# Configuration to enable internal subdomain for reverse proxy use

docker

InstantDreams 1 March 14, 2023, 4:19pm

My current hosting situation is similar to that of many:

1. Personal Domain Management
   1.1. Hosted on Azure DNS
   1.2. CNAME records pointed to DuckDNS domain

2. Dynamic DNS Management
   2.1. Hosted on DuckDNS
   2.2. CNAME requests via Azure forwarded to local IP address

3. Gateway Management
   3.1. Fibre ONT in bridge mode
   3.2. Netgear Orbi RBK852 as router
   3.3. Router configured to forward requests on port 80 and 443
   3.4. Target is edge server

4. Edge Management
   4.1. Nginx Proxy Manager is reverse proxy hosted on edge server
   4.2. Receives requests on port 80 and port 443
   4.3. Forwards [subdomain].[domain] requests to appropriate local services

There are many different ways to do this and multiple hosting sites or services that can be used, but this approach is working very well for me.

Azure knows the CNAME records to forward. DuckDNS knows where roughly to send them, and the router makes sure they get to the right place. NPM then routes the requests to the correct services.

How can I best use Pi-hole DNS / dnsmasq to do this for an internal domain with subdomains?

Let's say I have the following:

External Domain: **example.com**
Internal Domain: **example.lan**

Router / Gateway: **192.168.1.1/router**
Edge Server 1: **192.168.1.91/edge1**
Edge Server 2: **192.168.1.92/edge2**

NPM Address: `http://192.168.1.91:81/` or `http://edge1:81/`
Pi-hole1 Address: `http://192.168.1.91/8080/admin/` or `http://edge1:8080/admin/`
Pi-hole2 Address: `http://192.168.1.92/8080/admin/` or `http://edge2:8080/admin/`

Local queries are resolved via DNS, which means they should be forwarded to the two Pi-hole servers.

Environment variables have been set to enable the following configurations:

**01-pihole.conf**

```
domain-needed
expand-hosts
bogus-priv
dnssec
```

**02-pihole-dhcp.conf**

```
domain=example.lan
local=/example.lan/
```

**05-pihole-custom-cname.conf**

```
cname=pihole1.example.lan,example.lan
cname=pihole2.example.lan,example.lan
cname=homeassistant.example.lan,example.lan
```

I have Nginx Proxy Manager configured with a proxy host like this:

1. Add Proxy Host
   1.1. Domain Names: **pihole1.example.lan**
   1.2. Scheme: **http**
   1.3. Forward Hostname / IP: **edge1**
   1.4. Forward Port: **8080**
   1.5. Cache Assets: **Disabled**
   1.6. Block Common Exploits: **Disabled**
   1.7. Websockets Support: **Enabled**

I am missing something, and I don't know what it is. I have a custom configuration file in /etc/dnsmasq.d/ where I've tried the following settings:

07-custom.conf

```
server=/example.lan/192.168.1.91
```

or

```
address=/example.lan/192.168.1.91
```

But this doesn't seem to be working because the step to route the CNAME record to the reverse proxy isn't there:

```
$ docker exec pihole dig @192.168.1.1 pihole1.example.lan

; <<>> DiG 9.16.37-Debian <<>> @192.168.1.1 pihole1.example.lan
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18175
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;pihole1.example.lan.     IN      A

;; ANSWER SECTION:
pihole1.example.lan. 0    IN      CNAME   example.lan.

;; Query time: 0 msec
;; SERVER: 192.168.1.1#53(192.168.1.1)
;; WHEN: Tue Mar 14 10:15:24 MDT 2023
;; MSG SIZE  rcvd: 85
```

Has anyone done this, or is there any tutorial or explanation available?

Thank you!

