

Building a Privacy Box with a Raspberry Pi

Privacy for you Internet access plus a monitor for your devices, a Wi-Fi/LAN intruder detector and a VPN Server for remote access with a Raspberry Pi + bonus track: password manager

Daniel Alomar

daniel.alomar@pm.me

20230103



Building a Privacy Box with a Raspberry Pi © 2022 by Daniel Alomar is licensed under CC BY-NC-SA 4.0. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/4.0/>

The latest version of this guide can be found at <https://coneixement.info/blog/building-a-privacy-box-with-a-raspberry-pi/>

Changelog

Data	Changes
03/01/23	Added Netdata as a monitoring solution recommended instead RPi-monitor

Index

Introduction.....	7
Background.....	7
What is a Privacy Box.....	8
Objective.....	9
Requirements.....	10
Setup of the Privacy Box.....	11
Why I choose Debian instead Raspbian or RaspberryPi OS.....	11
Identifying the device.....	11
Download and flash an image.....	12
Configuring remote access with SSH.....	13
Enable SSH on Raspberry Pi in headless mode without keys (easy way).....	13
Pre-configuration and enabling ssh remote connection using SSH key.....	13
Basic configurations.....	15
Setting the host name.....	15
Updating the system.....	16
Adding user to sudo and video groups.....	16
Lock down the SSH service.....	17
Setting the time zone.....	19
Installing unattended upgrades package (recommended).....	19
Installing Fail2Ban (optional).....	21
Installing a firewall (optional).....	22
Install the packages.....	22
Configuring the firewall.....	22
Enabling the firewall.....	22
Installing log2ram to expand SSD life (recommended).....	23
Configuring log2ram.....	23
Installing a DDNS service (in case you do not have a static IP address).....	25
Installing and configuring Pi-hole.....	26
Tweaking Pi-hole.....	33
Installing Unbound.....	34
Enhancing Pi-hole Security (optional if we are not fine with Cloudflare DNS).....	34
Creating a configuration for Pi-hole.....	34
Testing Unbound.....	37
Keeping unbound updated.....	38
Keeping updated Pi-hole (optional).....	39
Configuring Pi-hole to use unbound.....	39

Improving our Blocklists (Blacklist, Adlist and Whitelist).....	41
Removing existing blocklist.....	46
Backup Pi-hole configuration.....	46
Additional functionalities.....	46
Installing WireGuard (light, secure and fast VPN).....	47
Introduction.....	47
Configuring repositories.....	47
Installing applications.....	48
Set Up and Configuring the WireGuard VPN Server.....	48
Generating security keys.....	48
Generating server configuration.....	49
Gotchas.....	50
Enabling IP Forwarding on the Server.....	50
Securing access to sensitive files.....	51
Set Up the WireGuard Client for each client.....	52
Phone access.....	52
Laptop access.....	52
Set Up the WireGuard Server.....	54
Adding unattended upgrades (optional).....	55
Installing Pi.Alert, a Wi-Fi/LAN intruder detector (optional).....	57
Monitoring tool (optional).....	62
Installing RPi-Monitor (deprecated).....	62
Installing Netdata (recommended).....	64
Grafana.....	67
Securing the Raspberry.....	68
Backup & restore.....	69
Method 1: Copy the SD Card Image.....	69
Method 2: Zip the Home Directory.....	69
Method 3: Scheduled Backups.....	69
Bonus track. Password manager: Vaultwarden.....	70
How to update components.....	70
Portainer.....	70
Containers.....	71
Updating through portainer.....	71
Updating command line.....	71
Updating Nginx.....	72
Bibliography.....	73
Base.....	73
Pi-hole.....	73

Unbound.....73

Fail2ban.....74

Unattended-Upgrades.....74

Pi.Alert.....74

WireGuard.....74

Monitoring tools.....75

Backup.....75

Vaultwarden.....75

Others.....75

Introduction

Background

In these uncertainty and strange times we are living - it seems to be - forever, we **should** take care of our privacy. The saying "if you are not paying for it, then you are the product"^{1,2} it is absolutely true.

We have not been trained at school in the use of Information and Communications Technologies (ICT). Most of us are not digital natives, and even if we were, the evolution of technology is faster than the speed of adaptation of training. Those of us who are lucky enough to like technology and to be early adapters of it, we have 'somewhat less difficult', like an intuition, to manage ICT, but that does not mean we are safe from the dangers of exposing our lives to the Internet, where different companies are on the lookout to collect our data in order to create the most reliable profile they can from us, a detailed description of our tastes, our preferences, our habits... they know us much better than we know ourselves, and that is not a cliché, **it is a reality**.

This data is collected by companies called [Data Brokers](#)³ through multiple techniques. The data is used to create profile of us to different purposes, like marketing and advertising, risk-mitigation, people-search services^{4,5}, etc.... Regarding marketing and advertising, while this could sound nice for someone who wants to have personalizing advertisements, we have to know some side effect, such as the price not being the same for everyone, and other worse effects. Since companies have a wide information from us, better than us, they will know also our risks and the potential collateral effects from our habits. All this information let us calculate, with a high accuracy, prices for the services or products we want to acquired or advertisements we see. For example, the price for a health insurance will not the same if the company knows, through the information from us acquired from the Data Brokers. The same apply to the advertisements we see in our devices. Here you have an example from Signal: <https://signal.org/blog/the-instagram-ads-you-will-never-see/>

It is common to hear someone say: "I do not care about privacy concerns. I do not have nothing to hide"... The quick answer you could provide, with a smile on your face: "If you

do not have nothing to hide, then you can give me your email password...". For sure he/she will not, so EVERYONE have some information to protect.

What is a Privacy Box

What I call Privacy Box is a device that will provide privacy to your Internet access. Various tools are installed and configured within this device. It will block advertisements, through the pi-hole application, to all your network and for all kind of devices without to have to install software on each one. With unbound we are going to have a validating, recursive, and caching DNS resolver locally. This means that our Internet Service Providers (ISP) will not see what we are searching for, the Domain Name Resolution will be locale, and even faster, within our Raspberry Pi. Along with the previous tools we will also add WireGuard, a VPN Server that will allow us a remote and secure connection to our network and continue provide privacy when we are outside our local network.

All this functionalities are going to be installed inside a Raspberry Pi device in order to have it running 24x7 at low cost.



Figure 1: Raspberry Pi 3 Model B

The main characteristics of the privacy box are:

- Blocking unwanted contend to all devices connected, without installing any client-side software (Pi-hole)
- Network-wide protection (Pi-hole)
- Improve network performance (Pi-hole)
- Secure open-source recursive DNS server for local resolution (unbound)
- Network intrude detector (Pi.Alert)

- Device monitoring (RPI-Monitor)
- Secure and remote access through a VPN (WireGuard)

As a bonus track I have added a tutorial to setup a raspberry pi to host a password manager and how to access it from Internet. I have choose **Vaultwarden**, based on the well know solution **Bitwarden**. Due to technical reasons, it is easier to run this server on another raspberry pi. You can try to setup on the same device where you have Pi-hole, but I will not recommend you.

If you do not have enough technological knowledge to follow this guide and build this Privacy Box for yourself, ask a friend with more knowledge in technology (maybe a [geek](#)) to help you to setup it. Those of us who are techies love to help others. My will has been to create a very simple tutorial, with step-by-step instructions, explaining the reason for each step so that we can understand what we are doing.

Before starting, I will like to thanks Mr.Smashy ([@THESMASHY](#)) who wrote a guide⁶, origin and source of inspiration of this one.

Note 1: Commands are identified using a different text style framed within a grey box like this:

```
$ls -l
```

The command will start with a \$ or # symbol. That means the command is executed without administrator privileges (\$ symbol) or with administrator privileges (# symbol). To elevate privileges (from \$ to #) we must run 'sudo -s' command or start the command with sudo. In both cases you will need the administrator password. Examples:

```
$sudo -s  
#
```

Running ls command with elevated privileges

```
$sudo ls -l
```

Objective

As I mention in the previous section, I am going to show also how to set up a password manager. This tool it will installed on another Raspberry Pi device.

Requirements

This is the list of requirements:

- Raspberry Pi 3 Model B (or higher)
- Computer to write the image and connect to the Raspberry to configure it
- SD Card or USB memory stick (capacity: 16 GB or higher)
- Internet connection
- Basic knowledge of computers or have a geek friend on hand
- Power supply
- Curiosity
- Time

Note 1: Regarding the computer, I have used my laptop with GNU/Linux (Manjaro flavour) to write the image and connect to the Raspberry, so the commands you will see belongs to the GNU/Linux operating system. The remaining instructions are independent of the operating system you use. If you are a Windows or Mac user, you will find several alternatives easily on Internet to write the image to the SD Card or USB stick and connect to the Raspberry.

Note 2: Whether to choose SD Card or USB memory stick? People says the lifetime of a SD Card is shorter than the USB memory stick, so many people boot from SD Card and use the USB memory stick (or even a SSD hard drive) to run the operating system. From Raspberry Pi 3 and up the operating system can be booted and run directly from the USB. In this manual I have added some tools to decrease the write cycles to the disk using the RAM memory. Which one to choose? There is not a right answer.. well, yes..

MAKE REGULAR BACKUPS to ensure you have a plan B for any incident related the device.

Setup of the Privacy Box

This section will explain how to setup Pi-hole and unbound plus another optional and recommended tools.

Why I choose Debian instead Raspbian or RaspberryPi OS

There are several reasons why I choose a Debian image instead Raspbian or RaspberryPi OS. The main reason is freedom. Debian is a full GNU/Linux flavour, with no commercial nor proprietary software.

A second reason is the incident related with the internal repository files modification without notification Raspbian did in February 2021^{7,8,9}. A Microsoft repository pointing to a Microsoft server was added secretly without any notification. The reason was to provide Visual Studio Code for some scenarios. This modification without consent crossed the boundaries of (my) trust and made me decide to move to a full open source distribution like Debian. If they changed this without notification, what could they do next time?

Identifying the device

First we have to check which raspberry model we have. If using visual inspection we are not sure, we can do a «logical inspection», asking the device through a command.

Run the command below will be the first option. The output will be the Raspberry Pi Model

```
$cat /proc/device-tree/model
```

```
daniel@Anuk:~$ cat /proc/device-tree/model  
Raspberry Pi 3 Model B Plus Rev 1.3daniel@Anuk:~$
```

In case we have installed Raspbian as operating system, we can run the following command to check the model. It will return us information from our device:


```
$rev=$(awk '/^Revision/ { print $3 }' /proc/cpuinfo) && curl -L  
perturb.org/rpi?rev=$rev
```

```
daniel@anuk:~$ rev=$(awk '/^Revision/ { print $3 }' /proc/cpuinfo) && curl -L perturb.org/rpi?rev=$rev
Revision : a020d3
Model    : 3 Model B+
Memory   : 1 GB
Overvolt  : No
Released  : Q1 2018
Notes    : (Mfg by Sony)
```

Download and flash an image

Debian image for raspberry pi can be downloaded from: <https://raspi.debian.net/>. For a production environment I would recommend to use a Tested image

Choose the (xz-compressed) image according to the hardware you have

<div>  Home Download tested Images Daily auto-built images Defaults and Settings FAQ About Debian </div>						
Tested images						
Build date	Release	Family	Tested hardware	File links	Tested OK	Failed tests
2021.08.23	11 (Bullseye)	4	p400	xz-compressed image <small>349.93 MB, 2021-08-23 12:48:0500</small> sha256sum GPG-signed sha256sum	Boots OK, HDMI, built-in keyboard	In order to use serial console, set <code>GPU_FREQ=360</code> in <code>/etc/defaults/raspi-firmware</code> and run <code>dpkg-reconfigure raspi-firmware</code> . Wireless interface not detected.
2021.08.23	11 (Bullseye)	4	4 (4GB)	xz-compressed image <small>349.93 MB, 2021-08-23 12:48:0500</small> sha256sum GPG-signed sha256sum	Boots OK, HDMI, USB keyboard, wireless networking.	In order to use serial console, set <code>GPU_FREQ=360</code> in <code>/etc/defaults/raspi-firmware</code> and run <code>dpkg-reconfigure raspi-firmware</code> .
2021.08.23	11 (Bullseye)	3	3B+	xz-compressed image <small>353.56 MB, 2021-08-23 12:47:0500</small> sha256sum GPG-signed sha256sum	Boots OK, serial console, HDMI, USB keyboard, wireless networking	
2021.8.23	11 (Bullseye)	2	2B	xz-compressed image <small>300.24 MB, 2021-08-23 12:47:0500</small> sha256sum GPG-signed sha256sum	Boots OK, serial console, HDMI, USB keyboard	
2021.08.23	11 (Bullseye)	0/1	1B (512MB)	xz-compressed image <small>232.05 MB, 2021-08-23 12:47:0500</small> sha256sum GPG-signed sha256sum	Boots OK, serial console, HDMI, USB keyboard	
2021.08.23	11 (Bullseye)	0/1	0W	xz-compressed image <small>232.05 MB, 2021-08-23 12:47:0500</small> sha256sum GPG-signed sha256sum	Boots OK, HDMI, USB keyboard, wireless networking	

Locate where the file was downloaded and open a console session into that folder. Decompress the image downloaded using the following unxz command:

```
$unxz 20210823_raspi_3_bullseye.img.xz
```

You will get a img file. In my case 20210823_raspi_3_bullseye.img

Plug a SD card or USB memory stick into your laptop and flash the image to the device with the following command (sdb is how the SD Card or USB has been identified. Check the partition in your case)

```
$sudo dd bs=4M if=20210823_raspi_3_bullseye.img of=/dev/sdb conv=fdatasync status=progress
```

If you have choose to boot and run the operating system from the USB stick, you have to configure the device in order to boot from USB. Setting the boot from USB can be found at Raspberry site¹⁰

If you have a RaspberryPi 2 version 1.1 or lower you can only boot from SD but then you switch to the USB.

Configuring remote access with SSH

The secure way to connect to your Raspberry Pi is through a SSH connection. This can be done in two ways:

1. Using a login and password we set. This will let connect to the device anyone who knows the user and password from anyplace
2. Using a SSH key, which are more secure. Our public key is stored on the remote machine and a private key is stored on our machine. The two SSH keys are required to make a secure connection

In this guide I will show you how to set the second one.

Enable SSH on Raspberry Pi in headless mode without keys (easy way)

First we have to enable the SSH connection, disabled by default for security reasons.

1. Turn off the device and remove the card or USB stick
2. Put the microSD card in the card reader or USB stick into the computer
3. Create an empty file inside boot partition called SSH

Pre-configuration and enabling ssh remote connection using SSH key

We are going to generate ssh keys on our computer and copy the public key inside sysconf.txt (raspi-firmware partition)

```
$ssh-keygen -t rsa
```

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/daniel/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/daniel/.ssh/id_rsa
Your public key has been saved in /home/daniel/.ssh/id_rsa.pub
The key fingerprint is:
SHA256: [REDACTED] DltM daniel@xiuxiueig
The key's randomart image is:
+----[RSA 3072]-----+
|
|  .
| o .
| o o
| = S . +
| + +ooo ..BE
| [REDACTED]
| . + ==+ .* +O.+
| [REDACTED] =
+-----[SHA256]-----+
```

Edit `sysconf.txt`, uncomment the `"root_authorized_key"` entry and paste the public key generated in previous step (located at `id_rsa.pub` file). We can also modify the hostname of the Raspberry (I have chosen Anuk)

```
# root_pw - Set a password for the root user (by default, it allows
# for a passwordless login)
#root_pw=FooBar

# root Authorized key - Set an authorized key for a root ssh login
root_authorized_key=ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQDhQTSaa0rTBjRA/AwbQ/+FJ8fpm/
0xP38M57-A0000AD-4786-1A75F5E
i
p
0
X
U
8iX3J2ehndpR0u0v7Rm7Y4d8pL7And702eLatt70CLLhND77q7t290p7p7BEAw027/
QcpWDtxf4SyushLoAr8KPaIXpwdhAya/YBQRk= daniel@xiuxiueig

# hostname - Set the system hostname.
Anuk
```

Go to the RASPIROOT partition (has this name) and set an static IP address modifying the eth0 file located at the following path: `/etc/network/interfaces.d/eth0`. Set the IP address you want and the IP of the router gateway (192.168.1.1 in my case). As a DNS we are going to set the cloudflare one (1.1.1.1):

```
iface eth0 inet static
    address 192.168.1.10
    netmask 255.255.255.0
    gateway 192.168.1.1
    dns-nameservers 1.1.1.1
```

Create the file `/etc/resolv.conf` with the content:

```
nameserver 1.1.1.1
```

Put the SD Card or USB stick back to the Raspberry Pi and boot it.

Now we can try to connect to the Raspberry using the username `root` and IP we set.

```
$ssh root@192.168.1.10
```

```
Enter passphrase for key '/home/daniel/.ssh/id_rsa':
Linux Anuk 5.10.0-8-arm64 #1 SMP Debian 5.10.46-4 (2021-08-03) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Sep  4 16:44:33 2021 from 192.168.1.83
daniel@Anuk:~$
```

Basic configurations

Setting the host name

Set the hostname at the file `/etc/hostname`

```
#nano /etc/hostname
```

and add a hostname entry to the hosts file (Anuk in my case)

```
#nano /etc/hosts
```

```
127.0.0.1    localhost
127.0.0.1    Anuk
#1          localhost:ip6 localhost:ip6 localhost
```

In case you are using Raspbian, you can set hostname through '*raspi-config*' application

Updating the system

Let's going to update the system to grab the latest uptades

```
#apt update && apt-get upgrade -y
```

Install some additional software stuff we will need

```
#apt install sudo dnstools gnupg wget curl git
```

Add a non root user and set a password for it

```
#adduser daniel
```

```
root@Anuk:~# adduser daniel
Adding user `daniel' ...
Adding new group `daniel' (1000) ...
Adding new user `daniel' (1000) with group `daniel' ...
Creating home directory `/home/daniel' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for daniel
Enter the new value, or press ENTER for the default
  Full Name []:
  Room Number []:
  Work Phone []:
  Home Phone []:
  Other []:
Is the information correct? [Y/n]
root@Anuk:~#
```

Adding user to sudo and video groups

```
# adduser daniel video
# adduser daniel sudo
```

Attention: In case we are using a Raspbian OS, the default user is 'pi'. I will recommend to create another user and remove the default one once you have created the new one with the following command:

```
$sudo pkill -u pi
$sudo deluser -remove-home pi
```


Lock down the SSH service

Edit the SSH config file. We recommend to use the ssh keys generated previously and disable password access

```
$nano /etc/ssh/sshd_config
```

Uncomment the lines of the following image that are in white

```
# Authentication:

#LoginGraceTime 2m
PermitRootLogin prohibit-password
#StrictModes yes
MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2
AuthorizedKeysFile .ssh/authorized_keys

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
```

and copy-paste the pub key we have generated previously.

```
$mkdir -p ~/.ssh
```

```
$nano ~/.ssh/authorized_keys
```

```
GNU nano 3.2                                .ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQDhQTSaa0rTBjRA/AWbQ/+FJ8fpm/oxPr8QM6ls9qaq/ADshZ06jA
```

Save changes and exit the editor. Restart SSH:

```
$sudo service ssh restart
```

Restart the service We are going to be disconnected in case we were connected through ssh.

