Ubuntu-MD Nov 17, 2018 Presentation

Pi3 as a Wireless Access Point, USB Network Print, Proxy (squid), OpenVPN (PiVPN), Media (Kodi), Security motion detection or Webserver.

We will cover in detail the wireless access point and network print server (image of this server setup with Raspian is available).

Documentation for Proxy, OpenVPN, Media, Security motion detection (motioneye on dedicated Pi3), media or Webserver will be provided. I am currently using some of them and will provide examples of my setups.

Using Raspberry Pi 3 running Stretch for a WiFi router

There are several tutorials out there about turning your Rapberry Pi into a WiFi access point, but they all seem to written for Raspbian Jessie or earlier. There are a few changes in Stretch, the most recent version of the Pi's operating system, that seem to break these tutorials.

Here I will describe what I did to turn my Raspberry Pi 3 into a WiFi router. This should all work for any Pi running Stretch, though I have only tested it on the 3. My desired routing setup is to plug an Ethernet cable from the Pi into a modem, so that the Internet connection through that modem can be shared by several wireless devices.

Differences between what I found in previous tutorials and what I needed to do related mainly to the switch from configuring the WiFi interface device via the `/etc/network/interfaces` file (used under Jessie and prior versions of Raspbian) to using the `/etc/dhcpcd.conf` file (as used by Stretch). Here's how I set up my router:

Software

Over the course of this tutorial, you'll need to `sudo apt-get install` three packages. But don't install them just yet! Installing `iptables-persistent` as the final step causes it to notice you've changed the defaults already, and prompts you to save your work. It's very convenient! The three packages are:

- hostapd
- dnsmasq
- iptables-persistent

The first is the access point software, which creates and broadcasts a wireless network to which other computers can connect. The second manages the addresses that will be handed out to computers connecting on the new network. And the third makes sure that settings about how to route packets between the wired Ethernet device and the WiFi device are saved, so that the connection will be restored when the Pi is restarted.

Configure hostapd

First, install `hostapd` via

```
sudo apt-get install hostapd
```

You'll configure two files to make your Pi into an access point. First, create the file `/etc/hostapd/hostapd.conf` for editing via `nano`:

```
sudo nano /etc/hostapd/hostapd.conf
```

This file will hold the settings for your access point. Just copy paste this code, but put your own choices in for `ssid` (this will be the name of the network you're creating) and `wpa_passphrase` (this is the password for connecting to your new network):

```
interface=wlan0
ssid=network_name_here
hw_mode=g
channel=7
ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]
wmm_enabled=0
macaddr_acl=0
auth_algs=1
wpa=2
ignore_broadcast_ssid=0
wpa_passphrase=network_password_here
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

If you'd like to learn more about these configuration options, check the documentation.

Now you need to tell `hostapd` to read this configuration file when it starts. To do so, edit the `/etc/default/hostapd` file:

```
sudo nano /etc/default/hostapd
```

and change the line that reads

```
DAEMON_CONF=""
```

to

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

Configure dnsmasq

Install `dnsmasq` via

```
sudo apt-get install dnsmasq
```

You have to configure two files in order for `dnsmasq` to start assigning IP adresses to devices that connect to your newly configured access point. First, open the `dhcpcd` configuration file in order to assign your Pi a static IP address on the WiFi device. This might not be necessary, but it does give you an address where you can wirelessly SSH into the Pi, so you'll no longer need it to be connected to a monitor and keyboard.

Note that before you can SSH into your Pi, you must enable it for SSH, either by creating an empty file called `ssh` at the top level of the `boot` partition, or by connecting the Pi to a keyboard and monitor, opening a terminal, and using the menu under the `sudo raspi-config` command. Googling "headless Raspberry Pi ssh" will get you answers here.

Back to the main story. At the terminal, enter `sudo nano /etc/dhcpcd.conf`. At the bottom of the file, enter the following lines, but change the `ip_address` and `routers` to whatever address you want to assign to your Pi. (works on 192.168.0.0 network)

```
interface wlan0
static ip_address=10.0.0.1
static routers=10.0.0.1,192.168.0.1
static domain_name_servers=8.8.8.8,8.8.4.4
```

Now, edit the `dnsmasq` config file so it knows which addresses to assign to devices that connect on WiFi. Open the file with the command `sudo nano /etc/dnsmasq.conf`, and add the following lines at the end, except that `dhcp-range` should indicate whatever addresses you want to be assignable on your network:

```
interface=wlan0
domain-needed
bogus-priv
dhcp-range=~~10.0.1.20,10.0.1.20,12h~~
```
(incorrect should be 10.0.0.20,10.0.0.25,12h)

Configure IP routing

At this point you have a wireless access point, but it doesn't connect to your Pi's wired Ethernet port, which is its connection to the Internet. Making this final link is pretty easy. First, enable routing by opening the config file with `sudo nano /etc/sysctl.conf`, and add the following line:

```
net.ipv4.ip_forward=1
```

Then, at the command line, tell the Pi how to decide which packets get routed. That's done by executing these three commands:

```
sudo iptables -t nat -A  POSTROUTING -o eth0 -j MASQUERADE
sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j
ACCEPT
sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

Finally, make these routing settings permanent by installing the `iptables-persistent` package, and saying yes when prompted to save the current settings:

```
sudo apt-get install iptables-persistent
```

Setting up Network Printer on Raspberry Pi:

This section will carry us through a series of steps that will culminate in the installation of CUPS on your Raspberry Pi.

Step 1: Upgrade the Pi

Kind of a ritual, first thing for all of my projects is updating the Raspberry Pi, by doing this you ensure your pi has all the latest updates to the OS you are working with.

To do this we use;

```
sudo apt-get update
sudo apt-get upgrade
```

With this done, reboot the pi using;

```
sudo reboot
```

Wait for the reboot process and login again

Step 2: Install Print Server Software CUPS

With the update done the next line of action is to install our print server software CUPS.

To do this run;

```
sudo apt-get install cups
```

This will take some time but will install CUPS and other dependencies like Samba, perl and several other software or libraries.

Step 3: Configure CUPS

With Installation done, its time to check out the configuration file of CUPS. Several settings that generally affect how cups works, like the port on which cups communicate which is by default 631, **port can be changed** here.

The config file can be accessed using;

```
sudo nano /etc/cups/cupsd.conf
```

Change/add the following lines to the configuration file.

```
# Only listen for connections from the local machine.

#Listen localhost:631

#CHANGED TO LISTEN TO LOCAL LAN

Port 631
```

```
# Restrict access to the server...
<Location />
  Order allow,deny
  Allow @Local
</Location>


# Restrict access to the admin pages...
<Location /admin>
  Order allow,deny
  Allow @Local
</Location>


# Restrict access to configuration files...
<Location /admin/conf>
  AuthType Default
  Require user @SYSTEM
  Order allow,deny
  Allow @Local
</Location>
```

Save the file using ctrl+X followed by y and then enter.

After saving, restart CUPS to effect the changes to the configuration file using;

```
sudo service cups restart
```

Step 4: User Access Settings

Next we add the Pi user to the *Ipadmin* group. This gives the Raspberry Pi the ability to perform administrative functions of CUPS without necessarily being a super user.

```
sudo usermod -a -G Ipadmin pi
```

Step 5: Network Accessibility

Next we need to ensure that CUPS can be connected to on the home network and its also accessible across the entire network.

To get it to allow all connections on the network, run;

```
sudo cupsctl –remote-any
```

After this we then restart cups to effect changes using;

```
sudo /etc/init.d/cups  restart
```

With this done we can proceed to test if it works effectively by checking out the CUPS homepage.

Open a web browser and type in your Pi's IP address, indicating the cups port.

e.g 192.168.137.147:631

631 is the cups port.

You should see the cups homepage like the image below.

Please note that your browser may warn you about the security certificate of the website but just click on ignore and proceed. Is that right?, I know, I had doubts too while trying, but haven't had any security breach since then so…

With this done we are ready to move to the next step.

Step 6: Setting Up Samba on Raspberry pi

Samba is an interoperability tool that allows for easy communication between windows and linux or unix programs and it will be used to allow our windows based system to communicate with **CUPS running on the Raspberry Pi to print**.

While cups is being installed, it installs other dependencies like samba, but just in case it wasn't installed, you can install it by following the procedure below.

Run:

```
sudo apt-get install samba
```

Wait for the installation to run its course then proceed to configure samba**.**

Step 7: Configure Samba

Configure samba by opening the configuration file using;

```
sudo nano /etc/samba/samba.conf
```

In the conf file, scroll to the print section and change the; ***guest ok = no* to *guest ok = yes***

```
guest ok = yes
```

Also under the printer driver section, change the; ***read only = yes* to *read only = no***

```
read only  = no
```

With this all done save the file using ctrl+X followed by y and enter.

After saving the file restart samba to effect the changes using;

```
sudo /etc/init.d/samba restart
```

With samba installed, our **Raspberry Pi is finally ready to be attached to a printer** so we take the final step which is adding a printer to cups.

Adding a Printer to CUPS

Adding a printer to cups is officially one of the easiest thing to do, go to the CUPS homepage once more by entering your PI's IP address into a web browser followed by ":631" which is port address on which CUPS is communicating, your Pi's IP address can be gotten easily by running the command;

```
hostname –I
```

Now on the home page, click on the administration tab.

This will take you to the administration page where you will see add new printer. Follow the prompts, select your printer server and continue.

On the final stage before clicking on continue, ensure you checked the ***"share this printer"*** check box.

With this you are all done, move the printer to the preferred location, fire up your Raspberry Pi and print away.

Oh before I forget (rushing to get some chicken, it's Christmas), to **add the new created network printer on your Windows PC**, go to devices and printers, select "*Add a printer*"

Squid Setup

Install Squid:

Enter the following in the shell

```
sudo apt-get install squid
```

Configuring Squid:

Backup the original Squid config file:

```
sudo cp /etc/squid/squid.conf /etc/squid/squidoriginal.conf.bak
```

Edit the config file:

sudo nano /etc/squid/squid.conf

use Ctrl + W to find each section:

`http_access allow localnet = remove the # symbol`

Find: **acl localnet** section

add the following:
acl localnet src **YOUR CIDR IP RANGE** # Description

ie:

```
acl localnet src 192.168.5.0/24 # Home Network
```

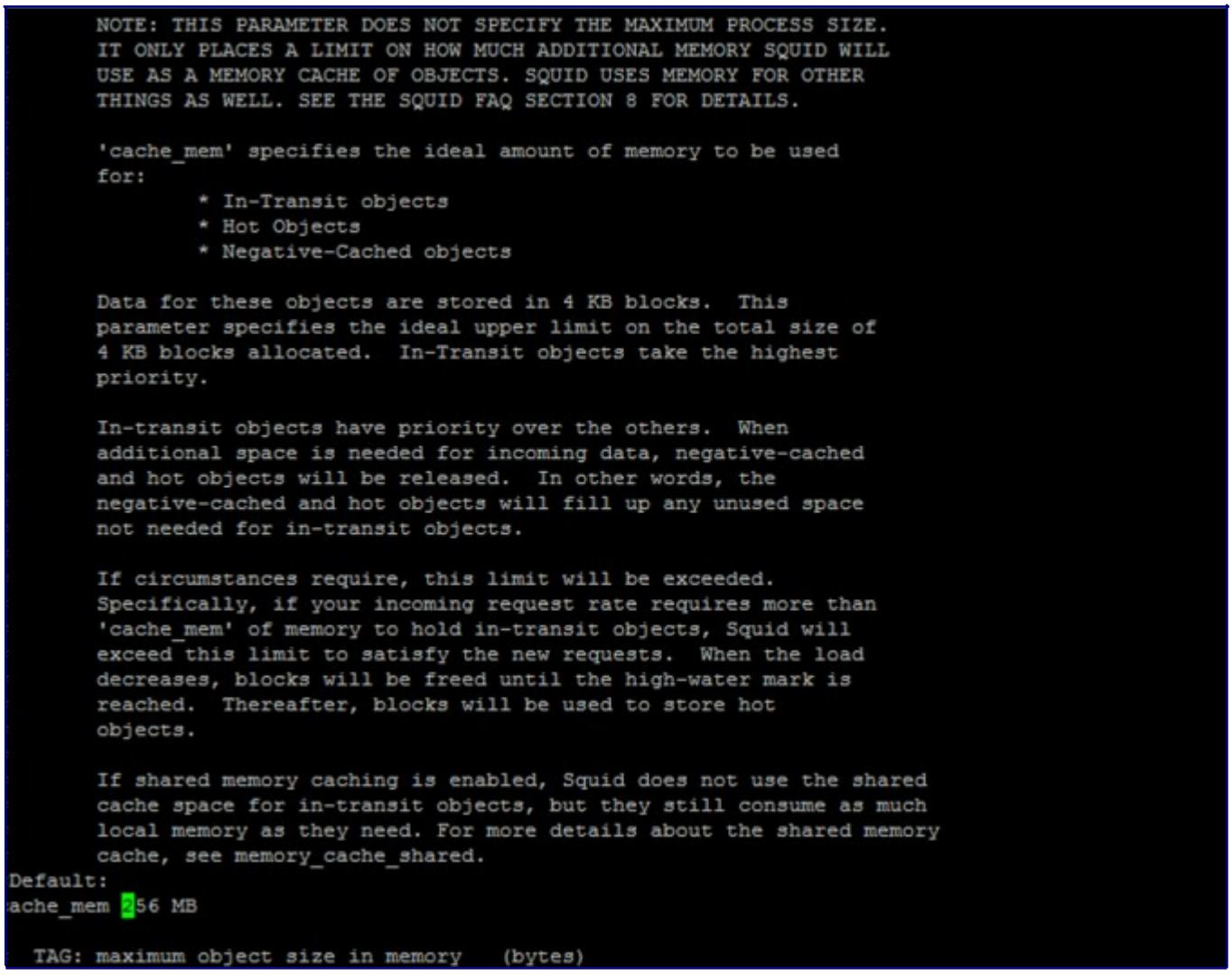
Make sure the ip range/cidr matches your networks range

```
#Default:
# ACLs all, manager, localhost, and to_localhost are predefined.
#
#
# Recommended minimum configuration:
#

# Example rule allowing access from your local networks.
# Adapt to list your (internal) IP networks from where browsing
# should be allowed
#acl localnet src 10.0.0.0/8     # RFC1918 possible internal network
#acl localnet src 172.16.0.0/12  # RFC1918 possible internal network
#acl localnet src 192.168.0.0/16         # RFC1918 possible internal network
#acl localnet src fc00::/7       # RFC 4193 local private network range
#acl localnet src fe80::/10      # RFC 4291 link-local (directly plugged) machines
acl localnet src 192.168.5.0/24 # Home network

acl SSL_ports port 443
acl Safe_ports port 80          # http
```

