
Internet Pi



A Raspberry Pi Configuration for Internet connectivity

I have had a couple Pis doing random Internet-related duties for years. It's finally time to formalize their configs and make all the DNS/ad-blocking/monitoring stuff encapsulated into one Ansible project.

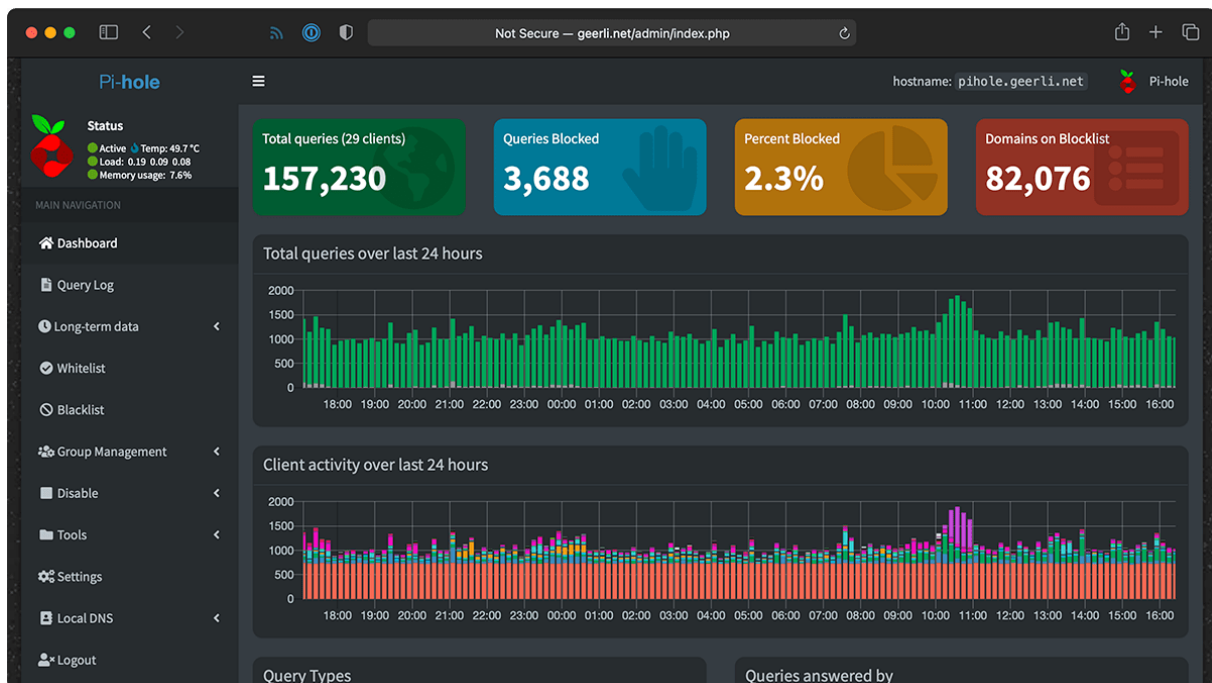
So that's what this is.

Features

Internet Monitoring: Installs Prometheus and Grafana, along with a few Docker containers to monitor your Internet connection with Speedtest.net speedtests and HTTP tests so you can see uptime, ping stats, and speedtest results over time.



Pi-hole: Installs the Pi-hole Docker configuration so you can use Pi-hole for network-wide ad-blocking and local DNS. Make sure to update your network router config to direct all DNS queries through your Raspberry Pi if you want to use Pi-hole effectively!



Other features:

- **Shelly Plug Monitoring:** Installs a `shelly-plug-prometheus` exporter and a Grafana dashboard, which tracks and displays power usage on a Shelly Plug running on the local network. (Disabled by default. Enable and configure using the `shelly_plug_*` vars in `config.yml`.)
- **AirGradient Monitoring:** Configures `airgradient-prometheus` and a Grafana dashboard, which tracks and displays air quality over time via one or more AirGradient DIY monitors. (Disabled by default. Enable and configure using the `airgradient_enable` var in `config.yml`. See example configuration for ability to monitor multiple AirGradient DIY stations.)
- **Starlink Monitoring:** Installs a `starlink` prometheus exporter and a Grafana dashboard, which tracks and displays Starlink statistics. (Disabled by default. Enable and configure using the `starlink_enable` var in `config.yml`.)

IMPORTANT NOTE: If you use the included Internet monitoring, it will download a decently-large amount of data through your Internet connection on a daily basis. Don't use it, or tune the `internet-monitoring` setup to not run the speedtests as often, if you have a metered connection!

Recommended Pi and OS

You should use a Raspberry Pi 4 model B or better. The Pi 4 and later generations of Pi include a full gigabit network interface and enough I/O to reliably measure fast Internet connections.

Older Pis work, but have many limitations, like a slower CPU and sometimes very-slow NICs that limit the speed test capability to 100 Mbps or 300 Mbps on the Pi 3 model B+.