```
$sudo service networking restart
```

In case you have assigned previously this IP address, you will get this message

```

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@  WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!  @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ECDSA key sent by the remote host is
SHA256:HF.....Y/U0.
Please contact your system administrator.
Add correct host key in /home/daniel/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /home/daniel/.ssh/known_hosts:1
ECDSA host key for 192.168.1.10 has changed and you have requested strict checking.
Host key verification failed.

```

Just delete the entry in your know hosts data base:

```
$nano ~/.ssh/known_hosts
```

Logout root & login as the new user (daniel in my case)

```
$ssh daniel@192.168.1.10
```

Check IP configuration (static IP and DNS configuration)

```

daniel@Anuk:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
   link/ether ..... brd ff:ff:ff:ff:ff:ff
   inet 192.168.1.10/24 brd 192.168.1.255 scope global eth0
       valid_lft forever preferred_lft forever
   inet6 fe80::ba27:ebff:fe71:a552/64 scope link
       valid_lft forever preferred_lft forever
3: wlan0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
   link/ether ..... brd ff:ff:ff:ff:ff:ff
daniel@Anuk:~$ 

```

Setting the time zone

Let's go and set our time zone. We can check all the timezones listed with the command:

```
$timedatectl list-timezones
```

Choose the one that fits you best. In my case I choose Europe/Madrid

```
$sudo timedatectl set-timezone Europe/Madrid
```

Once set, I can retrieve the status with the following command:

```
$timedatectl status
```

```
daniel@rpi3bp:~$ timedatectl status
          Local time: Sat 2021-02-13 16:40:47 CET
          Universal time: Sat 2021-02-13 15:40:47 UTC
              RTC time: n/a
              Time zone: Europe/Madrid (CET, +0100)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

We are going to set the time automatically, using the NTP protocol who help us to change and synchronize periodically the date and time.

```
$sudo nano /etc/systemd/timesyncd.conf
```

set NTP to "time.cloudflare.com" and uncomment the FallbackNTP and PollIntervalMaxSec lines

```
[Time]
NTP=time.cloudflare.com
FallbackNTP=0.debian.pool.ntp.org 1.debian.pool.ntp.org 2.debian.pool.ntp.org 3.debian.pool.ntp.org
#RootDistanceMaxSec=5
#PollIntervalMinSec=32
PollIntervalMaxSec=2048
```

Installing unattended upgrades package (recommended)

To have unattended upgrades, we need to install an additional package

```
$sudo apt install unattended-upgrades
```

The configuration of unattended upgrades is set inside this file:

```
$sudo nano /etc/apt/apt.conf.d/50unattended-upgrades
```

You may want to update some settings, I recommend uncomment and change “Unattended-Upgrade::Remove-Unused-Dependencies” to “true”. Exit and save the file.

```
// Do automatic removal of unused packages after the upgrade
// (equivalent to apt-get autoremove)
Unattended-Upgrade::Remove-Unused-Dependencies "true";
```

Basically, we commented out those type of upgrade we want to apply. The second last line allows the system to email us the status. We must install mailutils or mailx first in Raspbian for the email notification to be effective. The last line allow the system to reboot automatically. Please also make sure that update-notifier-common has been installed.

There are more option that we can set such as reboot time and log file in the configuration file. Uncomment any option when necessary.

Create a periodic upgrade file with the following command:

```
$sudo nano /etc/apt/apt.conf.d/02periodic
```

And the following content:

```
// Control parameters for cron jobs by /etc/cron.daily/apt-compat //
```



```
// Enable the update/upgrade script (0=disable)
APT::Periodic::Enable "1";
```



```
// Do "apt-get update" automatically every n-days (0=disable)
APT::Periodic::Update-Package-Lists "1";
```



```
// Do "apt-get upgrade --download-only" every n-days (0=disable)
APT::Periodic::Download-Upgradeable-Packages "1";
```



```
// Run the "unattended-upgrade" security upgrade script
// every n-days (0=disabled)
// Requires the package "unattended-upgrades" and will write
// a log in /var/log/unattended-upgrades
APT::Periodic::Unattended-Upgrade "1";
```

```
// Do "apt-get autoclean" every n-days (0=disable)
APT::Periodic::AutocleanInterval "7";

// Send report mail to root
//      0:  no report                (or null string)
//      1:  progress report          (actually any string)
//      2:  + command outputs        (remove -qq, remove 2>/dev/null, add -d)
//      3:  + trace on
APT::Periodic::Verbose "2";
```

Check your unattended upgrades by running this command to debug your configuration:

```
$sudo unattended-upgrades -d
```

```
daniel@Anuk:~$ sudo unattended-upgrades -d
Checking if system is running on battery is skipped. Please install powermgmt-base package to check power status and skip installing updates when the system is running on battery.
Starting unattended upgrades script
Allowed origins are: origin=Debian,codename=bullseye,label=Debian, origin=Debian,codename=bullseye,label=Debian-Security, origin=Debian,codename=bullseye-security,label=Debian-Security
Initial blacklist:
Initial whitelist (not strict):
Using (^linux-.*-[1-9][0-9]*\.[0-9]+\.[0-9]+-[0-9]+(-.+)?$|^kfreebsd-.*-[1-9][0-9]*\.[0-9]+\.[0-9]+-[0-9]+(-.+)?$|^gnumach-.*-[1-9][0-9]*\.[0-9]+\.[0-9]+-[0-9]+(-.+)?$|^.*-modules-[1-9][0-9]*\.[0-9]+\.[0-9]+-[0-9]+(-.+)?$|^.*-kernel-[1-9][0-9]*\.[0-9]+\.[0-9]+-[0-9]+(-.+)?$|^linux-.*-[1-9][0-9]*\.[0-9]+\.[0-9]+-[0-9]+(-.+)?$|^kfreebsd-.*-[1-9][0-9]*\.[0-9]+\.[0-9]+-[0-9]+(-.+)?$|^gnumach-.*-[1-9][0-9]*\.[0-9]+\.[0-9]+-[0-9]+(-.+)?$|^.*-modules-[1-9][0-9]*\.[0-9]+\.[0-9]+-[0-9]+(-.+)?$|^.*-kernel-[1-9][0-9]*\.[0-9]+\.[0-9]+-[0-9]+(-.+)?$) regexp to find kernel packages
Using (^linux-.*-5\.[0-9]\.[0-9]-arm64$|^linux-.*-5\.[0-9]\.[0-9]-arm64$|^kfreebsd-.*-5\.[0-9]\.[0-9]-arm64$|^kfreebsd-.*-5\.[0-9]\.[0-9]-arm64$|^gnumach-.*-5\.[0-9]\.[0-9]-arm64$|^gnumach-.*-5\.[0-9]\.[0-9]-arm64$|^.*-modules-5\.[0-9]\.[0-9]-arm64$|^.*-modules-5\.[0-9]\.[0-9]-arm64$|^.*-kernel-5\.[0-9]\.[0-9]-arm64$|^.*-kernel-5\.[0-9]\.[0-9]-arm64$|^linux-.*-5\.[0-9]\.[0-9]-arm64$|^linux-.*-5\.[0-9]\.[0-9]-arm64$|^kfreebsd-.*-5\.[0-9]\.[0-9]-arm64$|^kfreebsd-.*-5\.[0-9]\.[0-9]-arm64$|^gnumach-.*-5\.[0-9]\.[0-9]-arm64$|^gnumach-.*-5\.[0-9]\.[0-9]-arm64$|^.*-modules-5\.[0-9]\.[0-9]-arm64$|^.*-modules-5\.[0-9]\.[0-9]-arm64$|^.*-kernel-5\.[0-9]\.[0-9]-arm64$|^.*-kernel-5\.[0-9]\.[0-9]-arm64$) regexp to find running kernel packages
pkgs that look like they should be upgraded:
Fetched 0 B in 0s (0 B/s)
fetch.run() result: 0
Packages blacklist due to conffile prompts: []
No packages found that can be upgraded unattended and no pending auto-removals
Extracting content from /var/log/unattended-upgrades/unattended-upgrades-dpkg.log since 2021-04-10 23:07:10
daniel@Anuk:~$
```

Installing Fail2Ban (optional)

Fail2ban is an intrusion prevention software designed to prevent against brute-force attacks. First we need to install the package:

```
$sudo apt install fail2ban -y
```

Fail2ban will block attackers IP if they fail to login after 5 failures for 10 minutes.

Note: Fail2Ban installed from the repository will only provide security on IPv4 protocol. If you want Fail2Ban to support IPv6, please look at this [guide](#).

The configuration of fail2ban is set in the following file:

```
/etc/fail2ban/jail.conf
```

If you make any config changes, restart the service via:

```
$sudo service fail2ban restart
```

If you make any config changes, restart the service via:

```
$sudo service fail2ban restart
```

In order to recover acces

```
$ssh -o PreferredAuthentications=password -o PubkeyAuthentication=no  
user@your.vps.ip
```

Installing a firewall (optional)

Install the packages

```
$sudo apt install ufw
```

Configuring the firewall

Create your access list to the ports you need

```
$sudo ufw allow 80  
$sudo ufw allow 443  
$sudo ufw allow 53  
$sudo ufw allow 8888  
$sudo ufw allow 22
```

You can even be more restrictive with extended parameters on the rules, like SSH for example. You can only allow access on port 22 from your computer's IP address:

```
$sudo ufw allow from 192.168.1.120 port 22
```

Enabling the firewall

```
$sudo ufw enable
```

To show rules once the firewall is enabled, run the following command:

```
$sudo ufw status verbose
```

Installing log2ram to expand SSD life (recommended)

SSD Disks, SD Cards and USB sticks have a SSD inside, have a life span determined by the write cycles mainly (times we write something to the disk). To reduce the times we write to the SSD memory, we can derive the writing of the system logs to RAM memory using log2ram. To do that we have to install the log2ram application.

First we need to force a log reduction before to start to use log2ram

```
$sudo journalctl --vacuum-size=16M
```

Let's gonna add the repository where we are going to install the application and his key. Please check the Debian flavour you are using (bullseye in my case)

```
$sudo echo "deb http://packages.azlux.fr/debian/ bullseye main" | sudo  
tee /etc/apt/sources.list.d/azlux.list  
$sudo wget -qO - https://azlux.fr/repo.gpg.key | sudo apt-key add -
```

Let's gonna update the system database and install the application.

```
$sudo apt-get update  
$sudo apt install log2ram -y
```

Once installed we need a reboot

```
$sudo reboot
```

Configuring log2ram

We need to configure log2ram to increase the size

```
$sudo nano /etc/log2ram.conf
```

Increase the SIZE parameter to 128MB, disable the mail notification and increase the LOG_DISK_SIZE to 200M. Exit and save.

```

GNU nano 5.4 /etc/log2ram.conf
# Configuration file for Log2Ram (https://github.com/azlux/log2ram) under MIT license.
# This configuration file is read by the log2ram service

# Size for the ram folder, it defines the size the log folder will reserve into the R
# If it's not enough, log2ram will not be able to use ram. Check you /var/log size fo
# The default is 40M and is basically enough for a lot of applications.
# You will need to increase it if you have a server and a lot of log for example.
SIZE=128M

# If there are some errors with available RAM space, a system mail will be send
# Change it to false and you will have only a log if there is no place on RAM anymore.
MAIL=false

# Variable for folders to put in RAM. You need to specify the real folder `/path/fold
# example : PATH_DISK="/var/log;/home/test/FolderInRam"
PATH_DISK="/var/log"

# ***** Zram backing conf *****

# ZL2R Zram Log 2 Ram enables a zram drive when ZL2R=true ZL2R=false is mem only tmpfs
ZL2R=false
# COMP_ALG this is any compression algorithm listed in /proc/crypto
# lz4 is fastest with lightest load but deflate (zlib) and Zstandard (zstd) give far
# lzo is very close to lz4 and may with some binaries have better optimisation
# COMP_ALG=lz4 for speed or Zstd for compression, lzo or zlib if optimisation or avail
COMP_ALG=lz4
# LOG_DISK_SIZE is the uncompressed disk size. Note zram uses about 0.1% of the size
# LOG_DISK_SIZE is expected compression ratio of alg chosen multiplied by log SIZE
# lzo/lz4=2.1:1 compression ratio zlib=2.7:1 zstandard=2.9:1
# Really a guesstimate of a bit bigger than compression ratio whilst minimising 0.1% m
LOG_DISK_SIZE=200M

```

Restart log2ram

```
$sudo service log2ram restart
```

And check that log2ram is running.

```
$df -h
```

```

daniel@Anuk:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            406M    0  406M   0% /dev
tmpfs           91M   580K   91M   1% /run
/dev/mmcblk0p2  30G   3.8G   25G  14% /
tmpfs           454M   1.4M   452M   1% /dev/shm
tmpfs           5.0M    0   5.0M   0% /run/lock
/dev/mmcblk0p1  299M  135M  165M  46% /boot/firmware
tmpfs           91M    0   91M   0% /run/user/998
tmpfs           91M    0   91M   0% /run/user/1000
log2ram         128M   51M   78M  40% /var/log

```


To access to your network from outside through a VPN (WireGuard) you will need to know the public IP address of your local network (or a domain name associated). Usually this IP address is dynamic and can change. Static IP addresses are limited in number and are more expensive. One solution is to use a DDNS service. This service provide a free domain that will point to a public IP address. A script will update periodically my public IP to this service, so in case my IP changes, the domain will point to the new IP.

If you have a static public IP address, you can skip this step. I will use [duckdns](#). They have detailed instructions and provide scripts to install to several devices, including raspberry pi.

raspberrypi

if you are running <http://www.raspbmc.com/> then you must follow the [raspbmc](#) instructions
first lets login to your raspberrypi over ssh

```
ssh pi@raspberrypi
```

then lets make a directory to put your files in, move into it and make our main script

```
mkdir duckdns
cd duckdns
vi duck.sh
```

now copy this text and put it into the file (in vi you hit the i key to insert, **ESC** then u to undo) The example below is for the domain **xiuxiueig**
if you want the configuration for a different domain, use the drop down box above
you can pass a comma separated (no spaces) list of domains
you can if you need to hard code an IP (best not to - leave it blank and we detect your remote ip)
hit **ESC** then use use arrow keys to move the cursor x deletes, i puts you back into insert mode

```
echo url="https://www.duckdns.org/update?domains=xiuxiueig&token=&ip=0.0.0.0&txt=duckdns.org" | curl -k -o
~/duckdns/duck.log -K -
```

now save the file (in vi hit **ESC** then :wq! then ENTER)
this script will make a https request and log the output in the file duck.log
now make the duck.sh file executable

```
chmod 700 duck.sh
```

next we wil be using the cron process to make the script get run every 5 minutes

```
crontab -e
```

copy this text and paste it at the bottom of the crontab

```
*/* * * * * ~/duckdns/duck.sh >/dev/null 2>&1
```

now save the file (**CTRL+o** then **CTRL+x**)
lets test the script

```
./duck.sh
```

In our case we are going to create the cron with the sudo command and execute the script at 5 minutes past every hour. I think it is enough update every hour and not every 5 minutes. My script has *xiuxiueig.sh* as its name since I will be adding more scripts.

```
$sudo crontab -e
```

This is how the crontab looks

```
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
5 */1 * * * ~/duckdns/xiuxiueig.sh >/dev/null 2>&1
```

With crontab guru you can play with different combinations: <https://crontab.guru/>

Installing and configuring Pi-hole

Now that system is configured and secured, we can install Pi-hole. The install process is very simple, we just to execute the following command and the script downloaded it will start to install the application:

```
$sudo curl -sSL https://install.pi-hole.net | bash
```

```
-----
Selecting previously unselected package dhcpcd5.
(Reading database ... 24788 files and directories currently installed.)
Preparing to unpack .../dhcpcd5_7.1.0-2_arm64.deb ...
Unpacking dhcpcd5 (7.1.0-2) ...
Setting up dhcpcd5 (7.1.0-2) ...
Created symlink /etc/systemd/system/multi-user.target.wants/dhcpcd.service → /lib/syst
Processing triggers for systemd (241-7~deb10u6) ...
-----
[✓] Supported OS detected
[i] SELinux not detected
Cancel was selected, exiting installer
daniel@rpi3bp:~$ sudo curl -sSL https://install.pi-hole.net | bash

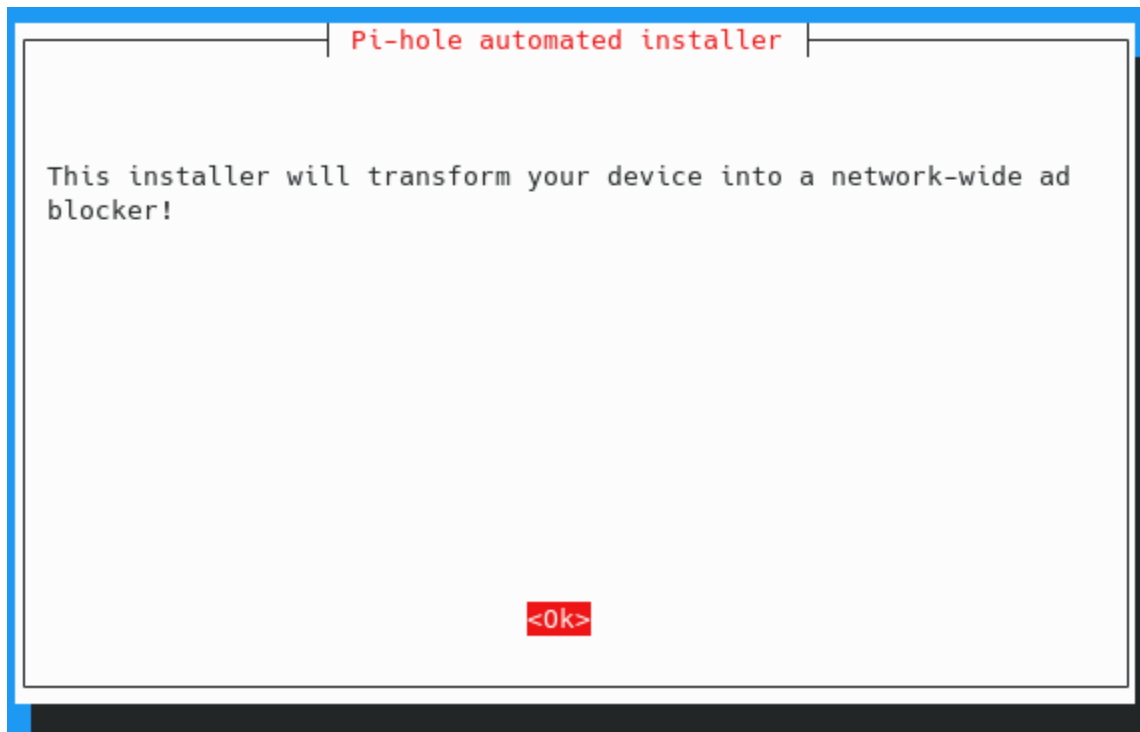
[i] Root user check
[i] Script called with non-root privileges
The Pi-hole requires elevated privileges to install and run
Please check the installer for any concerns regarding this requirement
Make sure to download this script from a trusted source

[✓] Sudo utility check

[✓] Root user check

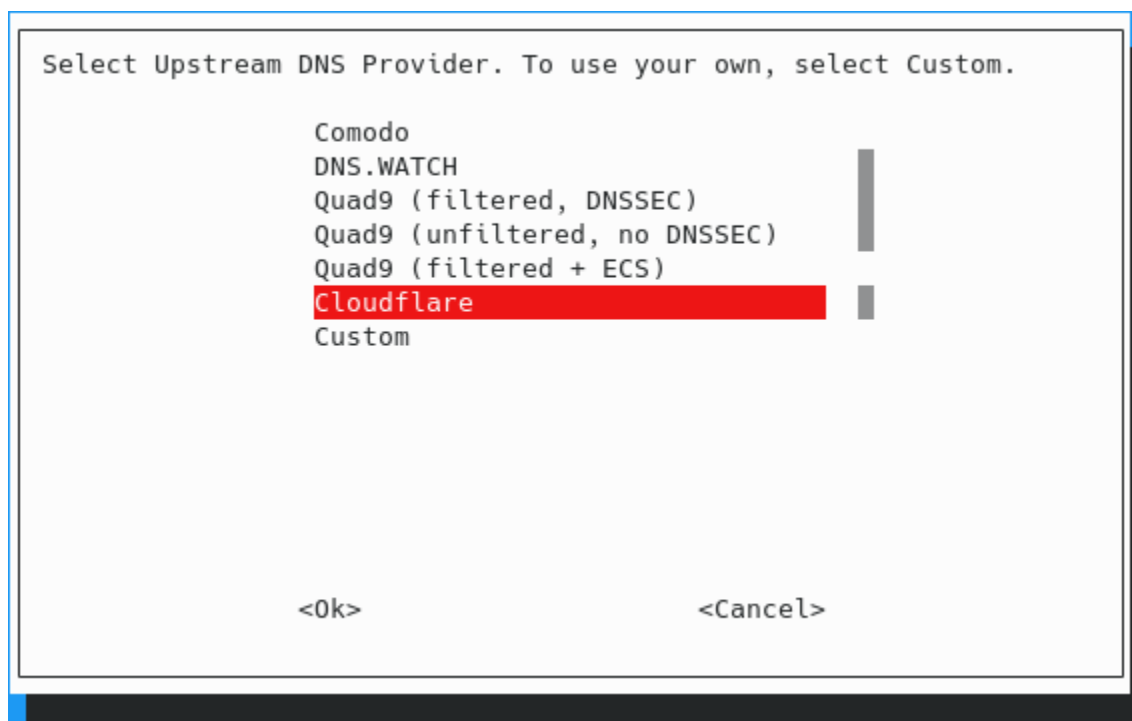
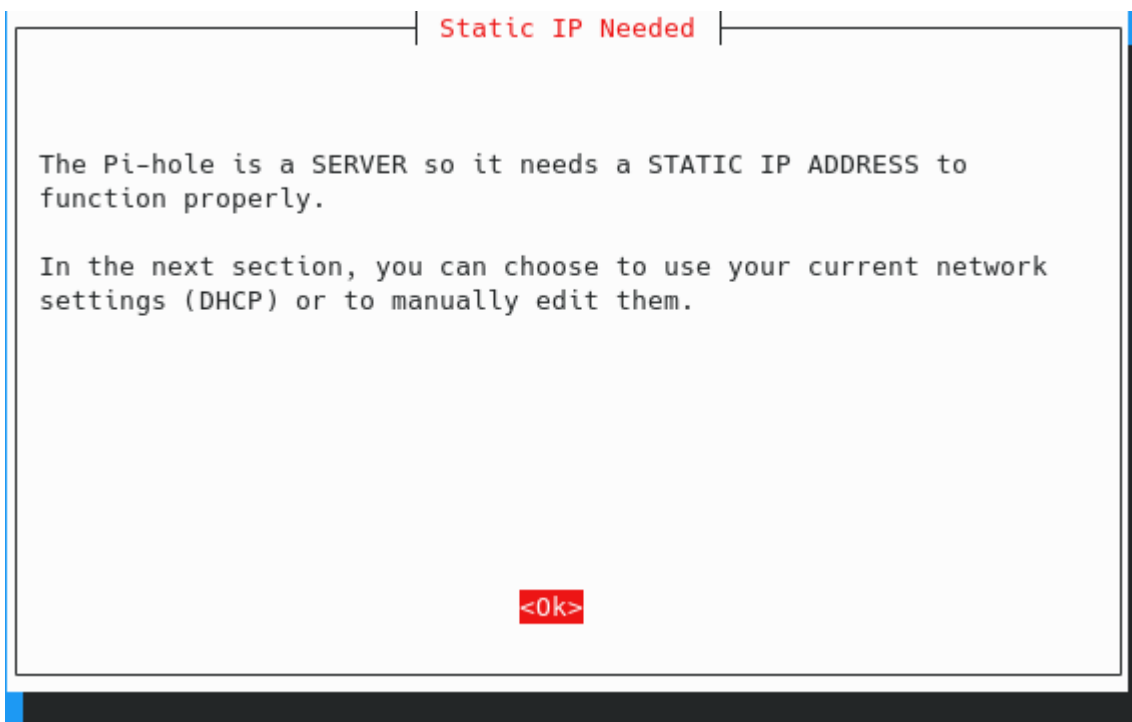
.;;,.
.cccc:,.
:cccclll:.    .,,
:cccclll.    ;ooodc
:cccclll.    ;ooodc
```