Find: # **dns_v4_first off** remove the # symbol and change **off** to **on**.

```
#Example:
# append_domain .yourdomain.com
#Default:
# Use operating system definitions

#  TAG: ignore_unknown_nameservers
#       By default Squid checks that DNS responses are received
#       from the same IP addresses they are sent to.  If they
#       don't match, Squid ignores the response and writes a warning
#       message to cache.log.  You can allow responses from unknown
#       nameservers by setting this option to 'off'.
#Default:
# ignore_unknown_nameservers on

#  TAG: dns_v4_first
#       With the IPv6 Internet being as fast or faster than IPv4 Internet
#       for most networks Squid prefers to contact websites over IPv6.
#
#       This option reverses the order of preference to make Squid contact
#       dual-stack websites over IPv4 first. Squid will still perform both
#       IPv6 and IPv4 DNS lookups before connecting.
#
#       WARNING:
#         This option will restrict the situations under which IPv6
#         connectivity is used (and tested), potentially hiding network
#         problems which would otherwise be detected and warned about.
#Default:
dns_v4_first on

#  TAG: ipcache_size    (number of entries)
#       Maximum number of DNS IP cache entries.
#Default:
# ipcache_size 1024
```

Cache_mem 256 MB



Maximum_object_size 4096 MB

```
#  TAG: maximum_object_size     (bytes)
#       Set the default value for max-size parameter on any cache_dir.
#       The value is specified in bytes, and the default is 4 MB.
#
#       If you wish to get a high BYTES hit ratio, you should probably
#       increase this (one 32 MB object hit counts for 3200 10KB
#       hits).
#
#       If you wish to increase hit ratio more than you want to
#       save bandwidth you should leave this low.
#
#       NOTE: if using the LFUDA replacement policy you should increase
#       this value to maximize the byte hit rate improvement of LFUDA!
#       See cache_replacement_policy for a discussion of this policy.
#Default:
maximum_object_size 4096 MB

#  TAG: cache_dir
#       Format:
```

Maximum_object_size_in_memory 8192 KB

```
#  TAG: maximum_object_size_in_memory   (bytes)
#       Objects greater than this size will not be attempted to kept in
#       the memory cache. This should be set high enough to keep objects
#       accessed frequently in memory to improve performance whilst low
#       enough to keep larger objects from hoarding cache_mem.
#Default:
# maximum_object_size_in_memory 8192 KB
```

Cache_dir ufs /var/spool/squid3 = 8192 (1st variable - this is 8192 MB) #my is pointing to mounted thumb drive.

```
#Default:
# No disk cache. Store cache ojects only in memory.
#

# Uncomment and adjust the following to add a disk cache directory.
cache_dir ufs /var/spool/squid3 8192 16 256

#  TAG: store_dir_select_algorithm
```

Ctrl + X and Y to save & exit.

Backup your altered squid config file and restart the Squid service:

sudo cp /etc/squid/squid.conf /etc/squid/mysquid.conf.bak

sudo service squid restart

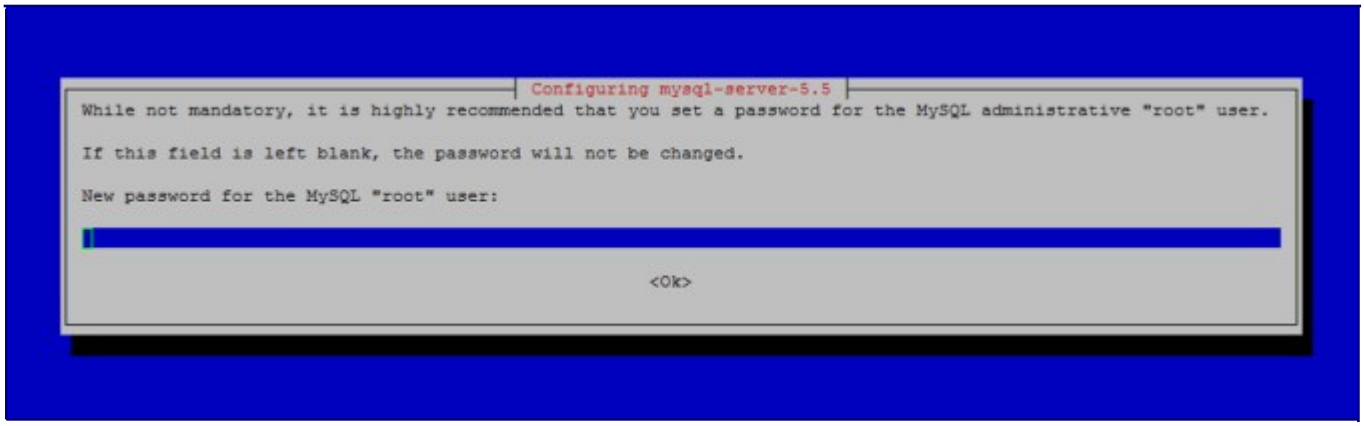
```
pi@raspberrypi:~ $ sudo nano /etc/squid3/squid.conf
pi@raspberrypi:~ $ sudo cp /etc/squid3/squid.conf /etc/squid3/squid.conf.bak
pi@raspberrypi:~ $ sudo service squid3 restart
```

Make managing Squid easier with Webmin:

First, install webmins prereqs; open a shell and enter:

```
sudo apt-get -f install

sudo apt-get -y install apache2 apache2-suexec-custom libnet-ssleay-perl libauthen-pam-perl libio-pty-perl apt-show-versions samba bind9 webalizer locate mysql-server

sudo apt-get install squid-cgi
```

Enter a secure password for MySQL when prompted.:

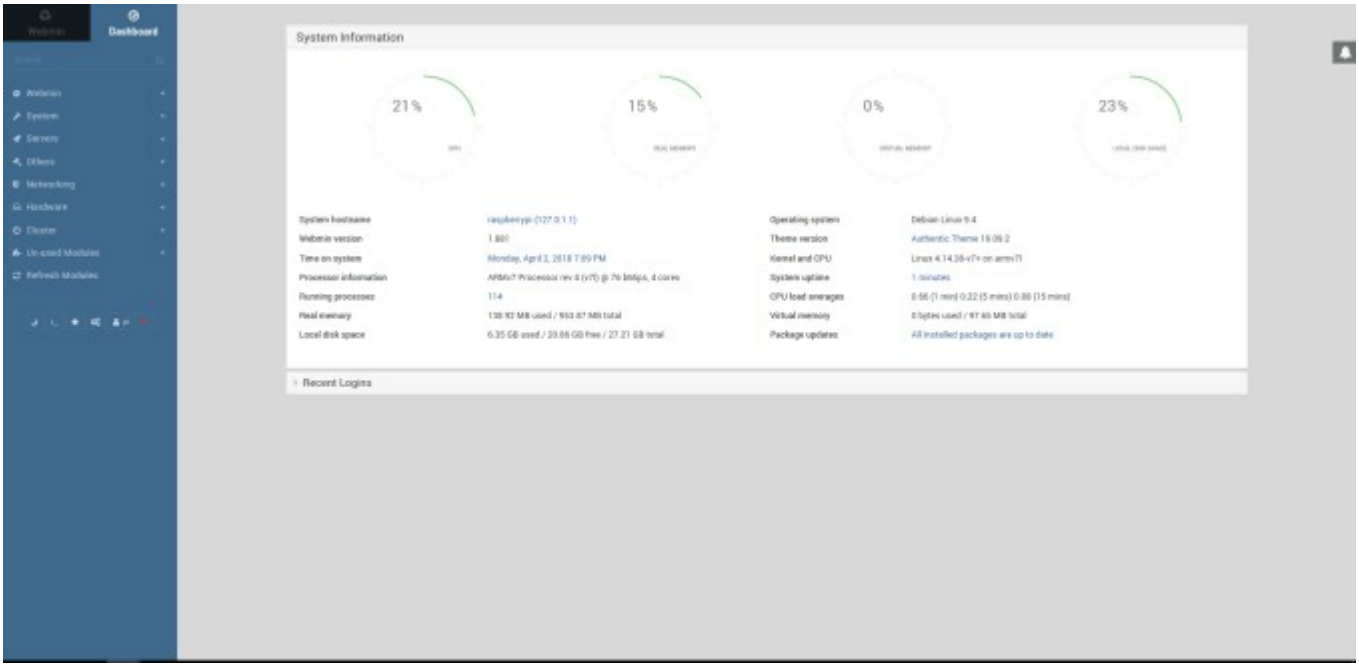


From the shell enter these commands in turn:

```
cd

check your current path should read as /home/pi
pwd

sudo mkdir installed-packages

cd installed-packages

Download the Webmin interface package:
sudo wget http://www.webmin.com/download/deb/webmin-current.deb

Install Webmin:
sudo dpkg -i webmin-current.deb
```

Once Webmin has been installed; open a browser on your pc https://192.168.5.250:10000

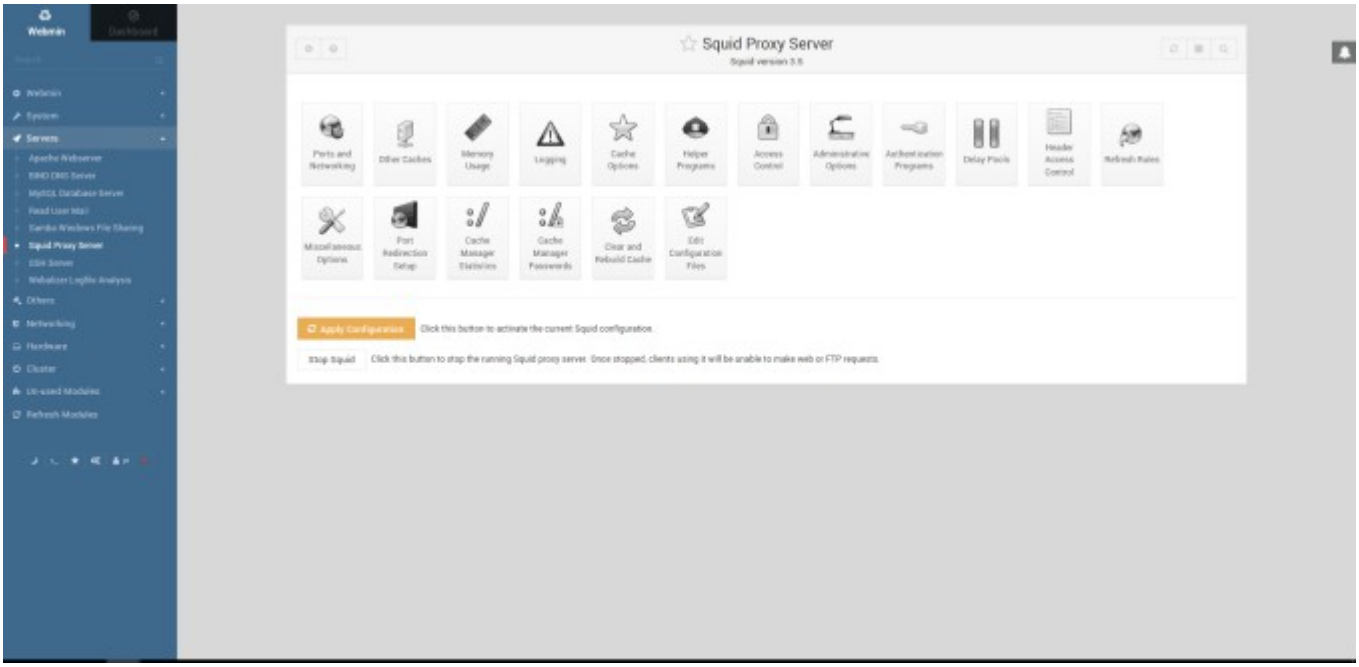
Login using the raspberry pi login (default is pi and raspberry).



In webmin; you'll be able to adjust Squid settings through webmin. Look under servers; Squid proxy server.

