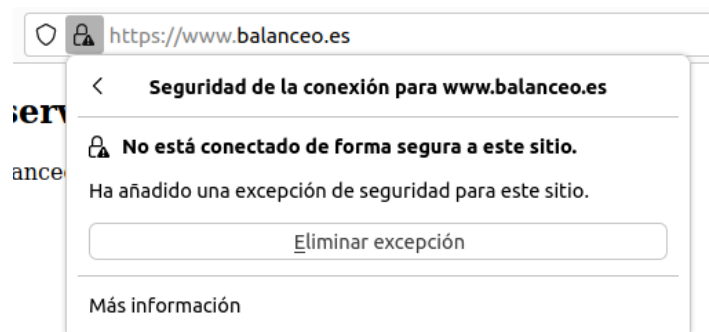
Un aviso del navegador informándonos de que el sitio es potencialmente peligroso al tratarse de un certificado autofirmado y no uno emitido oficialmente.

Para poder acceder debemos hacer clic en “Avanzado...” y en “Aceptar el riesgo y continuar”:

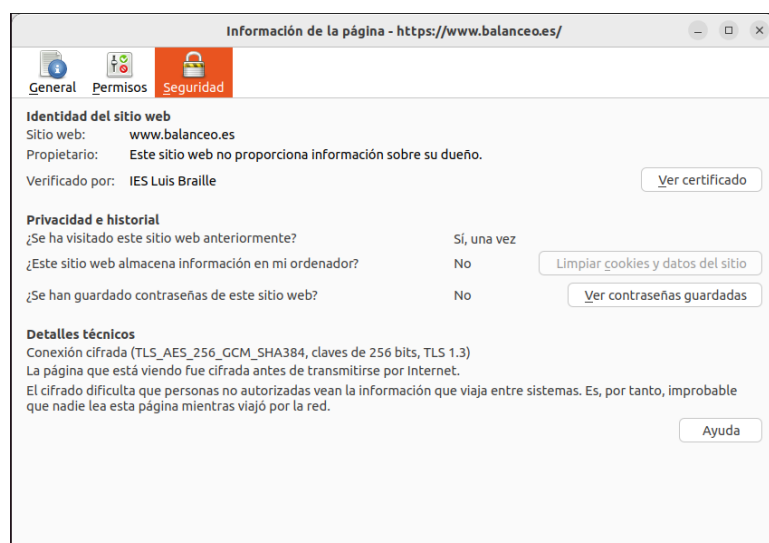


Una vez dentro podremos hacer F5 varias veces para comprobar que el balanceador de carga sigue haciendo su función derivándonos al servidor 1 o al 2 de forma aleatoria.

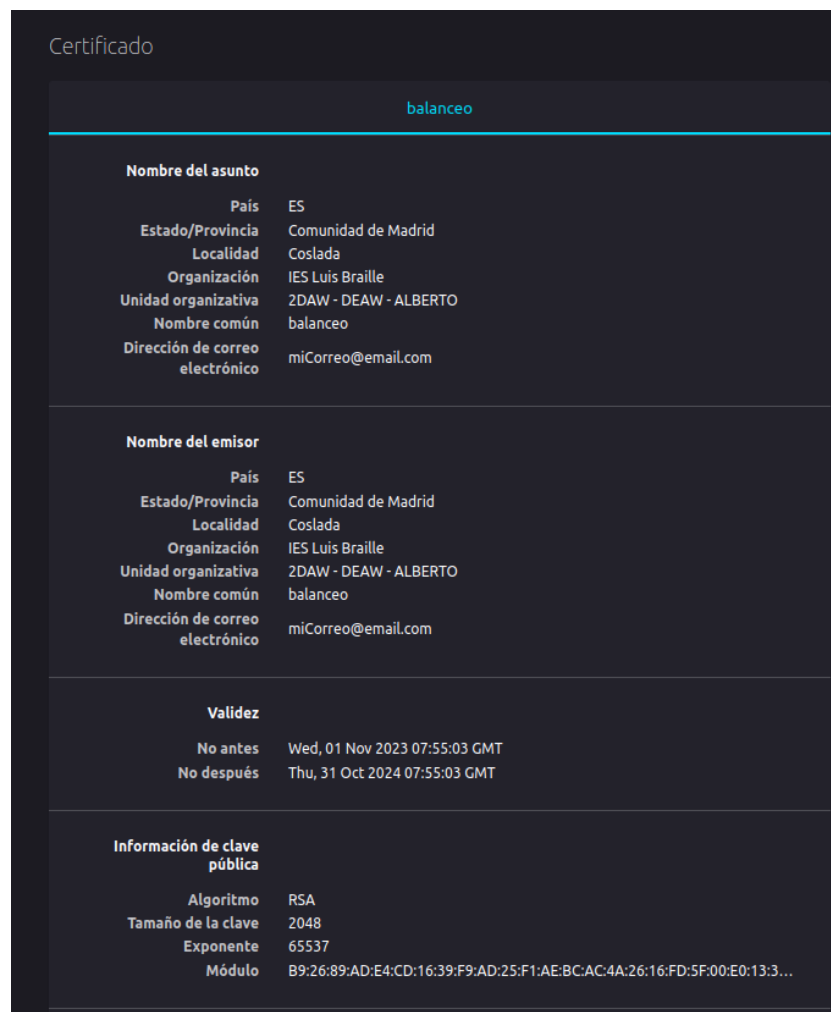
Podemos comprobar los datos del certificado haciendo clic en el candado de la barra del navegador y haciendo clic en “Más información”:



