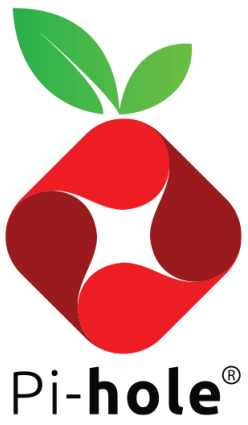
```
docker pull pihole/pihole
```

Docker Pi-hole



Quick Start

1. Copy docker-compose.yml.example to docker-compose.yml and update as needed. See example below: Docker-compose example:

```
version: "3"
# More info at https://github.com/pi-hole/docker-pi-hole/ and https://docs.pi-hole.net/
services:
  pihole:
    container_name: pihole
    image: pihole/pihole:latest
    ports:
      - "53:53/tcp"
      - "53:53/udp"
      - "67:67/udp"
      - "80:80/tcp"
    environment:
      TZ: 'America/Chicago'
      # WEBPASSWORD: 'set a secure password here or it will be random'
    # Volumes store your data between container upgrades
    volumes:
      - './etc-pihole/:/etc/pihole/'
      - './etc-dnsmasq.d/:/etc/dnsmasq.d/'
    # Recommended but not required (DHCP needs NET_ADMIN)
    #   https://github.com/pi-hole/docker-pi-hole#note-on-capabilities
    cap_add:
      - NET_ADMIN
    restart: unless-stopped
```

2. Run `docker-compose up --detach` to build and start pi-hole

Here is an equivalent docker run script.

Upgrade Notes

v5.8+

A check has been added in v5.8 to ensure that the `PIHOLE_DNS_` value is correct, a common error is that quotes are passed through to the script, usually because the environment variable has been defined as, e.g. `- PIHOLE_DNS_='10.0.0.2#5053;1.1.1.1'` . The following declarations are valid:

- `PIHOLE_DNS_: 10.0.0.2#5053;1.1.1.1`
- `PIHOLE_DNS_: '10.0.0.2#5053;1.1.1.1'`
- `- PIHOLE_DNS_=10.0.0.2#5053;1.1.1.1`

See Docker documentation for more detail, and discussion in this issue thread

Overview

Renamed from `diginc/pi-hole` to `pihole/pihole`

A Docker project to make a lightweight x86 and ARM container with Pi-hole functionality.

1) Install docker for your x86-64 system or ARMv7 system using those links. Docker-compose is also recommended. 2) Use the above quick start example, customize if desired. 3) Enjoy!

Running Pi-hole Docker

This container uses 2 popular ports, port 53 and port 80, so **may conflict with existing applications ports**. If you have no other services or docker containers using port 53/80 (if you do, keep reading below for a reverse proxy example), the minimum arguments required to run this container are in the script docker_run.sh

If you're using a Red Hat based distribution with an SELinux Enforcing policy add `:z` to line with volumes like so:

```
-v "$(pwd)/etc-pihole/:/etc/pihole/:z" \
-v "$(pwd)/etc-dnsmasq.d/:/etc/dnsmasq.d/:z" \
```

Volumes are recommended for persisting data across container re-creations for updating images. The IP lookup variables may not work for everyone, please review their values and hard code IP and IPv6 if necessary.

You can customize where to store persistent data by setting the `PIHOLE_BASE` environment variable when invoking `docker_run.sh` (e.g. `PIHOLE_BASE=/opt/pihole-storage ./docker_run.sh`). If `PIHOLE_BASE` is not set, files are stored in your current directory when you invoke the script.

Automatic Ad List Updates - since the 3.0+ release, `cron` is baked into the container and will grab the newest versions of your lists and flush your logs. **Set your TZ** environment variable to make sure the midnight log rotation syncs up with your timezone's midnight.

