El Reverse Proxy

Qué es un reverse proxy?

Sinopsis

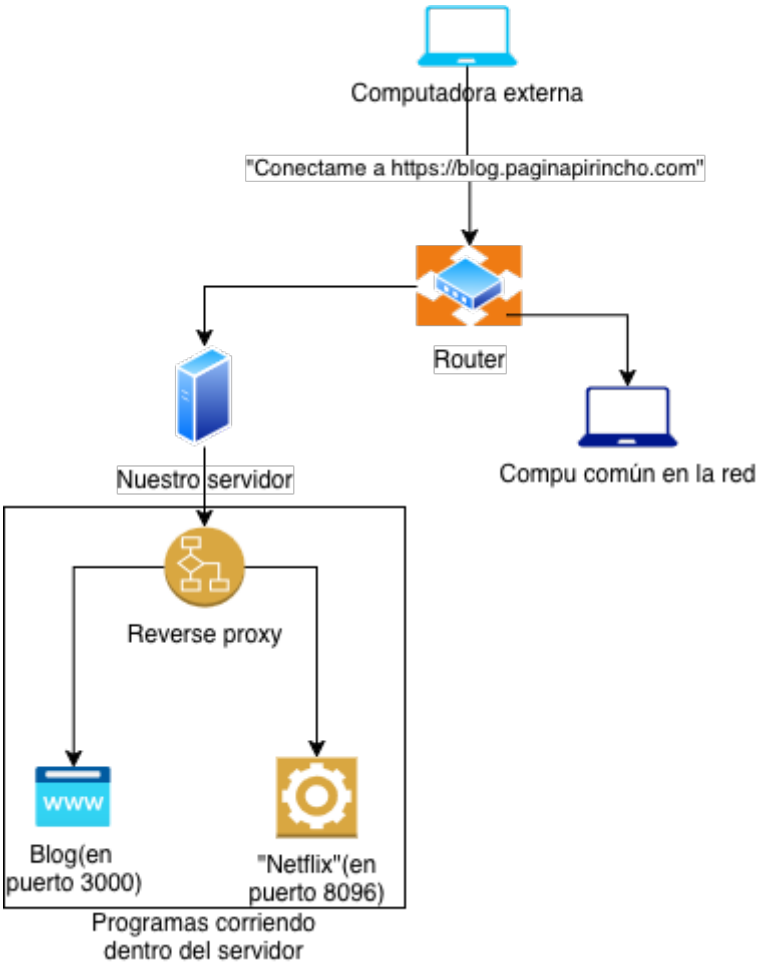
En el artículo sobre el router hablamos sobre cómo, cuando una computadora de afuera quiere conectarse con nuestro servidor, el router se encarga de "dirigir" la compu de afuera a nuestro server en la red local. Si nuestro server sólo proporciona un servicio, esto es ideal: Alguien se conecta, por ejemplo, a `https://paginapirincho.com` y pirincho, que en su server tiene un netflix privado, "sirve" ese contenido.

Pero qué pasa si, por ejemplo, la misma computadora sirve un netflix privado y, además un blog de cocina? `https://peliculas.paginapirincho.com` debería llevarnos al netflix, y `https://blog.paginapirincho.com` al blog. Pero si están en la misma computadora, el router "dirige" pedidos de ambas páginas hacia la misma computadora. Cómo sabe la computadora qué servicio servir? Para esto existen los **reverse proxies**.

Paso a paso

Un reverse proxy es un programa que ponemos al frente de nuestra computadora, que recibe todos los pedidos que vayan a esa compu, y se encarga de "dirigirlos", igual que el router, pero a nivel interno de la computadora. Además de esto, el reverse proxy cumple varias funciones de seguridad para nuestro server y para la gente que lo usa, y también nos resuelve algunas complejidades vinculadas con el problema (que quizás ya notaron, o quizás no) de no poder acceder cómodamente a páginas que no tengan "https" en lugar de "http" en la dirección (para más data, googlea "TLS" o "SSL", pero te advierto que es un tema complejo).