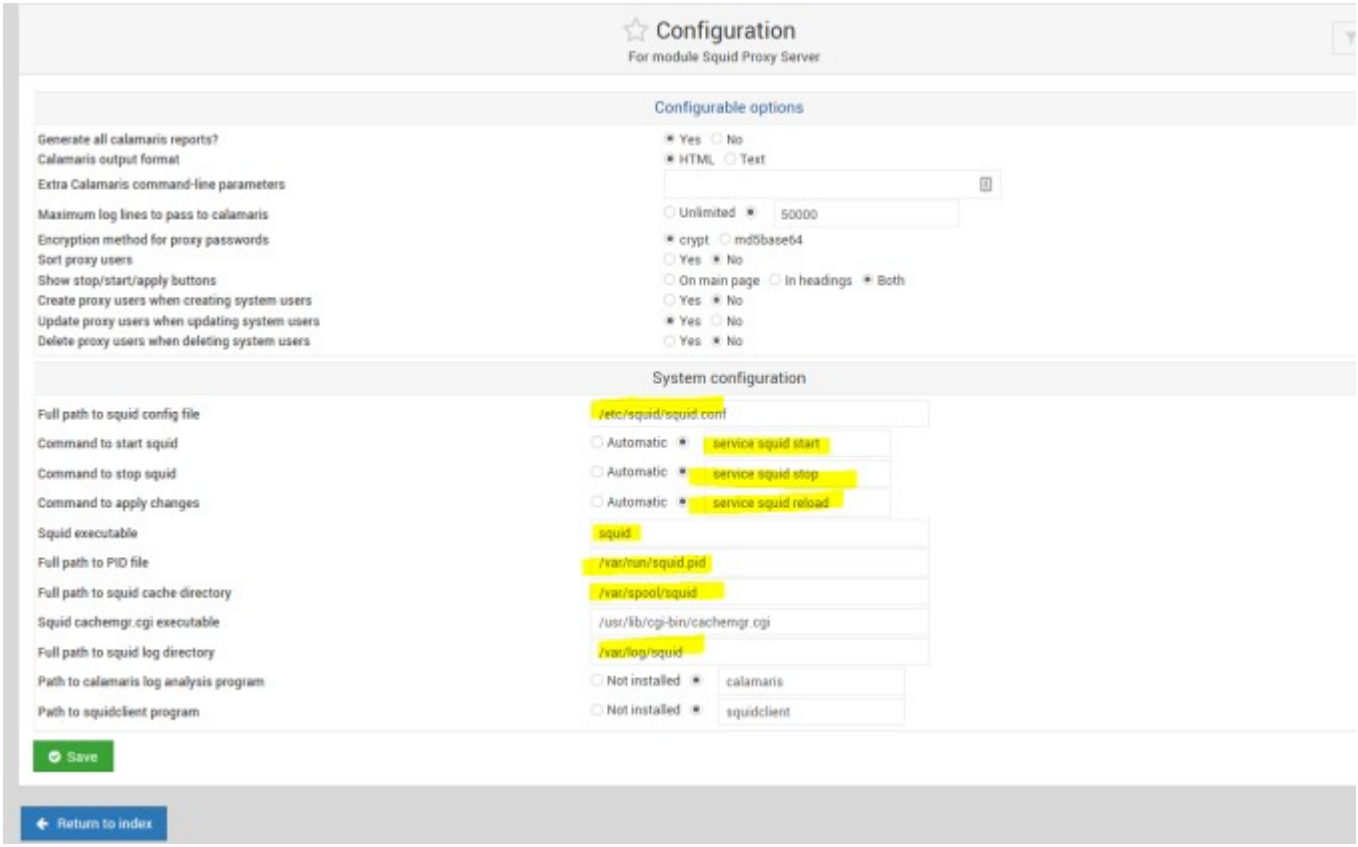
At this stage, its highly likely your webmin config isn't configured for Squid Pi (should be fine if your following this guide on a Linux 'Intel / AMD' PC…

You'll find SQUID hidden under the In-used modules menu.



Click on the edit config button, change squid3 to squid where highlighted.

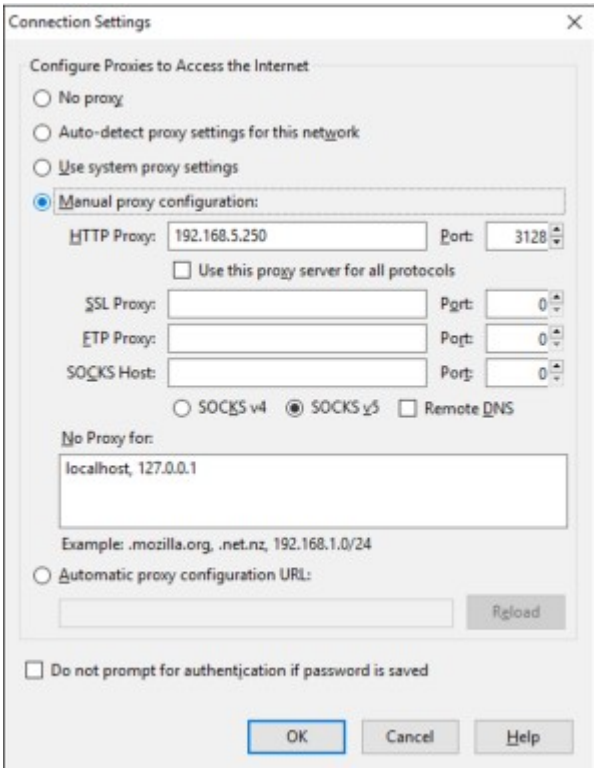
Hit save, then hit the orange apply config button. After a minute of so, the Squid services will be restarted and Webmin will work.



If you have an error relating to the cache manager statistic icon, ssh back onto the Pi, use sudo nano to edit the config file, make the required change and save the file. You may need to reboot the pi.

Configuring the client:

Set browser proxy: Enter the ip address of the Raspberry Pi (192.168.5.250) and port 3128. Restart browser.



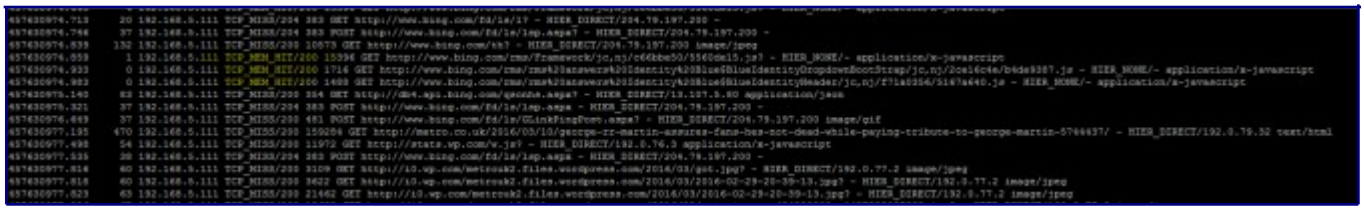
Clear your browser cache and restart the browser. You should now be using the Squid Proxy server on your Raspberry Pi.

Check the cache log:

To check the squid cache logs, open a new shell window and enter:

```
sudo tail -f /var/log/squid/access.log
```

Hits are items being pulled from the Squid Cache rather than the internet.

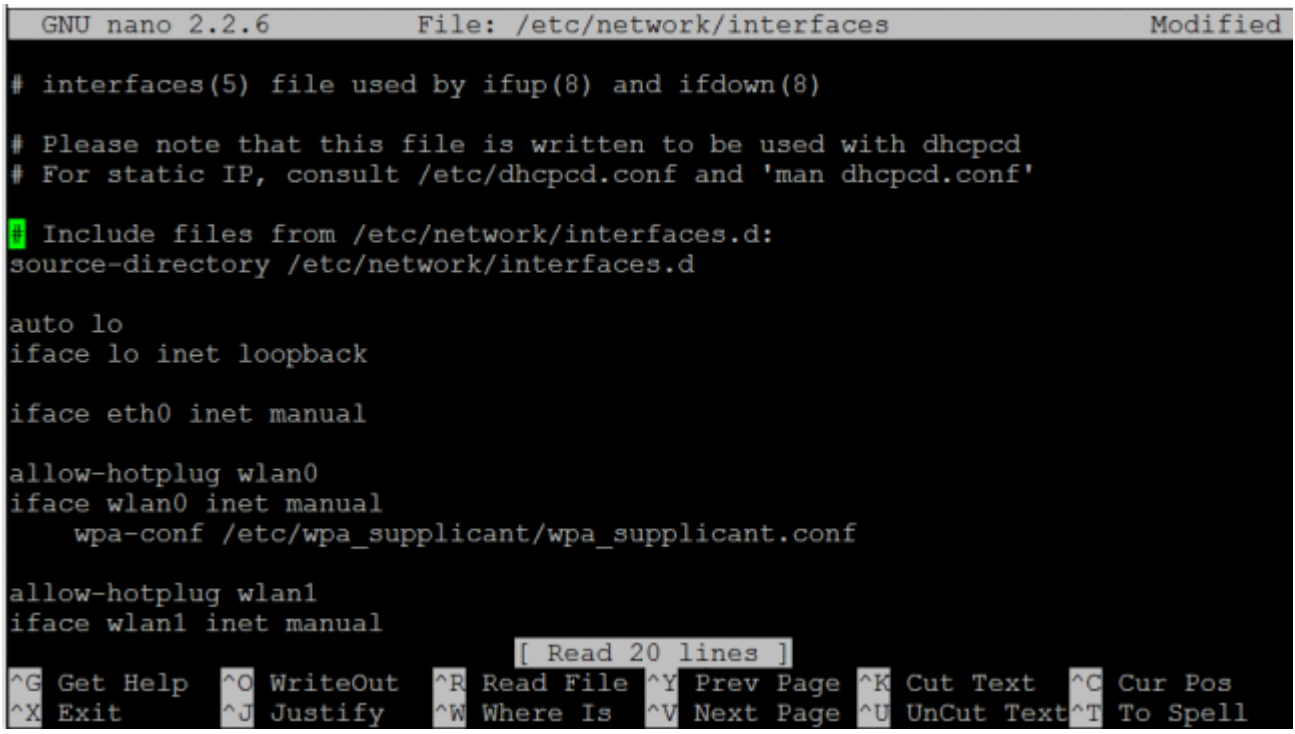


Summary:

If your unlucky enough to have a slow or laggy internet connection, one possible solution for you is to build and test a Squid proxy server. However, bear in mind, your mileage may vary as not all objects are cacheable, and certainly any improvement is less noticiable on fast internet connections such as BT infinity.

I performed some “not very scientific” tests using OpenOffice.org. I found that the download speed of the OpenOffice installer on the first try was 3.9mbs, jumping to 7.9Mb/s after caching once, then

maxing out at 9.8Mb/s on the second and subsequent runs (likely a limitation of the Raspberry Pi's network card – which is limited to 100mbs – *UPDATE: Raspberry Pi3B+ has a much faster NIC card*).



If your still running Wheezy, or another Linux distro; this is the equivalent method of setting up static ip:

sudo nano /etc/network/interfaces

Remove the line that reads: iface eth0 inet dhcp

Add the following:

iface eth0 inet static
address 192.168.5.250
netmask 255.255.255.0
network 192.168.5.0
broadcast 192.168.5.255
gateway 192.168.5.254

Check DNS:
sudo nano /etc/resolv.conf
set to: 208.67.222.222, 208.67.220.220

Reboot the Pi using: sudo reboot

PiVPN

Installation of PiVPN (The software we will be using as our VPN server) is a breeze. You simply have to run just one command to install PiVPN. I will assume you already have the Raspbian OS up and running. You only need the lite version if you will be running headless, that's how I am installing it since I will have PiVPN running along side PiHole, my network wide ad blocker.

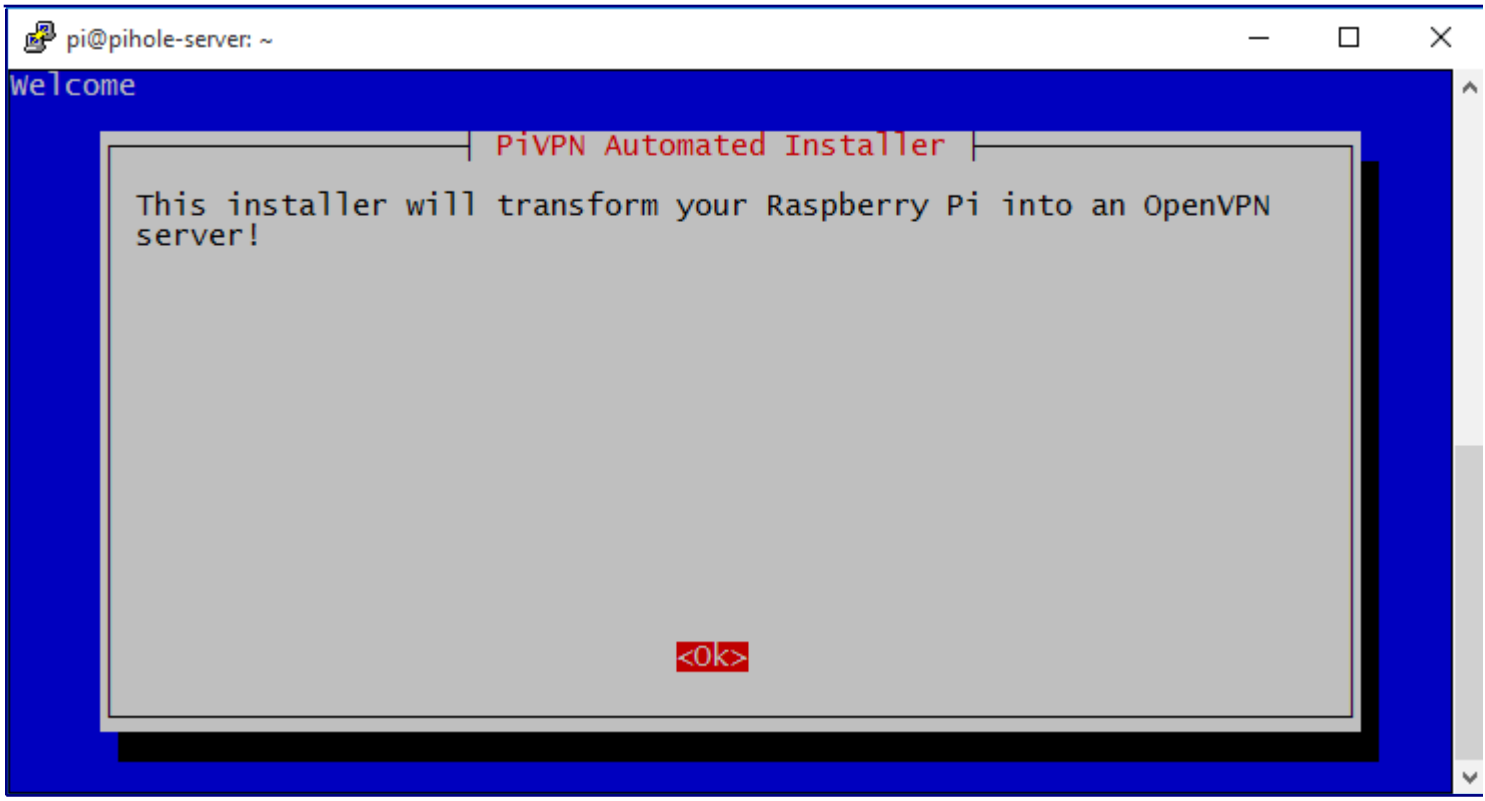
Run the following command in a terminal window or use SSH to install PiVPN:

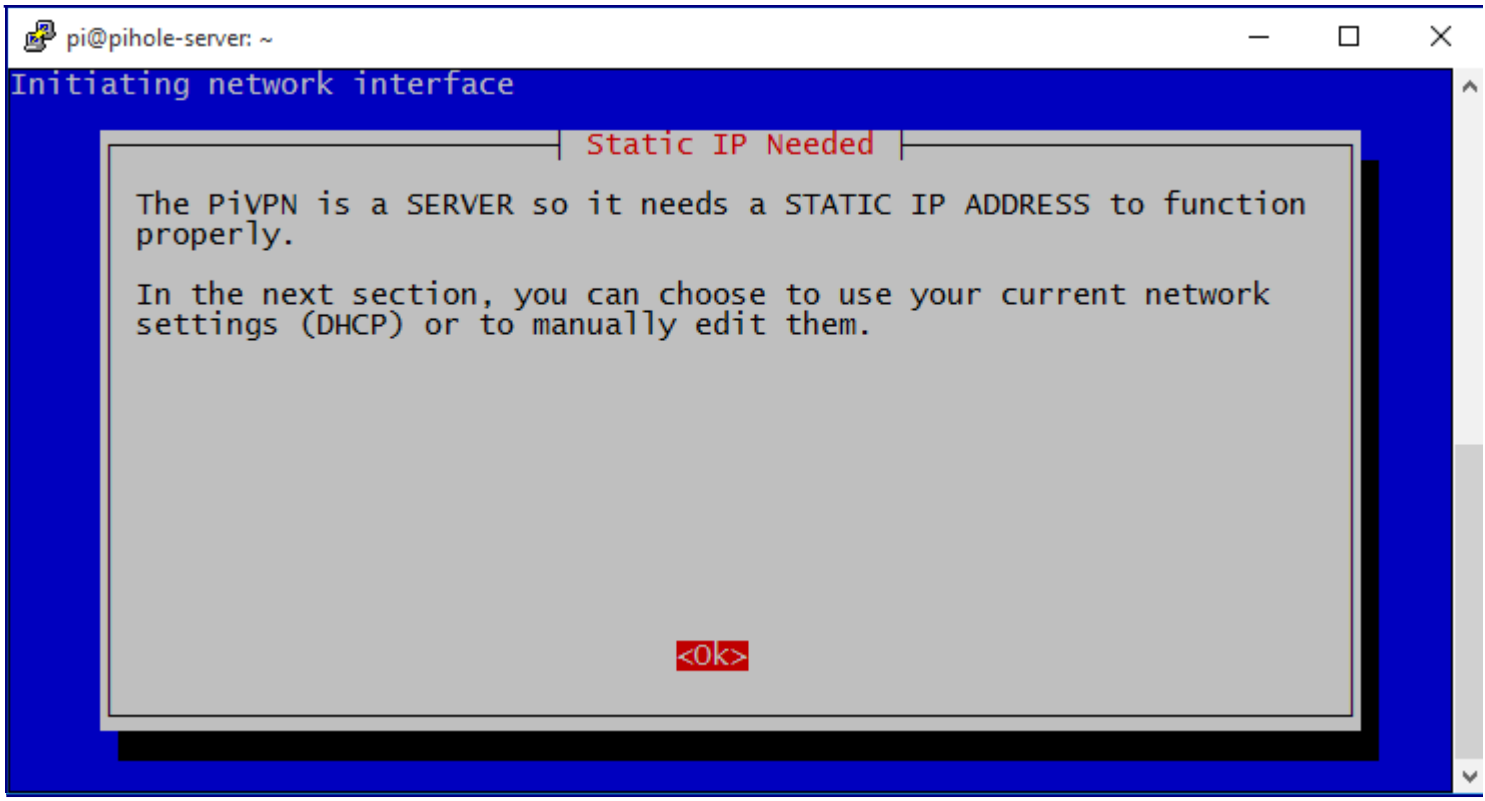
```
curl -L https://install.pivpn.io | bash
```

```
1 curl -L https://install.pivpn.io | bash
```

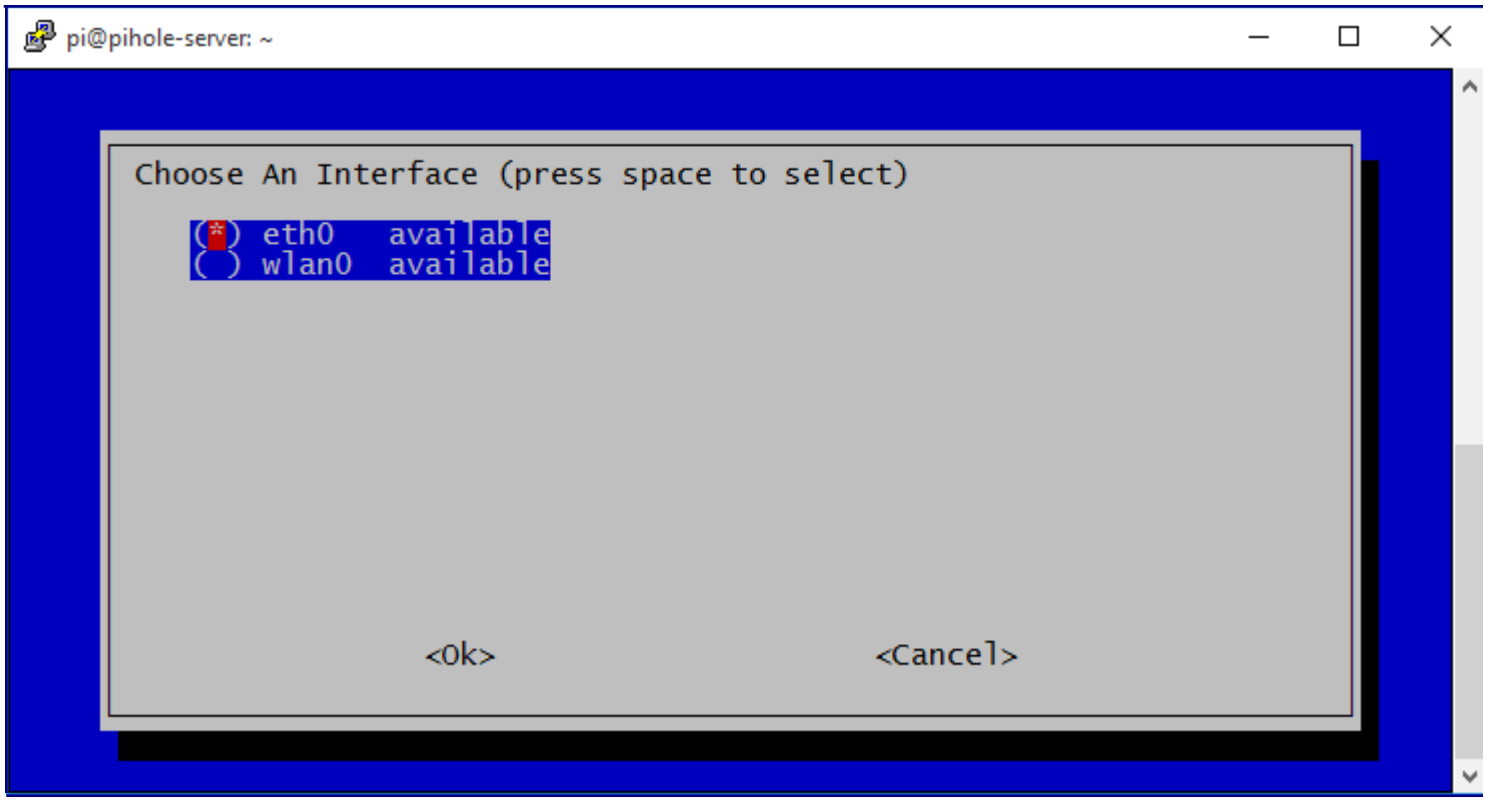
Just a quick side-note, running a command like this is dangerous. Basically what the command being run is doing is going to http://install.pivpn.io and parsing the data then running it in the command line. If you run a similar command from an untrusted source you can do some damage and it is very dangerous to do so. You can type https://install.pivpn.io in your browser to see the exact commands being run.

After you run the command above you should get the window below after a few minutes, hit enter to continue:

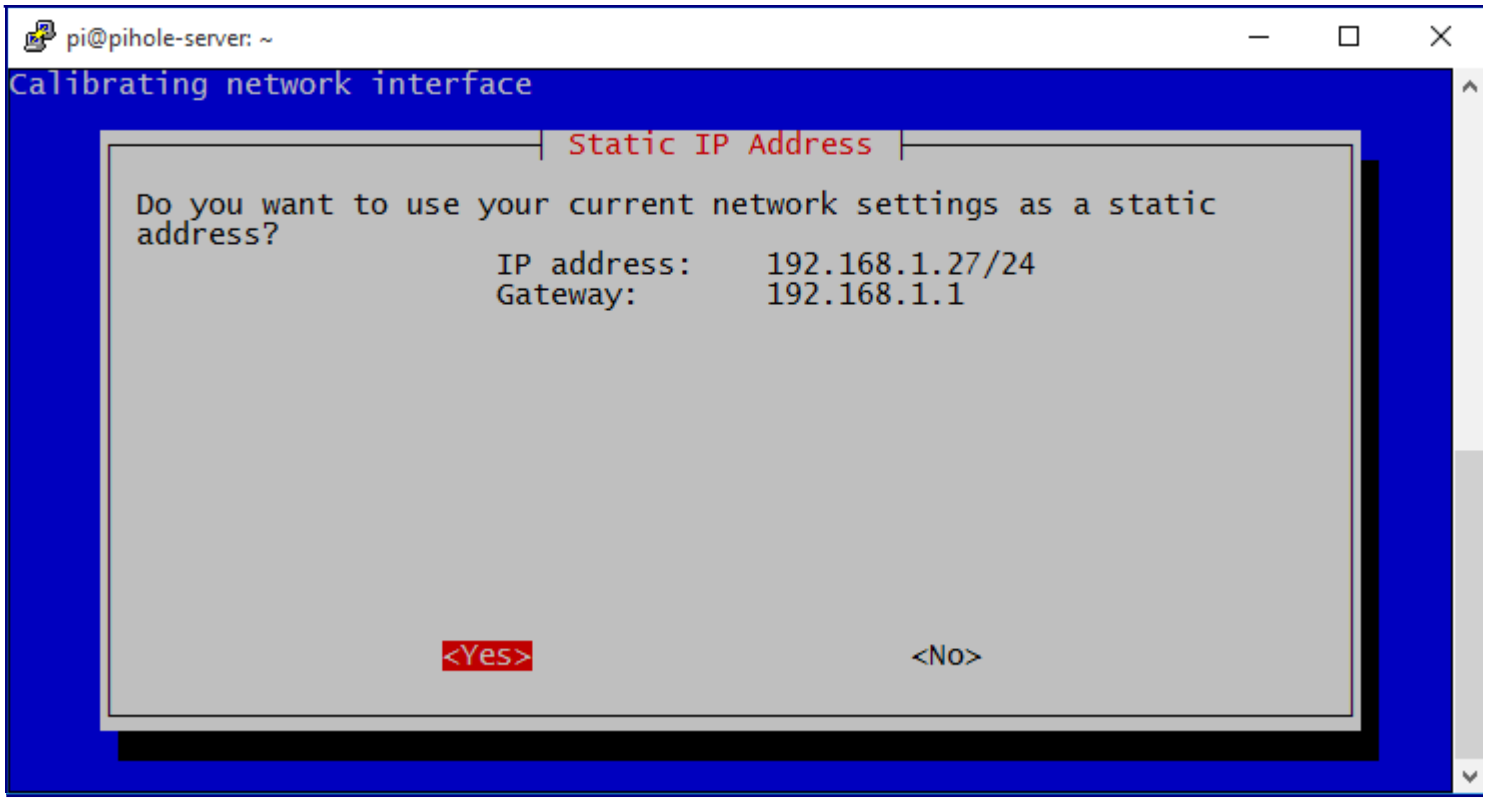


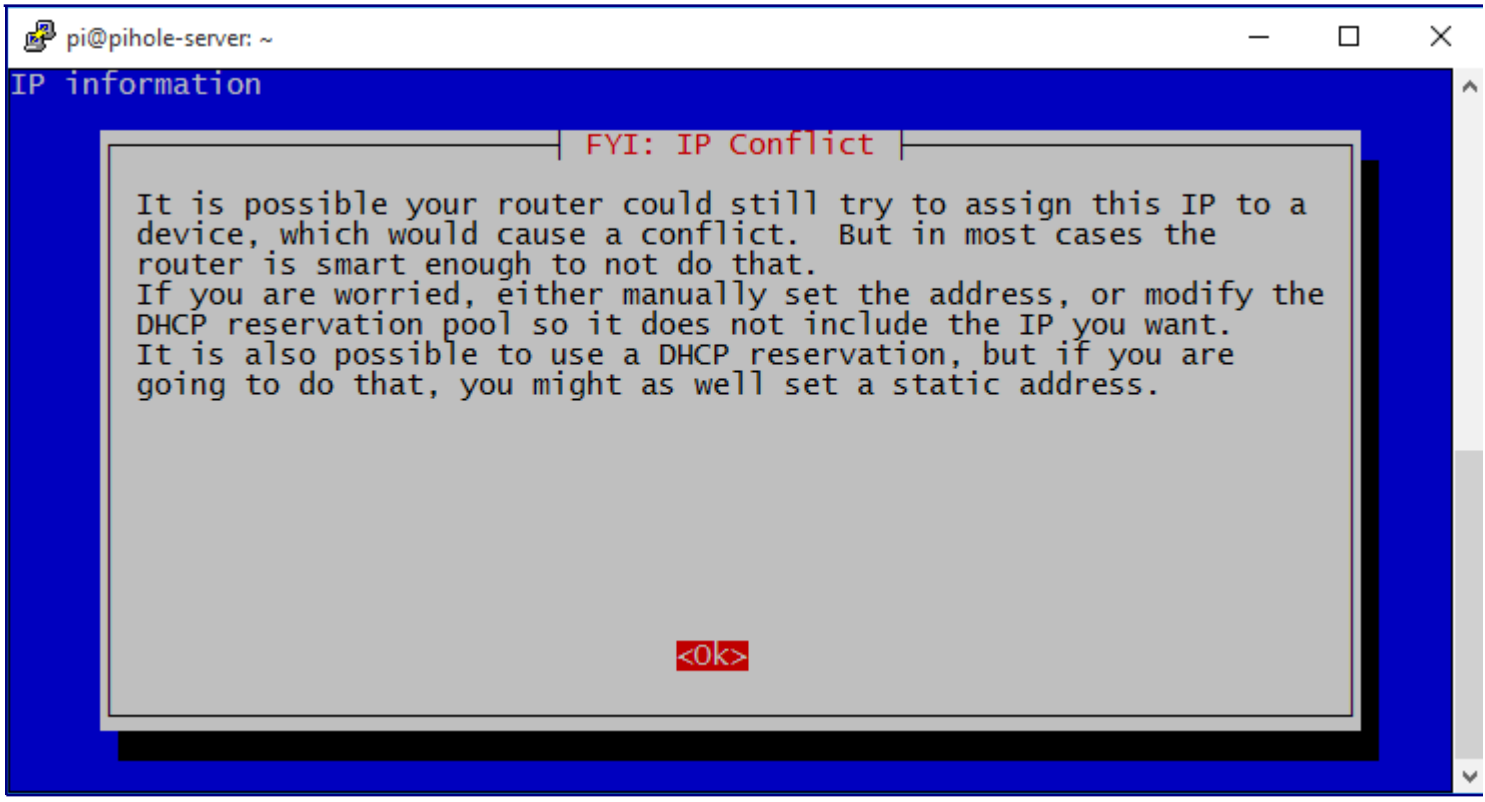


You will get a windows asking you to select which network interface you would like to use. I use my Ethernet connection in this example which is labeled eth0. I suggest using an ethernet connection since it will work a lot faster.

