

Tutorial proxy inverso con Nginx

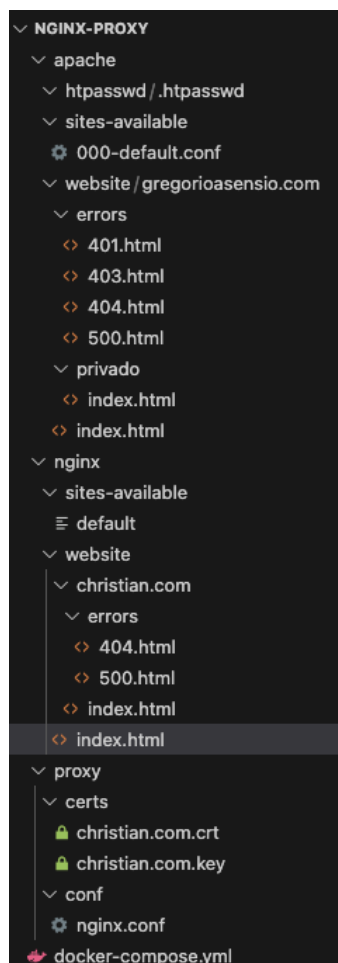
Un proxy inverso es un servidor que actúa como intermediario entre los clientes y varios servidores de origen. A diferencia de un proxy normal, que conecta a los clientes con un único servidor, el proxy inverso distribuye las solicitudes entre múltiples servidores.

Usos de un proxy inverso:

- Equilibrio de carga: Distribuye el tráfico entre varios servidores, evitando sobrecargas.
- Alta disponibilidad: Asegura que el servicio esté siempre disponible, redirigiendo solicitudes a servidores activos.
- Protección: Defiende los servidores de origen contra ataques, como los de denegación de servicio.
- Cache: Almacena contenido estático para mejorar el rendimiento.
- Privacidad: Oculta las direcciones IP tanto del servidor de origen como del cliente.

Un proxy inverso es clave para manejar eficientemente el tráfico y mejorar la seguridad y rendimiento de los servicios web.

La estructura resultante tras el tutorial será la siguiente:



Dentro del directorio donde vayamos a crear la estructura para nuestro proxy inverso, lo iniciaremos creando la carpeta “proxy” que contendrá a su vez la carpeta “conf” donde colocar nuestro archivo de configuración llamado nginx.conf, este fichero indicará los servidores de origen a los que se va a redirigir el tráfico.

Quedará configurado de la siguiente manera:

nginx.conf

```
1  events {}
2
3  http {
4      server {
5          listen 80;
6          server_name christian.com;
7
8          # Redirigir todo el tráfico HTTP a HTTPS
9          return 301 https://$host$request_uri;
10     }
11
12     server {
13         listen 443 ssl;
14         server_name christian.com;
15
16         # Configuración de SSL
17         ssl_certificate /etc/nginx/certs/christian.crt;
18         ssl_certificate_key /etc/nginx/certs/christian.key;
19         # Configuraciones SSL adicionales
20         ssl_protocols TLSv1.2 TLSv1.3;
21         ssl_ciphers 'EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH';
22         ssl_prefer_server_ciphers on;
23
24         # Configuración de proxy para la raíz
25         location / {
26             proxy_pass http://nginx_server;
27             proxy_set_header Host $host;
28             proxy_set_header X-Real-IP $remote_addr;
29             proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
30             proxy_set_header X-Forwarded-Proto $scheme;
31         }
32
33         # Configuración de proxy para /christian
34         location /one {
35             proxy_pass http://nginx_server;
36             proxy_set_header Host $host;
37             proxy_set_header X-Real-IP $remote_addr;
38             proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
39             proxy_set_header X-Forwarded-Proto $scheme;
40         }
41
42         # Configuración de proxy para /gregorioasensio
43         location /two {
44             proxy_pass http://apache_server;
45             proxy_set_header Host $host;
46             proxy_set_header X-Real-IP $remote_addr;
47             proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
48             proxy_set_header X-Forwarded-Proto $scheme;
49         }
50     }
51 }
52 }
```

Dentro de la carpeta proxy dejamos ya creada la carpeta certs para más tarde añadir certificado y keys.

En el directorio principal creamos la carpeta nginx que contendrá la configuración del propio proxy y de las webs con nginx, incluyendo las carpetas sites-available y website:

Webs de /nombre con /christian desde Nginx y /apellidos desde Apache:

Para Nginx:

En website añadimos los archivos de nuestra web christian.com que se devolverá en /christian en su directorio y a su vez un index.html por defecto.