After some checks, you'll be greeted with the install screen:

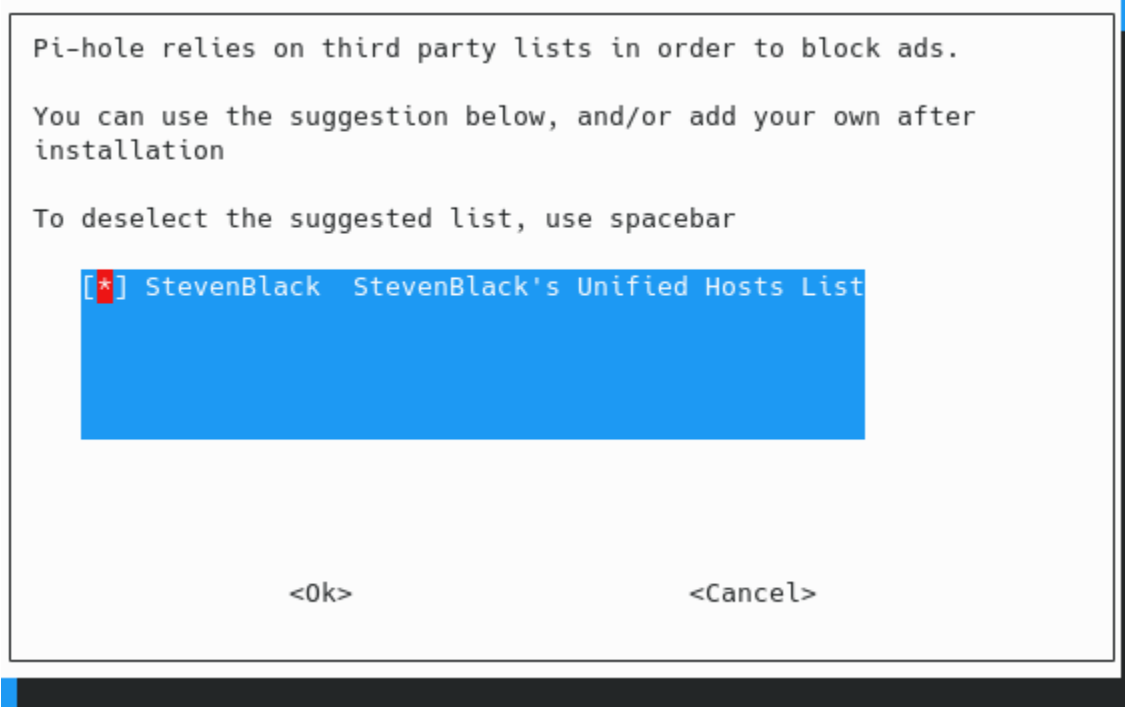


Remember to give a donation to the project if you find useful (I did it)

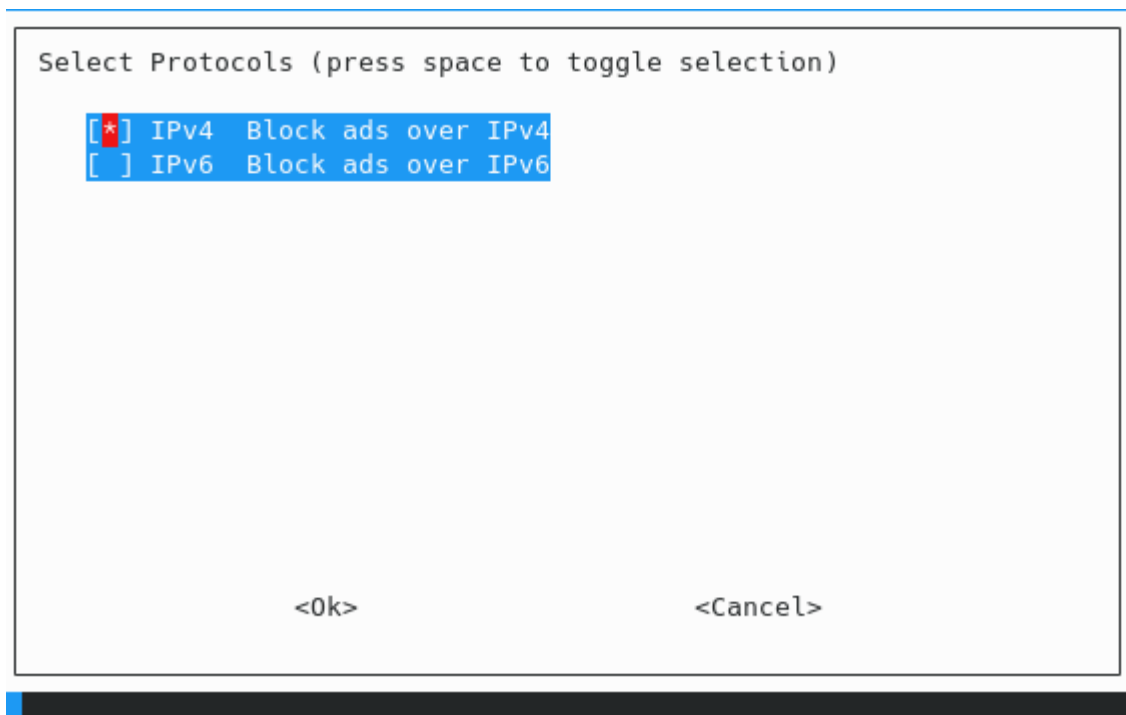


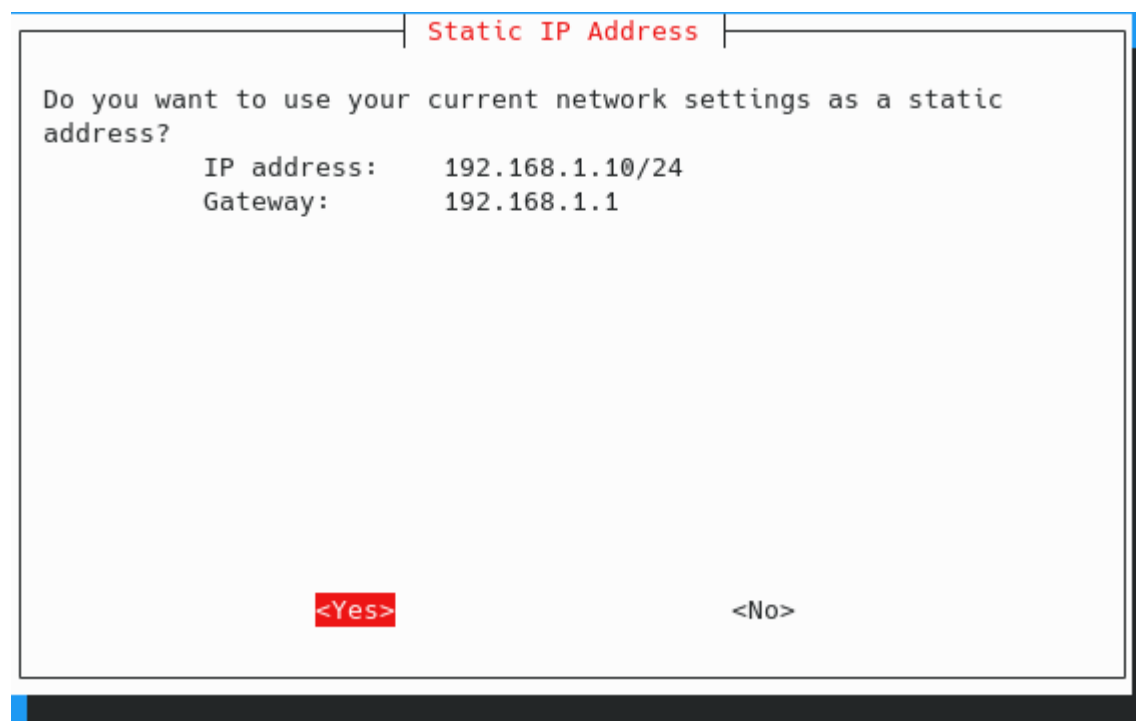


I recommend to select all the third-party list listed. We can add additional sources later.

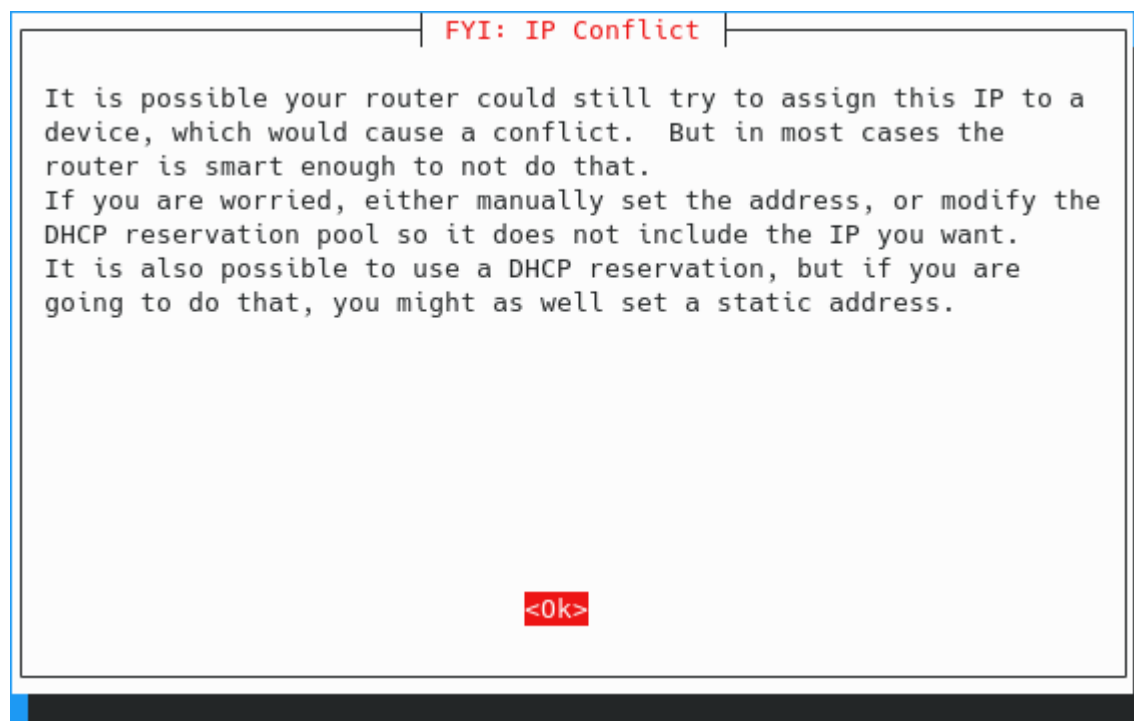


Choose the protocols you have in your network

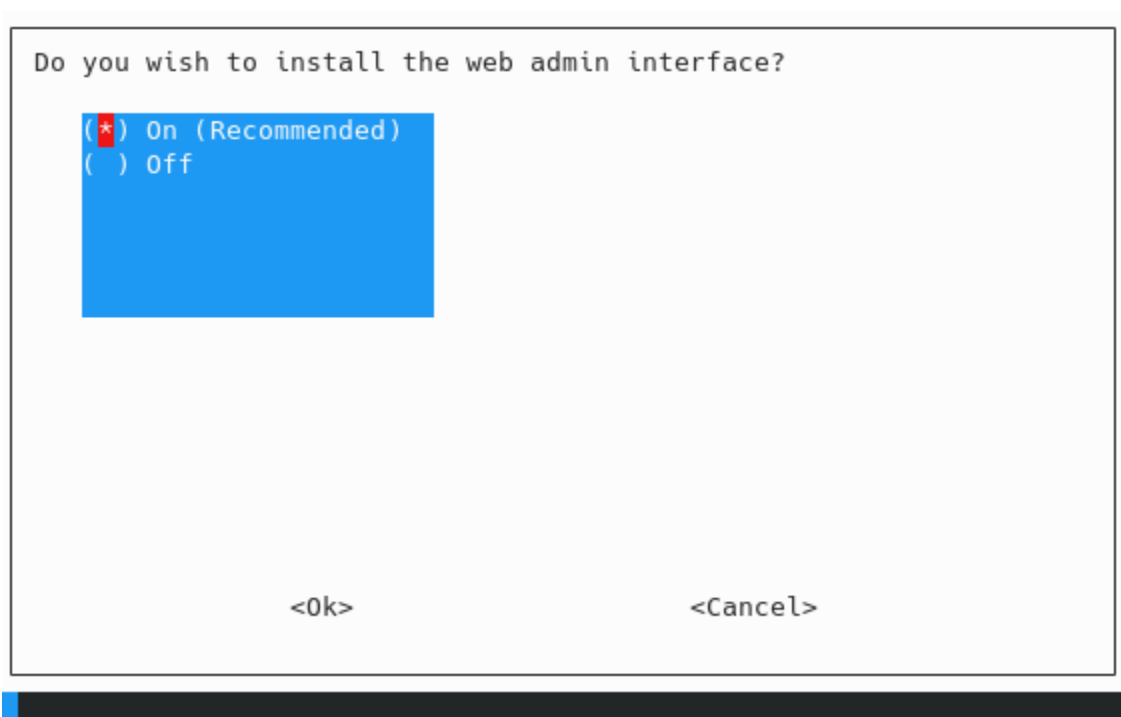




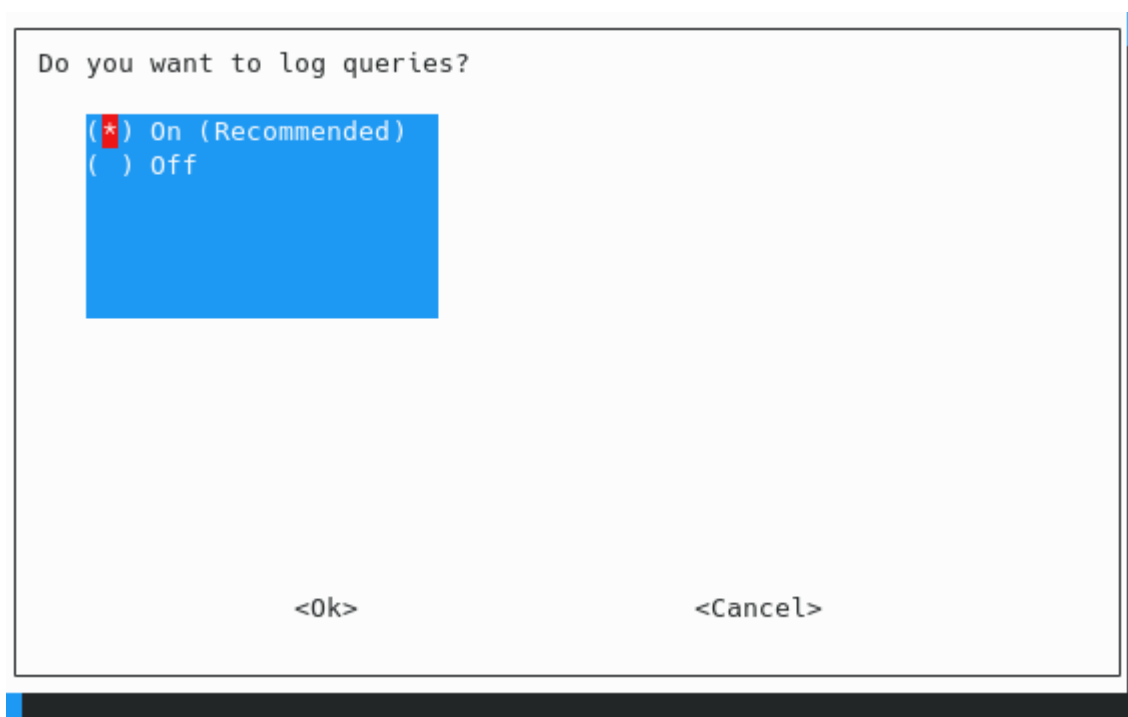
Ensure you have a IP reservation for your raspberry

