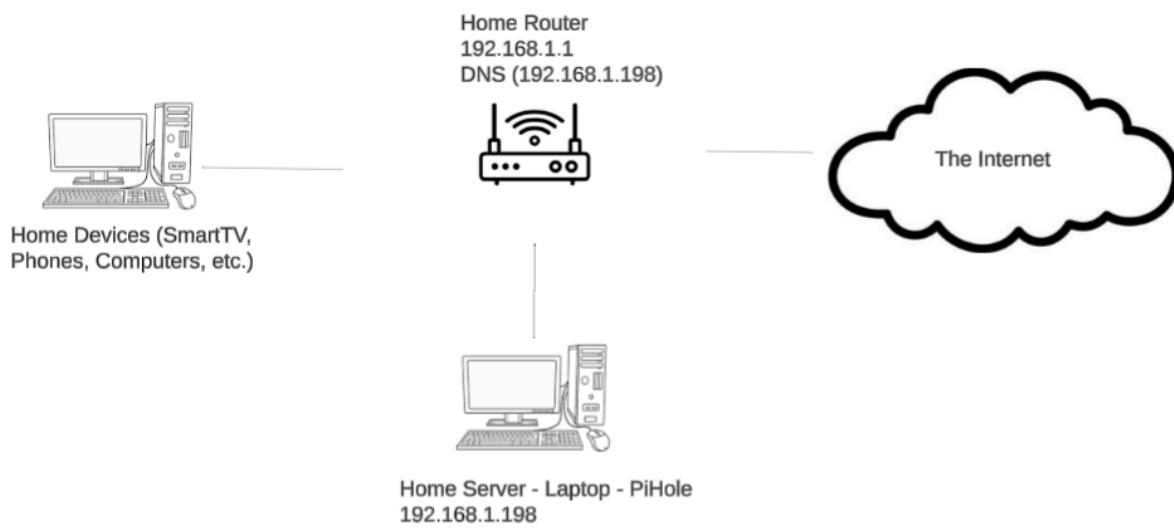


## PiHole Project

**Description:** In this project I will repurpose an old laptop that was laying around and put it to use. The first service I want on this home server is PiHole so that I can have a network wide ad and tracker blocker.

### Topology:

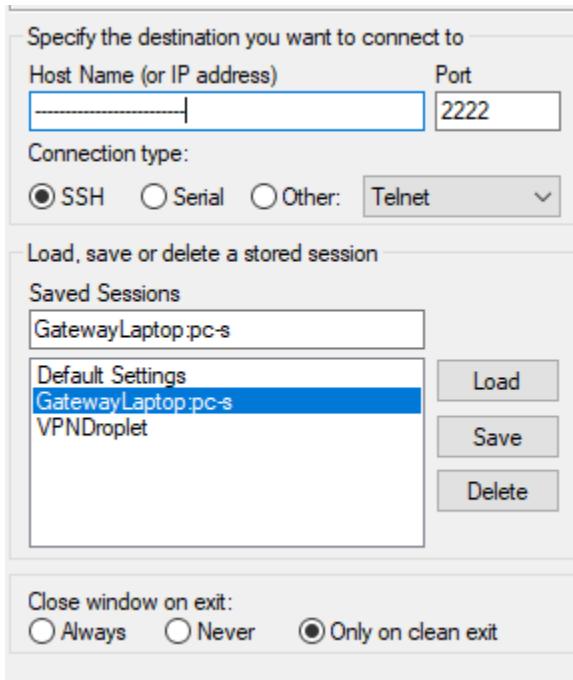


### Key Syntax:

Term	Definition
pihole -g	Update PiHole blocklists
pihole restartdns	Restarts the PiHole DNS service
pihole -t	Tails the PiHole log watching it in real time
Nslookup flurry.com 192.168.1.198	Tests a known ad domain to see if it is blocked
nano/etc/netplan/01-netcfg.yaml	Edits the netplan for Ubuntu

### Verification:

1. I started off by installing ubuntu server 24.04.1 onto a flash drive, booting to the flash drive on the laptop and installing ubuntu server that way.
2. I then setup ssh via putty so that I could remotely setup the PiHole from my main PC.