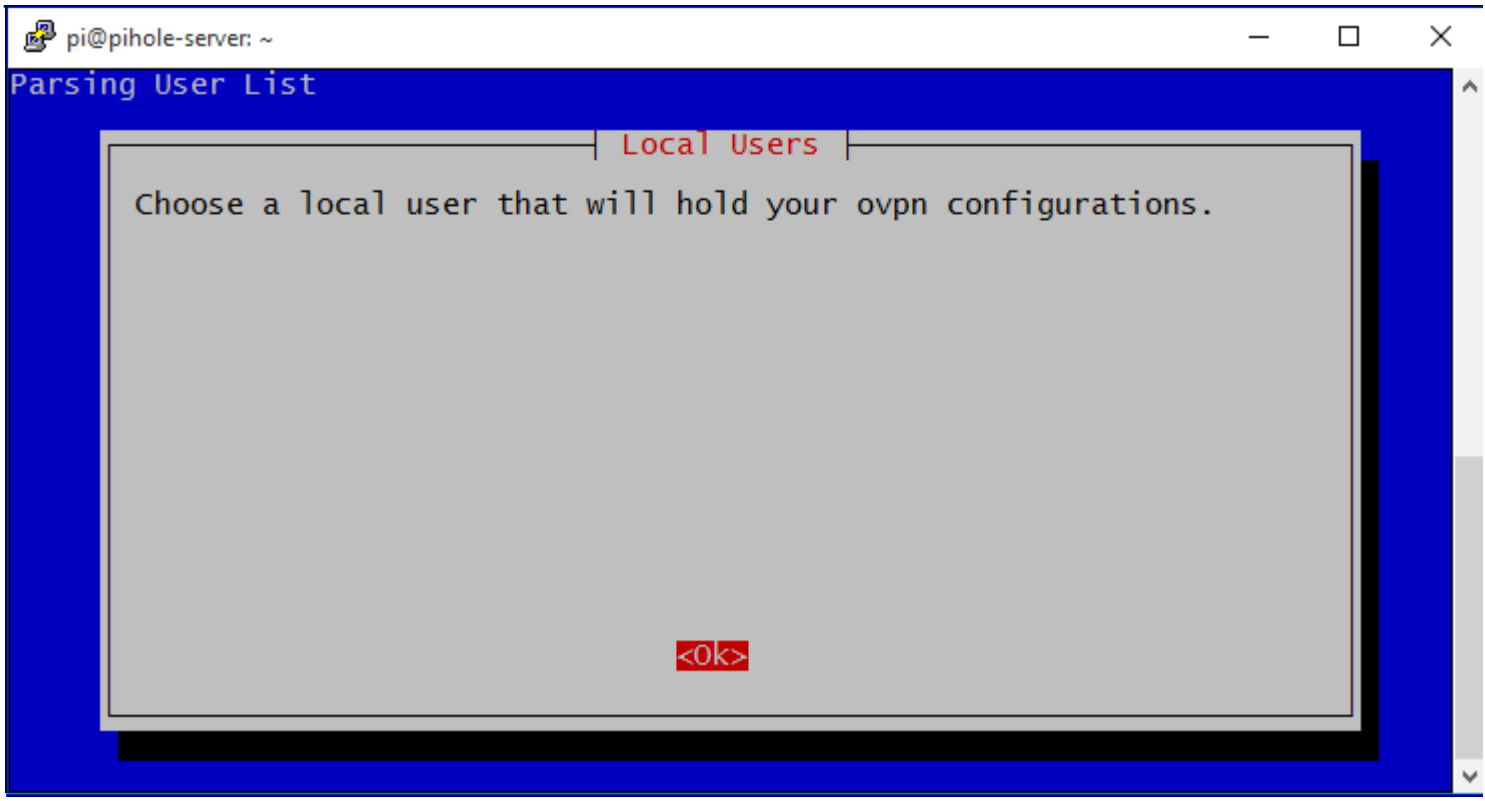


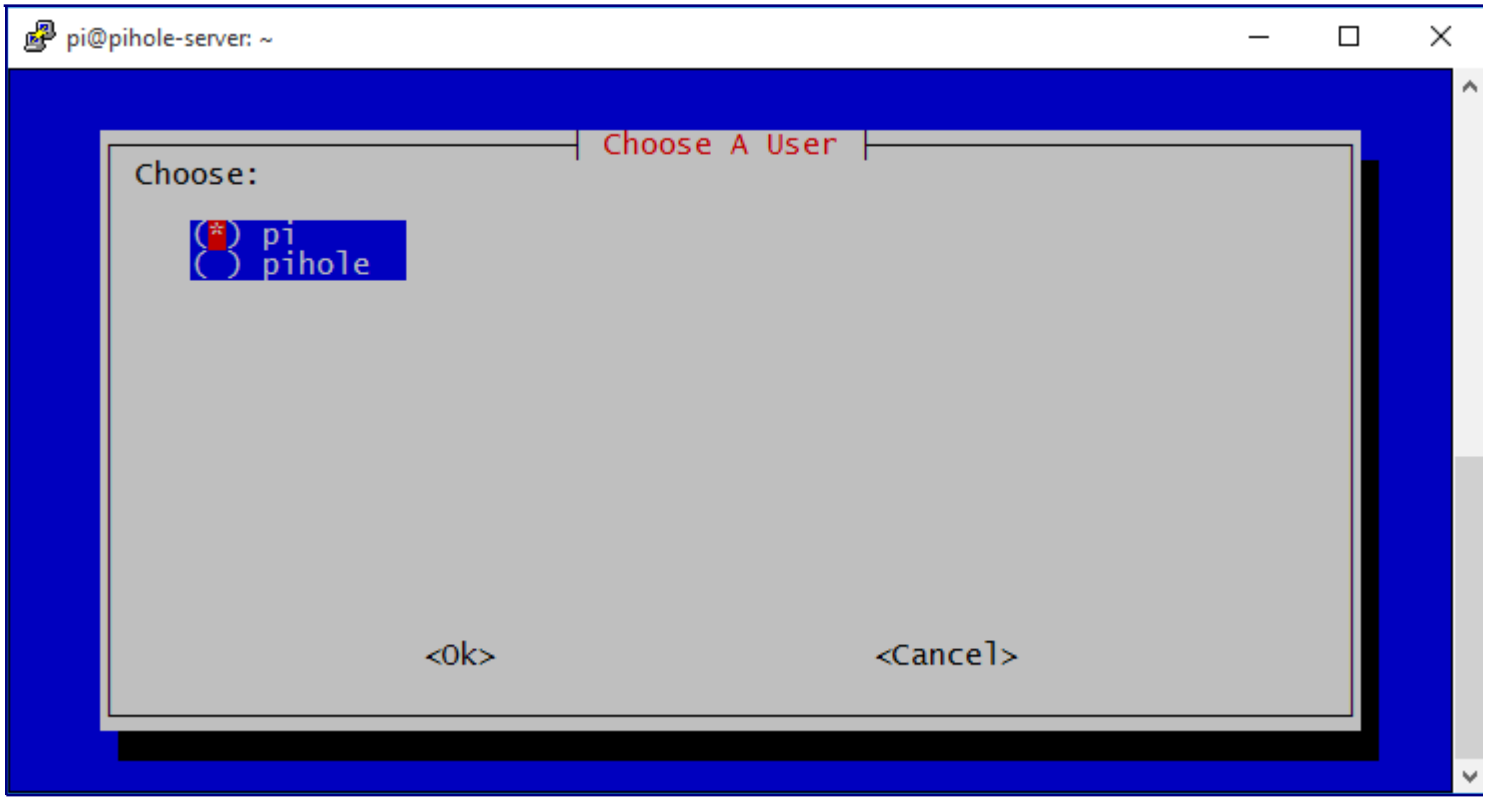
Once you select your network interface it will ask you if you would like to setup the interface to have a static IP Address. I highly suggest to setup the IP Address to have a static IP Address. This will ensure that your internal IP Address doesn't change if you restart your Raspberry
Pi.


pi@pihole-server: ~
Calibrating network interface
Static IP Address
Do you want to use your current network settings as a static address?
IP address: 192.168.1.27/24
Gateway: 192.168.1.1
<Yes>
<No>

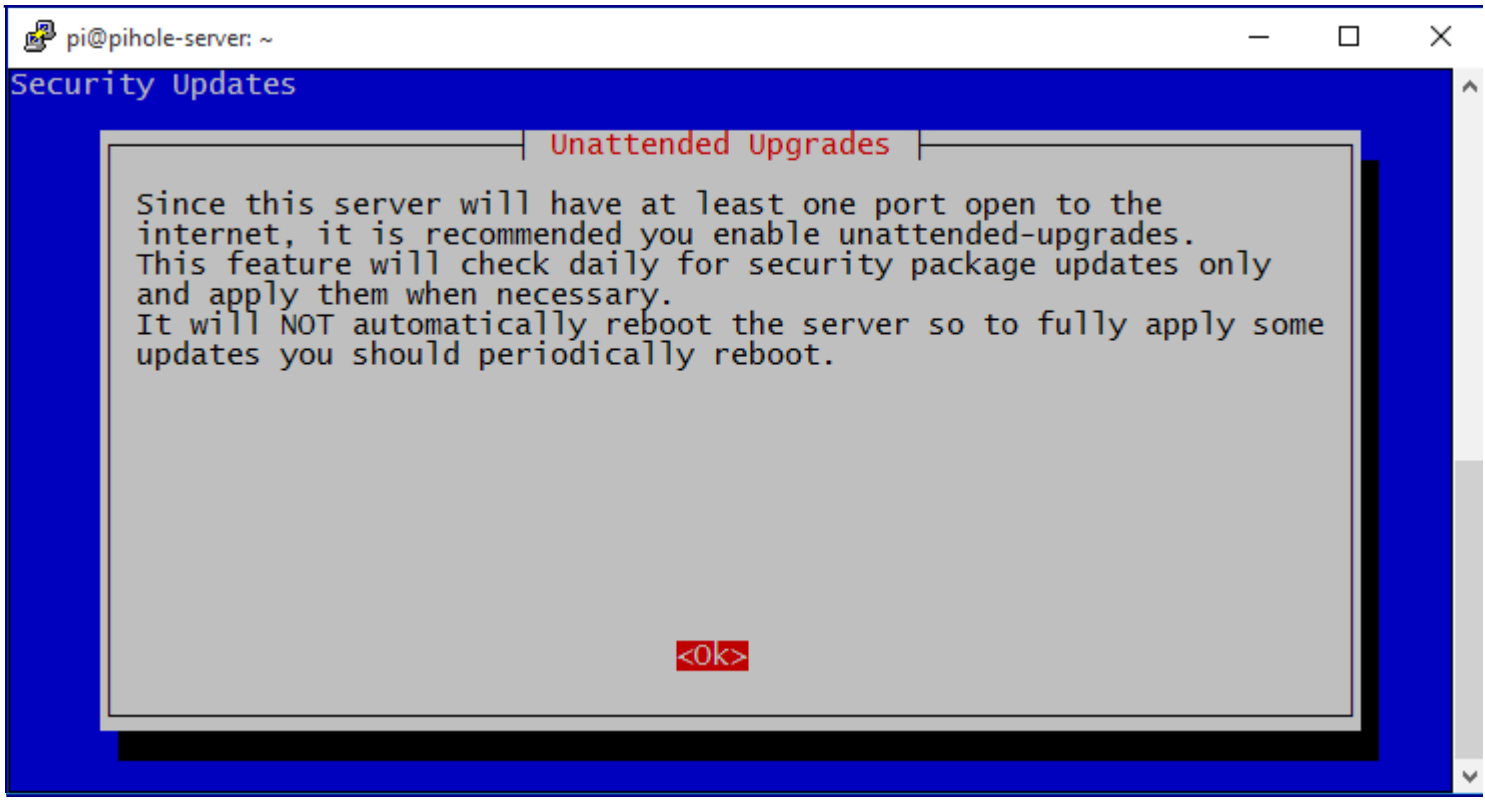

pi@pihole-server: ~
IP information
FYI: IP Conflict
It is possible your router could still try to assign this IP to a device, which would cause a conflict. But in most cases the router is smart enough to not do that.
If you are worried, either manually set the address, or modify the DHCP reservation pool so it does not include the IP you want.
It is also possible to use a DHCP reservation, but if you are going to do that, you might as well set a static address.
<Ok>