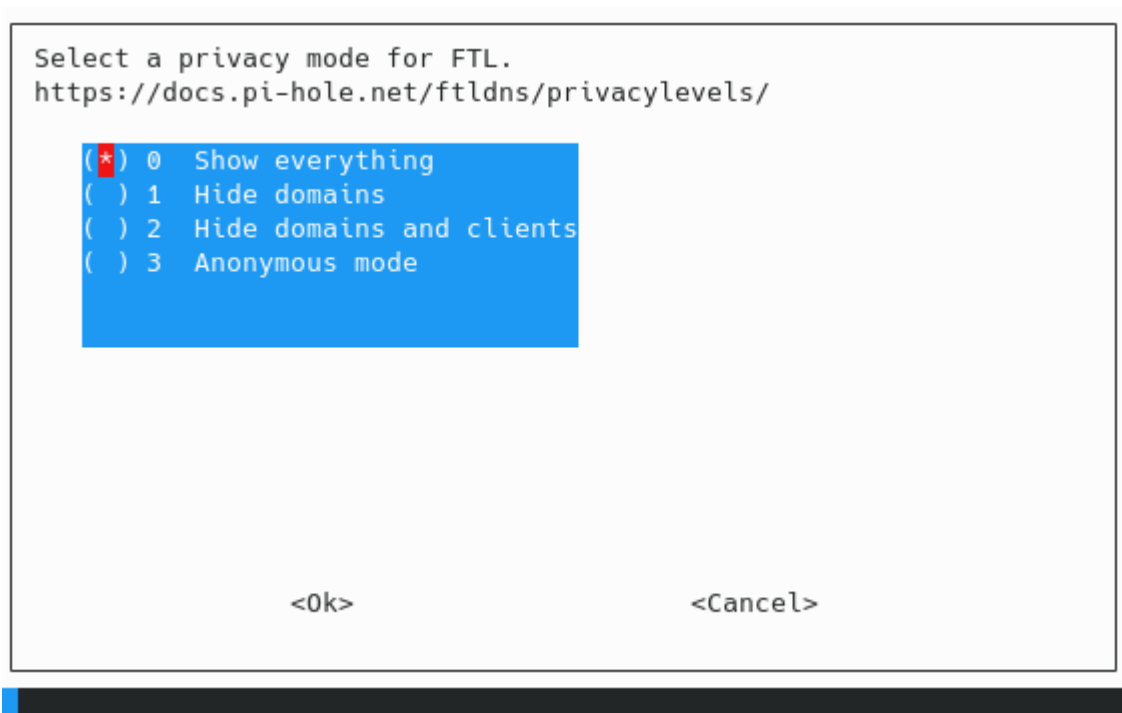


I will recommend to install the web interface

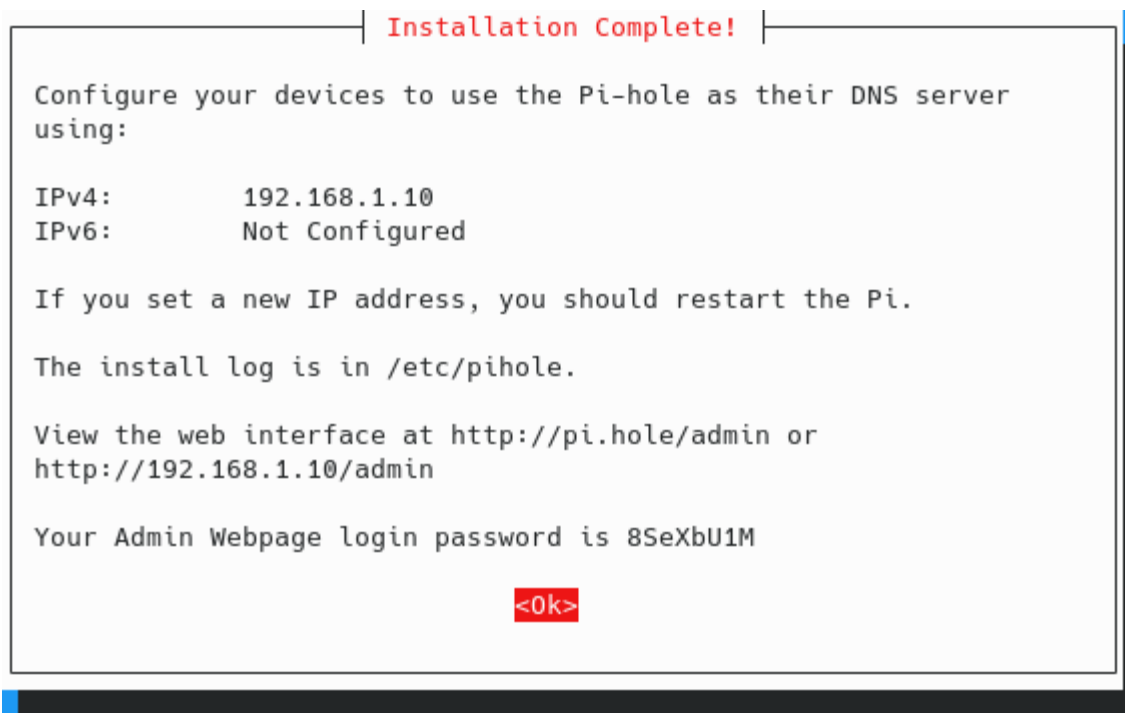


Let the log enable





When the installation is complete you will get a final screen with some important info. Save this information to access to the Pi-hole server:



Save the admin webpage password in your password manager for now, it should be changed later.

This same info is displayed once you return to the shell, note the command to change the web admin password (*pihole -a -p*)

```
[✓] DNS service is listening
    [✓] UDP (IPv4)
    [✓] TCP (IPv4)
    [✓] UDP (IPv6)
    [✓] TCP (IPv6)

[i] Pi-hole blocking will be enabled
[i] Enabling blocking
[✓] Flushing DNS cache
[✓] Pi-hole Enabled
[i] Web Interface password: 8SeXbU1M
[i] This can be changed using 'pihole -a -p'

[i] View the web interface at http://pi.hole/admin or http://192.168.1.10/admin

[i] You may now configure your devices to use the Pi-hole as their DNS server
[i] Pi-hole DNS (IPv4): 192.168.1.10
[i] If you set a new IP address, please restart the server running the Pi-hole

[i] The install log is located at: /etc/pihole/install.log
Installation Complete!
```

Tweaking Pi-hole

To change the privacy setting from Pi-hole application, we have to edit the following file:

```
$sudo nano /etc/pihole/pihole-FTL.conf
```

Set the privacy level and the days to store the queries in the database.

```
# Which privacy level is used?. More info:
https://docs.pi-hole.net/ftldns/privacylevels/
PRIVACYLEVEL=0

# How long should queries be stored in the database? Setting this to 0
disables the database. Default 365
MAXDBDAYS=30
```

Installing Unbound

Enhancing Pi-hole Security (optional if we are not fine with Cloudflare DNS)

So now we have a working Pi-hole, but it has minimal blocking and just forwards lookup to Google DNS. We can change our upstream DNS provider, but that is just changing who we trust with our DNS. What if we don't trust anyone? We can install Unbound and resolve DNS ourselves using root servers to recursively resolve DNS names. A more in depth explanation of how this works can be found here: <https://docs.pi-hole.net/guides/dns/unbound/> but essentially Unbound will look up a DNS query by asking TLD servers for DNS in a recursive manner. The benefit is more security; you do not have to trust an upstream provider with your DNS traffic. The drawback is performance for initial lookup, as they need to traverse and this takes time. Pi-hole and Unbound can both be configured with caching, which will help mitigate this for subsequent lookup.

```
$sudo apt install unbound -y
$wget https://www.internic.net/domain/named.root -qO- | sudo tee
/var/lib/unbound/root.hints
```

Creating a configuration for Pi-hole

```
$sudo nano /etc/unbound/unbound.conf.d/pi-hole.conf
```

Paste into the file this configuration. This is different than the one in Pi-hole's documentation. It includes caching configuration that will improve performance.

```
server:
    # If no logfile is specified, syslog is used
    # logfile: "/var/log/unbound/unbound.log"
    verbosity: 0

interface: 127.0.0.1
    port: 5335
    do-ip4: yes
    do-udp: yes
```

```
do-tcp: yes

# May be set to yes if you have IPv6 connectivity
do-ip6: no

# You want to leave this to no unless you have *native* IPv6. With 6to4 and
# Terredo tunnels your web browser should favor IPv4 for the same
reasons
prefer-ip6: no

# Use this only when you downloaded the list of primary root servers!
# If you use the default dns-root-data package, unbound will find it
automatically
root-hints: "/var/lib/unbound/root.hints"

# Trust glue only if it is within the server's authority
harden-glue: yes

# Require DNSSEC data for trust-anchored zones, if such data is absent, the
zone becomes BOGUS
harden-dnssec-stripped: yes

# Don't use Capitalization randomization as it known to cause DNSSEC issues
sometimes
# see https://discourse.pi-hole.net/t/unbound-stubby-or-dnscrypt-proxy/9378 for further details
use-caps-for-id: no

# Reduce EDNS reassembly buffer size.
# Suggested by the unbound man page to reduce fragmentation reassembly
problems
edns-buffer-size: 1472

# Perform prefetching of close to expired message cache entries
# This only applies to domains that have been frequently queried
# This refreshes expiring cache entries if they have been accessed with
# less than 10% of their TTL remaining
```

```
prefetch: yes

# This attempts to reduce latency by serving the outdated record before
# updating it instead of the other way around. Alternative is to
increase
# cache-min-ttl to e.g. 3600.
cache-min-ttl: 0
serve-expired: yes
# I had best success leaving this next entry unset.
# serve-expired-ttl: 3600 # 0 or not set means unlimited (I think)

# Use about 2x more for rrset cache, total memory use is about 2-2.5x
# total cache size. Current setting is way overkill for a small
network.
# Judging from my used cache size you can get away with 8/16 and still
# have lots of room, but I've got the ram and I'm not using it on
anything else.
# Default is 4m/4m
msg-cache-size: 128m
rrset-cache-size: 256m

# One thread should be sufficient, can be increased on beefy machines. In
reality for most users running on small networks or on a single machine, it
should be unnecessary to seek performance enhancement by increasing num-
threads above 1.
num-threads: 1

# Ensure kernel buffer is large enough to not lose messages in traffic
spikes
so-rcvbuf: 1m

# Ensure privacy of local IP ranges
private-address: 192.168.0.0/16
private-address: 169.254.0.0/16
private-address: 172.16.0.0/12
private-address: 10.0.0.0/8
private-address: fd00::/8
```

```
private-address: fe80::/10

# To get unbound stats (sudo unbound-control stats_noreset)
remote-control:
    control-enable: yes
```

Let's gonna check the unbound configuration

```
$sudo unbound-checkconf
```

```
daniel@Anuk:~$ sudo unbound-checkconf
unbound-checkconf: no errors in /etc/unbound/unbound.conf
```

Last step: restart the service

```
$sudo service unbound restart
```

Testing Unbound

Now we are going to test unbound, measuring the time it spend to reach a domain, coneixement.info in this example.

```
daniel@Anuk:~$ dig coneixement.info @127.0.0.1 -p 5335

; <<>> DiG 9.11.5-P4-5.1+deb10u3-Debian <<>> coneixement.info @127.0.0.1 -p 5335
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50124
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1472
;; QUESTION SECTION:
;coneixement.info.          IN      A

;; ANSWER SECTION:
coneixement.info.          600     IN      A      91.199.120.7

;; Query time: 390 msec
;; SERVER: 127.0.0.1#5335(127.0.0.1)
;; WHEN: Sun Apr 11 12:28:12 CEST 2021
;; MSG SIZE rcvd: 61

daniel@Anuk:~$
```

As you can see in the second test, the time decrease, since unbound has a cache.

```
daniel@Anuk:~$ dig coneixement.info

; <<>> DiG 9.16.13-Debian <<>> coneixement.info
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55473
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;coneixement.info.                IN      A

;; ANSWER SECTION:
coneixement.info.                600     IN      A      91.199.120.7

;; Query time: 43 msec
;; SERVER: 1.1.1.1#53(1.1.1.1)
;; WHEN: Sat Apr 10 22:59:30 CEST 2021
;; MSG SIZE rcvd: 61

daniel@Anuk:~$
```

Keeping unbound updated

Let's setup some cron jobs to keep unbound updated

```
$sudo crontab -e
```

Add a new line at the end and paste the following:

```
01 02 03 */4 * wget -N -O -q /var/lib/unbound/root.hints
https://www.internic.net/domain/named.root
```

Exit and save.

With the -O option we have to tell it the path to store the file. Also the -N will only update it if the remote file is newer than the file you have. The -q option will keep it quiet so it doesn't dump a bunch of output in your logs needlessly.

The update will be done at 02:01 on day-of-month 3 every 4 months. I think it is not necessary to update more often.

Keeping updated Pi-hole (optional)

Let's setup some cron jobs to keep Pi-hole updated. In case we want to keep Pi-hole updated, we have to run periodically the 'pihole -up' command.

```
$sudo crontab -e
```

Paste the following to update every sunday at 2:30 AM

```
30 2 * * SUN pihole -up
```

Warning: The PiHole team does not recommend updating Pi-hole via cron jobs. Be aware that with the previous configuration your server will update Pi-hole every Sunday via cron, and stay up-to-date on patch notes. If there is a major change, and you don't want to update, run 'sudo crontab -e' again and comment out the line to update Pi-hole (place a # at the beginning of the line).

Configuring Pi-hole to use unbound

Login to your Pi-hole admin page at <http://pi.hole/admin> and use the password you saved from the install. Navigate to Settings, and click on the DNS tab. Uncheck the DNS Servers checked and check custom 1 and enter `127.0.0.1#5335`. Click Save button at the bottom of the page.

Upstream DNS Servers

IPv4	IPv6	Name
<input type="checkbox"/>	<input type="checkbox"/>	Google (ECS)
<input type="checkbox"/>	<input type="checkbox"/>	OpenDNS (ECS, DNSSEC)
<input type="checkbox"/>	<input type="checkbox"/>	Level3

Upstream DNS Servers

Custom 1 (IPv4)
☒

Custom 3 (IPv6)
☐

As things get queried initial performance will be slow but quickly improve because of the caching nature of Pi-hole and the cache that has been configured for Unbound. Here is an example:

```
daniel@Anuk:~$ dig www.google.es

; <<>> DiG 9.11.5-P4-5.1+deb10u3-Debian <<>> www.google.es
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 20742
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;www.google.es.                IN      A

;; ANSWER SECTION:
www.google.es.                137     IN      A      216.58.215.163

;; Query time: 24 msec
;; SERVER: 1.1.1.1#53(1.1.1.1)
;; WHEN: Sun Apr 11 12:31:10 CEST 2021
;; MSG SIZE rcvd: 58

daniel@Anuk:~$ dig www.google.es @192.168.1.10

; <<>> DiG 9.11.5-P4-5.1+deb10u3-Debian <<>> www.google.es @192.168.1.10
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26182
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1472
;; QUESTION SECTION:
;www.google.es.                IN      A

;; ANSWER SECTION:
www.google.es.                300     IN      A      142.250.185.3

;; Query time: 449 msec
;; SERVER: 192.168.1.10#53(192.168.1.10)
;; WHEN: Sun Apr 11 12:31:23 CEST 2021
;; MSG SIZE rcvd: 58

daniel@Anuk:~$
```



```
daniel@Anuk:~$ dig www.google.es @192.168.1.10

; <<>> DiG 9.11.5-P4-5.1+deb10u3-Debian <<>> www.google.es @192.168.1.10
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5481
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

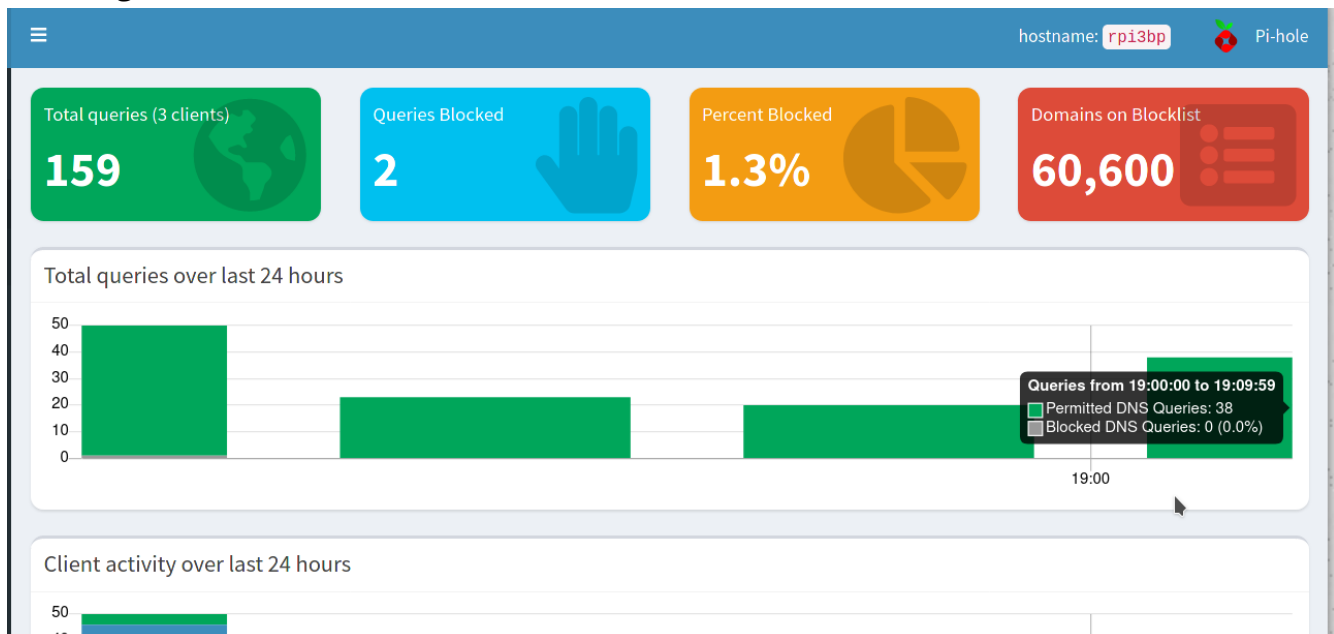
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:;, udp: 4096
;; QUESTION SECTION:
;www.google.es.                IN      A

;; ANSWER SECTION:
www.google.es.                196     IN      A      142.250.185.3

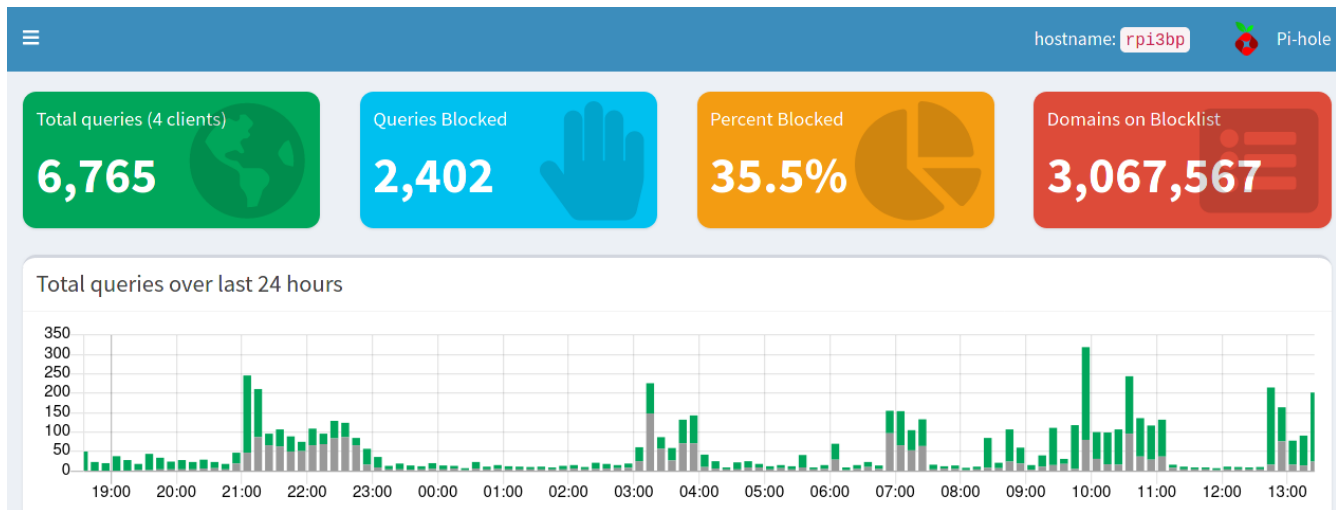
;; Query time: 0 msec
;; SERVER: 192.168.1.10#53(192.168.1.10)
;; WHEN: Sun Apr 11 12:33:07 CEST 2021
;; MSG SIZE rcvd: 58

daniel@Anuk:~$
```

If we login into the web interface, we will see some statistics.