**DanSchaper** 2 March 14, 2023, 6:32pm

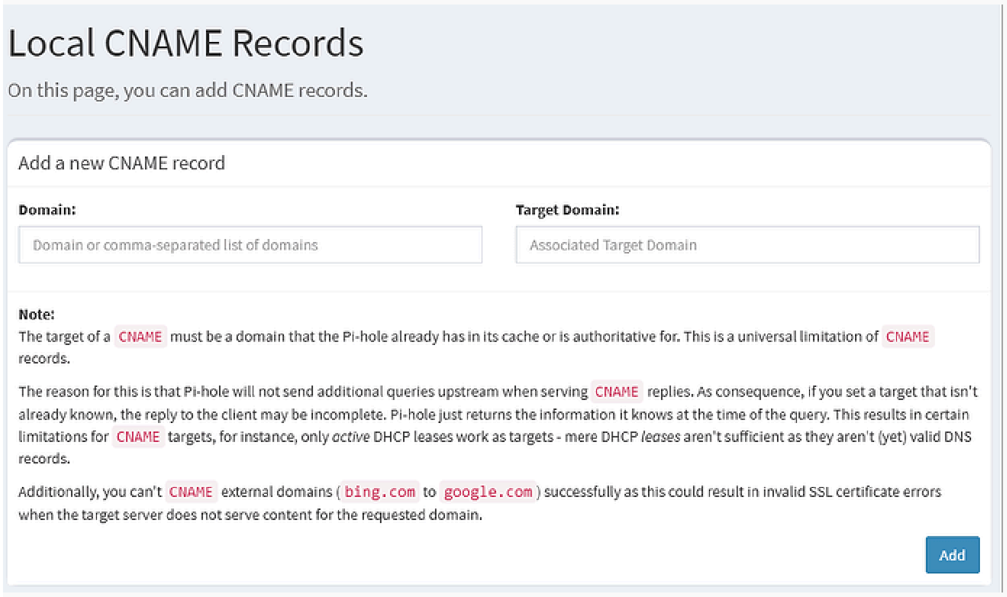
CNAMEs are pretty limited with `dnsmasq`, which is the base for `FTL`.

From the dnsmasq man page:

> `--cname=<cname>,[<cname>,]<target>[,<TTL>]`
> Return a CNAME record which indicates that `<cname>` is really `<target>`. There is a significant

> limitation on the target; it must be a DNS record which is known to dnsmasq and NOT a DNS record which comes from an upstream server. The cname must be unique, but it is permissible to have more than one cname pointing to the same target. Indeed it's possible to declare multiple cnames to a target in a single line, like so: --cname=cname1,cname2,target
>
> If the time-to-live is given, it overrides the default, which is zero or the value of --local-ttl. The value is a positive integer and gives the time-to-live in seconds.

This is also mentioned on the web interface.



Local CNAME Records

On this page, you can add CNAME records.

Add a new CNAME record

Domain: Domain or comma-separated list of domains

Target Domain: Associated Target Domain

Note:
The target of a `CNAME` must be a domain that the Pi-hole already has in its cache or is authoritative for. This is a universal limitation of `CNAME` records.

The reason for this is that Pi-hole will not send additional queries upstream when serving `CNAME` replies. As consequence, if you set a target that isn't already known, the reply to the client may be incomplete. Pi-hole just returns the information it knows at the time of the query. This results in certain limitations for `CNAME` targets, for instance, only *active* DHCP leases work as targets - mere DHCP *leases* aren't sufficient as they aren't (yet) valid DNS records.

Additionally, you can't `CNAME` external domains (`bing.com` to `google.com`) successfully as this could result in invalid SSL certificate errors when the target server does not serve content for the requested domain.

You can try a line in `/etc/hosts` that contains the `example.lan` to IP address mapping, then restarting the DNS resolver to pick up that change.

---

InstantDreams 3 March 15, 2023, 12:47am

Thank you. I shall try the following:

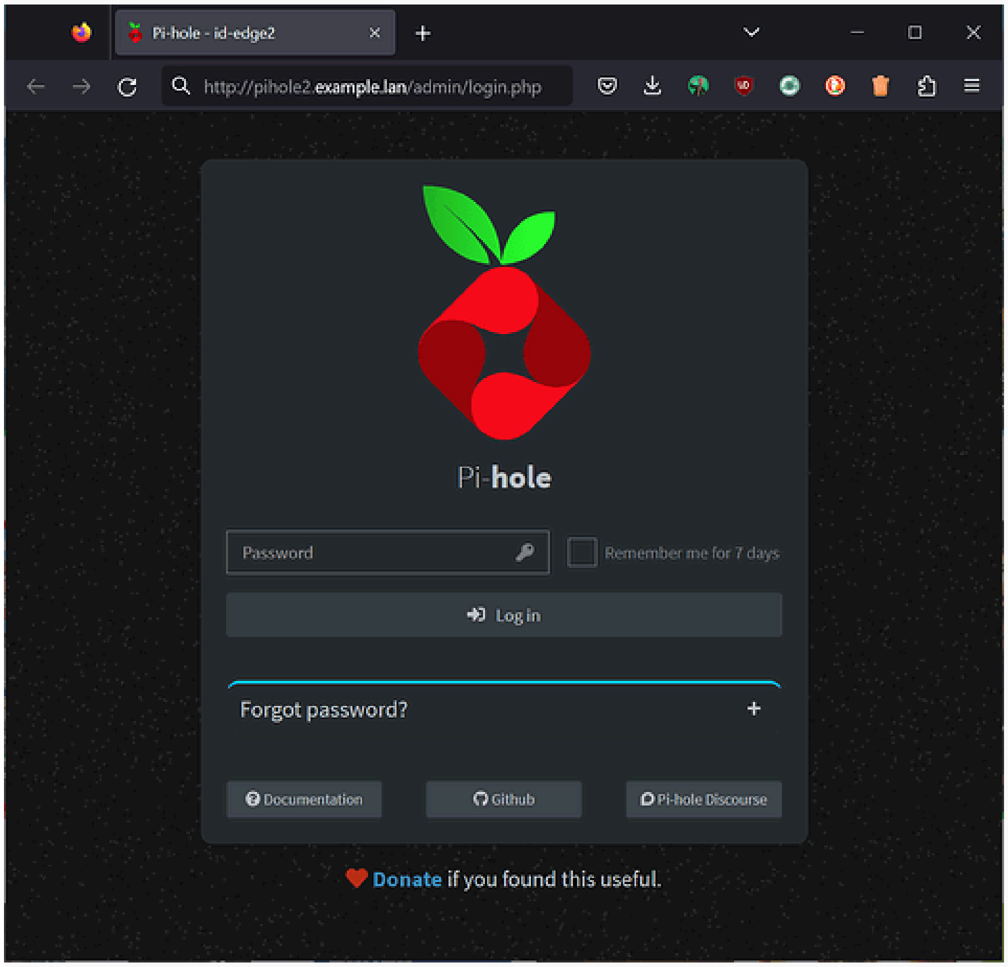
1. Edit /etc/pihole/custom.list
2. Add line entry for `192.168.1.91 example.lan`
3. Restart Pi-hole docker container
4. Test the dig commands

I will post my findings.

---

InstantDreams 4 March 15, 2023, 12:57am

That worked perfectly, thank you - it closed the resolution loop.



Results of dig:

```
 $ docker exec pihole dig @192.168.1.1 pihole2.example.lan

; <<>> DiG 9.16.37-Debian <<>> @192.168.1.1 pihole2.example.lan
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5984
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;pihole2.example.lan.      IN      A

;; ANSWER SECTION:
pihole2.example.lan. 0     IN      CNAME   example.lan.
```

```
example.lan.        0       IN      A       192.168.1.91

;; Query time: 3 msec
;; SERVER: 192.168.1.1#53(192.168.1.1)
;; WHEN: Tue Mar 14 18:56:41 MDT 2023
;; MSG SIZE  rcvd: 101
```

Thank you @DanSchaper !

**InstantDreams** 5 March 15, 2023, 2:20am

I have also found that using the following format also works:

```
cname=pihole-1.example.lan,edge-1
cname=pihole-2.example.lan,edge-1
```

Where edge-1 is the edge server running the reverse proxy. Easier to use this than add other A records via the `custom.list` file, perhaps.

**Bucking_Horn** 6 March 15, 2023, 7:35am

CNAME records define an alias, not a forward or redirection.

> InstantDreams:
>
> Edge Server 1: **192.168.1.91/edge1**
> Edge Server 2: **192.168.1.92/edge2**

> InstantDreams:
>
> Pi-hole1 Address: `http://192.168.1.91/8080/admin/` or `http://edge1:8080/admin/`
> Pi-hole2 Address: `http://192.168.1.92/8080/admin/` or `http://edge2:8080/admin/`

> InstantDreams:

```
cname=pihole-1.example.lan,edge-1
cname=pihole-2.example.lan,edge-1
```

Your above configuration is configuring the aliases `pihole-1.example.lan` and `pihole-3.example.lan` for the canonical name `edge-1`.

This would result in DNS requests for `pihole-2.example.lan` returning an A record of `192.168.1.91`, i.e. your second Pi-hole at `192.168.1.92` would not be reachable by name.

Is that your intended outcome?

---

InstantDreams 7 March 15, 2023, 8:02am

> Bucking_Horn:
>
> Is that your intended outcome?

Yes, because `edge-1` is running the active Nginx Proxy Manager instance, so all requests to the CNAME records will be processed there.