Running DHCP from Docker Pi-Hole

There are multiple different ways to run DHCP from within your Docker Pi-hole container but it is slightly more advanced and one size does not fit all. DHCP and Docker's multiple network modes are covered in detail on our docs site: Docker DHCP and Network Modes

Environment Variables

There are other environment variables if you want to customize various things inside the docker container:

Recommended Variables

Variable	Default	Value	Descrption
`TZ`	UTC	`<Timezone>`	Set your timezone to make sure logs rotate at local midnight instead of at UTC midnight.
`WEBPASSWORD`	random	`<Admin password>`	http://pi.hole/admin password. Run `docker logs pihole | grep random` to find your random pass.
`ServerIP`	unset	`<Host's IP>`	Set to your server's LAN IP, used by web block modes and lighttpd bind address

Optional Variables

Variable	Default	Value	Descrption
`ADMIN_EMAIL`	unset	email address	Set an administrative contact address for the Block Page
`PIHOLE_DNS_`	`8.8.8.8;8.8.4.4`	IPs delimited by `;`	Upstream DNS server(s) for Pi-hole to forward queries to, seperated by a semicolon (supports non-standard ports with `#[port number]`) e.g `127.0.0.1#5053;8.8.8.8;8.8.4.4`
`DNSSEC`	`false`	`<"true"|"false">`	Enable DNSSEC support
`DNS_BOGUS_PRIV`	`true`	`<"true"|"false">`	Enable forwarding of reverse lookups for private ranges
`DNS_FQDN_REQUIRED`	`true`	`<"true"|"false">`	Never forward non-FQDNs
`REV_SERVER`	`false`	`<"true"|"false">`	Enable DNS conditional forwarding for device name resolution
`REV_SERVER_DOMAIN`	unset	Network Domain	If conditional forwarding is enabled, set the domain of the local network router
`REV_SERVER_TARGET`	unset	Router's IP	If conditional forwarding is enabled, set the IP of the local network router
`REV_SERVER_CIDR`	unset	Reverse DNS	If conditional forwarding is enabled, set the reverse DNS zone (e.g. `192.168.0.0/24`)
`DHCP_ACTIVE`	`false`	`<"true"|"false">`	Enable DHCP server. Static DHCP leases can be configured with a custom `/etc/dnsmasq.d/04-pihole-static-dhcp.conf`
`DHCP_START`	unset	`<Start IP>`	Start of the range of IP addresses to hand out by the DHCP server (mandatory if DHCP server is enabled).
`DHCP_END`	unset	`<End IP>`	End of the range of IP addresses to hand out by the DHCP server (mandatory if DHCP server is enabled).
`DHCP_ROUTER`	unset	`<Router's IP>`	Router (gateway) IP address sent by the DHCP server (mandatory if DHCP server is enabled).
`DHCP_LEASETIME`	24	`<hours>`	DHCP lease time in hours.
`PIHOLE_DOMAIN`	`lan`	`<domain>`	Domain name sent by the DHCP server.
`DHCP_IPv6`	`false`	`<"true"|"false">`	Enable DHCP server IPv6 support (SLAAC + RA).
`DHCP_rapid_commit`	`false`	`<"true"|"false">`	Enable DHCPv4 rapid commit (fast address assignment).
`VIRTUAL_HOST`	`$ServerIP`	`<Custom Hostname>`	What your web server 'virtual host' is, accessing admin through this Hostname/IP allows you to make changes to the whitelist / blacklists in addition to the default 'http://pi.hole/admin/' address
`IPv6:`	`true`	`<"true"|"false">`	For unraid compatibility, strips out all the IPv6 configuration from DNS/Web services when false.
`TEMPERATUREUNIT`	`c`	`<c|k|f>`	Set preferred temperature unit to `c` : Celsius, `k` : Kelvin, or `f` Fahrenheit units.
`WEBUIBOXEDLAYOUT`	`boxed`	`<boxed|traditional>`	Use boxed layout (helpful when working on large screens)
`QUERY_LOGGING`	`true`	`<"true"|"false">`	Enable query logging or not.

Advanced Variables

Variable	Default	Value	Descrption
`ServerIPv6`	unset	`<Host's IPv6>`	**If you have a v6 network** set to your server's LAN IPv6 to block IPv6 ads fully
`INTERFACE`	unset	`<NIC>`	The default works fine with our basic example docker run commands. If you're trying to use DHCP with `--net host` mode then you may have to customize this or DNSMASQ_LISTENING.
`DNSMASQ_LISTENING`	unset	`<local|all|single>`	`local` listens on all local subnets, `all` permits listening on internet origin subnets in addition to local, `single` listens only on the interface specified.