And here how the statistics of queries blocked increase over time

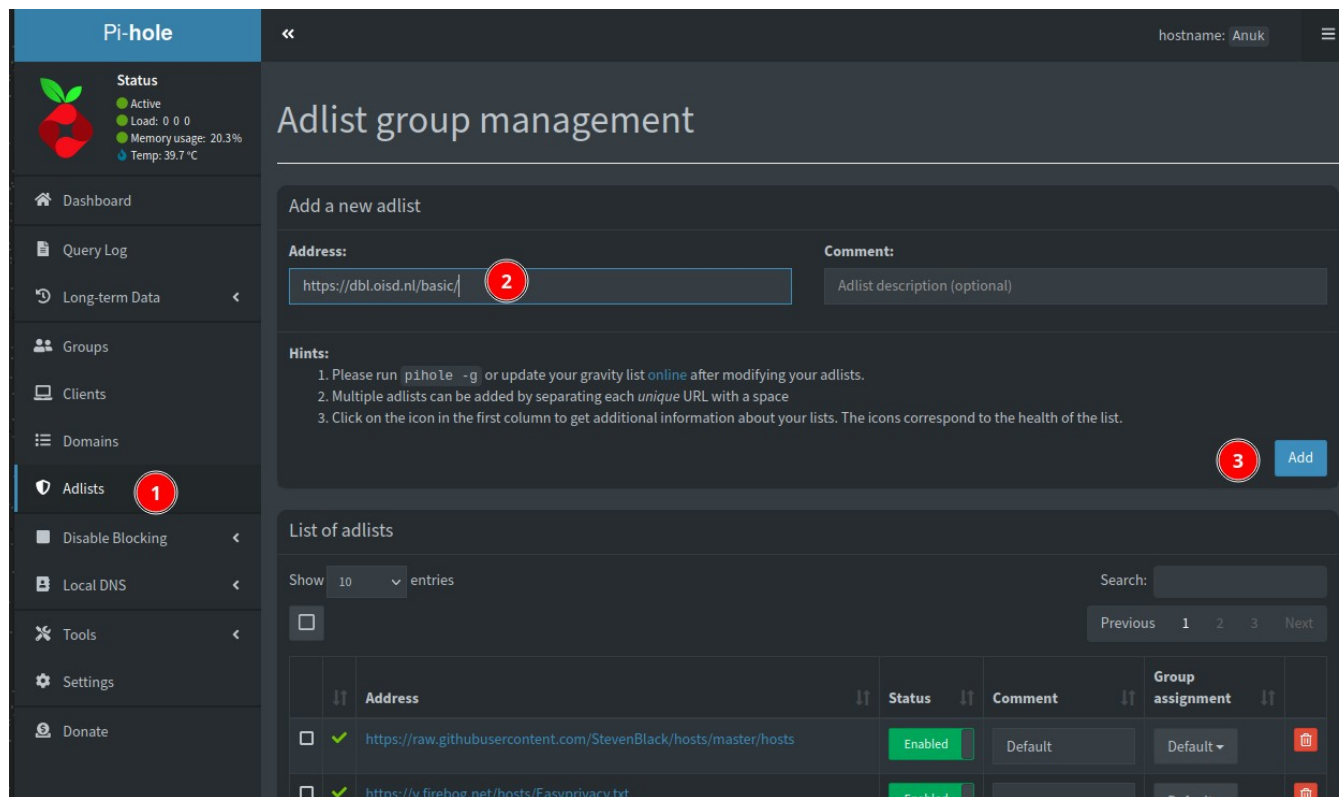


Improving our Blocklists (Blacklist, Adlist and Whitelist)

I strongly recommend to understand which kind of block do you want (advertising, telemetry, parental control, NSFW, malware domain, etc..) and add list from each category. Pi-hole comes out-the-box with an optional blocklist (in case we have selected them during the installation process). This list is maintained and updated regularly. In case we want to add additional blocklist, we can check several sources. [Firebog](#) is the main reference (The Big Blocklist Collection) of blocklist where we can find several lists separated into categories:

- Suspicious
- Advertising
- Tracking & Telemetry
- Malicious
- Other

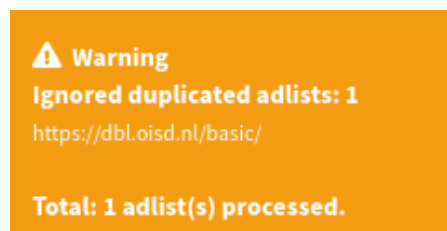
We can start adding from one to three list from each category you are interested to block, but before starting to add sources, read the points regarding each list from Firebog page. To add a new list, once logged into the Pi-hole web interface, you should to the Adlists option and paste the url list.



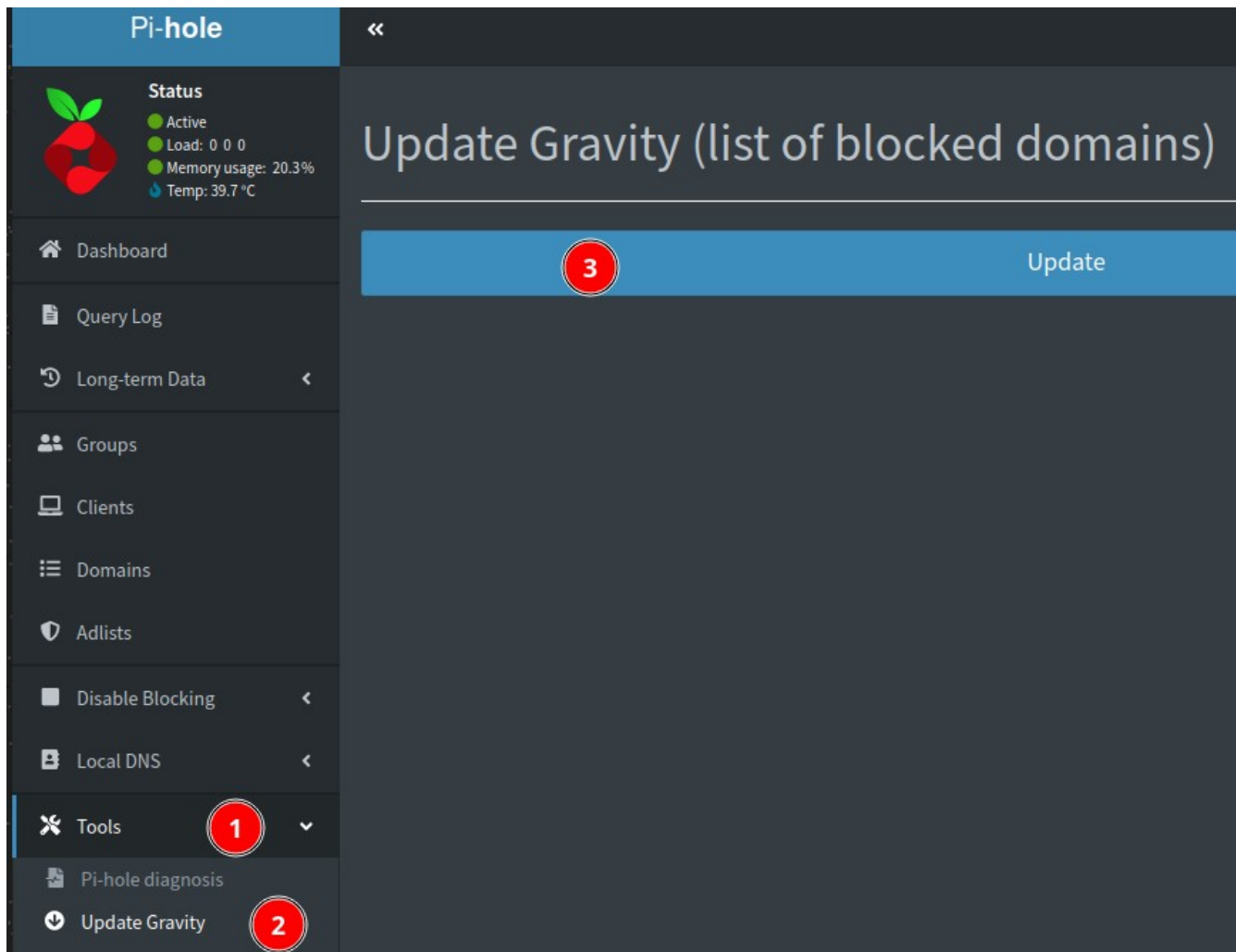
If the list has been successfully added, a message like this will appear:



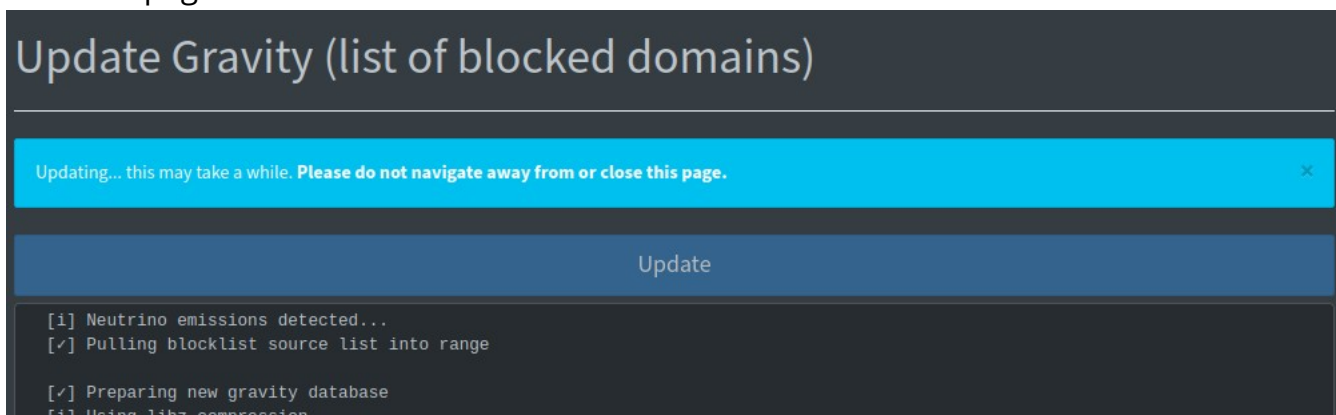
In case we have added previously this list, this will be ignored and a warning message like this will appear:



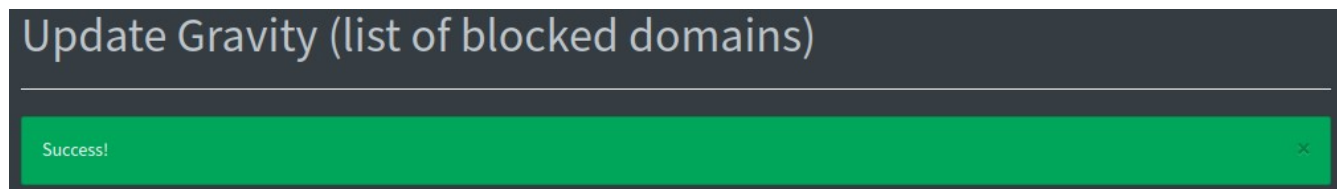
Once we have added all the lists, we have to update the internal database to have applied the new blocklists. This can be done running 'pihole -g' at the command line or through the web interface, clicking the update button we can find under 'Update Gravity'.



This will take a while . Be patient. As appear on the screen, do not navigate away from or close the page



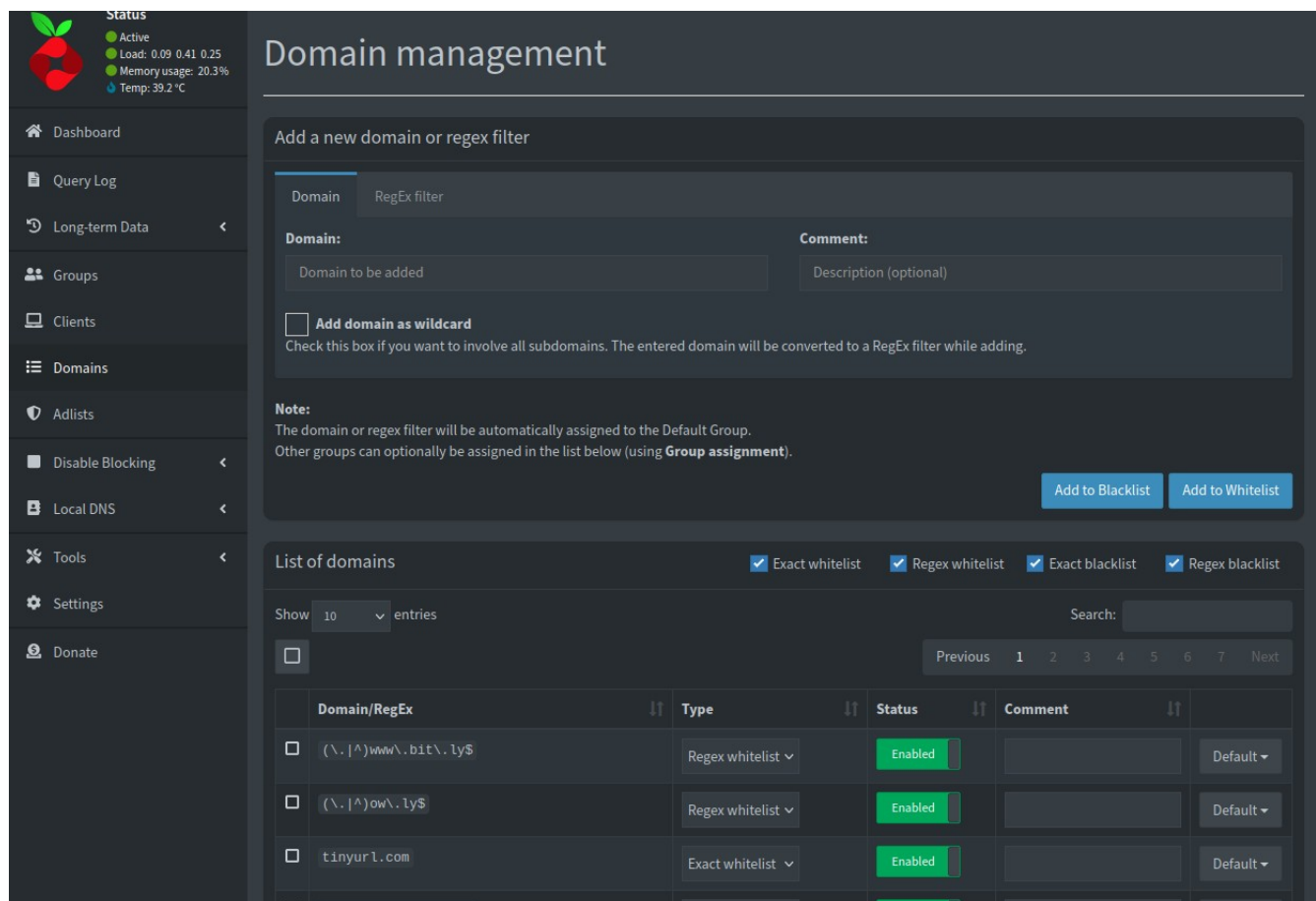
Wait until a success message appears



Remember “The more is not always the better”. If you add too much lists, you will get [false positives](#), which implies you will get troubles while you are surfing Internet, being some services inaccessible, unreachable or not fully functional

Some domains should be added to the Whitelist to avoid malfunctions or troubles while browsing.

You can also add domains to the Blacklist or Whitelist (Domains menu). Here you have an example of domains enabled:



For example, if you have problems with gmail icons (does not appear), you should add the domain gstaticadssl.l.google.com to the whitelist.

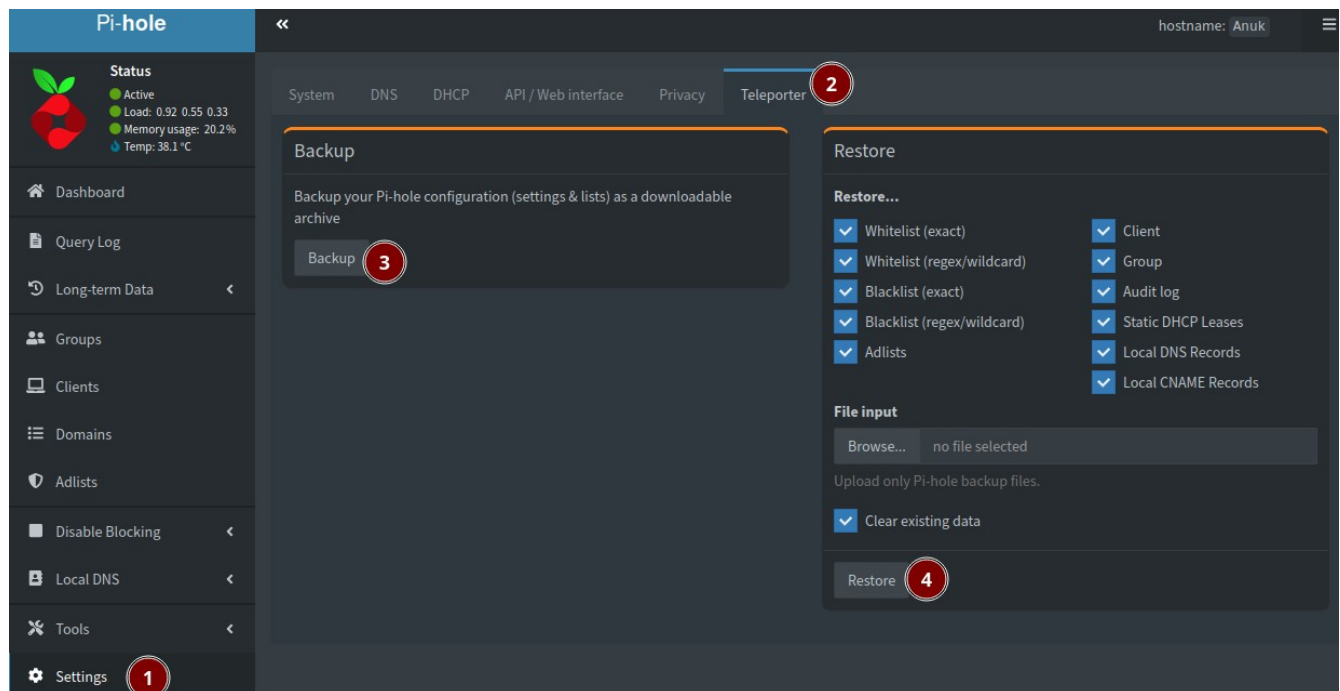
Removing existing blocklist

To remove the existing blocklist, we have to run this command:

```
$sudo sqlite3 /etc/pihole/gravity.db "DELETE FROM adlist"
```

Backup Pi-hole configuration

After all the tuning done, I will recommend to make a backup. This can be done through the web interface, creating a file (steps 1 to 3) that can be imported (4) in the same client or another pi-hole, saving configuration time:



Additional functionalities

Here's a list of additional functionalities, among others, that Pi-hole can do:

- Transforming Pi-hole to our DHCP service provider
- Managing clients and groups
- Disable blocking
- Query log to review black and white list, allowing to add or remove from black or whitelist

Installing WireGuard (light, secure and fast VPN)

A light, secure and fast VPN Server to allow remote and secure access.