En la nueva ventana que se nos abre debemos hacer clic en “Ver certificado”:



En la nueva pestaña que se nos habrá abierto en el navegador podemos comprobar la información:



Como podemos ver, coincide con la información que introdujimos cuando creamos el certificado y el par de claves pública y privada.

Si ahora accedemos al log que hemos creado veremos los diferentes accesos que hemos hecho para comprobar el funcionamiento del balanceador de carga.

```
sudo tail /var/log/nginx/https_access.log
```

```
albertom-servidor@debian-deaw:~$ sudo tail /var/log/nginx/https_access.log
10.0.2.6 - - [01/Nov/2023:09:13:38 +0100] "GET / HTTP/1.1" 200 257 "-" "Mozilla/5
10.0.2.6 - - [01/Nov/2023:09:13:41 +0100] "GET / HTTP/1.1" 200 258 "-" "Mozilla/5
10.0.2.6 - - [01/Nov/2023:09:13:42 +0100] "GET / HTTP/1.1" 200 257 "-" "Mozilla/5
10.0.2.6 - - [01/Nov/2023:09:13:43 +0100] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0
10.0.2.6 - - [01/Nov/2023:09:13:43 +0100] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0
10.0.2.6 - - [01/Nov/2023:09:13:44 +0100] "GET / HTTP/1.1" 200 258 "-" "Mozilla/5
10.0.2.6 - - [01/Nov/2023:09:13:44 +0100] "GET / HTTP/1.1" 200 257 "-" "Mozilla/5
10.0.2.6 - - [01/Nov/2023:09:13:45 +0100] "GET / HTTP/1.1" 200 258 "-" "Mozilla/5
10.0.2.6 - - [01/Nov/2023:09:13:45 +0100] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0
10.0.2.6 - - [01/Nov/2023:09:13:46 +0100] "GET / HTTP/1.1" 200 257 "-" "Mozilla/5
```

## REDIRECCIÓN FORZOZA A HTTPS

Ahora vamos a hacer que indistintamente de la forma por la que accedamos al sitio web `www.balaneo.es` (bien por `http://www.balaneo.es` o `https://www.balaneo.es`) todas las peticiones vayan siempre por HTTPS.

Para ello en el archivo de configuración de `sites-available/` vamos a añadir un segundo bloque `server` que debe estar separado del otro:

```
server {  
    listen 80;  
    listen [::]:80;  
    server_name www.balaneo.es  
    access_log /var/log/nginx/http_access.log;  
    return 301 https://www.balaneo.es$request_uri;  
}
```

Los parámetros son los siguientes:

- `listen`: El puerto de escucha 80 (por defecto en HTTP) tanto para IPv4 como IPv6.
- `server_name`: El nombre de dominio de nuestro sitio web.
- `access_log`: Vamos a crear un nuevo archivo de logs para almacenar los accesos que se produzcan por HTTP.
- `return 301`: Cuando se reciba una petición a nuestro sitio por HTTP se devolverá el código 301. 301 es un código de estado que indica que el host ha sido capaz de comunicarse con el servidor pero que el recurso solicitado ha sido movido a otra dirección de forma permanente. Esto nos va a permitir redirigir a los usuarios del sitio web http al https.