Variable	Default	Value	Descrption
`WEB_PORT`	unset	`<PORT>`	**This will break the 'webpage blocked' functionality of Pi-hole** however it may help advanced setups like those running synology or `--net=host` docker argument. This guide explains how to restore webpage blocked functionality using a linux router DNAT rule: Alternative Synology installation method
`SKIPGRAVITYONBOOT`	unset	`<unset|1>`	Use this option to skip updating the Gravity Database when booting up the container. By default this environment variable is not set so the Gravity Database will be updated when the container starts up. Setting this environment variable to 1 (or anything) will cause the Gravity Database to not be updated when container starts up.

Experimental Variables

Variable	Default	Value	Descrption
`DNSMASQ_USER`	unset	`<pihole|root>`	Allows running FTLDNS as non-root.

Deprecated environment variables:

While these may still work, they are likely to be removed in a future version. Where applicible, alternative variable names are indicated. Please review the table above for usage of the alternative variables

Docker Environment Var.	Description	Replaced By
`CONDITIONAL_FORWARDING`	Enable DNS conditional forwarding for device name resolution	`REV_SERVER`
`CONDITIONAL_FORWARDING_IP`	If conditional forwarding is enabled, set the IP of the local network router	`REV_SERVER_TARGET`
`CONDITIONAL_FORWARDING_DOMAIN`	If conditional forwarding is enabled, set the domain of the local network router	`REV_SERVER_DOMAIN`
`CONDITIONAL_FORWARDING_REVERSE`	If conditional forwarding is enabled, set the reverse DNS of the local network router (e.g. `0.168.192.in-addr.arpa`)	`REV_SERVER_CIDR`
`DNS1`	Primary upstream DNS provider, default is google DNS	`PIHOLE_DNS_`
`DNS2`	Secondary upstream DNS provider, default is google DNS, `no` if only one DNS should used	`PIHOLE_DNS_`

To use these env vars in docker run format style them like: `-e DNS1=1.1.1.1`

Here is a rundown of other arguments for your docker-compose / docker run.

Docker Arguments	Description
`-p <port>:<port>` **Recommended**	Ports to expose (53, 80, 67, 443), the bare minimum ports required for Pi-holes HTTP and DNS services
`--restart=unless-stopped` **Recommended**	Automatically (re)start your Pi-hole on boot or in the event of a crash
`-v $(pwd)/etc-pihole:/etc/pihole` **Recommended**	Volumes for your Pi-hole configs help persist changes across docker image updates
`-v $(pwd)/etc-dnsmasq.d:/etc/dnsmasq.d` **Recommended**	Volumes for your dnsmasq configs help persist changes across docker image updates
`--net=host` *Optional*	Alternative to `-p <port>:<port>` arguments (Cannot be used at same time as -p) if you don't run any other web application. DHCP runs best with --net=host, otherwise your router must support dhcp-relay settings.
`--cap-add=NET_ADMIN` *Recommended*	Commonly added capability for DHCP, see Note on Capabilities below for other capabilities.
`--dns=127.0.0.1` *Optional*	Sets your container's resolve settings to localhost so it can resolve DHCP hostnames from Pi-hole's DNSMasq, may fix resolution errors on container restart.
`--dns=1.1.1.1` *Optional*	Sets a backup server of your choosing in case DNSMasq has problems starting
`--env-file .env` *Optional*	File to store environment variables for docker replacing `-e key=value` settings. Here for convenience

Tips and Tricks

- A good way to test things are working right is by loading this page: http://pi.hole/admin/
- How do I set or reset the Web interface Password?
    - `docker exec -it pihole_container_name pihole -a -p` - then enter your password into the prompt
- Port conflicts? Stop your server's existing DNS / Web services.
    - Don't forget to stop your services from auto-starting again after you reboot
    - Ubuntu users see below for more detailed information
- You can map other ports to Pi-hole port 80 using docker's port forwarding like this `-p 8080:80` if you are using the default blocking mode. If you are using the legacy IP blocking mode, you should not remap this port.
    - Here is an example of running with jwilder/proxy (an nginx auto-configuring docker reverse proxy for docker) on my port 80 with Pi-hole on another port. Pi-hole needs to be `DEFAULT_HOST` env in jwilder/proxy and you need to set the matching `VIRTUAL_HOST` for the Pi-hole's container. Please read jwilder/proxy readme for more info if you have trouble.