Introduction

WireGuard is a communication protocol and free and open-source software that implements encrypted virtual private networks (VPNs), and was designed with the goals of ease of use, high speed performance, and low attack surface. It aims for better performance and more power than others VPN Servers like IPsec and OpenVPN. (From the Wikipedia :-))

To install the most recent version of WireGuard, we'll need packages from the Debian unstable release. Add the Debian unstable release, and [pin the Debian unstable priority behind Raspbian stable](#). This allows us to install packages that are not available in Debian stable, while keeping the "stable" versions of everything else.

Configuring repositories

By default, Raspbian doesn't trust the Debian package repository. We need to add Debian's public keys to the trusted set of keys.

```
$sudo apt-key adv --keyserver http://p80.pool.sks-keyservers.net:80 --recv-keys 04EE7237B7D453EC 648ACFD622F3D138
```

And

```
$sudo sh -c "echo 'deb http://deb.debian.org/debian/ unstable main' >> /etc/apt/sources.list.d/unstable.list"
```

And prevent RPi from using the Debian distro for normal Raspbian packages to avoid conflicts.

```
$sudo sh -c "printf 'Package: *\nPin: release a=unstable\nPin-Priority: 90\n' >> /etc/apt/preferences.d/limit-unstable"
```

And

```
$wget -O - https://ftp-master.debian.org/keys/archive-key-$(lsb_release -sr).asc | sudo apt-key add -
```

Installing applications

We need some additional stuff along WireGuard, so let's gonna update the system database

```
$sudo apt-get update
```

And install all the packages needed.

```
$sudo apt-get install wireguard wireguard-dkms wireguard-tools linux-headers-$(uname r) qrencode linux-headers
```

Note: In case you are using a raspbian OS, you will need the kernel headers:

```
$sudo apt-get install raspberrypi-kernel-headers
```

Set Up and Configuring the WireGuard VPN Server

We are going to configure 2 access, from a phone and from a laptop.

Generating security keys

To ensure that not just anyone gets access to our network and ensure a secure connection, we'll first need to generate a set of public/private key pairs with the following commands (execute them one line at a time in your RPi):

```
$sudo su -  
#cd /etc/wireguard  
#umask 077  
#wg genkey | tee server_private_key | wg pubkey > server_public_key  
#wg genkey | tee phone_private_key | wg pubkey > phone_public_key  
#wg genkey | tee laptop_private_key | wg pubkey > laptop_public_key  
#wg genpsk < phone_private_key > phone_preshared_key  
#wg genpsk < laptop_private_key > laptop_preshared_key
```

This is the list of files created:


```

root@Anuk:/etc/wireguard# ls -l
total 32
-rw----- 1 root root 45 Apr 11 13:53 laptop_preshared_key
-rw----- 1 root root 45 Apr 11 13:53 laptop_private_key
-rw----- 1 root root 45 Apr 11 13:53 laptop_public_key
-rw----- 1 root root 45 Apr 11 13:53 phone_preshared_key
-rw----- 1 root root 45 Apr 11 13:53 phone_private_key
-rw----- 1 root root 45 Apr 11 13:53 phone_public_key
-rw----- 1 root root 45 Apr 11 13:53 server_private_key
-rw----- 1 root root 45 Apr 11 13:53 server_public_key
root@Anuk:/etc/wireguard#

```

This is the content of each file (in this particular case, you will get another result)

```

server_public o70mx+P/xRTtIAYw04msRW2IU31lreJ/EZ2ZLVeTEA8=
server_private yL3BldjULCz/zqitAReoPLTfDTcM8khZb0V1+g0ZtHg=
phone_public M0bXWPhF/qbDJeKP52gAM+igeR6csNnKf4pWZexYynM=
phone_private 4LGoHHvFUacRvXC19LvMldEKZWsLpY3SupcLg5Dx/V8=
phone_preshared RP37rPp6ARczPuL0j8NDMB41vEOLqHvtSjmVA1PYzoI=
laptop_public PQ1k3QVZoV100CKHBG8PGH5XIsakI+44W4aKYp/IKic=
laptop_private IMT6LFk00+iJbRrY0/yWTUnfh/rOb1AHrXtoh/yfV1A=
laptop_preshared CeBp3tPfoDBRX+5qpvguNVajs217nFeeJg30jJAYomg=

```

Generating server configuration

We need to create the server configuration:

```
# nano /etc/wireguard/server.conf
```

With this content

```

[Interface]
PrivateKey = <insert private server key>
# SaveConfig = false
ListenPort = 51900
PostUp = iptables -A FORWARD -i %i -j ACCEPT; iptables -A FORWARD -o %i -j
ACCEPT; iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables -D FORWARD -o %i -
j ACCEPT; iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE
### begin phone configuration ###
[Peer]
PublicKey = <insert phone public key>
# PresharedKey = <insert phone preshared key>
# AllowedIPs = 10.6.0.2/32

```

```
AllowedIPs = 0.0.0.0/0
PersistentKeepalive = 25
### end phone ###
### begin laptop configuration ###
[Peer]
PublicKey = <insert laptop public key>
# PresharedKey = <insert laptop preshared key>
AllowedIPs = 10.6.0.3/32
PersistentKeepalive = 25
### end laptop ###
```

In case you need to edit the server configuration later, you will have to stop the interface with this command:

```
#systemctl stop wg-quick@wg0.service
```

Gotchas

1. Be sure to replace the key values in the configuration for PrivateKey and PublicKey.
2. Note that the above configuration assumes you are using a wired ethernet connection on your RPi WireGuard server. **If you instead wish to use wifi (wlan0)**, change the above config to use **-o wlan0** in PostUp and PostDown. [See this forum post for additional information.](#)

Enabling IP Forwarding on the Server

Edit sysctl.conf on the Raspberry Pi with:

```
#nano /etc/sysctl.conf
```

Uncomment the line with "net.ipv4.ip_forward=1" and save changes.

```
# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
```

And enable the interface

```
#systemctl enable wg-quick@wg0
```

```
root@Anuk:/etc/wireguard# systemctl enable wg-quick@wg0
Created symlink /etc/systemd/system/multi-user.target.wants/wg-quick@wg0.service → /lib/systemd/system/wg-quick@.service.
```

Securing access to sensitive files

Ensure sensitive files are protected (root rw only)

```
# chown -R root:root /etc/wireguard/
# chmod -R og-rwx /etc/wireguard/*
```

```
root@Anuk:/etc/wireguard# ls -l /etc/wireguard/
total 36
-rw----- 1 root root 45 Apr 11 13:53 laptop_preshared_key
-rw----- 1 root root 45 Apr 11 13:53 laptop_private_key
-rw----- 1 root root 45 Apr 11 13:53 laptop_public_key
-rw----- 1 root root 45 Apr 11 13:53 phone_preshared_key
-rw----- 1 root root 45 Apr 11 13:53 phone_private_key
-rw----- 1 root root 45 Apr 11 13:53 phone_public_key
-rw----- 1 root root 45 Apr 11 13:53 server_private_key
-rw----- 1 root root 45 Apr 11 13:53 server_public_key
-rw----- 1 root root 507 Apr 11 14:03 wg0.conf
root@Anuk:/etc/wireguard#
```

Reboot the raspberry

```
$sudo reboot
```

Access to your raspberry and check the WireGuard interface is correctly created with the command 'ip addr'

```
daniel@Anuk:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether [REDACTED] brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.10/24 brd 192.168.1.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::ba27:ebff:fe71:a552/64 scope link
        valid_lft forever preferred_lft forever
3: wlan0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN group default qlen 1000
    link/ether [REDACTED] brd ff:ff:ff:ff:ff:ff
4: wg0: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1420 qdisc noqueue state UNKNOWN group default qlen 1000
    link/none
    inet 10.6.0.1/24 scope global wg0
        valid_lft forever preferred_lft forever
```

You need to access to your router to forwards 51900 to the internal IP address and port 51900 of the Raspberry protocol UDP. This is an example of configuration:

Port Forwarding List (Max Limit : 32)						
Service Name	Source IP	Port Range	Local IP	Local Port	Protocol	Add / Delete
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	TCP	+
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	TCP	-
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	TCP	-
WireGuard		51900	192.168.1.10	51900	UDP	-

Apply

Set Up the WireGuard Client for each client

Create a file for each client with the content

Phone access

Creating the file for the phone client

```
$sudo nano /etc/wireguard/phone.conf
```

With this content

```
[Interface]
Address = 10.6.0.2/24
PrivateKey = <insert client_private_key>
DNS = 10.6.0.1

[Peer]
PublicKey = <insert server_public_key>
#PresharedKey = <insert phone_preshared_key>
Endpoint = <insert vpn_server_address>:51900
AllowedIPs = 0.0.0.0/0, ::/0
```

Laptop access

Creating the file for the laptop client

```
$sudo nano /etc/wireguard/laptop.conf
```

With this content

```
[Interface]
Address = 10.6.0.3/24
PrivateKey = <insert client_private_key>
DNS = 10.6.0.1

[Peer]
PublicKey = <insert server_public_key>
#PresharedKey = <insert laptop_preshared_key>
Endpoint = <insert vpn_server_address>:51900
AllowedIPs = 0.0.0.0/0, ::/0
```

With the qrencode command we generate a qr code to import the configuration easily into the wireguard app.

```
# qrencode -t ansiutf8 < /etc/wireguard/phone.conf
```



Set Up the WireGuard Server

To configure WireGuard we have to create the wg0 file:

```
$sudo nano /etc/network/interfaces.d/wg0
```

With this information

```
# indicate that wg0 should be created when the system boots, and on ifup -a
auto wg0

# describe wg0 as an IPv4 interface with static address
iface wg0 inet static

    # static IP address
    address 10.6.0.1/24

    # before ifup, create the device with this ip link command
    pre-up ip link add wg0 type wireguard

    # before ifup, set the WireGuard config from earlier
    pre-up wg setconf wg0 /etc/wireguard/server.conf

    # Routing
    pre-up iptables -A FORWARD -i wg0 -j ACCEPT
    pre-up iptables -A FORWARD -o wg0 -j ACCEPT
    pre-up iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

    # route packages when the VPN interface is up
    #post-up sysctl --write net.ipv4.ip_forward=1

    # and stop routing when stopping the VPN interface
    #post-down sysctl --write net.ipv4.ip_forward=0

    # Routing post-down
    post-down iptables -D FORWARD -i wg0 -j ACCEPT
    post-down iptables -D FORWARD -o wg0 -j ACCEPT
    post-down iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE

    # after ifdown, destroy the wg0 interface
    post-down ip link del wg0
```

Adding unattended upgrades (optional)

If you are using third-party packages (e.g. via [PPAs](#)), the system has no idea about security updates for those packages. So you need to take an additional step and get them included manually.

Remember to determine the PPA Origin and Suite

The first goal is to determine the details from the PPA (or other external package type). This can be done by peeking in the **/var/lib/apt/lists** directory. Use the related files ending with **InRelease**, to see more details about the specific package.

```
$less /var/lib/apt/lists/deb.debian.org_debian_dists_unstable_InRelease
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA256

Origin: Debian
Label: Debian
Suite: unstable
Codename: sid
Changelogs: https://metadata.ftp-master.debian.org/changelogs/@CHANGEPATH@_changelog
Date: Sat, 13 Feb 2021 14:09:17 UTC
Valid-Until: Sat, 20 Feb 2021 14:09:17 UTC
Acquire-By-Hash: yes
No-Support-for-Architecture-all: Packages
Architectures: all amd64 arm64 armel armhf i386 mips64el mipsel ppc64el s390x
Components: main contrib non-free
Description: Debian x.y Unstable - Not Released
MD5Sum:
 6bdaa4e5a3643799b0e167b9ea2ada5a 3552076 contrib/Contents-all
 f51342e0e5924fc74d59091df718a8fe 18459 contrib/Contents-all.diff/Index
 4576bbc743438d86356249df91c538f0 313053 contrib/Contents-all.gz
 abef11ca13bb83ef89ec9d6215958b91 866771 contrib/Contents-amd64
 f8855bb6826889a428d0552ad7a61de4 63339 contrib/Contents-amd64.diff/Index
```

The two things we need from this file is the field *Origin* and *Suite*. These two strings have to be combined and provided to *unattended-upgrade*. It then understands that this PPA should be upgraded automatically.

```
// ${distro_codename}      Installed codename (eg, "buster")
Unattended-Upgrade::Origins-Pattern {
    // Codename based matching:
    // This will follow the migration of a release through different
    // archives (e.g. from testing to stable and later oldstable).
    // Software will be the latest available for the named release,
    // but the Debian release itself will not be automatically upgraded.
    // "origin=Debian,codename=${distro_codename}-updates";
    // "origin=Debian,codename=${distro_codename}-proposed-updates";
    // "origin=Debian,codename=${distro_codename},label=Debian";
    // "origin=Debian,codename=${distro_codename},label=Debian-Security";

    // Archive or Suite based matching:
    // Note that this will silently match a different release after
    // migration to the specified archive (e.g. testing becomes the
    // new stable).
    // "o=Debian,a=stable";
    // "o=Debian,a=stable-updates";
    // "o=Debian,a=proposed-updates";
    // "o=Debian Backports,a=${distro_codename}-backports,l=Debian Backports";
    // "o=Debian,a=unstable";
};
```

