Pi-Hole

1. Introducción

Pi-hole actúa como un servidor DNS que bloquea publicidad y rastreadores antes de que lleguen a nuestros dispositivos. En esta práctica lo configuraremos para:

- Tener acceso web seguro mediante **Caddy**.
- Servir como DNS protegido para nuestra VPN **WireGuard**.

2. Configuración de Red Interna (Docker Bridge)

Para conectar los contenedores de VPN y DNS con IPs fijas, creamos una red interna.

Comando a ejecutar: "docker network create --driver bridge --subnet=172.19.0.0/16 wg_pihole"

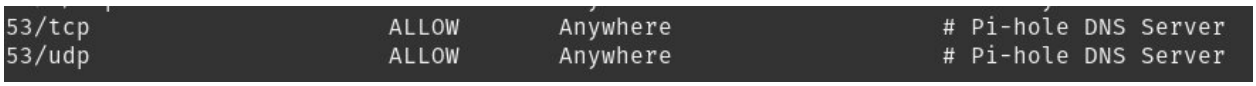
3. Estructura de Directorios

Preparamos la persistencia de datos creando las carpetas necesarias.

4. Configuración de Seguridad (UFW)

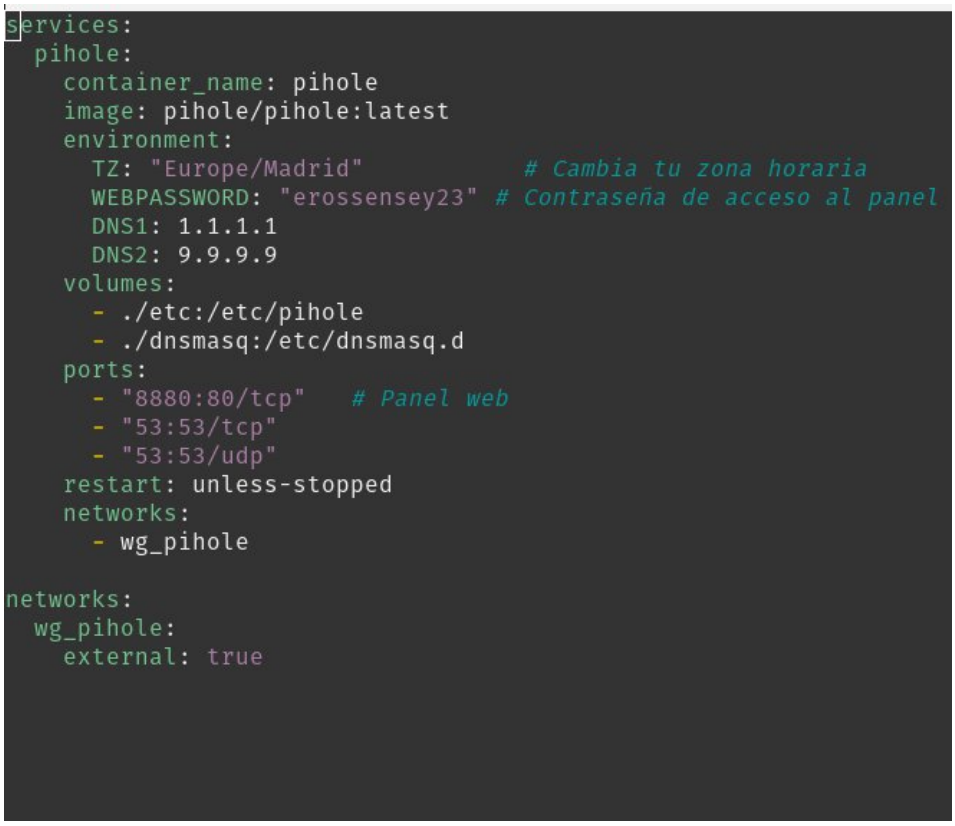
Abrimos los puertos necesarios para el servicio DNS y la gestión web local.

Comandos a ejecutar: "sudo ufw allow 53/tcp comment 'Pi-hole DNS Server'" "sudo ufw allow 53/udp comment 'Pi-hole DNS Server'" "sudo ufw allow 8081/tcp comment 'Pi-hole Web Interface (Local)'"



5. Archivo de Despliegue (docker-compose.yml)

Dejo el archivo en mi carpeta.



```yaml
services:
  pihole:
    container_name: pihole
    image: pihole/pihole:latest
    environment:
      TZ: "Europe/Madrid"           # Cambia tu zona horaria
      WEBPASSWORD: "erossensey23" # Contraseña de acceso al panel
      DNS1: 1.1.1.1
      DNS2: 9.9.9.9
    volumes:
      - ./etc:/etc/pihole
      - ./dnsmasq:/etc/dnsmasq.d
    ports:
      - "8880:80/tcp"    # Panel web
      - "53:53/tcp"
      - "53:53/udp"
    restart: unless-stopped
    networks:
      - wg_pihole

networks:
  wg_pihole:
    external: true
```

6. *Integración 1: Acceso Web vía Caddy*

Para acceder al panel de Pi-hole desde fuera usando nuestro dominio, añadimos estas líneas al archivo `Caddyfile` (ubicado en `~/docker/caddy`).

Código a añadir en Caddyfile: "# Panel de Pi-hole pihole.tu-subdominio.duckdns.org { reverse_proxy 172.19.0.2:80 }"

7. *Integración 2: Conexión con WireGuard (WG-Easy)*

Para que los dispositivos conectados a la VPN usen este DNS, editamos el `docker-compose.yml` de **WireGuard** para conectarlo a la red `wg_pihole` y asignarle el DNS.

Código modificado en WireGuard: "services: wg-easy: environment: - WG_DEFAULT_DNS=172.19.0.2 networks: default: wg_pihole:

networks: default: wg_pihole: external: true"

```yaml
volumes:
  etc_wireguard:

services:
  wg-easy:
    environment:
    #  Optional:
    #  - PORT=51821
    #  - HOST=0.0.0.0
      - INSECURE=true

    image: ghcr.io/wg-easy/wg-easy:15
    container_name: wg-easy
    network_mode: "host"
    volumes:
      - etc_wireguard:/etc/wireguard
      - /lib/modules:/lib/modules:ro
#    ports:
#      - "51820:51820/udp"
#      - "51821:51821/tcp"
    restart: unless-stopped
    cap_add:
      - NET_ADMIN
      - SYS_MODULE
      # - NET_RAW # ⚠️Uncomment if using Podman
networks:
  wg_pihole:
    external: true
```

8. *Despliegue y Verificación*

Finalmente, levantamos el servicio Pi-hole.

Comando a ejecutar: "docker compose up -d"