Como siempre que tocamos un archivo de configuración debemos llevar a cabo un reinicio del servicio Nginx.

Si ahora intentamos acceder a `http://www.balaneo.es` automáticamente nos redirige a `https://www.balaneo.es`

Podemos comprobar los accesos que se produzcan a nuestra web por HTTP en el log de `http_access.log` que hemos creado:

```
albertom-servidor@debian-deaw:~$ sudo tail -1 /var/log/nginx/http_access.log  
10.0.2.6 - - [01/Nov/2023:09:28:46 +0100] "GET / HTTP/1.1" 301 169 "-" "Mozilla"
```

Como podemos ver se ha producido una redirección 301 que provoca una conexión registrada en el `https_access.log`.

```
albertom-servidor@debian-deaw:~$ sudo tail -2 /var/log/nginx/https_access.log  
10.0.2.6 - - [01/Nov/2023:09:28:51 +0100] "GET / HTTP/1.1" 200 258 "-" "Mozilla"
```

## ELIMINACIÓN DEL LISTEN 80

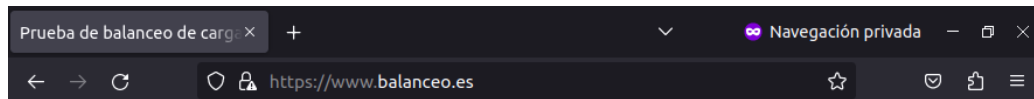
Vamos a eliminar del bloque server el listen 80 para ver qué ocurre si intentamos acceder a nuestro sitio web por HTTP. Por tanto, lo primero que debemos hacer es cambiar el archivo de configuración de balanceo eliminando el puerto de escucha 80 en el bloque server que hemos creado en el apartado anterior:

```
server {
    server_name www.balanceo.es;
    access_log /var/log/nginx/http_access.log;
    return 301 https://www.balanceo.es$request_uri;
}
```

Reiniciamos el servicio para que este cambio tenga efecto:

```
albertom-servidor@debian-deaw:~$ sudo systemctl restart nginx.service
albertom-servidor@debian-deaw:~$ sudo systemctl status nginx.service
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; preset: enabled)
   Active: active (running) since Fri 2023-11-03 16:24:22 CET; 1s ago
     Docs: man:nginx(8)
  Process: 2640 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
  Process: 2641 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
 Main PID: 2642 (nginx)
    Tasks: 3 (limit: 2284)
   Memory: 2.7M
      CPU: 24ms
  CGroup: /system.slice/nginx.service
          └─2642 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             └─2643 "nginx: worker process"
                └─2644 "nginx: worker process"
```

Y ahora en la máquina cliente intentamos un acceso a <http://www.balanceo.es>:



### Este es el servidor web 2

Comprueba el balanceo de carga con Nginx recargando esta página.

Se nos redirige de forma automática a <https://www.balanceo.es> y esta acción queda reflejada tanto en `http_access.log`:

```
albertom-servidor@debian-deaw:~$ sudo tail -1 /var/log/nginx/http_access.log
10.0.2.6 - - [03/Nov/2023:16:28:08 +0100] "GET / HTTP/1.1" 301 169 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/119.0"
```

Como en el `https_access.log`:

```
albertom-servidor@debian-deaw:~$ sudo tail -2 /var/log/nginx/https_access.log
10.0.2.6 - - [03/Nov/2023:16:28:13 +0100] "GET / HTTP/1.1" 200 257 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/119.0"
10.0.2.6 - - [03/Nov/2023:16:28:13 +0100] "GET /favicon.ico HTTP/1.1" 404 125 "https://www.balanceo.es/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/119.0"
```

Como podemos ver en el primer log (el del HTTP) se ha producido esa redirección 301 que nos lleva al sitio seguro.

## CUESTIONES FINALES

- a) Hemos configurado nuestro proxy inverso con todo lo que nos hace falta pero no nos funciona y da un error del tipo `This site can't provide a secure connection, ERR_SSL_PROTOCOL_ERROR`.