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

Origin: packages.azlux.fr
Label: packages.azlux.fr
Suite: stable
Codename: buster
Date: Sun, 07 Feb 2021 06:52:08 UTC
Architectures: i386 amd64 armhf arm64
Components: main
Description: Apt repository for Azlux projects
MD5Sum:
  21f8d385b524539fa4d0e3e37957162f 5897 main/binary-i386/Packages
  966ab7df05849ce1ca53677a8e86483c 2391 main/binary-i386/Packages.gz
  eaf5a21a0d5b1ea4adc7bd6816eeac0a 149 main/binary-i386/Release
  4a76514e6cbdb12a7945eba88015c039 8852 main/binary-amd64/Packages
  c2974416f6437c757f74b93c9378ca7c 3380 main/binary-amd64/Packages.gz
  799f5061c6a592928c521460be7d8953 150 main/binary-amd64/Release
  4d7aa3952933a55f0a04e6103bfbd99c 7424 main/binary-armhf/Packages
  18db2fad3c1fa05c0c014f20c97f08e7 2887 main/binary-armhf/Packages.gz
  8dde8c649c54f2e2191b61e170613c7b 150 main/binary-armhf/Release
```

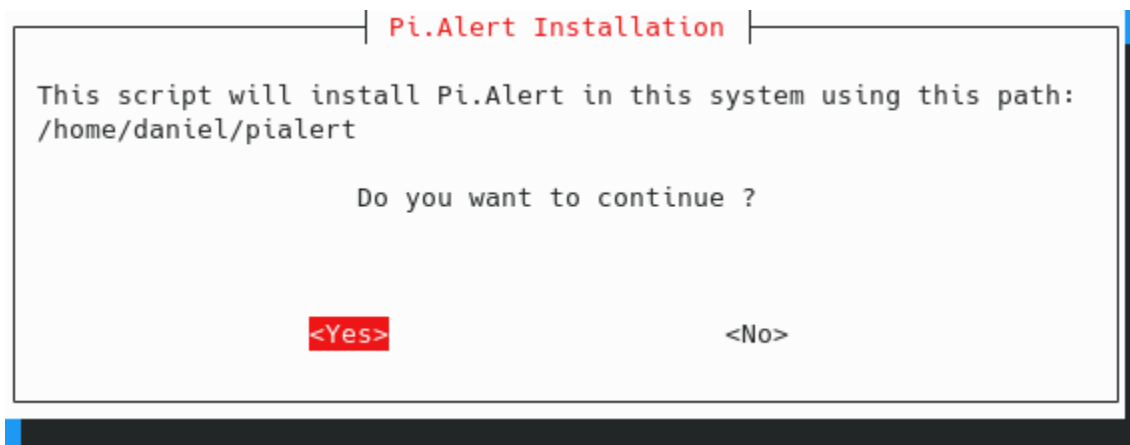

Installing Pi.Alert, a Wi-Fi/LAN intruder detector (optional)

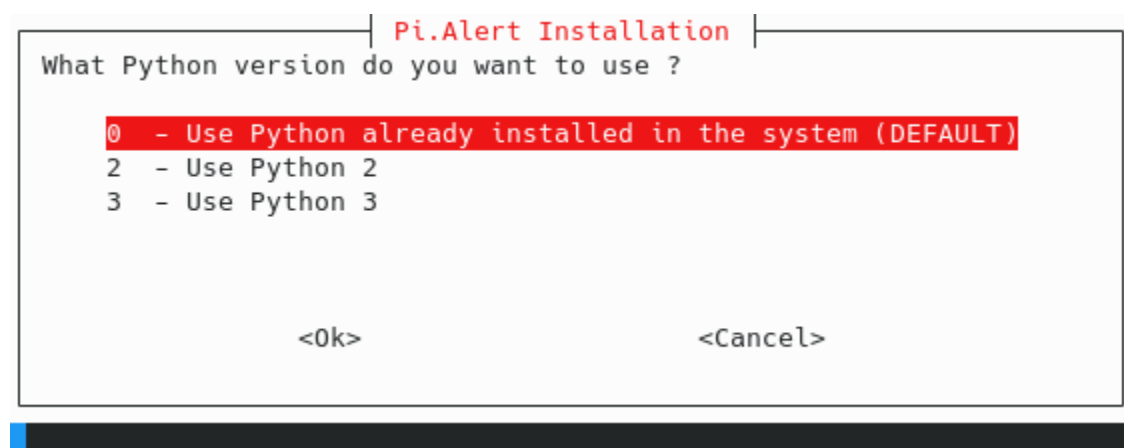
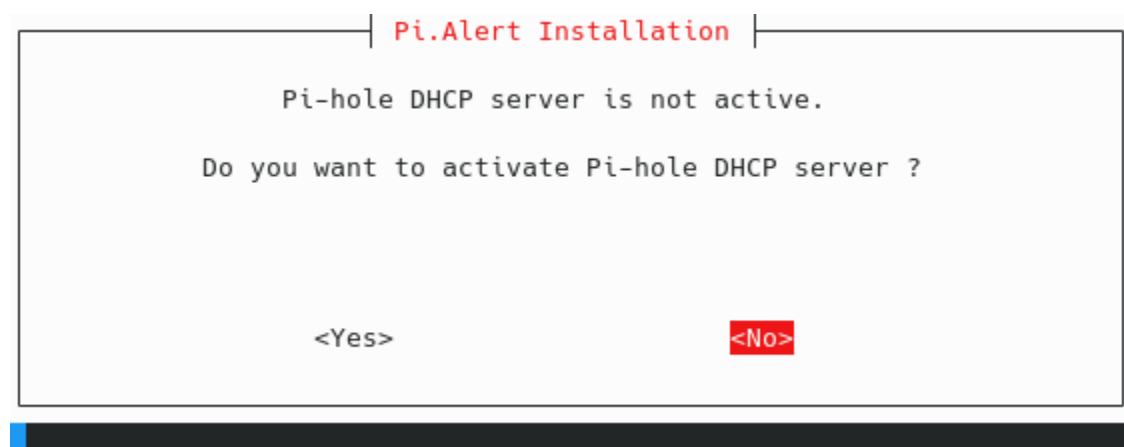
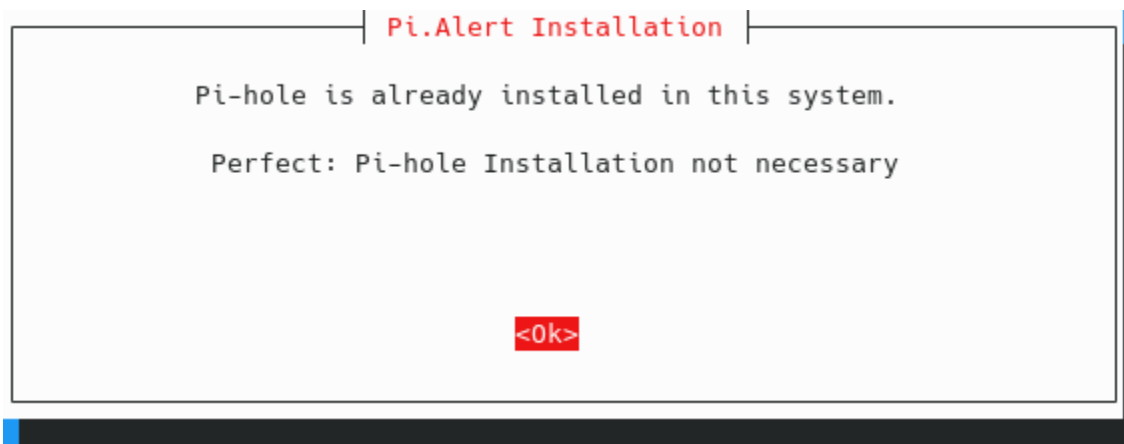
Pi.Alert is a nice and small project that provide a Wi-Fi and LAN intruder detector. Check the devices connected and alert you with unknown devices. It also warns of the disconnection of "always connected" devices. It can be located at <https://github.com/pucherot/Pi.Alert>

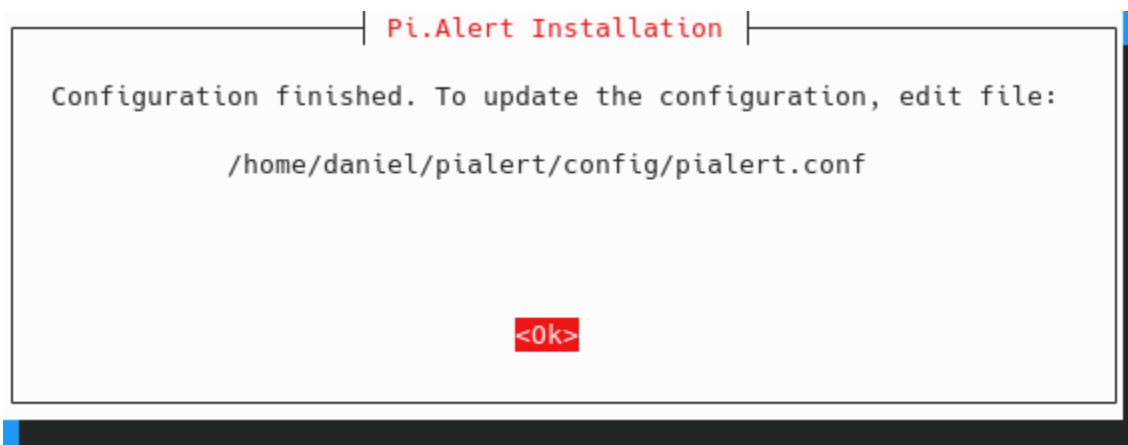
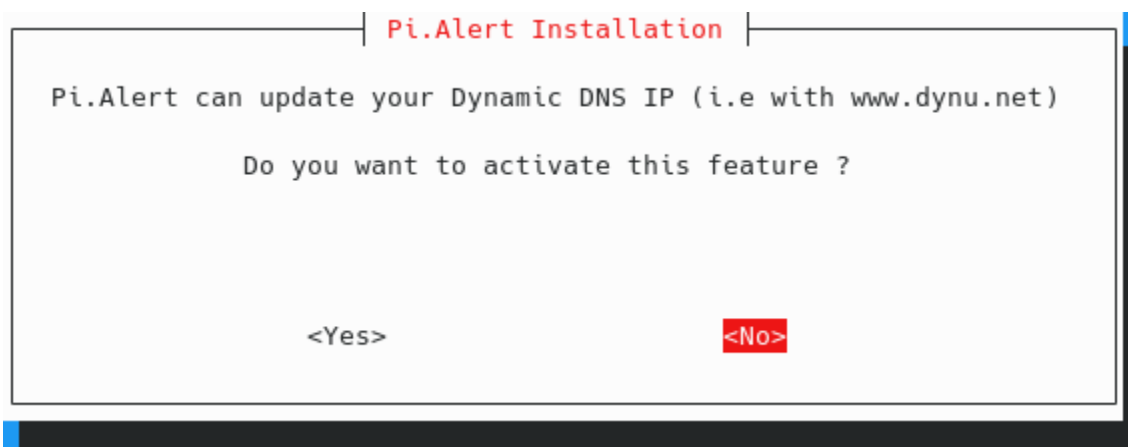
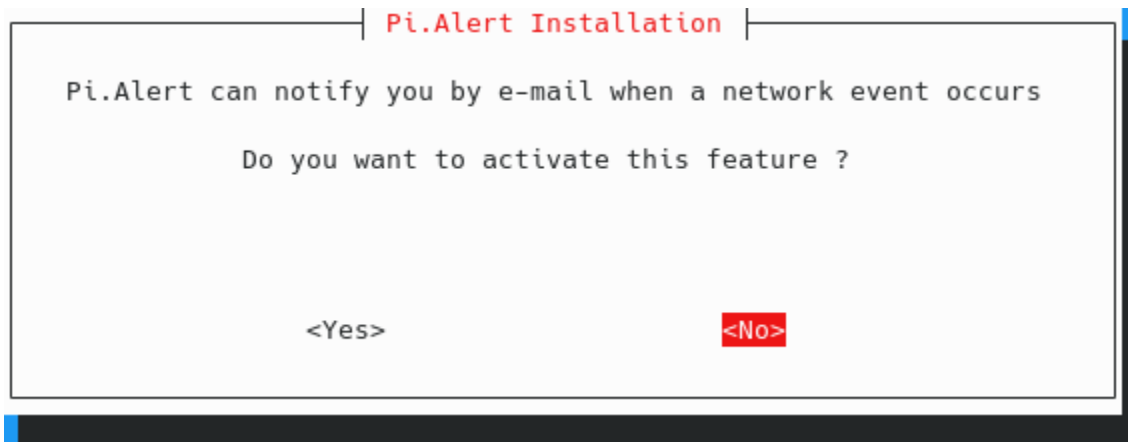
The installation is quite simple. Just run the following script (It will ask for the sudo password)

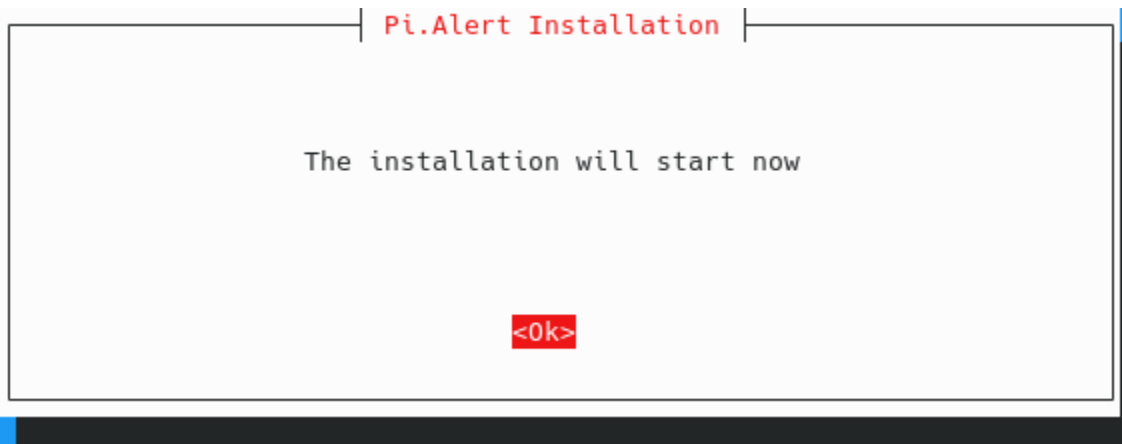
```
$curl -sSL  
https://github.com/pucherot/Pi.Alert/raw/main/install/pialert_install.sh |  
bash
```

Here you have the screens you will have with the options I have choose for my system:









Save this information to access to the Pi.Alert server:

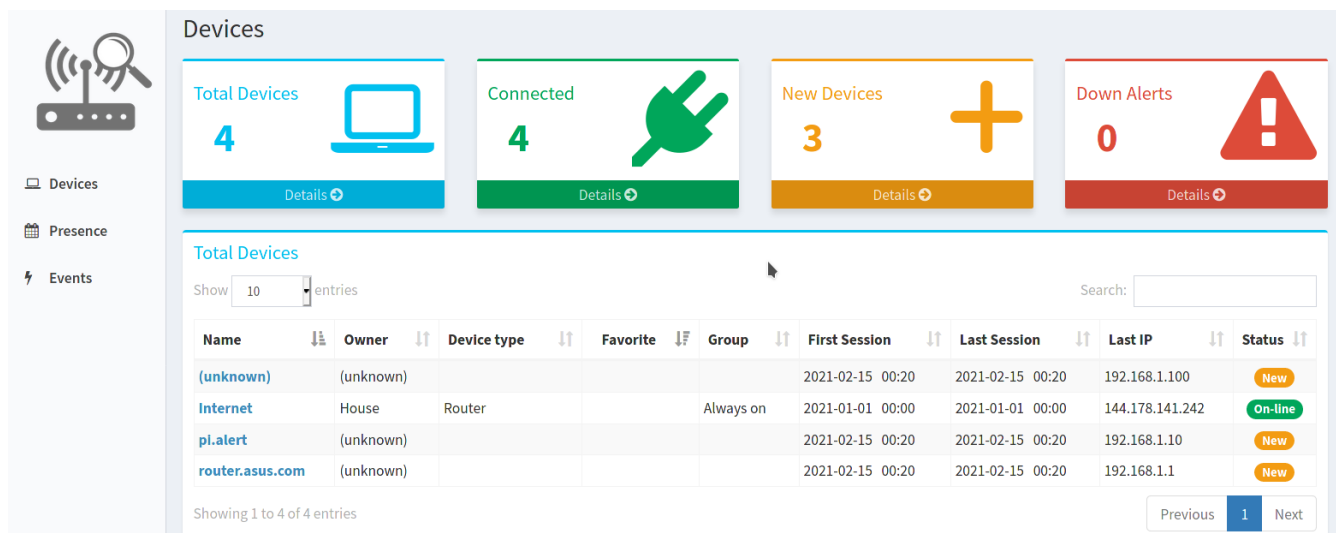
```
DONE!!!

- Adding jobs to the crontab...
- Setting permissions...
- Publishing Pi.Alert web...
- Configuring http://pi.alert/ redirection...
- Restarting lighttpd...

-----
Installation process finished
-----

Use: - http://pi.alert/
      - http://192.168.1.10/pialert/
To access Pi.Alert web
```

And this is how Pi.Alert looks like

A screenshot of the Pi.Alert web interface. The left sidebar has a router icon and a menu with "Devices", "Presence", and "Events". The main content area has a "Devices" header. Below it are four summary cards: "Total Devices 4" (blue), "Connected 4" (green), "New Devices 3" (orange), and "Down Alerts 0" (red). Below these is a "Total Devices" section with a search bar and a table of devices. The table has columns: Name, Owner, Device type, Favorite, Group, First Session, Last Session, Last IP, and Status. The status column has icons: "New" (orange), "On-line" (green), and "New" (orange).

Name	Owner	Device type	Favorite	Group	First Session	Last Session	Last IP	Status
(unknown)	(unknown)				2021-02-15 00:20	2021-02-15 00:20	192.168.1.100	New
Internet	House	Router		Always on	2021-01-01 00:00	2021-01-01 00:00	144.178.141.242	On-line
pi.alert	(unknown)				2021-02-15 00:20	2021-02-15 00:20	192.168.1.10	New
router.asus.com	(unknown)				2021-02-15 00:20	2021-02-15 00:20	192.168.1.1	New

Here you will find more information regarding device management:

https://github.com/pucherot/Pi.Alert/blob/main/docs/DEVICE_MANAGEMENT.md

To have Pi.alert updated, we have to add to the cron

```
$sudo crontab -e
```

And paste this command at the end of the file

```
3 1 4 */1 * curl -sSL  
https://github.com/pucherot/Pi.Alert/raw/main/install/pialert_update.sh |  
bash
```

Attention: Since I start to write this guide, it seems Pi.Alert project has not quite activity, and others users has started to create forks that are more updated:

- Installation in a server: [leiweibau/Pi.Alert](#)
- Container version: [jokob-sk/Pi.Alert](#)

Monitoring tool (optional)

We have several options to monitor our system. We can use light applications or heavier ones with a lot of functionalities but with more resource consumptions.

When I started to write this guide I choose [RPi-Monitor](#). An simple and light monitor with all the basic parameters. The problem I found is that is not longer maintained. The last version is from august 2017, so I decided to find another light solution and I found [NetData](#), an open source tool designed to collect real-time metrics, such as CPU usage, disk activity, bandwidth usage, website visits, etc., and then display them in live, easy-to-interpret charts. It offers the possibility to create a free cloud account to complement the netdata agent to provide:

- Infrastructure level dashboards (each chart aggregates data from multiple nodes)
- Central dispatch of alert notifications
- Custom dashboards editor
- Intelligence assisted troubleshooting, to help surface the root cause of issues

The offer some paid functionalities but they claim the free account will be free forever.

In case you need a powerful solution to monitor devices with a lot of integrations and functionalities like Machine Learning, you can use [Grafana](#). There is a [sandbox](#) where you can play with Dashboards.

Installing RPi-Monitor (deprecated)

Rpi-Monitor let us to monitor basic parameters like temperature, CPU load, disk space, and packages upgradables. To install we need to execute the following commands:

```
$sudo apt-get install dirmngr
$sudo apt-key adv --recv-keys --keyserver hkp://keyserver.ubuntu.com:80
2C0D3C0F
$sudo wget http://goo.gl/vewCLL -O /etc/apt/sources.list.d/rpimonitor.list
$sudo apt-get update
$sudo apt-get install rpimonitor
```

Configure RPi-Monitor to show network statistics:

```
$sudo nano /etc/rpimonitor/template/network.conf
```

Uncomment the first two sections that start with “dynamic.10” and “dynamic.11”. Comment out the third, fourth and fifth lines in the next section that start with “web.status.1” and uncomment the last one. Uncomment the next section that starts with “web.statistics.1”. Exit and save.

```
dynamic.10.regex=(. *)
dynamic.10.postprocess=$1*-1
dynamic.10.rrd=DERIVE
dynamic.10.max=0

dynamic.11.name=net_send
dynamic.11.source=/sys/class/net/eth0/statistics/tx_bytes
dynamic.11.regex=(. *)
dynamic.11.postprocess=
dynamic.11.rrd=DERIVE
dynamic.11.min=0

web.status.1.content.8.name=Network
web.status.1.content.8.icon=network.png
#web.status.1.content.8.line.1="To activate network monitoring, edit and customize <font color='#AA0000'><b>network.conf</b></font>
#web.status.1.content.8.line.2="Help is available in man pages:"
#web.status.1.content.8.line.3="<font color='#AA0000'><b>man rpimonitor</b></font> or <font color='#AA0000'><b>man rpi</b></font>"
web.status.1.content.8.line.1="Ethernet Sent: <b>"+KMG(data.net_send)+"<i class='icon-arrow-up'></i></b> Received: <b>"+KMG(data.net_recv)+"<i class='icon-arrow-up'></i></b>"

web.statistics.1.content.2.name=Network
web.statistics.1.content.2.graph.1=net_send
web.statistics.1.content.2.graph.2=net_received
web.statistics.1.content.2.graph_options.yaxis={ tickFormatter: function (v) { if (Math.abs(v) > 1048576) return (Math.$
web.statistics.1.content.2.ds_graph_options.net_send.label=Upload bandwidth (bytes)
web.statistics.1.content.2.ds_graph_options.net_send.lines={ fill: true }
```

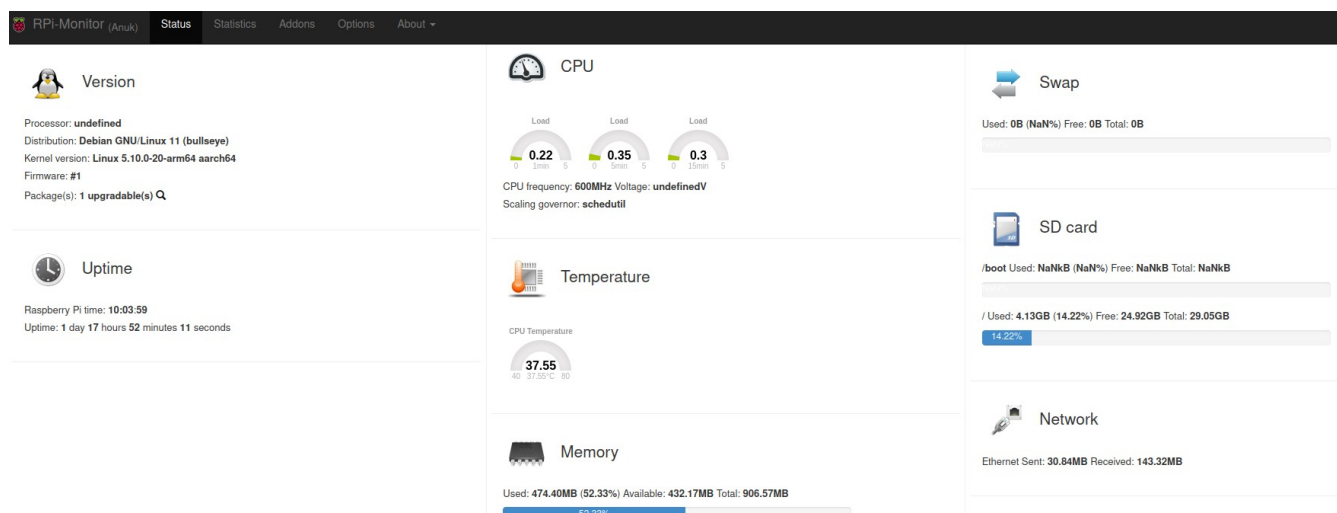
Restart RPi-Monitor.