El esquema terminaría siendo algo así:



Lo que pasaría, entonces:

- Computadora externa se conecta a nuestra IP pública pidiéndo la página `https://blog.paginapirincho.com`.
- Nuestro router recibe la conexión externa (en el puerto predeterminado que usamos para conexiones de http o https, las conexiones que usan nuestros navegadores de internet, que son el 80 y el 443) y la redirige a la dirección interna de nuestro server, pasándo el pedido de la página `https://blog.paginapirincho.com`.
- Dentro de nuestro servidor, nuestro reverse proxy recibe ese pedido en el puerto predeterminado (80 o 443, ambos se usan, y nuestro reverse proxy "presta atención" o "escucha" a ambos puertos), busca en su configuración las instrucciones de qué hacer con esa página, encuentra que hay que redirigir hacia el puerto 3000 y lo hace. Le pide al programa del blog que le dé el blog, el programa se lo da al reverse proxy, que se lo da al router, que finalmente le llega a la computadora, externa.

Guía: Instalar y configurar mi reverse proxy

Qué es Caddy?

Caddy es una aplicación de servidor de código abierto. Un programa "servidor" es esencialmente eso: Un programa que, cuando le hablan, sirve contenido. En este caso, lo vamos a utilizar para que sirva contenido actuando como "reverse proxy": No le vamos a dar el contenido directamente, sino que le vamos a pedir que "redirija" las conexiones a las aplicaciones que corramos en nuestro server y obtenga el contenido desde ahí.

> Para utilizar caddy, es muy recomendable que hayas ya configurado un dominio para tus servicios

Instalación

> Como en todos los tutoriales de esta guía, la instalación va a ser en sistemas Linux; específicamente, en sistemas de tipo "debian", como Debian, Ubuntu, o Linux Mint

Instalar Caddy, por suerte, es muy fácil (por lo menos, lo es en Linux ;)). Simplemente corré los siguientes comandos en una consola o terminal:

```
sudo apt update
sudo apt install caddy
```

> Recordá que, cuando uses comandos con [sudo](https://wiki.cuquiweb.xyz/link/14#bkmrk-sudo), el sistema operativo va a pedirte la contraseña, ya que estás instalando programas como administradorx.

En caso de que no funcione, las instrucciones oficiales indican estos comandos:

```
sudo apt install -y debian-keyring debian-archive-keyring apt-transport-https curl
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/gpg.key' | sudo gpg --dearmor -o
/usr/share/keyrings/caddy-stable-archive-keyring.gpg
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/debian.deb.txt' | sudo tee
/etc/apt/sources.list.d/caddy-stable.list
chmod o+r /usr/share/keyrings/caddy-stable-archive-keyring.gpg
chmod o+r /etc/apt/sources.list.d/caddy-stable.list
sudo apt update
sudo apt install caddy
```

Una vez que lo instalaste, fijate si está corriendo ya con el comando `systemctl status caddy`. Esto debería abrirte un cuadro similar a esto:

```
● caddy.service - Caddy
     Loaded: loaded (/lib/systemd/system/caddy.service; enabled; vendor pr>
     Active: active (running) since Fri 2025-10-24 13:55:58 -03; 1 week 1 >
       Docs: https://caddyserver.com/docs/
    Process: 2189865 ExecReload=/usr/bin/caddy reload --config /etc/caddy/>
   Main PID: 673034 (caddy)
      Tasks: 23 (limit: 9145)
     Memory: 55.6M
        CPU: 2h 30min 25.271s
     CGroup: /system.slice/caddy.service
             └─673034 /usr/bin/caddy run --environ --config /etc/caddy/Cad>
```

> Para volver a la consola, probá apretar `q`

Si te indica que no está activo, lo podés poner a andar escribiendo `sudo systemctl start caddy` (notá que este comando *lleva sudo*).

