Practical architecture for combining Pi-hole and Nginx Proxy Manager

Overview: Roles of Each Component

Pi-hole

- Runs local DNS (port 53)
- Serves as the DNS resolver for your network
- Provides local DNS records so devices resolve service names (e.g., `home.lab`, `jellyfin.lab`)
- Does *not* need to serve HTTP(80) or HTTPS(443)

Nginx Proxy Manager (NPM)

- Handles all HTTP/HTTPS traffic for services
- Manages SSL certificates (Let's Encrypt if you choose)
- Routes incoming application requests to internal services on different ports/IPs

Key Principle

> Pi-hole handles *DNS resolution* only. NPM handles *web traffic routing and HTTPS termination*.

This clear separation avoids conflicts and makes management scalable.

?? Best Architecture (Most Reliable)

1) Move Pi-hole's Web UI off Port 80/443

By default Pi-hole binds its web UI to port 80/443. This collides with NPM.

Change Pi-hole Admin UI to a different port (e.g., `8081`, `8888`):

- On Pi-hole host: change Lighttpd config so `/admin` listens on `8081`
- DNS on Pi-hole still runs on port 53 and is unaffected by this change

This allows NPM to own 80/443 exclusively. (bolet.io)

---

2) Local DNS Records in Pi-hole for All Reverse-Proxied Hosts

Create `A` records that point friendly domains to NPM's IP.

Example:

```
pihole.lab.zn80.net     ? 192.168.10.105
jellyfin.lab.zn80.net   ? 192.168.10.105
bookstack.lab.zn80.net  ? 192.168.10.105
```

Reasoning:

- Devices query Pi-hole DNS and get your internal names
- NPM then routes based on Host header

Pi-hole is the DNS source of truth for your LAN. (ReproDev)

---

3) Configure NPM Proxy Hosts for Each Service

Inside NPM **Proxy Hosts**:

Domain/Hostname	Forward To	Port	Notes
`pihole.lab.zn80.net`	192.168.10.105	8081	Pi-hole admin UI
`jellyfin.lab.zn80.net`	192.168.10.xxx	8096	Jellyfin service
`bookstack.lab.zn80.net`	…	…	Other services

Important details

- For Pi-hole you may need to forward `/admin` correct paths
- Enable SSL for all hosts (ideally) via Let's Encrypt (even internal DNS) *If Let's Encrypt can't validate via DNS challenge for internal domains, you can use a local CA or self-signed cert and manage trust in your clients.* (Reddit)

This makes everything *clean to access via Hostnames exclusively*:

```
https://jellyfin.lab.zn80.net
https://pihole.lab.zn80.net/admin
```

---

4) Maintain Internal DNS Instead of Wildcard DNS (Optional but safer)

You could be tempted to use a wildcard (`*.lab.zn80.net ? 192.168.10.105`), but that is **less safe and confusing** for certificate validation and service discovery.

Better pattern:

- Pi-hole local DNS ? explicit A records per hostname
- Pi-hole ? NPM IP
- NPM ? internal service

Explicit records help:

- TTL management
- SSL certificate issuance
- Troubleshooting

? Why This Is the Best Way

? DNS + Reverse Proxy are cleanly separated

Pi-hole isn't serving HTTP, so there's no port conflict. NPM can handle 80/443 without interference.

? You achieve real HTTPS for internal domains

Every service can have a valid certificate via NPM.

? Easy, scalable, and future-proof

Adding a new service is trivial:

1. Add A record in Pi-hole
2. Add Proxy Host in NPM
3. Done

? Works with VPN / local DNS clients

If your devices use Pi-hole's DNS (wired/wireless/VPN), they resolve domain names that NPM can proxy. (Pi-hole Userspace)

? Detailed Implementation

A) Change Pi-hole web interface port

Edit `/etc/lighttpd/lighttpd.conf`:

```
server.port = 8081
```

Restart:

```
sudo systemctl restart lighttpd
```

B) Add DNS entries in Pi-hole

In Pi-hole:

```
Local DNS ? DNS Records
```

Examples:

```
pihole.lab.zn80.net ? 192.168.10.105
jellyfin.lab.zn80.net ? 192.168.10.105
```

C) In NPM, add Proxy Hosts

For each service:

- Domain names: your local domain
- Forward hostname: actual service IP/port (Pi-hole admin UI at 8081)
- SSL:
  - Request new cert (if public domain or DNS challenge available)
  - Or use local CA / self-signed trusted cert

Set common options:

- Block exploits: enabled
- Expires headers: optional
- Websockets: if needed

? What *Not* to Do

? Don’t proxy Pi-hole DNS-port

Do not try to proxy port 53 through NPM. DNS must be direct.

? Don’t expose Pi-hole admin UI publicly

Pi-hole should only be accessible from your LAN or VPN.

? Don’t use wildcards unless you know certificate chain

Wildcard DNS can break HTTPS if certificates aren’t matching.

? Final Result You Should See

After setup:

```
nslookup pihole.lab.zn80.net      # ? 192.168.10.105
nslookup jellyfin.lab.zn80.net    # ? 192.168.10.105
```

```
Browser:
https://pihole.lab.zn80.net/admin  # SSL
https://jellyfin.lab.zn80.net      # SSL
```

Pi-hole DNS works, NPM handles web access, and everything is secure, consistent, and manageable.

Revision #5

Created 2025-12-14 20:26:50 UTC by Carsten

Updated 2026-02-14 15:36:51 UTC by Carsten