Dentro de nuestro server block tenemos esto:

```
server {
    listen 443;
    ssl_certificate /etc/nginx/ssl/enrico-berlinguer/server.crt;
    ssl_certificate_key /etc/nginx/ssl/enrico-berlinguer/server.key;
    ssl_protocols TLSv1.3;
    ssl_ciphers
ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:ECDH+3DES:DH+3DES:RSA+AESGCM:RSA+AES:RSA+3DES:!aNULL:!MD5:!DSS;
    server_name enrico-berlinguer;
    access_log /var/log/nginx/https_access.log;

    location / {
        proxy_pass http://red-party;
    }
}
```

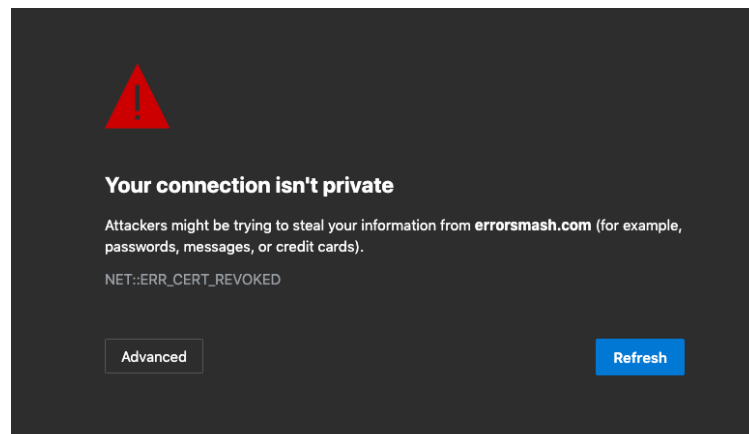
Falta especificar el protocolo SSL/TLS en el listen. Si se agrega la palabra “ssl”, se habilita el SSL/TLS en el servidor.

Por tanto, el código quedaría así:

```
server {
    listen 443 ssl;
    ssl_certificate /etc/nginx/ssl/enrico-berlinguer/server.crt;
    ssl_certificate_key /etc/nginx/ssl/enrico-berlinguer/server.key;
    ssl_protocols TLSv1.3;
    ssl_ciphers
ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:ECDH+3DES:DH+3DES:RSA+AESGCM:RSA+AES:RSA+3DES:!aNULL:!MD5:!DSS;
    server_name enrico-berlinguer;
    access_log /var/log/nginx/https_access.log;

    location / {
        proxy_pass http://red-party;
    }
}
```

- b) Imaginad que intentamos acceder a nuestro sitio web HTTPS y nos encontramos con el siguiente error:




*Investigad qué está pasando y como se ha de solucionar.*

El error `NET::ERR_CERT_REVOKED` implica que el certificado SSL utilizado para el sitio web ha sido revocado por la entidad emisora o por alguna razón de seguridad. Esto puede ocurrir por ejemplo cuando se detecta un compromiso de seguridad en el certificado.

Para solucionarlo deberíamos ponernos en contacto con la entidad emisora del certificado y, en el caso de que haya sido revocado se deberá solicitar y obtener uno nuevo.

Una vez obtenido habrá que actualizar el archivo de configuración de nuestro proxy para poder utilizar el nuevo certificado y clave correspondiente.

Por ejemplo:

A screenshot of a terminal window displaying an Nginx configuration file. The configuration is for an SSL server listening on port 443. It specifies the paths for the new certificate and key files, sets the protocols to TLSv1.3, and lists a comprehensive set of ciphers. The server name is set to 'www.mi-dominio.es' and the access log is located at '/var/log/nginx/https\_access.log'. A proxy\_pass directive is also present for the location '/'.

```
server {  
    listen 443 ssl;  
    ssl_certificate /etc/nginx/ssl/nuevoCertificado.crt;  
    ssl_certificate_key /etc/nginx/ssl/nuevaClave.key;  
    ssl_protocols TLSv1.3;  
    ssl_ciphers  
ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES128:ECDH+3DES:DH+3DES:RSA+AESGCM:RSA+AES:R  
SA+3DES:!aNULL:!MD5:!DSS  
    server_name www.mi-dominio.es;  
    access_log /var/log/nginx/https_access.log;  
  
    location / {  
        proxy_pass http://backend_hosts;  
    }  
}
```