Configuración

Dejo dos maneras de configurar caddy. La primera es la más simple. Si sólo vas a tener un servicio o dos corriendo en tu servidor, esta te puede alcanzar. Sin embargo, a medida que se agreguen servicios, puede llegar a complicar un poco la lectura y modificación de la configuración. Para esto, vamos a usar una segunda propuesta, un poco más avanzada y que requiere un par de pasos más, pero más estructurada y fácil de leer y modificar.

Puntos en común

El archivo `Caddyfile`

Si bien vamos a ver dos formas de armar nuestra configuración de caddy, ambas comparten un par de principios. El primero es que caddy guarda su configuración en el directorio `/etc/caddy/`, en un archivo llamado `Caddyfile`. En este archivo vamos a poder encontrar todas las "directivas" para el programa: Es decir, la "tablita" que el programa lee para ver qué hacer con cada dirección que reciba. Si no figura acá, el programa no va a hacer nada, y no va a "servir" ningún contenido.

Para abrir el archivo, primero debemos movernos al directorio de la configuración con el comando `cd /etc/caddy` (nótese la barra diagonal al comienzo de la dirección, esto es importante). De esta forma, todas las cosas que hagamos se van a llevar a cabo en este directorio.

Una vez en el directorio apropiado, vamos a abrir el Caddyfile con el comando `sudo nano Caddyfile`. Este comando nos va a abrir un editor de texto on el que vamos a poder editar el archivo. Ya que es un archivo del sistema, y no del usuario en específico, necesitamos utilizar `sudo` al comienzo del comando.

Cuando terminemos de editar el archivo, vamos a poder guardarlo con `Ctrl+o` (apretamos enter para confirmar que el nombre del archivo sigue siendo el mismo), y lo cerramos con `Ctrl+x`.

Aplicar cambios

Cada vez que actualicemos nuestro Caddyfile, nuestro archivo de configuración, para implementar sus cambios vamos a tener que escribir el comando `sudo systemctl reload caddy`. Si no nos dice nada, significa que se reinició bien. Si no, nos va a tirar un error. Es muy importante prestar atención a tener los archivos en el caddyfile "bien escritos"; si nos comemos algún símbolo, es muy posible que nos tire un error.

Ejemplo de Caddyfile

En un Caddyfile, unx escribe "bloques" de indicaciones para cada dominio y subdominio, por ejemplo:

```
blog.midominio.com {
  <qué "sirvo" para este dominio
}
```

Observemos el formato: Primero va el dominio, un espacio, y luego, entre llaves `{}`, qué hacer con las conexiones a ese dominio.

Configuración básica

Para nuestra configuración básica, vamos a escribir un Caddyfile que directamente va a contener los bloques de cada dirección a la que queremos que se pueda conectar la gente de afuera. En este ejemplo, voy a escribir un archivo Caddyfile que contendría dos servicios: Un blog en el puerto 3000, y un "netflix privado" en el puerto 8096:

```
blog.midominio.com {
  reverse_proxy localhost:3000
}

neflis.midominio.com {
  reverse_proxy localhost:8096
}
```

Nótense dos cosas: La dirección "localhost" es una dirección que refiere a que el servicio se encuentra en la misma computadora, en otro puerto (ya que Caddy "escucha" en los puertos estándar de conexión de internet http, `80` y `443`). (Sí, esto significa que podemos hacer que un dominio x sea "redirigido" mediante reverse proxy a otra IP o dominio *externos a nuestro server*, pero esto es mucho más avanzado y probablemente no lo necesitemos nunca).

Por otro lado, los dos puntos indican qué puerto servimos: En el caso del "netflix", que corre "atendiendo" el puerto 8096, usamos `localhost:8096`. Esto funciona en un navegador también. Si ponemos `https://undominio.com:8000`, estamos indicando que nos queremos conectar al puerto 8000 de la computadora conectada a ese dominio. Cuando no aclaramos nada, por defecto "pide" el puerto 443 (y, a veces, el puerto 80).

Una vez escrito este archivo Caddyfile, podemos cargar la nueva configuración con `sudo systemctl reload caddy`.