3. I went straight into setting a static IPv4 address for my ubuntu server by using the command sudo nano /etc/netplan/01-netcfg.yaml to edit the network configuration file.

```
GNU nano 7.2
network:
  version: 2
  renderer: networkd
  ethernets:
    enp2s0:
      dhcp4: no
      addresses:
        - 192.168.1.198/24
      routes:
        - to: default
          via: 192.168.1.1
  nameservers:
    addresses: [1.1.1.1, 8.8.8.8]
```

4. I then went to my router's configuration page and made a DNS entry to reserve an address so that my server always has the IP 192.168.1.198.

#### DNS Entry

Host Name:

---

IPv4 Address:  .  .  .

---

**Apply >** **Cancel >**

5. This screenshot shows that I then installed PiHole on my ubuntu server.

```
root@pc-s:~# curl -sSL https://install.pi-hole.net | bash
```

```
[+] Root user check
```



```
[i] SELinux not detected
```

```
[i] Update local cache of available packages...
```

6. This screenshot is showing that PiHole is properly running and blocking trackers/ads on my network.

```
root@pc-s:~# pihole status
[✓] FTL is listening on port 53
    [✓] UDP (IPv4)
    [✓] TCP (IPv4)
    [✓] UDP (IPv6)
    [✓] TCP (IPv6)

[✓] Pi-hole blocking is enabled
```

