

Nginx con Certificados

Paso 1

Creamos un contenedor con ubuntu

```
docker run -it ubuntu:latest
```

Paso 2

Nos conecta a la terminal del contenedor y ejecutamos

1. Actualizamos el contenedor de ubuntu

```
apt-get update
```

2. Instalamos los paquetes que vamos a necesitar

```
apt-get install -y certbot nginx nano
```

Paso 3

Pedimos un certificado a "Let's Encrypt" .



Previamente tenemos que hacer que el nombre del dominio (DNS) apunte a la ip del servidor

```
certbot certonly --standalone -d midominio.es
```

También podemos automatizar las peticiones de certificados sin que nos pida nada.

```
certbot certonly \
  -d midominio.es \
  --noninteractive \
  --standalone \
  --agree-tos \
  --register-unsafely-without-email
```

<note>Si usamos la opción **--standalone** previamente hay que asegurarse de que el servicio de nginx está parado ya que este método levanta un servidor web temporal y si ya tenemos nginx

ejecutandose no va a funcionar. Para para el servicio de nginx

```
service nginx stop
```

Paso 4

Si tenemos el servidor nginx arrancado el propio certbot nos lo puede configurar . Si prefieres hacerlo manualmente debes de editar el fichero **etc/nginx/conf.d/default.conf** y añadir/crear lo siguiente

```
# Redirecciona el puerto 80 a https
server {
    listen 80 default_server;
    client_max_body_size 500M;
    return 301 https://midominio.es$request_uri;
}

# Le indicamos donde están los certificados
server {
    ssl_stapling off;
    ssl_stapling_verify off;
    listen 443 ssl;
    client_max_body_size 1500M;
    server_name midominio.es;
    ssl_certificate
/etc/letsencrypt/live/registry.midominio.es/fullchain.pem;
    ssl_certificate_key
/etc/letsencrypt/live/registry.midominio.es/privkey.pem;
    ssl_protocols      TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers         HIGH:!aNULL:!MD5;
```

Aparte de lo anterior sería recomendable añadir las siguientes opciones de seguridad al fichero de configuración de nginx

```
# Deshabilitar las cabeceras inválidas y la versión de nginx
ignore_invalid_headers    on;
server_tokens              off;

#Evitar ataques de buffer overflow
client_body_buffer_size   100K;
client_header_buffer_size 1k;
client_max_body_size      100k;
large_client_header_buffers 2 1k;

#Controlar Time-outs
client_body_timeout        15;
client_header_timeout      15;
send_timeout               15;
keepalive_timeout          5 5;
```

```
#OCSP Stapling
ssl_stapling          on;
ssl_stapling_verify   on;
resolver              1.1.1.1 208.67.222.222 208.67.220.220 valid=60s;
resolver_timeout      2s;
```

Renovación del certificado

Manualmente

Para renovar el certificado manualmente

```
certbot renew
```

Automáticamente

Con un script en cron que ejecute el comando diariamente. Por ejemplo este que se ejecuta todos los días a las 6 de la mañana y a las 9 de la noche

```
0 6,21 * * * /usr/bin/certbot renew --quiet --post-hook "/bin/systemctl
restart nginx.service"
```

Si las renovaciones no funcionan debemos hacer lo siguiente: Editar el archivo `/etc/nginx/sites-available/example_com`, añadiendo:

```
server {
    listen 80;
    listen [::]:80;
    root /var/www/example_com/wordpress;
    index index.php index.html index.htm;
    server_name example.com www.example.com;
    #permitir la respuesta HTTP a todo lo que se encuentre en el directorio
    ".well-known/acme-challenge/", usado por Let's Encrypt para crear y renovar
    los #certificados, ya que para verificar que en nuestro servidor está
    configurado ese dominio y las DNS apuntan a ese servidor, lo que hace Let's
    Encrypt es #generar un archivo en el directorio ".well-known/acme-
    challenge/" (referenciado respecto a la raíz del directorio donde está
    instalado el sitio web, #indicado en el archivo de configuración de nginx
    por la variable "root") y tratar de acceder externamente navegando a ese
    archivo.
    location ~ /\.well-known/acme-challenge/ {
        allow all;
    }
    #Todas las peticiones que se soliciten en esa dirección no serán
    redireccionadas a HTTPS.
```

```
location / {  
    return 301 https://$host$request_uri;  
}  
}
```

location ~ /\.well-known/acme-challenge/ {

```
allow all;
```

```
}
```

El segundo elemento redirecciona las peticiones HTTP a HTTPS mediante una redirección 301 (elemento movido permanentemente).

location / {

```
return 301 https://$host$request_uri;
```

```
}
```

Referencias

- <https://www.albertcoronado.com/2020/01/22/configurar-un-contenedor-con-nginx-como-proxy-ssl-con-certificado-lets-encrypt/>
- <https://www.albertcoronado.com/2020/05/05/contratar-un-certificado-ssl-gratis-con-lets-encrypt-y-configurar-nginx/>
- <https://blog.ichasco.com/nginx-optimizar-y-securizar-un-despliegue-de-wordpress/>

From:

<http://wiki.intrusos.info/> - LCWIKI

Permanent link:

<http://wiki.intrusos.info/doku.php?id=aplicaciones:nginx:certificados&rev=1627467454>

Last update: **2023/01/18 14:13**