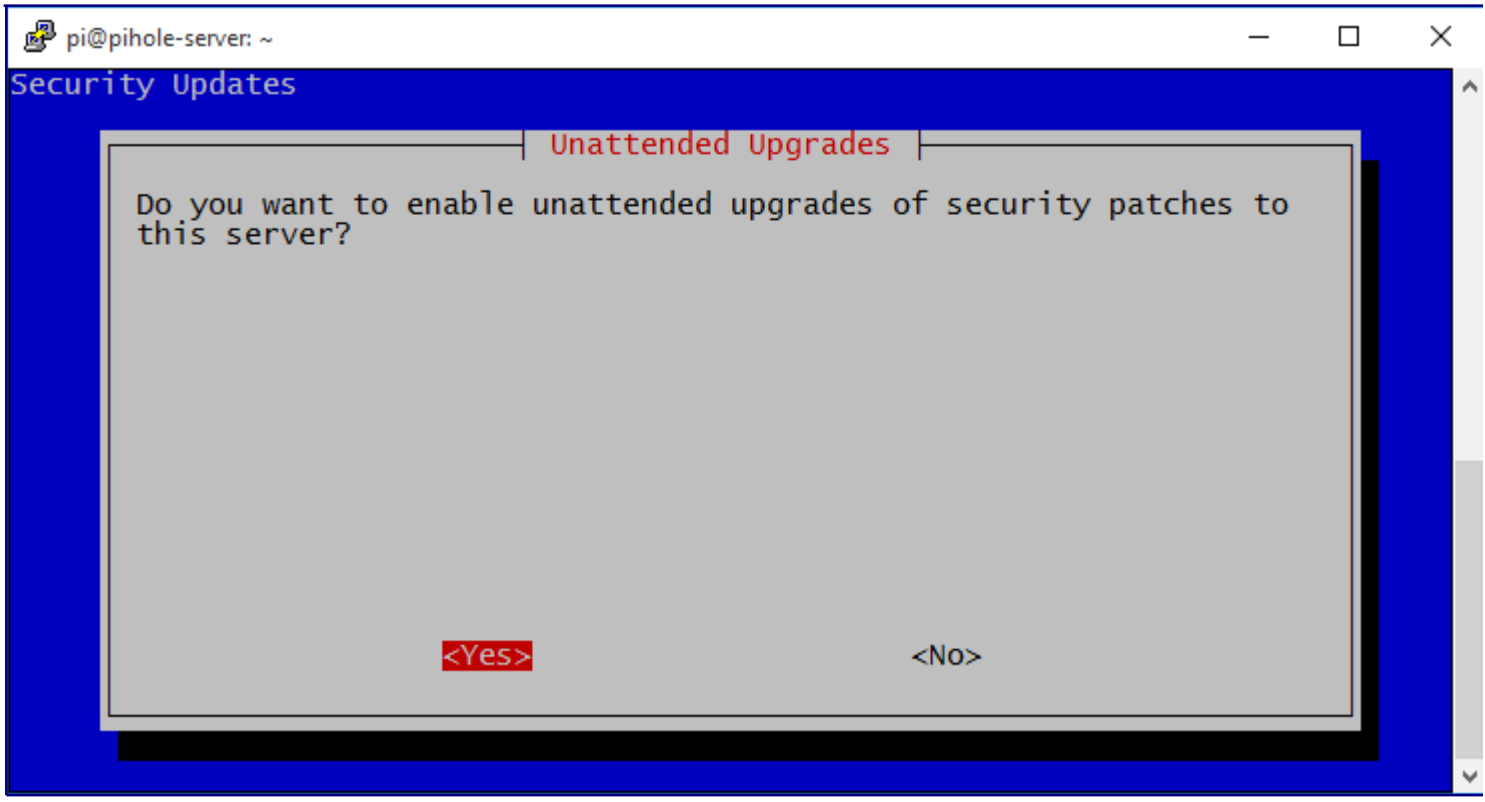
The next step will ask you to pick a user that will have the PiVPN configuration settings. If you created other users you can select them here. The user you pick is not really important. You can see in my image below I have 2 users. One is the original ‘pi’ user and the other is the ‘pihole’ user from my adblocker.


pi@pihole-server: ~
Parsing User List
Local Users
Choose a local user that will hold your ovpn configurations.
<Ok>

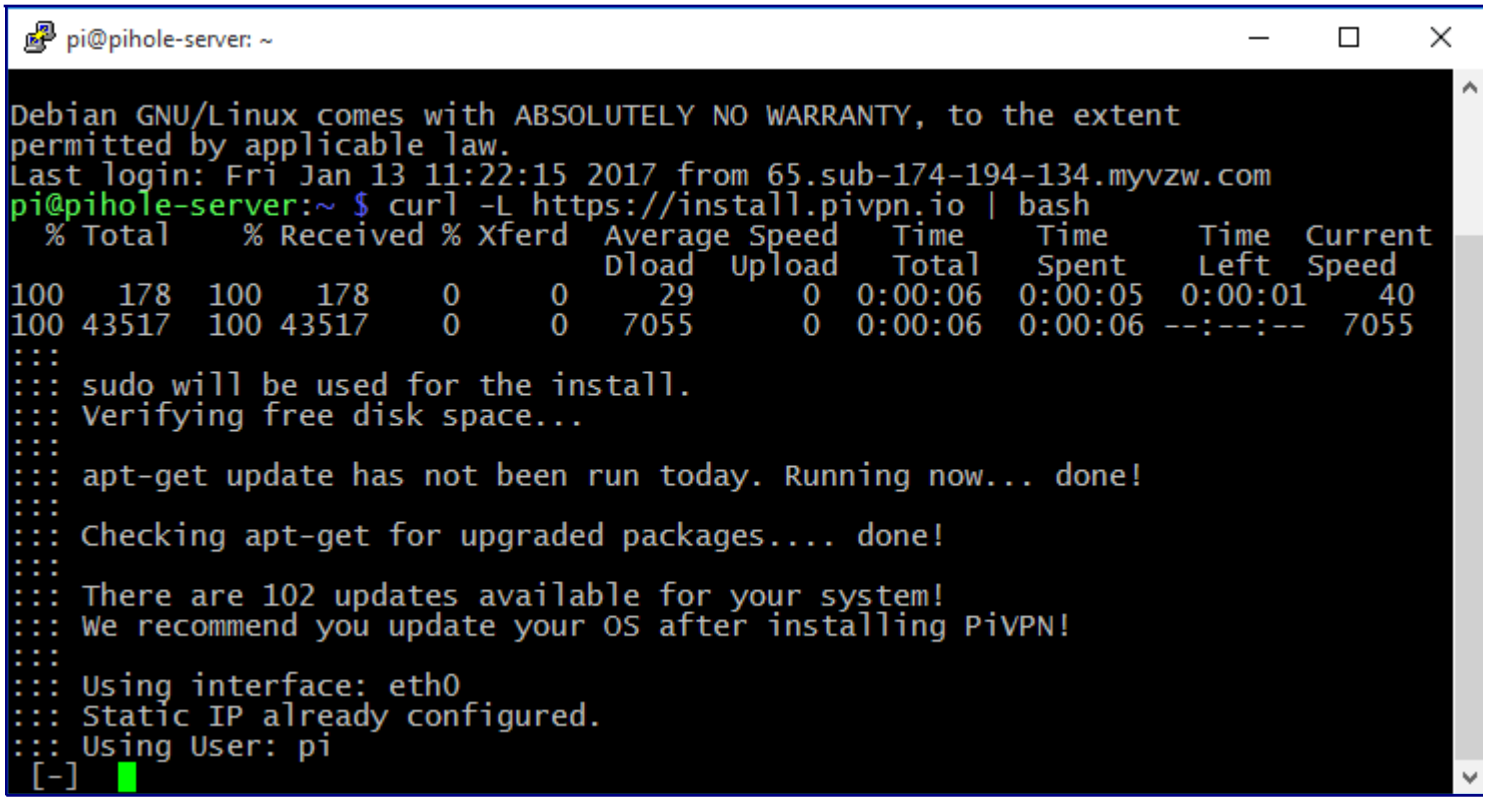

pi@pihole-server: ~
Choose A User
Choose:
(*) pi
( ) pihole
<Ok>
<Cancel>

The next step is another crucial step. Since we will be opening a port on our router to redirect to our Raspberry Pi we can be vulnerable to attacks since we are exposing our device to the internet. What this step will do is enable unattended upgrade of security patches. Basically your Raspberry Pi will check daily for new security updates and update itself. You should periodically reboot for some updates to apply. I would also suggest strong passwords on your users.

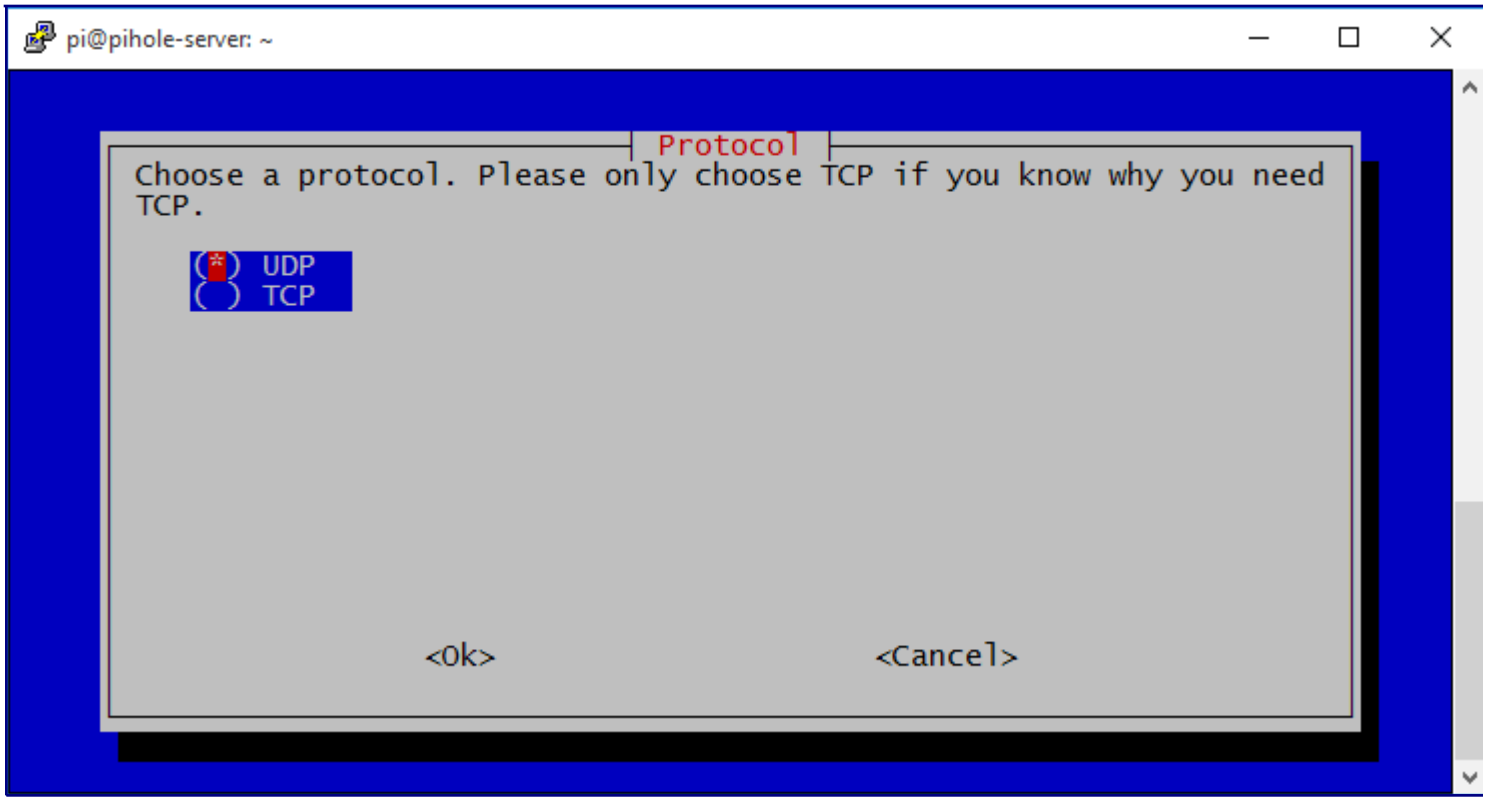

pi@pihole-server: ~
Security Updates
Unattended Upgrades
Since this server will have at least one port open to the
internet, it is recommended you enable unattended-upgrades.
This feature will check daily for security package updates only
and apply them when necessary.
It will NOT automatically reboot the server so to fully apply some
updates you should periodically reboot.
<Ok>


pi@pihole-server: ~
Security Updates
Unattended Upgrades
Do you want to enable unattended upgrades of security patches to
this server?
<Yes>
<No>