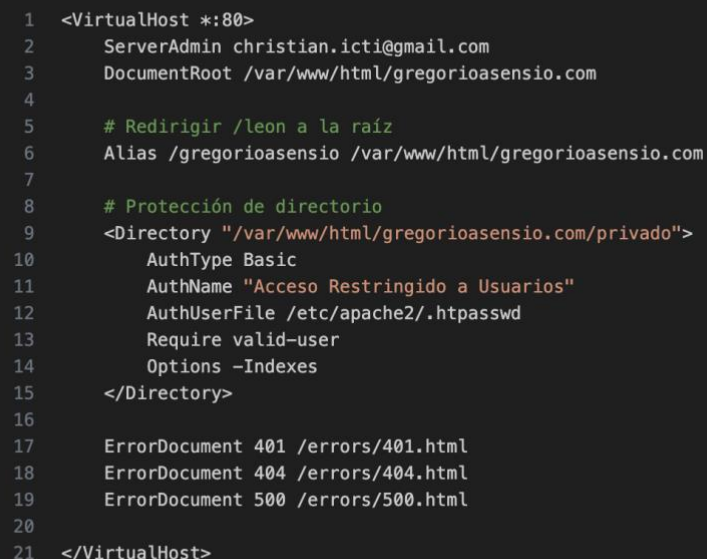
Para Apache:

Creamos el directorio Apache que contendrá las carpetas htpasswd, sites-available y website.

En sites-available añadimos 000-default.conf y website incluirá nuestra web con sus html para /apellidos y su propia web segura con /privado.

Debe contener el archivo .htaccess donde se guardarán nuestros datos de acceso.

000-default.conf



```
1 <VirtualHost *:80>
2     ServerAdmin christian.icti@gmail.com
3     DocumentRoot /var/www/html/gregorioasensio.com
4
5     # Redirigir /leon a la raíz
6     Alias /gregorioasensio /var/www/html/gregorioasensio.com
7
8     # Protección de directorio
9     <Directory "/var/www/html/gregorioasensio.com/privado">
10         AuthType Basic
11         AuthName "Acceso Restringido a Usuarios"
12         AuthUserFile /etc/apache2/.htpasswd
13         Require valid-user
14         Options -Indexes
15     </Directory>
16
17     ErrorDocument 401 /errors/401.html
18     ErrorDocument 404 /errors/404.html
19     ErrorDocument 500 /errors/500.html
20
21 </VirtualHost>
```

Añadimos el archivo default a sites-available con la configuración de proxy y web /christian con sus errores:

```
1 server {
2     listen 80;
3     listen [::]:80;
4
5     root /var/www/html; # Ruta de la carpeta raíz del dominio
6     index index.html; # Archivo por defecto
7
8     # Configuración para la raíz
9     location / {
10         try_files $uri $uri/ /index.html; # Intenta servir el archivo solicitado, si no existe, sirve
11     }
12
13     # Configuración para /christian
14     location /christian {
15         alias /var/www/html/christian.com;
16     }
17
18     # Personalizar la página de error 404
19     error_page 404 /404.html;
20     location = /404.html {
21         root /var/www/html/christian.com/errors; # Ruta donde se encuentra el archivo de error
22         internal; # Asegura que la página de error no sea accesible directamente
23     }
24
25     # la página de error 500
26     error_page 500 502 503 504 /500.html;
27     location = /500.html {
28         root /var/www/html/christian.com/errors; # Ruta donde se encuentra el archivo de error
29         internal; # Asegura que la página de error no sea accesible directamente
30     }
31 }
```