It closes the connection:

1. External request on 443
   1.1. Azure CNAME to DuckDNS
   1.2. DuxkDNS to Router
   1.3. Router to NPM on edge-1
   1.4. Request processed
2. Internal request on 80
   2.1. Pi-hole CNAME to NPM on edge-1
   2.2. Request processed

This seems to be exactly what I needed.

---

Bucking_Horn 8 March 15, 2023, 8:33am

As already mentioned, CNAMEs are aliases, not redirections.

It seems you are trying to accept external requests for `https://pihole-2.example.com` and redirect them to internal `http://pihole-2.example:8080`.

That redirection has to be configured in your `nginx`.

As the local DNS resolver, your Pi-hole would be only involved in providing the correct name resolution for `pihole-2.example`, and that should be `192.168.1.92` (if that IP is hosting your second Pi-hole). No CNAME records are required here, only A.

---

**InstantDreams** 9 March 15, 2023, 1:25pm

> Bucking_Horn:
>
> It seems you are trying to accept external requests

That is incorrect. I am trying to accept **internal** requests for `example.lan`. Using CNAME records is working.

---

**Bucking_Horn** 10 March 17, 2023, 7:28am

Ah, I see - your repeated mentioning of your external steps lured me into believing that you wanted to accept external requests in your step 1 and then redirect them to your internal servers in step 2.

A CNAME would be valid, but not be required.

Whether CNAME or not, in general I'd probably suggest to use a set of two distinct domains per target host in that case, to distinguish proxied (e.g. `pihole-1-ui`) from unproxied access (e.g. `pihole-1`) to that same host.
This would avoid a possible loop when configuring the redirection in nginx by hostnames as well as keeping access to unproxied services by name functional.

What's your motivation or proxying HTTP internally?
What's the benefit of that approach over having local clients access your Pi-holes directly?

You should also note that you do not have to edit files manually:
Pi-hole's UI allows simple configurations of both A and CNAME records via **Local DNS Records** and **Local CNAME records**.

InstantDreams 11 March 26, 2023, 6:50pm

My configuration seems rock solid right now. I exclusively use [subdomain].example.com.

External requests are resolved using this process:

- External Domain Name: **example.com**
  - Domain Name Host: **Azure DNS Zones**
    - CNAME Record Example: **nextcloud.example.com**
    - CNAME Record Alias: **example.duckdns.org**
  - Dynamic DNS Host: **Duck DNS**
    - Duck DNS Domain: **example**
    - Duck DNS IP: **[home network ip address]**
  - Internet Service Provider: **Fibre ONT**
    - ONT Configuration: **Bridge Mode to Personal Router**
  - Router: Netgear **RBR852**
    - Port Forwarding: **80 to server edge1**
    - Port Forwarding: **443 to server edge1**
  - Server: **edge1**
    - Docker Container: **duckdns**
    - Port: **None**
    - Function: **Refreshes Duck DNS IP regularly**
    - Docker Container: **nginx-proxy-manager**
    - Port: **443 & 80 (https and http traffic)**
    - Port: **81 (web interface)**
    - Function: **Reverse Proxy**
    - Configuration:
    - Domain Name: **nextcloud.example.com**
    - Scheme: **http**
    - Forward: **services**
    - Port: **port**
    - SSL Certificate: **wildcard example.com**

Requests for externally hosted subdomains are received through my DNS host and forwarded to my Dynamic DNS host which are then forwarded to my home IP. My router forwards requests from ports 443 and 80 to one of my edge servers, which is hosting Nginx Proxy Manager in a Docker container. NPM determines the request address and forwards to the appropriately hosted service.

Internal requests work this way:

- Internal Domain Name: **example.com**
  - Router: Netgear **RBR852**
    - Primary DNS **server edge2**