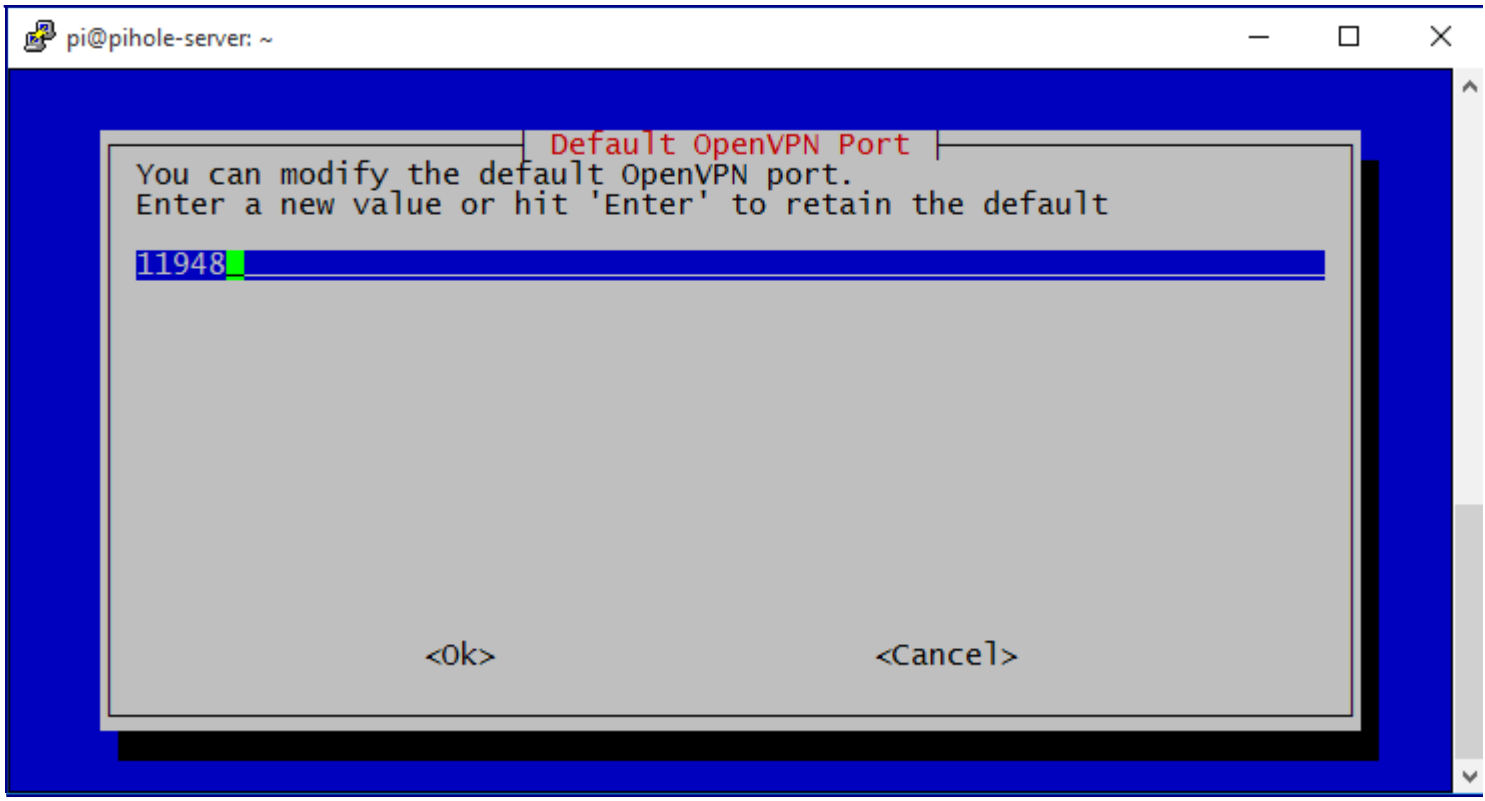
After you enable security updates you will get the following screen setting up PiVPN.

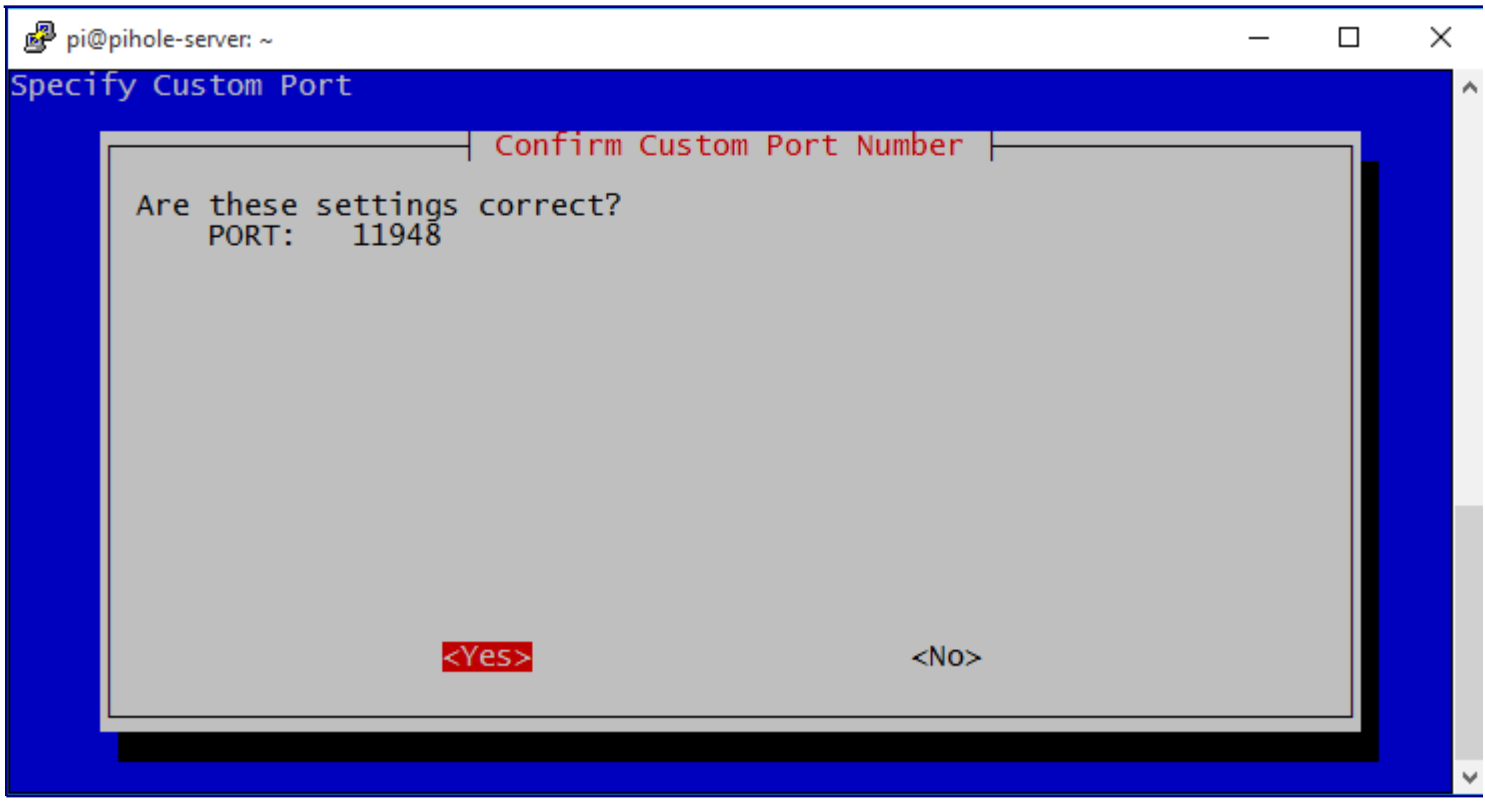


Simply pick UDP in this screen. There is no need for TCP.

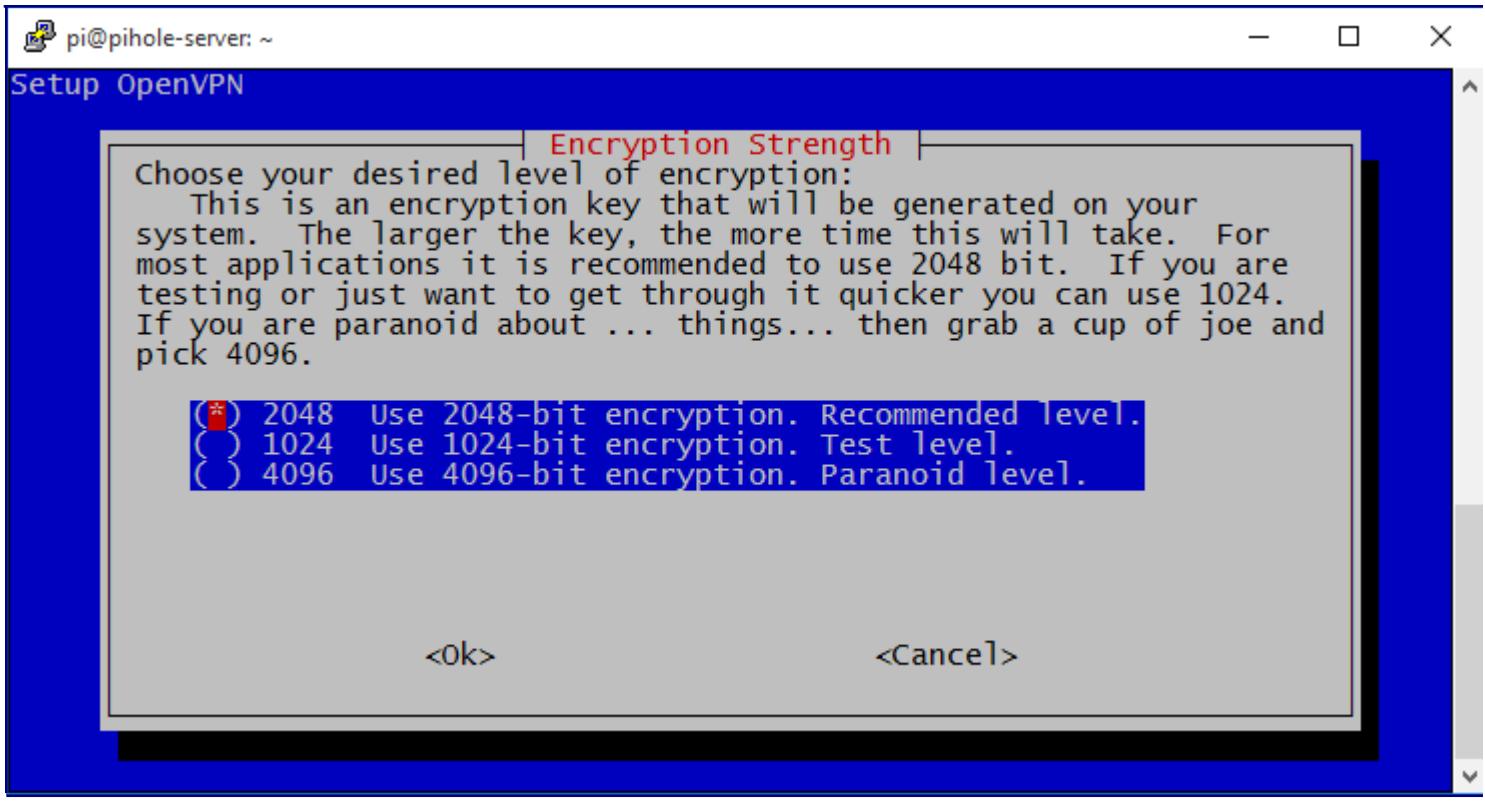


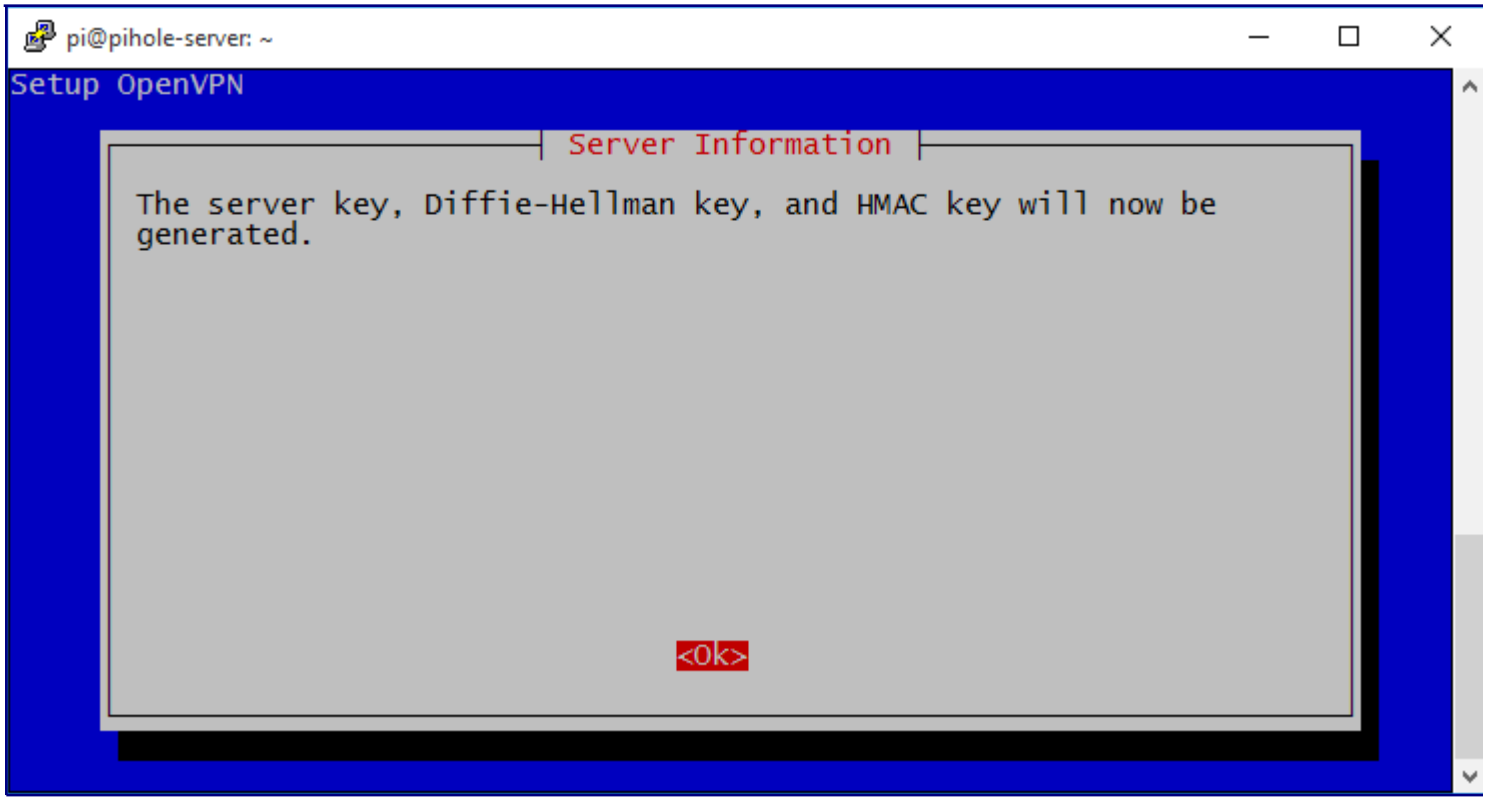
The next step we will pick our port for our VPN connections. The default port is 1194. As you can see I chose port 11948. You can leave the default VPN port of 1194 or change it to something else. My suggestion is changing it to enhance security. Changing your port won't turn your server into Fort Knox but it will not show up in default port scans of your IP Address assuming the attacker is scanning default ports only.