7. This screenshot shows that the blocklist was downloaded and correctly applied.

```
root@pc-s:~# pihole -g
[✓] DNS resolution is available

[i] Neutrino emissions detected...

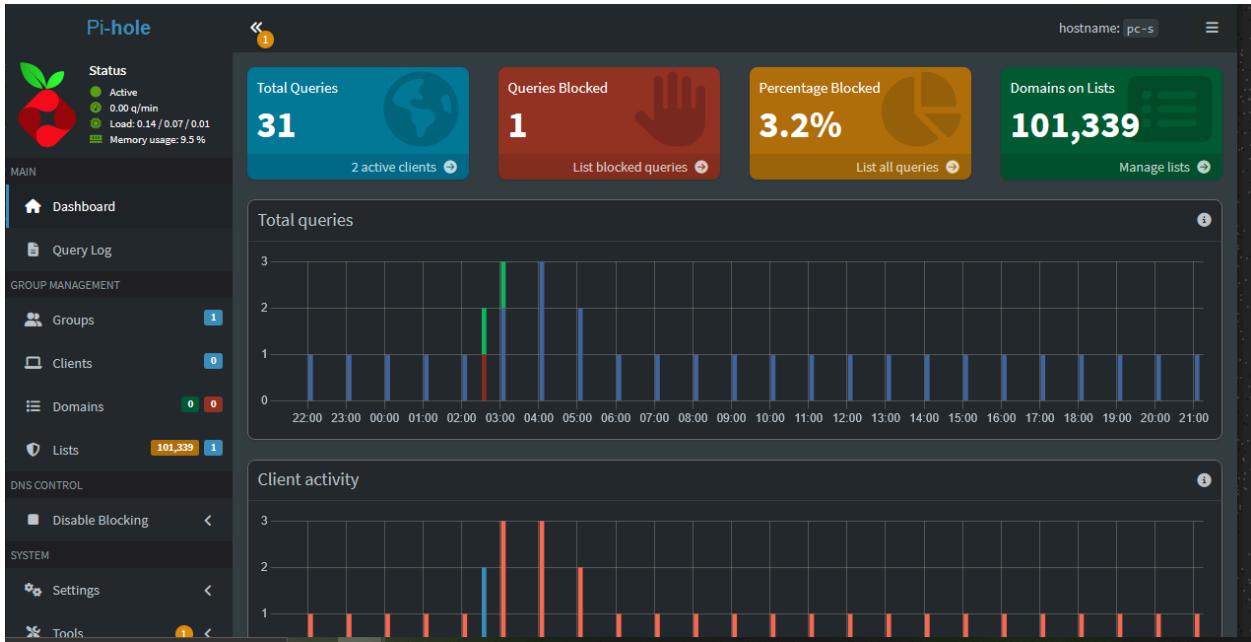
[✓] Preparing new gravity database
[✓] Creating new gravity databases
[✓] Pulling blocklist source list into range
[i] Using libz compression

[i] Target: https://raw.githubusercontent.com/StevenBlack/hosts/master/hosts
[✓] Status: Retrieval successful
[i] List has been updated
[✓] Parsed 101339 exact domains and 0 ABP-style domains (blocking, ignored 1 n
on-domain entries)
    Sample of non-domain entries:
        - fe80::1%lo0

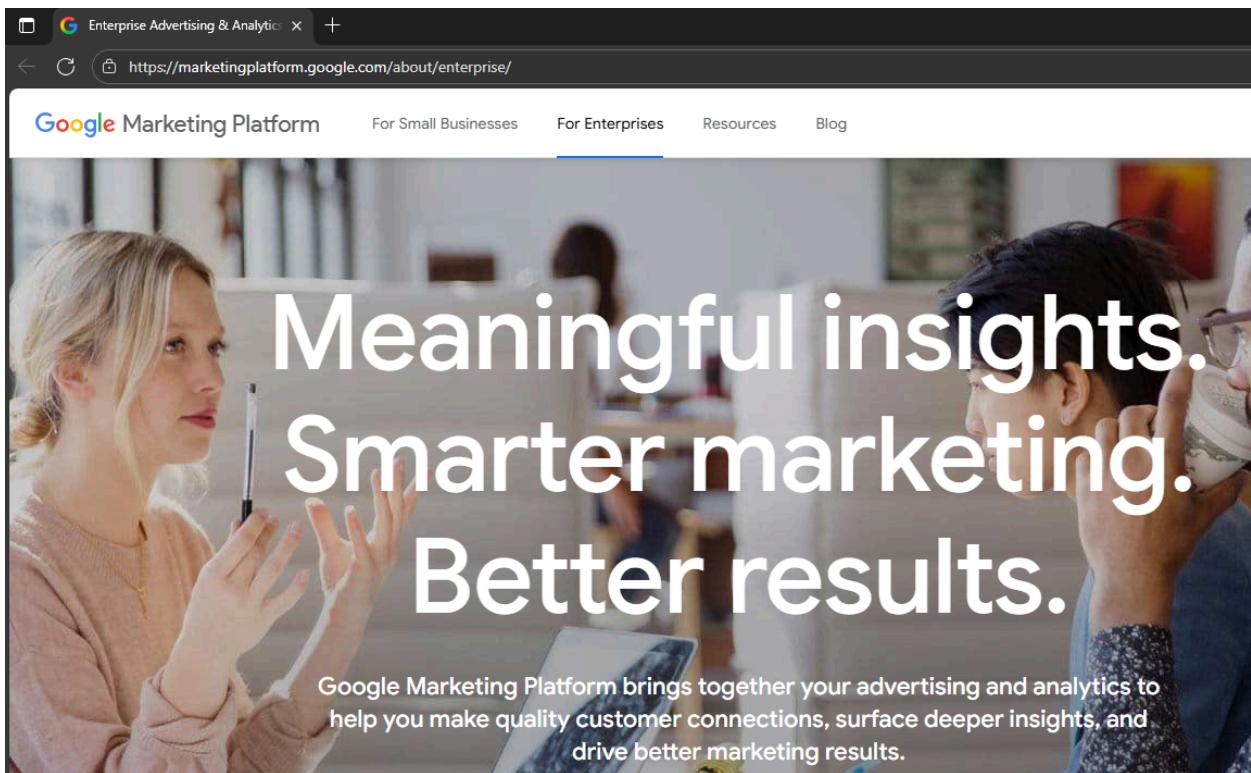
[✓] Building tree
[i] Number of gravity domains: 101339 (101339 unique domains)
[i] Number of exact denied domains: 0
[i] Number of regex denied filters: 0
[i] Number of exact allowed domains: 0
[i] Number of regex allowed filters: 0
[✓] Optimizing database
[✓] Swapping databases
[✓] The old database remains available
[✓] Cleaning up stray matter

[✓] Done.
root@pc-s:~#
```