- Secondary DNS: **server edge1**
- Server: **edges** & **edge1**
- Docker Container: **pihole-2** & **pihole-1**
- Port: **8080 (web interface)**
- Port: **53 (dns)**
- Port: **67 (dhcp)**
- Function: **ad blocker, dns management, dhcp management**
- Configuration:
- DHCP Pihole Domain Name: **example.com**
- DNS A Record: **edge1 to static ip address**
- DNS A Record: **edge2 to static ip address**
- DNS A Record: **services to static ip address**
- DNS CNAME Record: **pihole-1.example.com to edge1**
- DNS CNAME Record: **pihole-2.example.com to edge1**
- DNS CNAME Record: **homer.example.com to edge1**
- Docker Container: **nginx-proxy-manager**
- Port: **443 & 80 (https and http traffic)**
- Port: **81 (web interface)**
- Function: **Reverse Proxy**
- Configuration:
- Domain Name: **pihole-1.example.com**
- Scheme: **http**
- Forward: **edge1**
- Port: **8080**
- SSL Certificate: **wildcard example.com**
- Domain Name: **pihole-2.example.com**
- Scheme: **http**
- Forward: **edge2**
- Port: **8080**
- SSL Certificate: **wildcard example.com**
- Domain Name: **homer.example.com**
- Scheme: **http**
- Forward: **services**
- Port: **8888**
- SSL Certificate: **wildcard example.com**

Requests for internally hosted subdomains are received by my router and sent to my **DNS resolver**. My two pi-hole instances have A records that define the domain and ip of all my servers, and CNAME records that point all internal [subdomain].example.com requests to the edge server hosting my reverse proxy. Nginx Proxy Manager determines the request address and forwards to the appropriately hosted service - using an access list that only allows such requests internally.

The only thing I need to update for my configuration was a suggested by you, @Bucking_Horn - I need to move my domain details from the environment variables and into the custom configuration file.

I do appreciate that the use of two domains (`example.com` and `example.lan`, perhaps) may help identify internally vs externally hosted services, but within NPM I can determine this if the Access is set to "Public" or "Internal".

> What's your motivation or proxying HTTP internally?

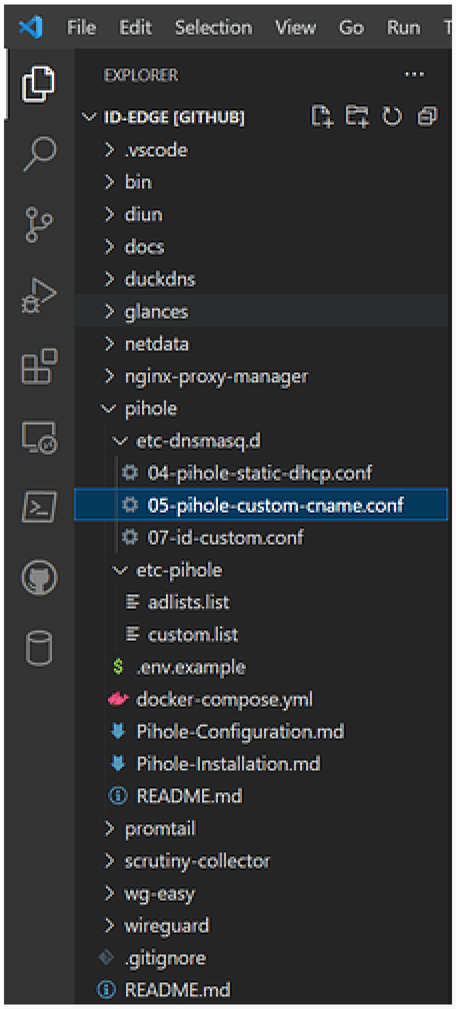
Having all of my services accessible by `[servicename].example.com` rather than having to remember `[servername]:[portnumber]`. I'm very glad I've been able to configure pi-hole - and by extension dnsmasq - to do this.

> What's the benefit of that approach over having local clients access your Pi-holes directly?

I can access my pi-hole instances with either **http://edge1:8080/admin/** or **https://pihole-1.example.com/admin/** and they both work very well. For other users in the house, any internally hosted sites are using a wildcard SSL/TLS certificate which provides encryption and a reasonable expectation of additional security.

> You should also note that you do not have to edit files manually

I absolutely love the fact that pi-hole supports the concept of configuration-as-code, as this is how I manage my docker containers across my servers. It allows me to tear down and stand up containers, and servers, very simply:



ID-EDGE [GITHUB]
- .vscode
- bin
- diun
- docs
- duckdns
- glances
- netdata
- nginx-proxy-manager
- pihole
  - etc-dnsmasq.d
    - 04-pihole-static-dhcp.conf
    - 05-pihole-custom-cname.conf
    - 07-id-custom.conf
  - etc-pihole
    - adlists.list
    - custom.list
  - .env.example
  - docker-compose.yml
  - Pihole-Configuration.md
  - Pihole-Installation.md
  - README.md
- promtail
- scrutiny-collector
- wg-easy
- wireguard
- .gitignore
- README.md

Thank you for your detailed responses!