Installing on Ubuntu

Modern releases of Ubuntu (17.10+) include `systemd-resolved` which is configured by default to implement a caching DNS stub resolver. This will prevent pi-hole from listening on port 53. The stub resolver should be disabled with: `sudo sed -r -i.orig 's/#?DNSStubListener=yes/DNSStubListener=no/g' /etc/systemd/resolved.conf`

This will not change the nameserver settings, which point to the stub resolver thus preventing DNS resolution. Change the `/etc/resolv.conf` symlink to point to `/run/systemd/resolve/resolv.conf`, which is automatically updated to follow the system's `netplan`: `sudo sh -c 'rm /etc/resolv.conf && ln -s /run/systemd/resolve/resolv.conf /etc/resolv.conf'` After making these changes, you should restart systemd-resolved using `systemctl restart systemd-resolved`

Once pi-hole is installed, you'll want to configure your clients to use it (see here). If you used the symlink above, your docker host will either use whatever is served by DHCP, or whatever static setting you've configured. If you want to explicitly set your docker host's nameservers you can edit the netplan(s) found at `/etc/netplan`, then run `sudo netplan apply`. Example netplan:

```
network:
    ethernets:
        ens160:
            dhcp4: true
            dhcp4-overrides:
                use-dns: false
            nameservers:
                addresses: [127.0.0.1]
    version: 2
```

Note that it is also possible to disable `systemd-resolved` entirely. However, this can cause problems with name resolution in vpns (see bug report). It also disables the functionality of netplan since systemd-resolved is used as the default renderer (see `man netplan`). If you choose to disable the service, you will need to manually set the nameservers, for example by creating a new `/etc/resolv.conf`.

Users of older Ubuntu releases (circa 17.04) will need to disable dnsmasq.

Docker tags and versioning

The primary docker tags / versions are explained in the following table. Click here to see the full list of tags, I also try to tag with the specific version of Pi-hole Core for version archival purposes, the web version that comes with the core releases should be in the GitHub Release notes.

tag	architecture	description	Dockerfile
`latest`	auto detect	x86, arm, or arm64 container, docker auto detects your architecture.	Dockerfile
`v5.0`	auto detect	Versioned tags, if you want to pin against a specific Pi-hole version, use one of these	
`v5.0-buster`	auto detect	Versioned tags, if you want to pin against a specific Pi-hole and Debian version, use one of these	
`v5.0-<arch>-buster`	based on tag	Specific architectures and Debian version tags	
`dev`	auto detect	like latest tag, but for the development branch (pushed occasionally)	

`pihole/pihole:latest`

This version of the docker aims to be as close to a standard Pi-hole installation by using the recommended base OS and the exact configs and scripts (minimally modified to get them working). This enables fast updating when an update comes from Pi-hole.

https://hub.docker.com/r/pihole/pihole/tags/

Upgrading, Persistence, and Customizations

The standard Pi-hole customization abilities apply to this docker, but with docker twists such as using docker volume mounts to map host stored file configurations over the container defaults. Volumes are also important to persist the configuration in case you have removed the Pi-hole container which is a typical docker upgrade pattern.

Upgrading / Reconfiguring

Do not attempt to upgrade (`pihole -up`) or reconfigure (`pihole -r`). New images will be released for upgrades, upgrading by replacing your old container with a fresh upgraded image is the 'docker way'. Long-living docker containers are not the docker way since they aim to be portable and reproducible, why not re-create them often! Just to prove you can.

