Pi-hole

- Pi-hole on Docker Compose
- Pi-hole Moving Port 53

Pi-hole on Docker Compose

.yaml:

```
services:
  pihole:
    container_name: pihole
    image: pihole/pihole:latest
    ports:
      # Web admin → map to a high port on host so NGINX can proxy to it without conflict
      - "80:80/tcp"   # Or pick any free port like 8080:80, 8000:80, etc.
      # DNS ports → these MUST stay exposed on the host (for ad-blocking to work network-wide)
      - "53:53/udp"
      - "53:53/tcp"
      # Uncomment if you want Pi-hole to handle DHCP too (requires NET_ADMIN cap below)
      #- "67:67/udp"
    environment:
      TZ: 'America/Los_Angeles'          # Update to your timezone (you're in Anaheim, CA)
      FTLCONF_webserver_api_password: 'correct horse battery staple'  # Change this!
      # Optional but useful for reverse proxy setups:
      VIRTUAL_HOST: pi.hole              # Helps some auto-proxy tools, but not required for
manual NGINX
      # If you need Pi-hole to listen on all interfaces for DNS (common in Docker bridge
network)
      FTLCONF_dns_listeningMode: 'all'
    volumes:
      - './data:/etc/pihole'
      # - './etc-dnsmasq.d:/etc/dnsmasq.d'   # Uncomment if you use custom dnsmasq configs
    cap_add:
      - NET_ADMIN                        # Required for DNS (and DHCP if enabled)
    restart: unless-stopped
```

Pi-hole Moving Port 53

Free up port 53 (for Pi-hole Docker)

In Linux, you need to tell systemd-resolved **not to listen on port 53** (via its stub listener), then restart it. This keeps local DNS working on the host while freeing the port for Pi-hole.

Step 1: Confirm what's using port 53

Run one of these commands (use sudo if needed):

```
sudo ss -tulpn | grep ':53'
```

or

```
sudo lsof -i :53
```

or the classic:

```
sudo netstat -tulpn | grep ':53'
```

Look for output like:

- systemd-resolved or systemd-resolve listening on 127.0.0.53:53 (or sometimes 0.0.0.0:53)
- Possibly dnsmasq, unbound, named (BIND), or another DNS service if you've installed one before.

If it's systemd-resolved (most common), proceed.

Step 2: Free up port 53 (for Pi-hole Docker)

You need to tell systemd-resolved **not to listen on port 53** (via its stub listener), then restart it. This keeps local DNS working on the host while freeing the port for Pi-hole.

1. Edit the config file:

   ```
   sudo nano /etc/systemd/resolved.conf
   ```

Find the line #DNSStubListener=yes (it might be commented out).
Change it to (uncomment and set):

text:

```
DNSStubListener=no
```

Save and exit.

2. Restart the service:

```
sudo systemctl restart systemd-resolved
```

(Or if it was fully disabled before, sudo systemctl enable --now systemd-resolved isn't needed — just restart.)

3. Verify port 53 is now free:
Run the check command from Step 1 again — no process should be listening on :53 anymore.
-If the command returns blank, that is good.

If still taken, double-check for other services (e.g., sudo systemctl stop unbound if you have Unbound installed, or sudo systemctl disable --now dnsmasq).

Step 3: Restart your Pi-hole container

Now try again:

```
sudo docker compose down
sudo docker compose up -d
```

It should start without the port error.

Important: Fix host DNS resolution after this change

Disabling the stub listener breaks the host's own DNS (it was using 127.0.0.53 as its resolver).

To restore it:

1. Remove the symlink to the stub resolver:

Bash

```
sudo rm /etc/resolv.conf
```

2. Create a new static /etc/resolv.conf with a working upstream DNS (temporarily use public ones; later change to your Pi-hole IP once it's up):
   Bash

```
sudo nano /etc/resolv.conf
```

   Put in something like:
   text

```
nameserver 1.1.1.1
nameserver 1.0.0.1
# or nameserver 8.8.8.8
```

   Save

Once its running: On your server (the host machine)

/etc/resolv.conf is currently:

```
nameserver 1.1.1.1
nameserver 1.0.0.1
```

(or whatever DNS you put in there)

Change it to this: (add nameserver 127.0.0.1)

```
nameserver 127.0.0.1
nameserver 1.1.1.1
nameserver 1.0.0.1
```

Commands:

```
sudo nano /etc/resolv.conf
```

Paste the three lines above, save and exit (Ctrl+O → Enter → Ctrl+X).

Then lock the file so it survives reboot:

```
sudo chattr +i /etc/resolv.conf
```

Test it works:

Bash

```
ping google.com
```

(Should resolve and ping.)

On your home router

1. Log in to your router's admin page.
2. Go to **LAN → DHCP Server** (or LAN Setup / DHCP Settings / Advanced Network).
3. Find the **DNS Server** fields.
4. Set **Primary DNS Server** to: **192.168.0.11** (this is the host/server IP – do **not** add any port here)
5. (Optional but recommended) Set **Secondary DNS Server** to: **9.9.9.9**
6. Click **Save** or **Apply**.
7. If prompted, reboot the router or wait 1–2 minutes.

After router changes

1. On your phone, laptop, or other devices (not the server): Turn Wi-Fi off and back on (or reboot the device) to pick up the new DHCP settings.
2. Test access to the Pi-hole dashboard from any device on your network:
   - Direct container access (bypassing NGINX, for testing):

     http://192.168.0.11:8081/admin (replace 8081 with the actual host port you mapped in docker-compose.yml under ports: for the web interface, e.g., if you have "8000:80", use :8000)
   - If you already set up NGINX reverse proxy for Pi-hole: http://192.168.0.11/admin (or

     https://192.168.0.11/admin if HTTPS is configured) → no custom port needed in the

URL, NGINX handles it on 80/443

- If you set up local DNS in Pi-hole (recommended): http://pi.hole/admin (or https://pi.hole/admin if HTTPS) → no IP or port needed

3. Verify it's working:
   - Dashboard loads.
   - Visit a site with lots of ads (news site, YouTube app on phone) → most ads/trackers blocked.
   - In Pi-hole dashboard → Query Log → see queries coming from multiple devices (phones, laptops, etc.).