Configuración avanzada

A medida que vayamos agregando nuevos servicios a nuestro servidor, este archivo inicial se puede llegar a hacer muy grande y difícil de navegar y leer. Además de esto, quizás queremos correr alguna de las otras directivas de Caddy, para varias cosas más avanzadas que hagamos, como logear, o correr algún tipo de protección como crowdsec.

Para "reconfigurar" nuestra configuración de Caddy, vamos a empezar por crear los directorios `sites-available` y `sites-enabled`. En `sites-available` vamos a ingresar la configuración de cada servicio de caddy, y los vamos a "conectar" con un link simbólico (algo así como un "acceso directo", un archivo que apunta a otro archivo) en la carpeta sites-enabled cuando queramos que esos sitios estén disponibles. De esta forma, si en algún momento queremos dejar de servir, por ejemplo, nuestro netflix privado, sólo tenemos que eliminar ese archivo en `sites-enabled`, sin borrar la configuración que teníamos armada. Cuando queramos volver a servirlo, sólo tenemos que crear el link de nuevo.

Para crear los directorios, vamos a usar los siguientes comandos:

```
cd /etc/caddy/ # Para movernos hacia la carpeta de la configuración
sudo mkdir sites-available
sudo mkdir sites-enabled # Crea ambos directorios
```

> No es necesario copiar lo que viene después del `#`, incluyendo el propio símbolo: Representa comentarios que el "intérprete" de la consola va a ignorar. Si los escribís, no cambia nada.

Una vez creados, la estructura local del directorio `/etc/caddy` debería ser la siguiente: (en este ejemplo, usamos los mismos dos servicios que teníamos antes)

```
Caddyfile
sites-available/
|- blog.midominio.com <-- Bloque de config de blog
L- neflis.midominio.com <-- Bloque de config de neflis
sites-enabled/
|- blog.midominio.com <-- Link al archivo en sites-avail
```

```
L- neflis.midominio.com <-- Link
```

> Los subdirectorios están indicados con una barra diagonal `/` al final del nombre. Si no tiene esta barra diagonal, es un archivo

Una vez creadas la carpetas, tenemos que tomar cada bloque de cada servicio y escribirlo en un archivo en el directorio `sites-available`. Por ejemplo, copiamos el bloque del blog, y lo copiamos en el archivo propio con el comando `sudo nano sites-available/blog.midominio.com`. Dentro del archivo quedaría:

```
blog.midominio.com {
  reverse_proxy localhost:3000
}
```

Hacemos lo mismo con el archivo del "neflis". Ambos bloques los borramos del Caddyfile.

En el Caddyfile, le indicamos que "incluya" todas las "sub-configuraciones" que se encuentran en la carpeta sites-enabled. Nuestro archivo Caddyfile, en principio, quedaría únicamente así:

```
import sites-enabled/*
```

> El * en la indicación significa "todo" lo que está en la carpeta sites-enabled

Por último, creamos los "accesos directos" dentro de la carpeta sites-enabled que apunten a las configuraciones individuales. En este ejemplo uso la configuración del blog:

`sudo ln -s /etc/caddy/sites-available/blog.midominio.com /etc/sites-enabled/`

(Nótese que uso "paths absolutos", que empiezan con una barra diagonal `/`).

Hacemos lo mismo con cada sitio que queremos que esté "activado".

Finalmente, refrescamos la nueva configuración con `sudo systemctl reload caddy`.

> Nótese que, mientras trabajamos en la carpeta /etc/caddy, necesitamos usar sudo *cada vez que queremos escribir o modificar* algo, pero no cada vez que queremos verlo. Esto es porque tenemos, como usuarios comunes no-administradores, permisos para ver pero no para modificar. Por esto con `cd` podemos "entrar" al directorio y ver sus contenidos, pero no podemos modificarlos sin usar `sudo`.

Revision #4
Created 1 November 2025 15:51:36 by Kvothe
Updated 1 November 2025 21:45:54 by Kvothe