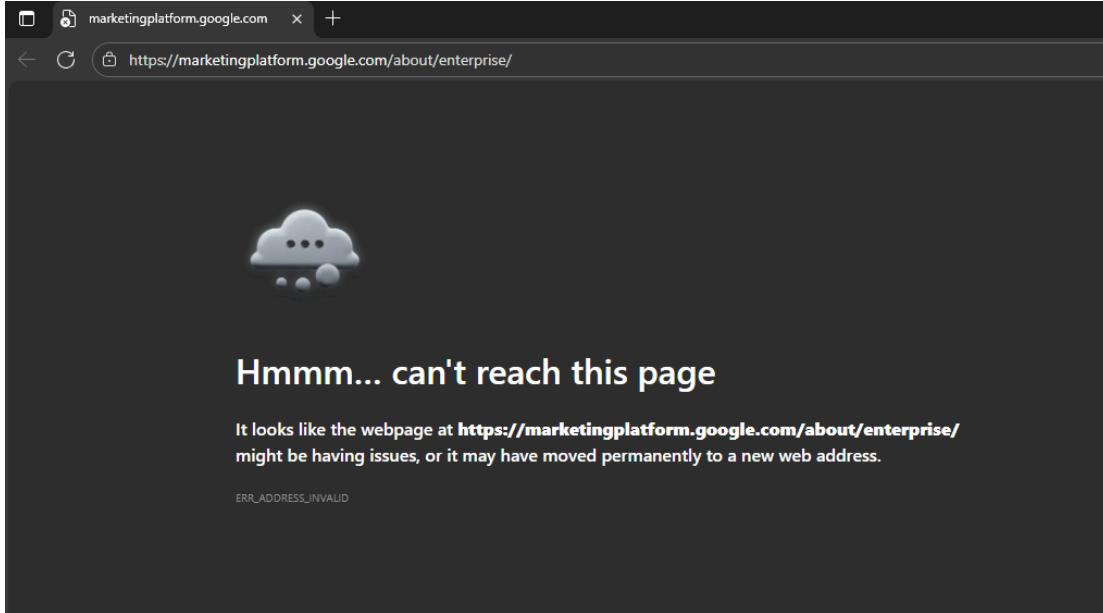
8. This is a screenshot of the web ui, allowing monitoring of all blocked traffic, also showing all clients using the PiHole.



- To verify that the PiHole properly functions, we can try to go to googles marketing platform site (which is included in the blocklist) while we do not have the PiHole as our DNS server. As you can see the site properly loads.



Once we change our DNS server to the PiHole, the site no longer loads which means that the PiHole blocklist is working properly.



10. We can check the web ui to make sure our query was blocked. It does show up here meaning that everything is working properly.

A screenshot of the Pi-hole web interface. On the left, there's a sidebar with various management sections like 'Status', 'Groups', 'Clients', 'Domains', 'Lists' (with 101 items), 'DNS Control', and 'System'. The main area is titled 'Advanced filtering' and shows a table of 'Recent Queries'. The table has columns for 'Time', 'Type', 'Domain', 'Client', and 'Response'. All the entries in the table show a red 'X' icon in the 'Type' column, indicating they were blocked. The 'Domain' column lists 'marketingplatform.google.com' multiple times. The 'Client' column shows '192.168.1.164' for all entries. The 'Response' column shows response times like '33.6 µs', '21.7 µs', etc., followed by a green checkmark and the word 'Allow'.

11. The final step is to make sure all of my network devices use the PiHole as their DNS server. I can do this by logging into my router and changing the main DNS server to the PiHole. Once this is complete, every device that is on my network will block ads and trackers making this project a complete success.

## IPv4 Address Distribution

IPv4 Address Distribution provides the ability to allocate IPv4 addresses and configuration parameters to selected hosts

Name	Service	Subnet Mask	Dynamic IPv4 Range	Action
<a href="#">Network (Home/Office)</a>	Disabled			<a href="#">Edit</a>

[DHCP Leases >](#)

[Close >](#)

## DHCP Settings

### DHCP Settings

#### **DHCP server enabled**

Make sure your router's DHCP server is disabled when using the Pi-hole DHCP server!

#### Range of IP addresses to hand out

Start 192.168.1.2

End 192.168.1.254

#### Router (gateway) IP address

Router 192.168.1.1

#### Netmask

Netmask automatic

## Conclusion:

This project was a fairly simple and fun way to repurpose a very old laptop that was not getting any practical use anywhere else. It is nice to know that all devices on my home network using my server for DNS are blocking ads and trackers. I am excited to add other services onto it and expand my home server even more.

## **References:**

1. <https://docs.pi-hole.net/>
2. <https://raw.githubusercontent.com/pi-hole/pi-hole/master/automated%20install/basic-install.sh>
3. <https://netplan.readthedocs.io/en/stable/>
4. <https://github.com/StevenBlack/hosts>