0. Read the release notes for both this Docker release and the Pi-hole release
    - This will help you avoid common problems due to any known issues with upgrading or newly required arguments or variables
    - We will try to put common break/fixes at the top of this readme too
1. Download the latest version of the image: `docker pull pihole/pihole`
2. Throw away your container: `docker rm -f pihole`
    - **Warning** When removing your pihole container you may be stuck without DNS until step 3; **docker pull** before **docker rm -f** to avoid DNS interruption **OR** always have a fallback DNS server configured in DHCP to avoid this problem altogether.
    - If you care about your data (logs/customizations), make sure you have it volume-mapped or it will be deleted in this step.
3. Start your container with the newer base image: `docker run <args> pihole/pihole` (`<args>` being your preferred run volumes and env vars)

Why is this style of upgrading good? A couple reasons: Everyone is starting from the same base image which has been tested to known it works. No worrying about upgrading from A to B, B to C, or A to C is required when rolling out updates, it reduces complexity, and simply allows a 'fresh start' every time while preserving customizations with volumes. Basically I'm encouraging phoenix server principles for your containers.

To reconfigure Pi-hole you'll either need to use an existing container environment variables or if there is no a variable for what you need, use the web UI or CLI commands.

Pi-hole features

Here are some relevant wiki pages from Pi-hole's documentation. The web interface or command line tools can be used to implement changes to pihole.

We install all pihole utilities so the the built in pihole commands will work via `docker exec <container> <command>` like so:

- `docker exec pihole_container_name pihole updateGravity`
- `docker exec pihole_container_name pihole -w spclient.wg.spotify.com`
- `docker exec pihole_container_name pihole -wild example.com`

Customizations

The webserver and DNS service inside the container can be customized if necessary. Any configuration files you volume mount into `/etc/dnsmasq.d/` will be loaded by dnsmasq when the container starts or restarts or if you need to modify the Pi-hole config it is located at `/etc/dnsmasq.d/01-pihole.conf`. The docker start scripts runs a config test prior to starting so it will tell you about any errors in the docker log.

Similarly for the webserver you can customize configs in /etc/lighttpd

Systemd init script

As long as your docker system service auto starts on boot and you run your container with `--restart=unless-stopped` your container should always start on boot and restart on crashes. If you prefer to have your docker container run as a systemd service instead, add the file pihole.service to "/etc/systemd/system"; customize whatever your container name is and remove `--restart=unless-stopped` from your docker run. Then after you have initially created the docker container using the docker run command above, you can control it with "systemctl start pihole" or "systemctl stop pihole" (instead of `docker start` / `docker stop`). You can also enable it to auto-start on boot with "systemctl enable pihole" (as opposed to `--restart=unless-stopped` and making sure docker service auto-starts on boot).

NOTE: After initial run you may need to manually stop the docker container with "docker stop pihole" before the systemctl can start controlling the container.

Note on Capabilities

DNSMasq / FTLDNS expects to have the following capabilities available:

- `CAP_NET_BIND_SERVICE` : Allows FTLDNS binding to TCP/UDP sockets below 1024 (specifically DNS service on port 53)
- `CAP_NET_RAW` : use raw and packet sockets (needed for handling DHCPv6 requests, and verifying that an IP is not in use before leasing it)
- `CAP_NET_ADMIN` : modify routing tables and other network-related operations (in particular inserting an entry in the neighbor table to answer DHCP requests using unicast packets)

This image automatically grants those capabilities, if available, to the FTLDNS process, even when run as non-root.
By default, docker does not include the `NET_ADMIN` capability for non-privileged containers, and it is recommended to explicitly add it to the container using `--cap-add=NET_ADMIN`.
However, if DHCP and IPv6 Router Advertisements are not in use, it should be safe to skip it. For the most paranoid, it should even be possible to explicitly drop the `NET_RAW` capability to prevent FTLDNS from automatically gaining it.

User Feedback

Please report issues on the GitHub project when you suspect something docker related. Pi-hole or general docker questions are best answered on our user forums. Ping me (@diginc) on the forums if it's a docker container and you're not sure if it's docker related.