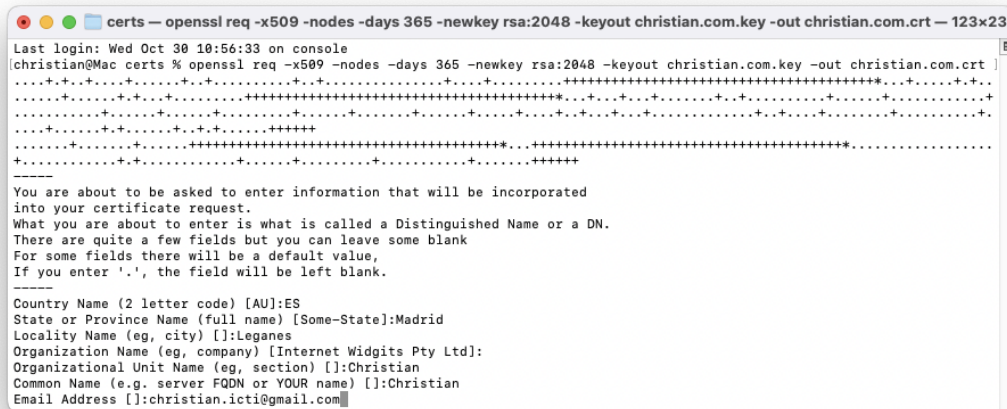
Creación del Docker-compose.yml

```
1 services:
2
3     nginx:
4         image: ubuntu/nginx # imagen de Nginx
5         container_name: nginx_server # nombre del contenedor
6         # ports:
7         # - "8081:80" # mapeo de puertos HTTP
8         volumes:
9             - ./nginx/sites-available:/etc/nginx/sites-available # archivos de configuración de hosts virtuales
10             - ./nginx/website:/var/www/html/ # directorio de los sitios web
11         restart: always # reinicio automático
12         networks:
13             - webnet # red de contenedores
14
15     apache:
16         image: ubuntu/apache2 # imagen de Apache
17         container_name: apache_server # nombre del contenedor
18         # ports:
19         # - "8082:80" # mapeo de puertos HTTP
20         volumes:
21             - ./apache/sites-available:/etc/apache2/sites-available # archivos de configuración de hosts virtuales
22             - ./apache/website:/var/www/html/ # directorio de los sitios web
23             - ./apache/httpd.conf:/etc/apache2/httpd.conf # archivo de configuración principal
24             - ./apache/htpasswd:/etc/apache2/htpasswd # archivo de contraseñas (en este caso lo uso)
25         restart: always # reinicio automático
26         networks:
27             - webnet # red de contenedores
28
29     proxy:
30         image: ubuntu/nginx # imagen de Nginx
31         container_name: proxy_server # nombre del contenedor
32         # ports:
33         # - "80:80" # mapeo de puertos HTTP
34         # - "443:443" # mapeo de puertos HTTPS
35         volumes:
36             - ./proxy/conf/nginx.conf:/etc/nginx/nginx.conf # archivo de configuración principal
37             - ./proxy/certs:/etc/nginx/certs # directorio de certificados (hechos con openssl)
38         restart: always # reinicio automático
39         depends_on:
40             - apache # dependencia de Apache
41             - nginx # dependencia de Nginx
42         networks:
43             - webnet # red de contenedores
44
45 networks:
46     webnet:
```

Certificados SSL

Los generamos desde la terminal y con el siguiente comando, añadiendo el nombre de nuestro servidor:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout christian.com.key -out christian.com.crt
```



Se habrán generado dentro de la carpeta certs si anteriormente nos hubicamos en ese directorio al generarlos.

Añadimos **Hosts** (en nuestro equipo):

Mediante la terminal añadiremos al archivo /etc/hosts las entradas:

```
127.0.0.1 christian.com
127.0.0.1 gregorioasensio.com
127.0.0.1 www.gregorioasensio.com
127.0.0.1 seguro.net
127.0.0.1 www.seguro.net
127.0.0.1 www.christian.com
```

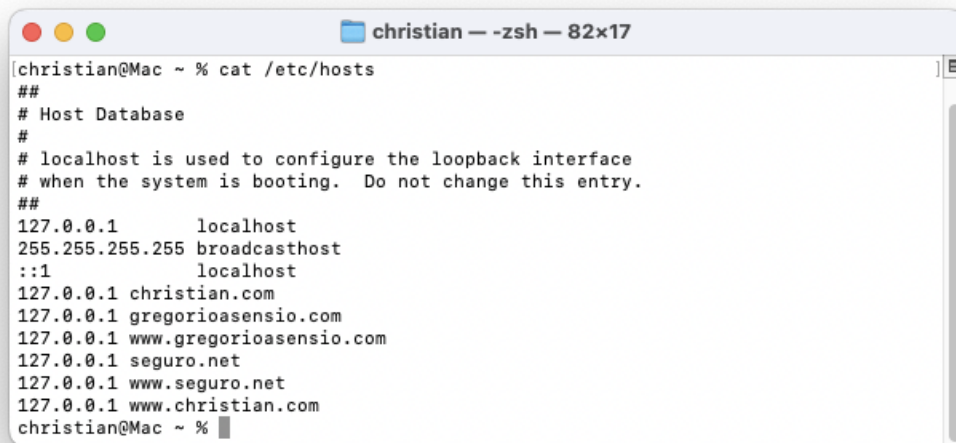
Accedemos con el commando:

```
sudo nano /etc/hosts
```

Y guardamos los cambios presionando Ctrl + O y Enter para confirmar.

Comprobamos los cambios con:

```
cat /etc/hosts
```

A terminal window titled "christian — zsh — 82x17" showing the output of the command "cat /etc/hosts". The output lists various IP addresses and their corresponding hostnames, including localhost, broadcasthost, and several domain names like christian.com and gregorioasensio.com.

