ies zaidín-vergeles
granada

PumukyChat

Adrián Bertos Gómez

2ASIRB

Fernando Raya Díaz

I.E.S. Zaidín-Vergeles (GRANADA)

27/05/2025

Anexo - 1

Índice

1. Introducción

Para garantizar que el dominio `pumukydev.com` y sus subdominios estén siempre accesibles desde Internet, hay una infraestructura Docker que combina un sistema de **DNS dinámico** con un **proxy inverso** gestionado por NGINX.

La infraestructura, al desplegarse con `Docker`, permite reiniciarse automáticamente ante cualquier problema como un fallo de conexión o un corte de luz. De esta manera, la IP pública del servidor está siempre actualizada y todo el tráfico entrante es redirigido al contenedor adecuado según el subdominio.

2. Despliegue de la infraestructura

Para desplegar los contenedores, se usa el siguiente archivo `docker-compose.yml`:

docker-compose.yml

```
services:
  nginx_proxy:
    build: ./nginx
    container_name: nginx_proxy
    restart: unless-stopped
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx/conf.d/default.conf:/etc/nginx/conf.d/default.conf
      - ./nginx/certs/:/etc/nginx/certs/
    networks:
      - proxy-network

  dyndns:
    build: ./dynamic-dns
    container_name: dyndns
    restart: unless-stopped

networks:
  proxy-network:
    external: true
```

Como se puede observar, el contenedor `nginx_proxy` tiene abiertos los puertos 80 (HTTP) y 443 (HTTPS), que a su vez están expuestos en el router. Esto permite que cualquier usuario acceda a los servicios desplegados desde el exterior mediante un navegador.

El contenedor `nginx_proxy` también está conectado a una red externa llamada `proxy-network`. Esta red permite que otros contenedores pertenecientes a otros archivos `docker-compose.yml` (por ejemplo, el contenedor del chat,streaming u otros proyectos) puedan ser accedidos por su nombre de servicio a través del proxy inverso.

La opción `restart: unless-stopped` asegura que los contenedores se reinicien automáticamente ante cualquier fallo o reinicio del sistema, salvo que hayan sido detenidos manualmente por mí.

2.1. nginx

Este servicio utiliza una imagen base de `nginx:alpine` y se complementa con un script de espera (`wait-for-it.sh`) para garantizar que no se arranque antes de que los servicios a los que debe redirigir estén activos.

Dockerfile

```
FROM nginx:alpine

RUN apk add --no-cache netcat-openbsd

COPY wait-for-it.sh /wait-for-it.sh
RUN chmod +x /wait-for-it.sh

ENTRYPOINT ["/wait-for-it.sh"]
CMD ["nginx", "-g", "daemon off;"]
```

Antes de arrancar NGINX, es necesario asegurarse de que los servicios a los que va a redirigir el tráfico (por ejemplo, nginx_chat o nginx_streaming) ya estén completamente operativos. Para ello, se utiliza un script llamado `wait-for-it.sh`, que utilizando `netcat` (nc) comprueba si estos servicios están escuchando correctamente en sus respectivos puertos.

El script recibe una lista de destinos en formato host:puerto y verifica su disponibilidad antes de ejecutar NGINX. Si algún servicio no responde en el tiempo establecido (60 segundos), el script finaliza con un error, evitando así que NGINX falle por iniciar antes de tiempo.

wait-for-it.sh

```
#!/bin/sh

# wait-for-it.sh: espera a que una lista de host:puerto esté disponible

set -e

PROXIED_TARGETS="${PROXIED_TARGETS:-nginx_chat:80 nginx_streaming:80}"
TIMEOUT="${TIMEOUT:-60}"
```

```
echo "[wait-for-it] Starting health checks..."

for target in $PROXIED_TARGETS; do
  host=$(echo "$target" | cut -d: -f1)
  port=$(echo "$target" | cut -d: -f2)

  echo "[wait-for-it] Checking $host:$port..."

  start_time=$(date +%s)
  while ! nc -z "$host" "$port"; do
    sleep 1
    now=$(date +%s)
    elapsed=$((now - start_time))

    if [ "$elapsed" -ge "$TIMEOUT" ]; then
      echo "[wait-for-it] ERROR: Timeout reached while waiting for $host:$port
(${TIMEOUT}s)"
      exit 1
    fi
  done

  echo "[wait-for-it] $host:$port is available."
done

echo "[wait-for-it] All targets are ready. Proceeding..."
exec "$@"
```

2.2. dyndns

Este servicio se encarga de actualizar la IP pública del dominio dinámicamente utilizando la API de IONOS. Está basado en Debian e incluye `cron`, `curl` y `jq` para gestionar peticiones y respuestas.