```
$sudo service rpimonitor restart
```

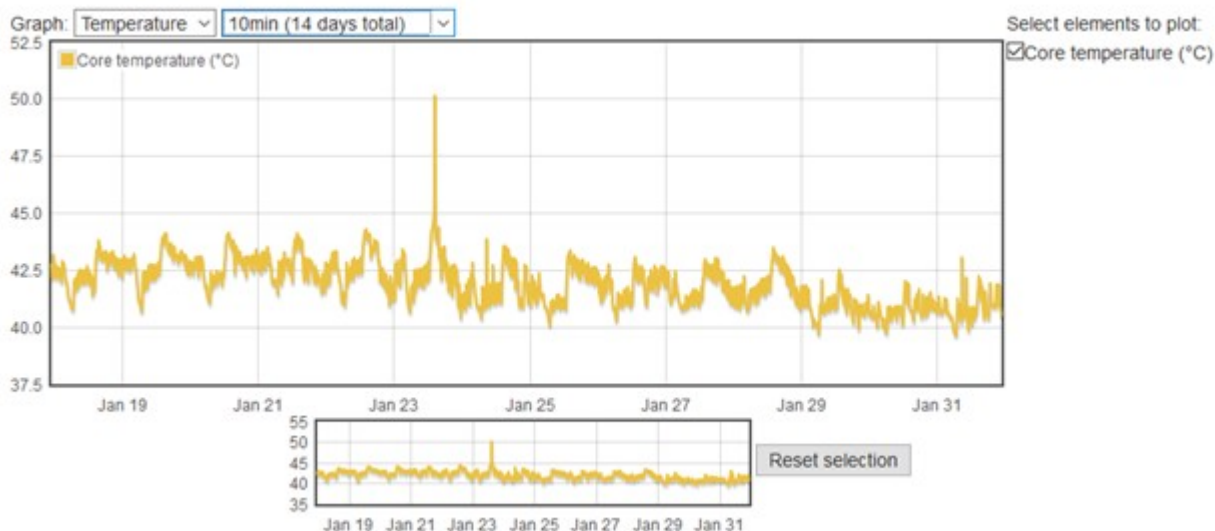
Update RPi-Monitor package status:

```
$sudo /etc/init.d/rpimonitor update
```

Check the RPi-Monitor web page at <http://<IPAddress>:8888>



You now have a web dashboard of your server's status, and there is a historical view under Statistics. This can be helpful for monitoring and troubleshooting. Here is a view in Statistics of temperature over 14 days:



Installing Netdata (recommended)

Netdata helps to monitor and troubleshoot several kind of devices and applications they run, like raspberry pi and Pi-hole.

After a quick installation and with no additional configuration required, you will be able to see device parameters like CPU load, memory and disk usage, bandwidth... Netdata collects about 1.500 metrics every second.

On Raspberry Pi, the installation is done by one command script. This script asks you to install dependencies and compile Netdata from the source:

```
$wget -O /tmp/netdata-kickstart.sh https://my-netdata.io/kickstart.sh && sh /tmp/netdata-kickstart.sh --disable-telemetry
```

Parameters:

- Use a stable release: '--stable-channel'
- Do not send anonymous statistics, '--disable-telemetry'
- No automatic updates: '--no-updates'

As you can see, the command line uses a nightly version (more updated), I do not want to send anonymous statistics and I want to receive automatic updates.

When we run this script, it will ask for you administrator account and install all the required packages.

Once finished, we have to modify a configuration file in order to enable the temperature sensor monitoring. We have to uncomment the `sensors=force` line from the `charts.d.conf` configuration file. The installation path differs if we have Debian or Raspbian

Debian:

```
$cd /etc/netdata
$sudo ./edit-config charts.d.conf
```

Raspbian:

```
$cd /opt/netdata
$sudo cp usr/lib/netdata/conf.d/charts.d.conf etc/netdata/
$cd etc/netdata
$sudo ./edit-config charts.d.conf
```

```
# ap=yes
# apcupsd=yes
# libreswan=yes
# nut=yes
# opensips=yes

# -----
# THESE NEED TO BE SET TO "force" TO BE ENABLED

# Nothing useful.
# Just an example charts.d plugin you can use as a template.
# example=force
sensors=force
```

Once modified we have to restart the service to enable Raspberry Pi temperature sensor monitoring:

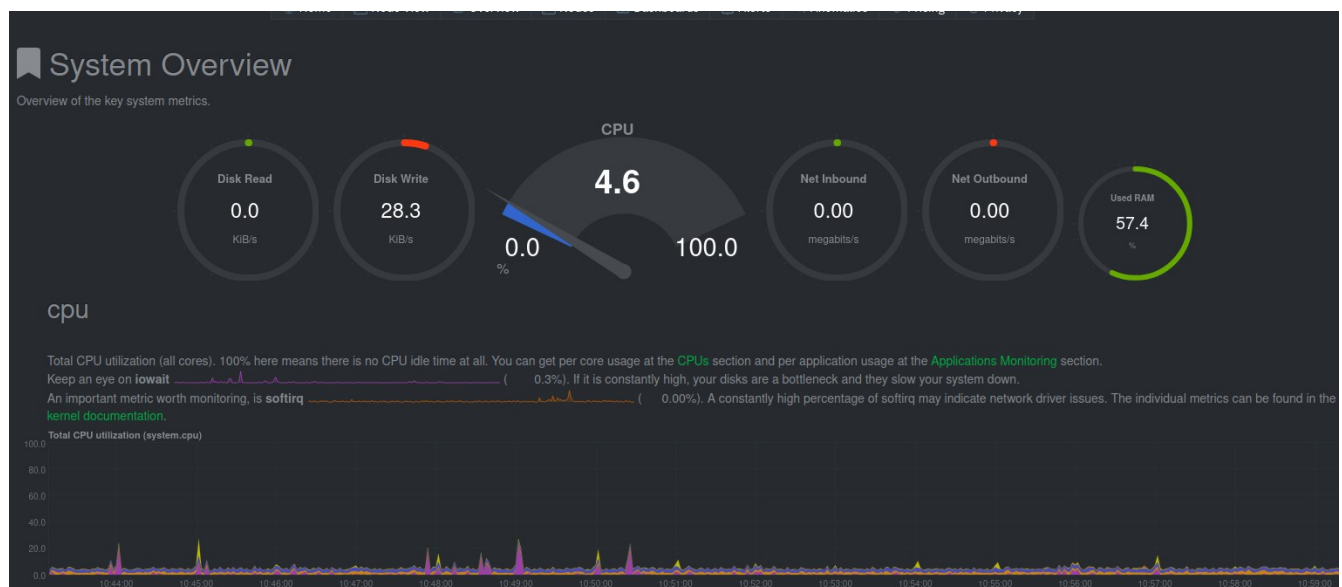
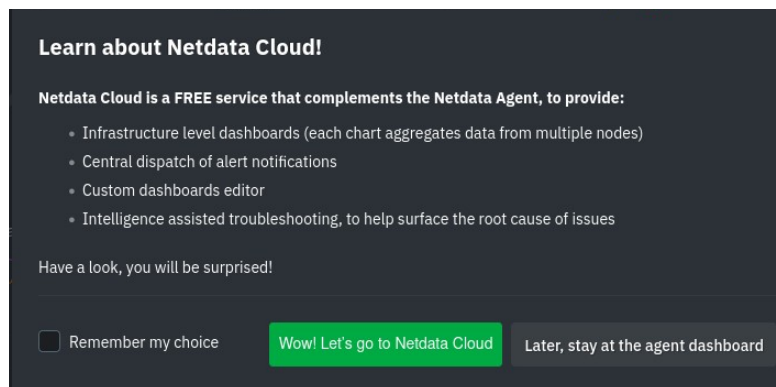
```
$sudo systemctl restart netdata
```

Another improvement suggested by Netdata is to increase the allocation to increase historical metrics. As they said on their website, I will recommend to use their [database sizing calculator](#) and [guide on storing historical metrics](#) to help you determine the right setting for your Raspberry Pi.

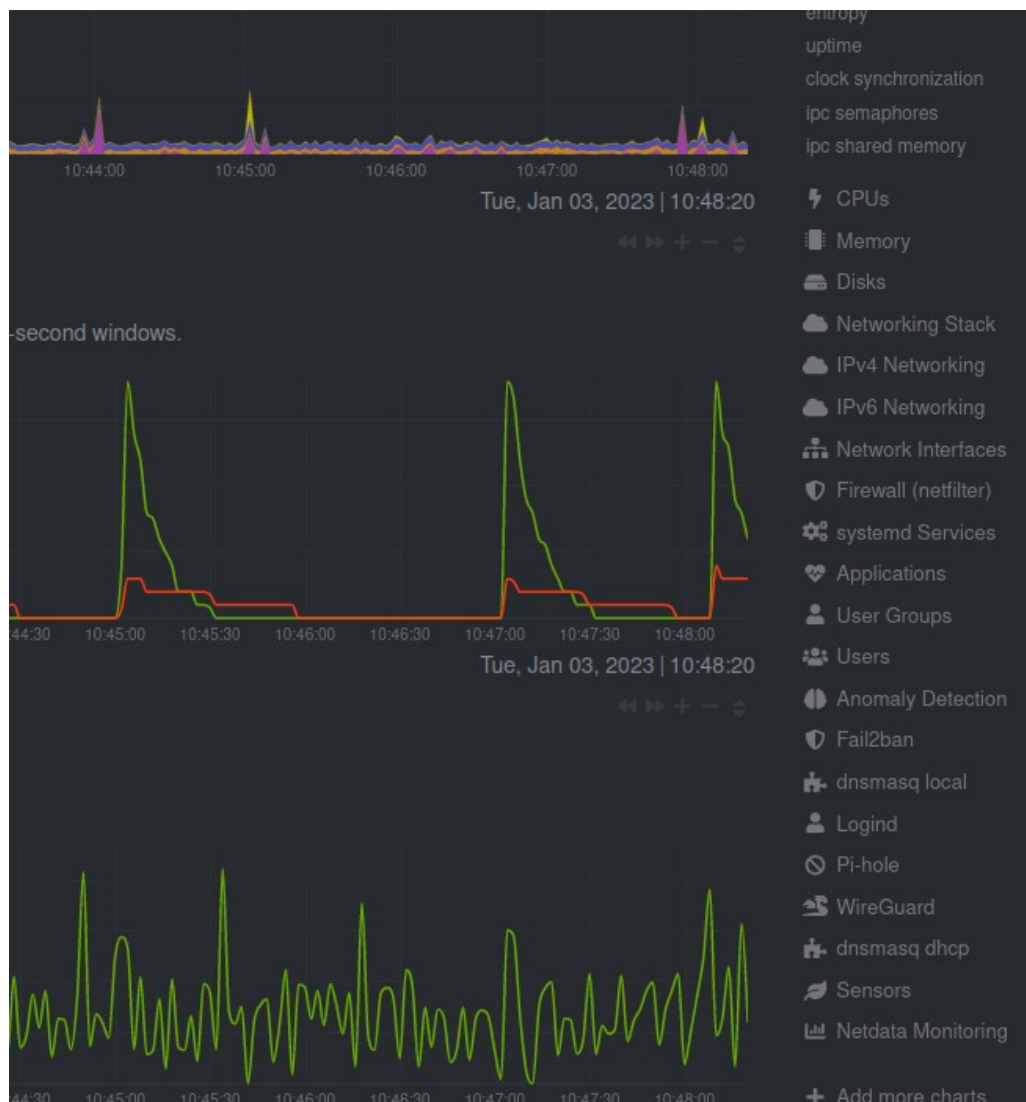
Once installed, we can point our browser to our raspberry pi IP with the port 19999 ([http://\[our raspberry pi IP\]:19999](http://[our raspberry pi IP]:19999))

The first login will show a reminder to create a cloud account to access your data through Internet and provide:

- Infrastructure level dashboards (each chart aggregates data from multiple nodes)
- Central dispatch of alert notifications
- Custom dashboards editor
- Intelligence assisted troubleshooting, to help surface the root cause of issues



Through the right menu you can browse different metrics Netdata collects, from the device like CPU, memory, Disks, network, temperature (under sensors Section)... to metrics from the applications installed, like Fail2ban, firewall, Pi-hole, WireGuard



In case you create a cloud account, a command with an agent to be installed is provided in order to grab the data and send to the cloud.

I will also strongly recommend the use of their app to monitoring remotely our systems.

Grafana

Grafana has a web where explain step by step how to install an agent for [raspberry pi](#).

Securing the Raspberry

In case we have a raspberry with a Wi-Fi interface and we do not use it, we can disable it. To completely disable the onboard WiFi from the firmware on the Pi3 / Pi4, add in /boot/config.txt

```
dtoverlay=disable-wifi  
dtoverlay=pi3-disable-wifi
```

Or can add to this two lines

```
blacklist brcmfmac  
blacklist brcmutil
```

to the module blacklist

```
$sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

Backup & restore

Method 1: Copy the SD Card Image

The easiest way is dump the SD Card or USB to an image. Let's assume the SD is at the sdb partition.

```
$sudo dd bs=4M if=/dev/sdb of=raspbian_bck.img conv=fdatasync  
status=progress
```

Method 2: Zip the Home Directory

(To Be Done)

Method 3: Scheduled Backups

(To Be Done)

Bonus track. Password manager: Vaultwarden

When I started to use password managers, I choose a light an open-source solution, [keepass](#). I have written on my blog an article a few years ago [how to set and use passwords in a safety way](#) using this solution. With the time, a new solution has appear with more functionalities and open-source also: [Bitwarden](#).

Instead Bitwarden I choose [Vaultwarden](#), an alternative implementation of the Bitwarden server API written in Rust and compatible with upstream [Bitwarden clients](#), perfect for self-hosted deployment where running the official resource-heavy service might not be ideal.

The basic installation of Vaultwarden can be done following the instructions from [here](#). I decided to use another raspberry pi since the first one has dedicated exclusively to privacy filter and VPN Server and I prefer to not overload with more services running in the same device.

To access to the password manager server a proxy manager is needed (Nginx). We will need a container manager, portainer, since Vaultwarden and Nginx (application and database) are running within containers.

As we did it before, a DDNS service is needed in order to reach the raspberry from Internet. This also will help us to have a secure connection since it is mandatory to access the password manager. It will be foolish to access your passwords with an http connection.

How to update components

Here I detailed the steps to update all the components we have inside this raspberry pi. The update procedure is not always explained in the installation pages, so I decided to describe how to update each component.

Portainer

Updating Portainer can be done in 4 steps from the raspberry pi command line:

1. stop portainer dock

```
$docker stop portainer
```

2. Remove the container

```
$docker rm portainer
```

3. Pull the new version

```
$docker pull portainer/portainer-ce:latest
```

4. Run the docker

```
$sudo docker run -d -p 9000:9000 --name=portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce:latest
```

Containers

Every certain time we should check if a new version has been released and evaluate if we want to update it. The update can be done in two ways, through portainer or by command line.

Updating through portainer

1. Select **Containers**, then **stop** the container that you'd like to update.
2. Select the container, and you'll see a button named **Recreate**. By selecting this button, the container will take the persistent data and recreate the container. Keep in mind that the only data that will stay on the container is data that was mapped to a volume. This is explained in greater detail in the video above.
3. Select **Pull latest image**, then **Recreate**.
4. When this process is finished, the container will be recreated with the latest image. Select the container and **Start** it. The status will change to **running**.
5. The container will now exist with the newest released version!

Note 1: It will take some time. Be patient.

Note 2: You can remove old container images in order to save space.

Updating command line

First we pull the latest image

```
$docker pull vaultwarden/server:latest  
sudo docker run -d --name bitwarden \
```

```
--restart=always \  
-v /bw-data/:/data/ \  
-p 127.0.0.1:8080:80 \  
-p 127.0.0.1:3012:3012 \  
vaultwarden/server:latest
```

And with this command we launch it removing the previous one

```
$docker run --rm -it --mount type=volume,source=vaultwarden-rclone-  
data,target=/config/ ttionya/vaultwarden-backup:latest rclone config
```

Updating Nginx

The reverse proxy should be updated in this order. First the app

```
$sudo docker update --restart always nginx_app_1
```

And later the database

```
$sudo docker update --restart always nginx_db_1
```


Bibliography

Base

1. <https://www.forbes.com/sites/marketshare/2012/03/05/if-youre-not-paying-for-it-you-become-the-product/>
2. <https://medium.com/change-your-mind/if-you-are-not-paying-for-the-product-you-are-the-product-4dbc15b9a3f2>
3. <https://clearcode.cc/blog/what-is-data-broker/>
4. <https://www.peakyou.com/>
5. <https://www.toptenreviews.com/best-people-search-services>
6. <https://thesmashy.medium.com/building-a-pihole-for-privacy-and-performance-f762dbcb66e5>
7. <https://www.thetechherald.com/tech-news/raspberry-pi-raspbian-os-gets-a-microsoft-repo-without-any-notification-heres-how-to-remove-concerns-of-telemetry-data-collection/>
8. <https://arstechnica.com/gadgets/2021/02/raspberry-pi-os-added-a-microsoft-repo-no-its-not-an-evil-secret/>
9. <https://betanews.com/2021/02/08/linux-based-raspberry-pi-os-secret-microsoft-repo/>
10. <https://www.raspberrypi.org/documentation/hardware/raspberrypi/bootmodes/msd.md>

Pi-hole

<https://dev.to/jldohmann/the-ultimate-ad-blocker-configuring-pi-hole-with-unbound-dns-20eo>
<https://www.bentasker.co.uk/blog/the-internet/703-scaling-pihole-to-cope-with-huge-query-rates>
<https://www.vcloudinfo.com/2019/02/my-pi-hole-is-out-of-space-how-to-free-up-space-to-upgrade.html>
<https://firebog.net>
<https://github.com/topics/pihole-ads-list>
<https://blog.mandos.io/p/ultimate-guide-setup-raspberry-pi-hole-boost-privacy-browsing-speed>
<https://avoidthehack.com/best-pihole-blocklists>

Unbound

<https://docs.pi-hole.net/guides/dns/unbound/>

<https://github.com/pi-hole/docs/issues/207>

[https://www.reddit.com/r/pihole/comments/d9j1z6/unbound as recursive dns server slow performance/](https://www.reddit.com/r/pihole/comments/d9j1z6/unbound_as_recursive_dns_server_slow_performance/)

Fail2ban

<https://www.niih.de/how-to-upgrade-fail2ban-to-support-ipv6/>

Unattended-Upgrades

<https://pimylifeup.com/unattended-upgrades-debian-ubuntu/>

<https://www.zealfortechology.com/2018/08/configure-unattended-upgrades-on-raspberry-pi.html>

<https://wiki.debian.org/UnattendedUpgrades>

<https://kb.iu.edu/d/aews>

<https://linux-audit.com/upgrading-external-packages-with-unattended-upgrade/>

<https://pimylifeup.com/unattended-upgrades-debian-ubuntu/>

Pi.Alert

<https://github.com/pucherot/Pi.Alert>

<https://github.com/jokob-sk/Pi.Alert>

<https://github.com/leiweibau/Pi.Alert>

WireGuard

<https://wireguard.how/server/debian/>

<https://engineerworkshop.com/blog/how-to-set-up-wireguard-on-a-raspberry-pi/>

<https://www.cyberciti.biz/faq/debian-10-set-up-wireguard-vpn-server/>

<https://serversideup.net/generating-wireguard-qr-codes-for-fast-mobile-deployments/>

<https://serversideup.net/courses/gain-flexibility-and-increase-privacy-with-wireguard-vpn/>

<https://www.procustodibus.com/blog/2020/12/wireguard-site-to-site-config/>

<https://wiki.debian.org/SimplePrivateTunnelVPNWithWireGuard>

<https://kirelos.com/set-up-your-own-wireguard-vpn-server-on-debian/>

Monitoring tools

<https://github.com/XavierBerger/RPi-Monitor>

<https://learn.netdata.cloud/guides/monitor/pi-hole-raspberry-pi>

<https://grafana.com/tutorials/install-grafana-on-raspberry-pi/>

Backup

<https://raspberrypexpert.com/how-to-backup-raspberry-pi/>

Vaultwarden

<https://www.wundertech.net/how-to-self-host-bitwarden-on-a-raspberry-pi/>

<https://medium.com/codex/complete-self-hosted-bitwarden-for-raspberry-pi-24b59c3b02df>

<https://pimylifeup.com/raspberry-pi-bitwarden/>

<https://phoenixnap.com/kb/update-docker-image-container>

<https://www.wundertech.net/how-to-update-a-docker-container-using-portainer/>

<https://github.com/NginxProxyManager/nginx-proxy-manager>

Others

<https://serverfault.com/questions/1006595/cannot-setup-wireguard-vpn>

<https://wireguard.how/server/debian/>

<https://raspberrytips.com/disable-wifi-raspberry-pi/>

<https://www.raspberrypi.org/forums/viewtopic.php?t=290611>

<https://www.raspberrypi.org/forums/viewtopic.php?f=63&t=138610>

<https://www.simonpreston.dev/2019/03/01/using-the-raspberry-pi-as-a-dhcp-server/>

<https://www.privacyguides.org/>