```
christian@Mac ~ % cat /etc/hosts
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting. Do not change this entry.
##
127.0.0.1        localhost
255.255.255.255 broadcasthost
::1             localhost
127.0.0.1 christian.com
127.0.0.1 gregorioasensio.com
127.0.0.1 www.gregorioasensio.com
127.0.0.1 seguro.net
127.0.0.1 www.seguro.net
127.0.0.1 www.christian.com
christian@Mac ~ %
```

Ejecución del contenedor en Docker:

Previamente podemos detener contenedores en ejecución y limpiarlos ejecutando:

```
docker stop $(docker ps -q)
```

Este comando usa `docker ps -q` para listar los IDs de todos los contenedores en ejecución, y luego pasa esos IDs al comando `docker stop`.

Ejecutar `prune`: Para eliminar contenedores detenidos, imágenes sin referencias, y redes no usadas, ejecuta:

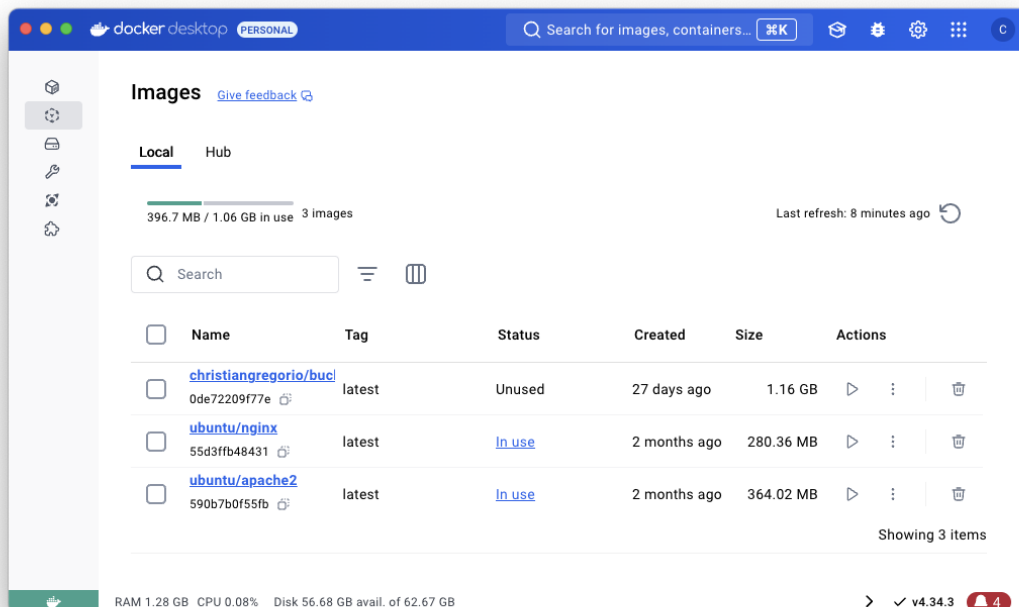
```
docker system prune -f
```

El flag `-f` (o `--force`) evita que Docker te pida confirmación antes de limpiar.

Con la terminal en el directorio donde se encuentra nuestro `Docker-compose.yml` ejecutamos el siguiente comando:

```
docker-compose up -d
```

```
christian@Mac nginx-proxy % docker-compose up -d
[+] Running 10/10
  ✓ nginx Pulled                                6.0s
  ✓ apache Pulled                               7.9s
  ✓ 5efb307521c7 Download complete              0.4s
  ✓ 5cce572b52b6 Download complete              1.5s
  ✓ b8142b74ea46 Download complete              2.3s
  ✓ proxy Pulled                                6.0s
  ✓ f75aa6f8f7e9 Download complete              2.1s
  ✓ 7dd0ab5a6d9d Download complete              3.2s
  ✓ 5a778534c419 Download complete              2.9s
  ✓ 42d6716dea8e Download complete              3.2s
[+] Running 4/4
  ✓ Network nginx-proxy_webnet Created          0.0s
  ✓ Container nginx_server Started              0.5s
  ✓ Container apache_server Started             0.5s
  ✓ Container proxy_server Started              0.4s
christian@Mac nginx-proxy %
```



Para entrar en la terminal del servidor ejecutado con Docker podemos usar:
docker exec -it nginx_server /bin/bash

Para detener el contenedor ejecutamos:
docker-compose down

Para crear el usuario y contraseña, introducimos en la terminal:

Teniendo ya creado el directorio htpasswd.

Dentro de la terminal del contenedor, debemos ejecutar el comando:

```
docker exec -it apache_server bash
```

```
htpasswd -c /etc/apache2/.htpasswd usuario
```

Siguiendo los pasos que nos indica, configuramos usuario y contraseña, al introducirla nos mostrará el html que hemos creado dentro del directorio Privado y que antes no era visible.

Pruebas en el navegador:

