Proxy Inverso y HTTPS

Indice

1. Definición proxy Inverso

Antes de nada vamos a ver que es un proxy inverso y en que se diferencia de un proxy tradicional:

Proxy (Directo):
Representa al **cliente** ante el servidor. Es el intermediario que realiza solicitudes en nombre del cliente y, a menudo, se usa para mejorar la privacidad, el control y la caché de las solicitudes salientes.

Proxy Inverso:
Representa al **servidor o servidores** ante el cliente. Es el intermediario que recibe las solicitudes entrantes y las distribuye a los servidores internos, ayudando en la seguridad, el balanceo de carga y la optimización de la entrega de contenido.

En resumen, un **proxy** protege o modifica la comunicación saliente del cliente, mientras que un **proxy inverso** protege y gestiona la comunicación entrante hacia los servidores internos.

2. Proxy Inverso: Configuración

Como ya hemos visto el proxy inverso nos va a permitir "ocultar" los distintos servicios y sus puertos en un servidor tras url definidas por nosostros. Por tanto la configuración generica de un proxy inverso sería la siguiente:

2.1 Default.conf

```
server {
   #puerto de escucha del servidor (el que está abierto y atiende peticiones)
   listen 80;

# ruta definida para cada servicio
   location / {

      # Redirige las solicitudes al servicio en su puerto
      proxy_pass http://servicio:puerto;

      #persistencia de la información original del cliente

      #pasa el nombre del host o dominio del cliente
      proxy_set_header Host $host;
      #pasa la ip original del cliente
      proxy_set_header X-Real-IP $remote_addr;
      #pasa las ip del cliente y las de los proxies intermedios si existen
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      #envia información sobre si la solicitud es http o https
      proxy_set_header X-Forwarded-Proto $scheme;

      #soporte para websockets y SSE para mantener la conexión abierta
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection "Upgrade";
```

```
        #soporte para tiempos de conexión y respuesta lentos
        proxy_connect_timeout 60s; #tiempo máximo de conexión
        proxy_send_timeout 60s; #tiempo máximo para enviar la solicitud
        proxy_read_timeout 60s; #tiempo máximo para recibir la respuesta
    }
#siguiente servicio
    location /ruta/ {
        proxy_pass http://servicio:puerto; # Redirige las solicitudes al servicio en su puerto

        #persistencia de la información original del cliente

        #pasa el nombre del host o dominio del cliente
        proxy_set_header Host $host;
        #pasa la ip original del cliente
        proxy_set_header X-Real-IP $remote_addr;
        #pasa las ip del cliente y las de los proxies intermedios si existen
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        #envia información sobre si la solicitud es http o https
        proxy_set_header X-Forwarded-Proto $scheme;

        #soporte para websockets y SSE para mantener la conexión abierta
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";

        #soporte para tiempos de conexión y respuesta lentos
        proxy_connect_timeout 60s; #tiempo máximo de conexión
        proxy_send_timeout 60s; #tiempo máximo para enviar la solicitud
        proxy_read_timeout 60s; #tiempo máximo para recibir la respuesta
    }
}
```

En resumen, definiendo cada servicio de nuestro servidor en este archivo no necesitamos publicar ningún puerto adicional y nuestro proxy inverso se encarga de redirigir el tráfico a las distintas url asociadas a cada servicio, ganando así seguridad (solo un puerto expuesto) y evitando errores de rutas generadas por los navegadores (CORS)

3. Certificados y tráfico HTTPS en nginx

Para completar la configuración de nuestro nginx como proxy inverso vamos a habilitar y configurar el tráfico https por el puerto 443 y redirigir todo nuestro tráfico http al https. Además crearemos nuestros certificados autofirmados de forma automática.

3.1 Creación automática del certificado

Para crear el certificado autofirmado de forma automática vamos a crear el siguiente sccript que vamos llamar entrypoint.sh:

```
#!/bin/sh
set -e

# Verifica si el certificado ya existe
if [ ! -f /etc/nginx/certs/selfsigned.crt ]; then
    echo "Generando certificado autofirmado..."
    openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
                                                                                -subj
"/C=US/ST=Estado/L=Ciudad/O=Organizacion/OU=Departamento/CN=localhost_IP_Dominio/
emailAddress=admin@miempresa.com" \
        -keyout /etc/nginx/certs/selfsigned.key \
        -out /etc/nginx/certs/selfsigned.crt
fi

# Ejecuta Nginx en primer plano
exec nginx -g "daemon off;"
```

Para personalizar nuestro certificado con nuestros datos tendremos que personalizar los datos del parámetro -subj

3.2 Dockerfile

Ahora vamos a personalizar nuestro dockerfile para que incluya el script e instale openssl para poder crear los certificados, con lo que nuestro dockerfile quedaría así:

```
FROM nginx:latest

# Instala openssl
RUN apt-get update && apt-get install -y openssl && rm -rf /var/lib/apt/lists/*

# Copia la configuración de Nginx
COPY default.conf /etc/nginx/conf.d/default.conf

# Copia el script de entrypoint y asigna permisos de ejecución
COPY entrypoint.sh /entrypoint.sh
RUN chmod +x /entrypoint.sh

# Define el script de entrada como comando predeterminado
CMD ["/entrypoint.sh"]
```

3.3. Actualizar el default.conf

Vamos a actualizar este archivo para que redirija todo el tráfico del puerto 80 (http) al 443 (https). Por esto ahora tendremos un bloque server con este contenido:

```
server {
   listen 80;
   server_name localhost; #tu_dominio_o_IP publica

   # Redirige todas las peticiones HTTP a HTTPS
   return 301 https://$host$request_uri;
}
```

Las reglas del proxy inverso ahora estarán en el bloque server del puerto 443 de la siguiente forma:

```
server {
   listen 443 ssl;

   server_name localhost;#tu_dominio_o_IP publica

   # Configuración SSL con las rutas a las claves y los certificados
   ssl_certificate /etc/nginx/certs/selfsigned.crt;
   ssl_certificate_key /etc/nginx/certs/selfsigned.key;

   # Opciones adicionales de seguridad
   ssl_protocols TLSv1.2 TLSv1.3;
   ssl_prefer_server_ciphers on;
   ssl_ciphers HIGH:!aNULL:!MD5;

#primer servicio
   location /ruta/ {
      proxy_pass http://servicio:puerto; # Redirige las solicitudes al servicio en su puerto
      #persistencia de la información original del cliente
      #pasa el nombre del host o dominio del cliente
      proxy_set_header Host $host;
      #pasa la ip original del cliente
      proxy_set_header X-Real-IP $remote_addr;
      #pasa las ip del cliente y las de los proxies intermedios si existen
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      #envia información sobre si la solicitud es http o https
      proxy_set_header X-Forwarded-Proto $scheme;

      #soporte para websockets y SSE para mantener la conexión abierta
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection "Upgrade";

      #soporte para tiempos de conexión y respuesta lentos
      proxy_connect_timeout 60s; #tiempo máximo de conexión
      proxy_send_timeout 60s; #tiempo máximo para enviar la solicitud
      proxy_read_timeout 60s; #tiempo máximo para recibir la respuesta
   }
#siguiente servicio
…..
}
```

3.4. Docker-Compose

Ahora tenemos que actualizar el apartado de nuestro proxy inverso en el docker compose, para publicar el puerto 443 y crear un nuevo bind mount para persistir los certificados:

```
 web:
   build:
     context: ./web
     dockerfile: DockerfileWeb
   ports:
     - "80:80"
     - "443:443"
   volumes:
     - ./web/certs:/etc/nginx/certs
   depends_on:
     - frontend
     - backend
   networks:
     - app-network
```

3.5. .gitIgnore para proteger los certificados

Evidentemente no queremos que nuestros certificados se copien a un repositorio publico de git, (también para esto hemos puesto que se generen automáticamente). Para esto debemos crear un fichero .gitignore dentro de la carpeta certs con el siguiente contenido:

```
*
!.gitignore
```