Dockerfile

```
FROM debian:12.8

# Install required packages
RUN apt-get update && apt-get -y install cron curl jq

# Set working directory to /opt/dynamic-dns
WORKDIR /opt/dynamic-dns

# Copy your script and necessary files into the container
COPY update_public_ip.sh /opt/dynamic-dns/
```

```
COPY dyndns-cronjob /etc/cron.d/
COPY .env /etc/environment

# Set correct permissions for the scripts and cronjob
RUN chmod +x /opt/dynamic-dns/update_public_ip.sh
RUN chmod 0644 /etc/cron.d/dyndns-cronjob

# Install the cron job
RUN crontab /etc/cron.d/dyndns-cronjob

CMD ["cron", "-f"]
```

3. Servicios desplegados

3.1. Nginx proxy inverso

Este contenedor actúa como proxy inverso. Redirige las solicitudes entrantes a los servicios correspondientes según el subdominio.

Por ejemplo:

- `chat.pumukydev.com` → redirige a `nginx_chat`
- `stream.pumukydev.com` → redirige a `nginx_streaming`

default.conf

```
server {
    listen 80;
    server_name pumukydev.com stream.pumukydev.com chat.pumukydev.com;

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

server {
    listen 443 ssl;
    server_name stream.pumukydev.com;

    ssl_certificate /etc/nginx/certs/pumukydev.com_ssl_certificate.cer;
    ssl_certificate_key /etc/nginx/certs/_.pumukydev.com_private_key.key;

    location / {
        proxy_pass http://nginx_streaming:80;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
```

```
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto https;
    }
}

server {
    listen 443 ssl;
    server_name chat.pumukydev.com;

    ssl_certificate /etc/nginx/certs/fullchain.pem;
    ssl_certificate_key /etc/nginx/certs/_.pumukydev.com_private_key.key;

    client_max_body_size 500M;

    location / {
        proxy_pass http://nginx_chat:80;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto https;

        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
    }
}
```

Este archivo de configuración permite la redirección segura con HTTPS, la compatibilidad con WebSocket (gracias a los headers `Upgrade` y `Connection`) y la carga de archivos de hasta 500MB en el chat.

3.2. DNS dinámico

Este servicio se encarga de mantener actualizados los registros DNS mediante la API de IONOS. Para ello, ejecuta cada minuto un script Bash que envía una petición POST a la API de DNS dinámico.

update_public_ip.sh

```
#!/bin/bash

source /etc/environment

# Realiza una petición POST a la API de IONOS para actualizar la IP
output=$(/usr/bin/curl -X 'POST' \
  'https://api.hosting.ionos.com/dns/v1/dyndns' \
```

```
  -H "accept: application/json" \
  -H "X-API-Key: $ID.$SecretKey" \
  -H "Content-Type: application/json" \
  -d '{
  "domains": [
    "pumukydev.com",
    "www.pumukydev.com",
    "stream.pumukydev.com",
    "chat.pumukydev.com"
  ],
  "description": "Dynamic DNS"
}')

# Extrae la URL de actualización y realiza la llamada
updateUrl=$(echo "$output" | /usr/bin/jq -r '.updateUrl')
/usr/bin/curl -f $updateUrl
```

La ejecución de este script está automatizada con el siguiente cronjob:

dyndns-cronjob

```
* * * * * /opt/dynamic-dns/update_public_ip.sh
```

De esta manera, cada minuto se comprueba y actualiza la dirección IP pública, garantizando que los subdominios sigan funcionando incluso tras un cambio de IP.

PumukyDev

PumukyDev

https://pumukydev.com

adrian.bertosgomez@gmail.com

Adrián Bertos Gómez

PumukyDev

PumukyDev