Caddy (Proxy Inverso)

1. Introducción

Para acceder a nuestros servicios desde fuera de casa sin recordar números complejos, necesitamos un nombre de dominio (ej. midominio.duckdns.org).

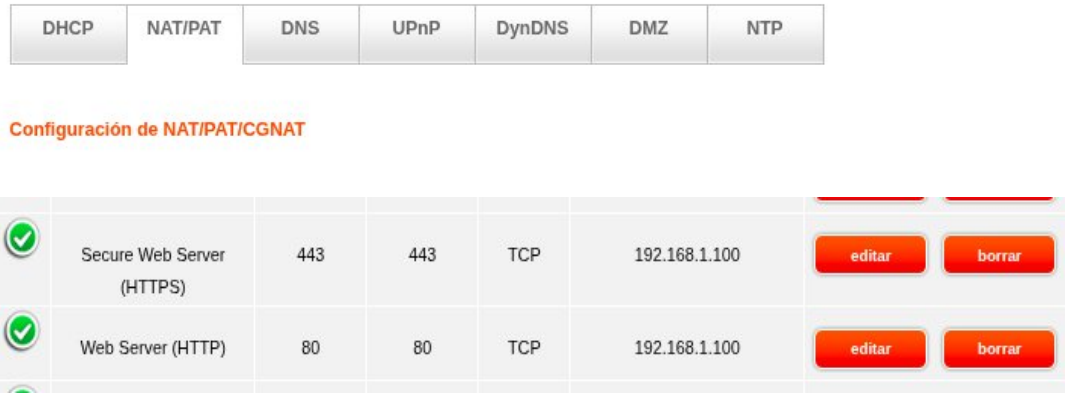
Nota sobre la IP Pública: En este despliegue utilizo DuckDNS.

- Mi caso (IP Fija): Yo dispongo de una IP pública fija, por lo que utilizo este servicio únicamente para asignar un nombre amigable a mi IP.
- Caso general (IP Dinámica): Si no tienes contratada una IP fija (la mayoría de conexiones domésticas), tu IP cambiará periódicamente. Para estos casos, desplegaremos el contenedor de DuckDNS, el cual actuará como un servicio DDNS (Dynamic DNS), encargándose de actualizar automáticamente la IP si esta cambia, asegurando que nunca pierdas la conexión.

2. Configuración de Red (Router)

Para que el tráfico llegue al servidor, se han abierto los siguientes puertos en el router apuntando a la IP local del servidor (`192.168.1.X`):

- Puerto 80 (TCP)
- Puerto 443 (TCP/UDP)



3. Estructura de Directorios

Organizamos los volúmenes para separar la configuración (`Caddyfile`) de los datos de certificados.

4. Archivos de Configuración

- Para el despliegue necesitamos dos archivos. A continuación se muestran las capturas con la configuración utilizada.

A. Archivo docker-compose.yml Este archivo define los servicios de DuckDNS y Caddy.

```yaml
services:
  caddy:
    image: caddy:latest
    container_name: caddy
    network_mode: "host"
    restart: unless-stopped
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./Caddyfile:/etc/caddy/Caddyfile
      - caddy_data:/data
      - caddy_config:/config
volumes:
  caddy_data:
  caddy_config:
```

B. Archivo Caddyfile Este archivo gestiona las redirecciones de los dominios hacia las IPs locales.

5. Despliegue y Pruebas

- Dejare los archivos a parte para que lo puedan copiar
- Ponemos el “docker compose up –d”
- Si agregamos un nuevo servicio a caddy no hace falta reiniciar el docker. Podemos poner, “docker exec -w /etc/caddy caddy caddy reload”