Other computers and VMs may run this configuration as well, but it is only regularly tested on a Raspberry Pi.

The configuration is tested against Raspberry Pi OS, both 64-bit and 32-bit, and runs great on that or a generic Debian installation.

It should also work with Ubuntu for Pi, or Arch Linux, but has not been tested on other operating systems.

Setup

1. Install Ansible. The easiest way (especially on Pi or a Debian system) is via Pip:
 1. (If on Pi/Debian): `sudo apt-get install -y python3-pip`
 2. (Everywhere): `pip3 install ansible`
 3. If you get an error like “externally-managed-environment”, follow this guide to fix it, then run `pip3 install ansible` again.
 4. Make sure Ansible is in your PATH: `export PATH=$PATH:~/.local/bin` (and consider adding it permanently).
2. Clone this repository: `git clone https://github.com/geerlingguy/internet-pi.git`, then enter the repository directory: `cd internet-pi`.
3. Install requirements: `ansible-galaxy collection install -r requirements.yml` (if you see `ansible-galaxy: command not found`, restart your SSH session or reboot the Pi and try again)
4. Make copies of the following files and customize them to your liking:
 - `example.inventory.ini` to `inventory.ini` (replace IP address with your Pi’s IP, or comment that line and uncomment the `connection=local` line if you’re running it on the Pi you’re setting up).
 - `example.config.yml` to `config.yml`
5. Run the playbook: `ansible-playbook main.yml`

If running locally on the Pi: You may encounter an error like “Error while fetching server API version” or “connect: permission denied”. If you do, please either reboot or log out and log back in, then run the playbook again.

Usage

Pi-hole

Visit the Pi's IP address (e.g. `http://192.168.1.10/admin`) and use the `pihole_password` you configured in your `config.yml` file. An existing pi-hole installation can be left unaltered by disabling the setup of this project's installation in your `config.yml` (`pihole_enable: false`)

Grafana

Visit the Pi's IP address with port 3030 (e.g. `http://192.168.1.10:3030/`), and log in with username `admin` and the password `monitoring_grafana_admin_password` you configured in your `config.yml`.

To find the dashboard, navigate to Dashboards, click Browse, then go to the Internet connection dashboard. If you star this dashboard, it will appear on the Grafana home page.

Note: The `monitoring_grafana_admin_password` is only used the first time Grafana starts up; if you need to change it later, do it via Grafana's admin UI.

Prometheus

A number of default Prometheus job configurations are included out of the box, but if you would like to add more to the `prometheus.yml` file, you can add a block of text that will be added to the end of the `scrape_configs` using the `prometheus_extra_scrape_configs` variable, for example:

```
1 prometheus_extra_scrape_configs: |
2   - job_name: 'customjob'
3     scrape_interval: 5s
4     static_configs:
5       - targets: ['192.168.1.1:9100']
```

You can also add more targets to monitor via the node exporter dashboard, say if you have a number of servers or other Pis you want to monitor on this instance. Just add them to the list, after the `nodeexp:9100` entry for the main Pi:

```
1 prometheus_node_exporter_targets:
2   - 'nodeexp:9100'
3   # Add more targets here
4   - 'another-server.local:9100'
```

Updating

pi-hole

To upgrade Pi-hole to the latest version, run the following commands:

```
1 cd ~/pi-hole #
2 docker compose pull          # pulls the latest images
3 docker compose up -d --no-deps # restarts containers with newer images
4 docker system prune --all     # deletes unused images
```

Configurations and internet-monitoring images

Upgrades for the other configurations are similar (go into the directory, and run the same `docker compose` commands. Make sure to `cd` into the `config_dir` that you use in your `config.yml` file.

Alternatively, you may update the initial `config.yml` in the the repo folder and re-run the main playbook: `ansible-playbook main.yml`. At some point in the future, a dedicated upgrade playbook may be added, but for now, upgrades may be performed manually as shown above.

Backups

A guide for backing up the configurations and historical data will be posted here as part of Issue #194: Create Backup guide.

Uninstall

To remove `internet-pi` from your system, run the following commands (assuming the default install location of `~`, your home directory):

```
1 # Enter the internet-monitoring directory.
2 cd ~/internet-monitoring
3
4 # Shut down internet-monitoring containers and delete data volumes.
5 docker compose down -v
6
7 # Enter the pi-hole directory.
8 cd ~/pi-hole
9
10 # Shutdown pi-hole containers and delete data volumes.
11 docker compose down -v
12
```

```
13 # Delete all the unused container images, volumes, etc. from the system
14 docker system prune -af
```

Do the same thing for any of the other optional directories added by this project (e.g. [shelly-plugin-prometheus](#), [starlink-exporter](#), etc.).

You can then delete the [internet-monitoring](#), [pi-hole](#), etc. folders and everything will be gone from your system.

License

MIT

Author

This project was created in 2021 by Jeff Geerling.