pi@pihole-server: ~
Default OpenVPN Port
You can modify the default OpenVPN port.
Enter a new value or hit 'Enter' to retain the default
11948
<Ok>
<Cancel>

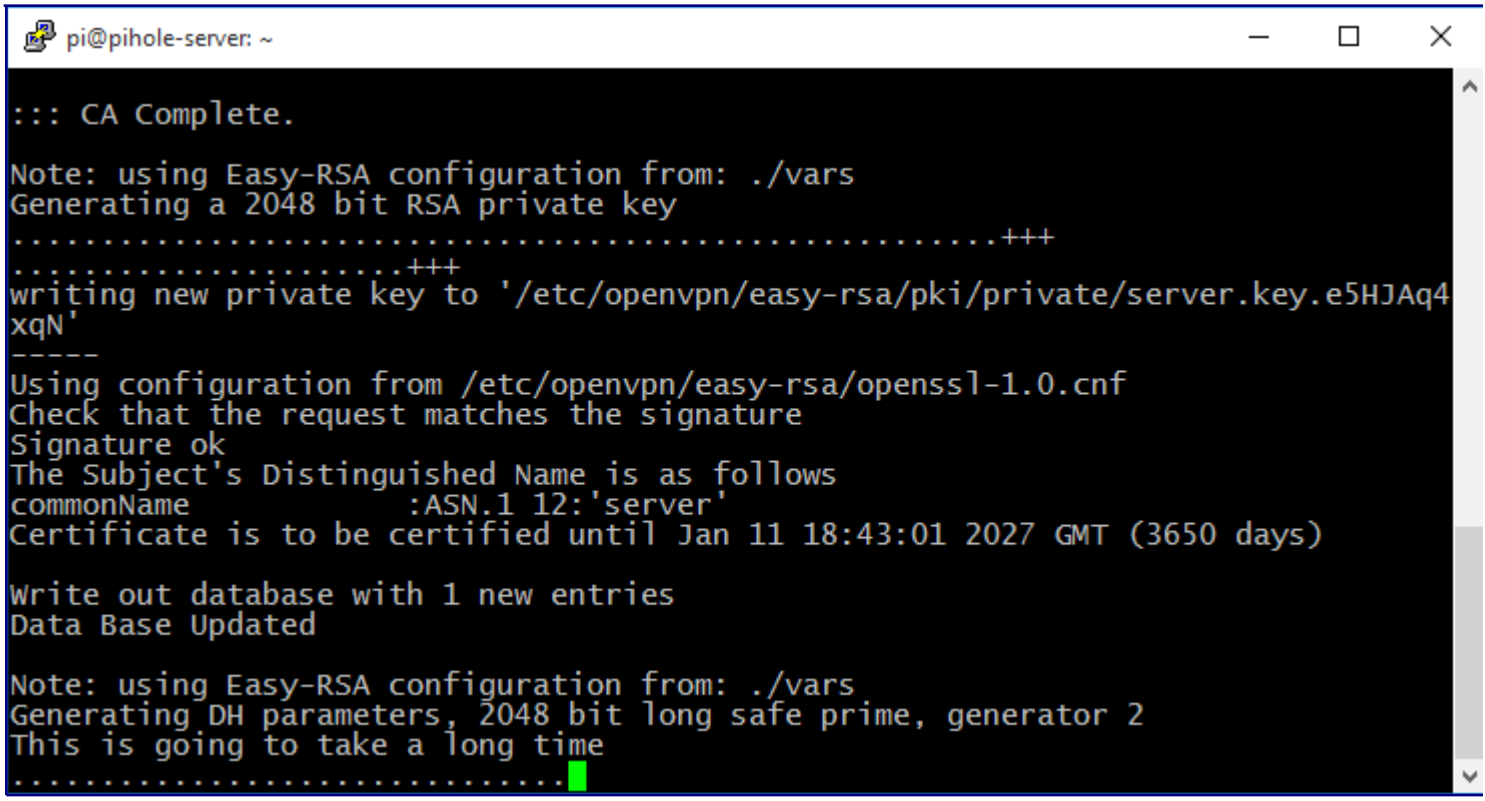

pi@pihole-server: ~
Specify Custom Port
Confirm Custom Port Number
Are these settings correct?
PORT: 11948
<Yes>
<No>

The next step is to set the size of your encryption key. I suggest the 2048 bit encryption only because its secure enough. I wouldn’t suggest dropping to 1024 bit encryption unless you are running a old Raspberry Pi. Since I am installing this on a Raspberry Pi 3 then 2048 bit encryption will be sufficient enough and will run with no issues. I have never tried the 4096 bit encryption, the only difference will be that it will take a longer time to create the encryption key and will be more secure if trying to crack it.


pi@pihole-server: ~
Setup OpenVPN
Encryption Strength
Choose your desired level of encryption:
This is an encryption key that will be generated on your system. The larger the key, the more time this will take. For most applications it is recommended to use 2048 bit. If you are testing or just want to get through it quicker you can use 1024. If you are paranoid about ... things... then grab a cup of joe and pick 4096.
(*) 2048 Use 2048-bit encryption. Recommended level.
( ) 1024 Use 1024-bit encryption. Test level.
( ) 4096 Use 4096-bit encryption. Paranoid level.
<Ok>
<Cancel>

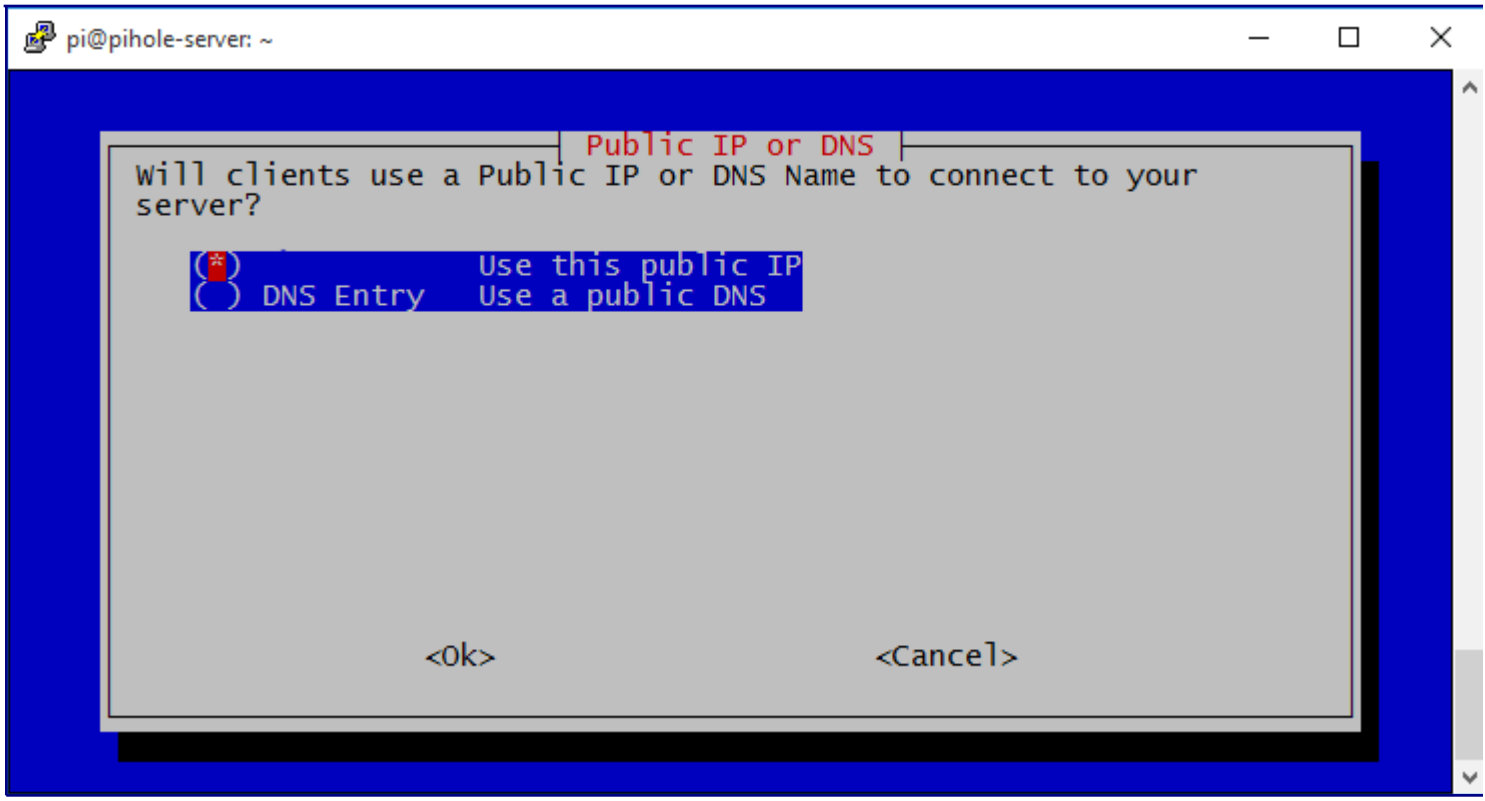

pi@pihole-server: ~
Setup OpenVPN
Server Information
The server key, Diffie-Hellman key, and HMAC key will now be generated.
<Ok>

You will get the following screen when your key is being generated. It will take a few minutes to generate. It took my Raspberry Pi 3 around 3 minutes to generate a 2048 bit encryption key.



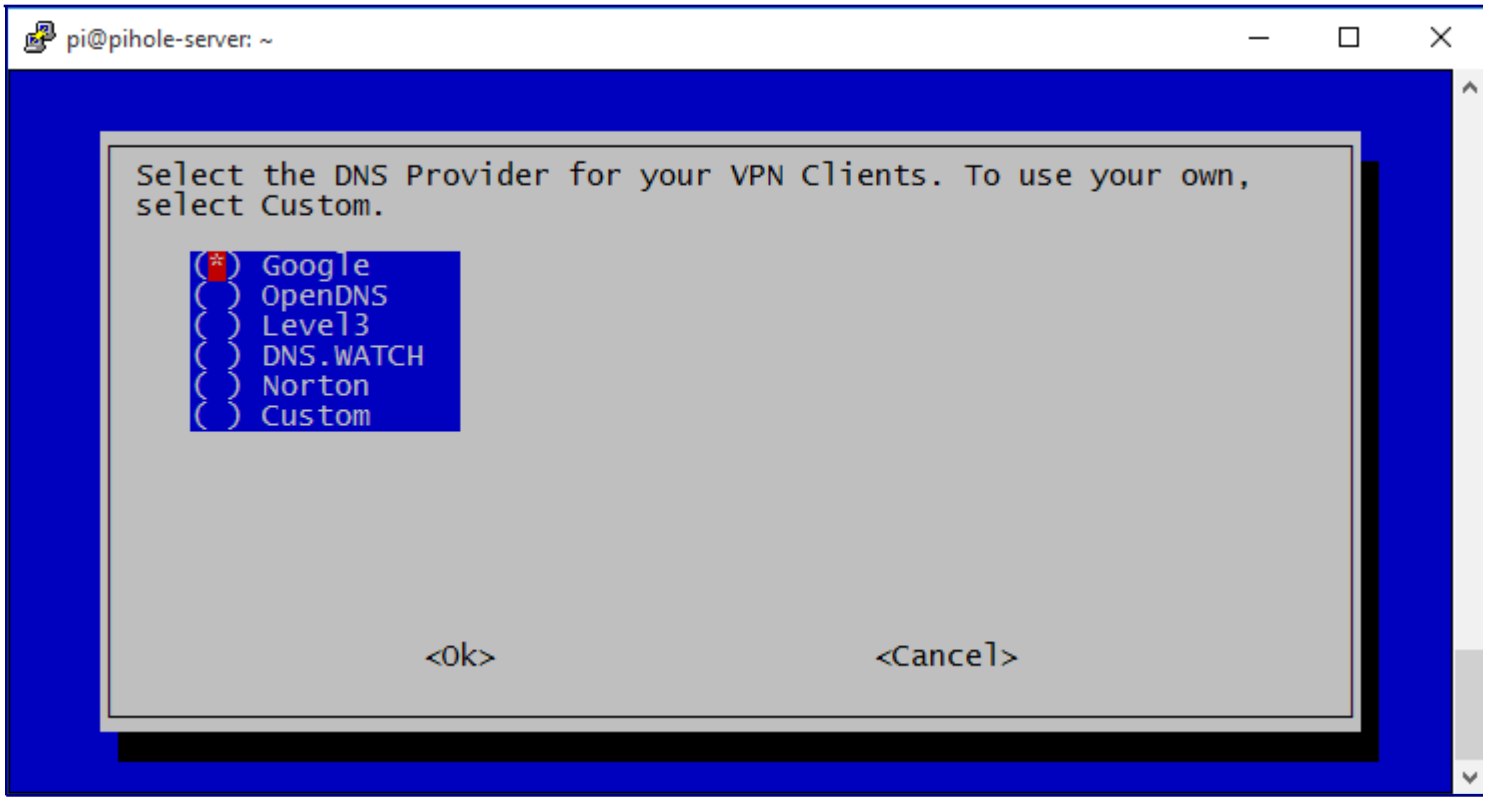
The next step will be to set up your DNS entry. I blanked out my IP Address since I did not want to expose it. If you have a static IP Address from your internet provider then I would use this IP Address. If you have an IP Address that changes randomly then you can use the DNS Entry screen. You will need to sign up for a DNS website like No-IP that will track your IP Address. You will get a name like

xxxx.noip.com which you will put in the DNS Entry screen.

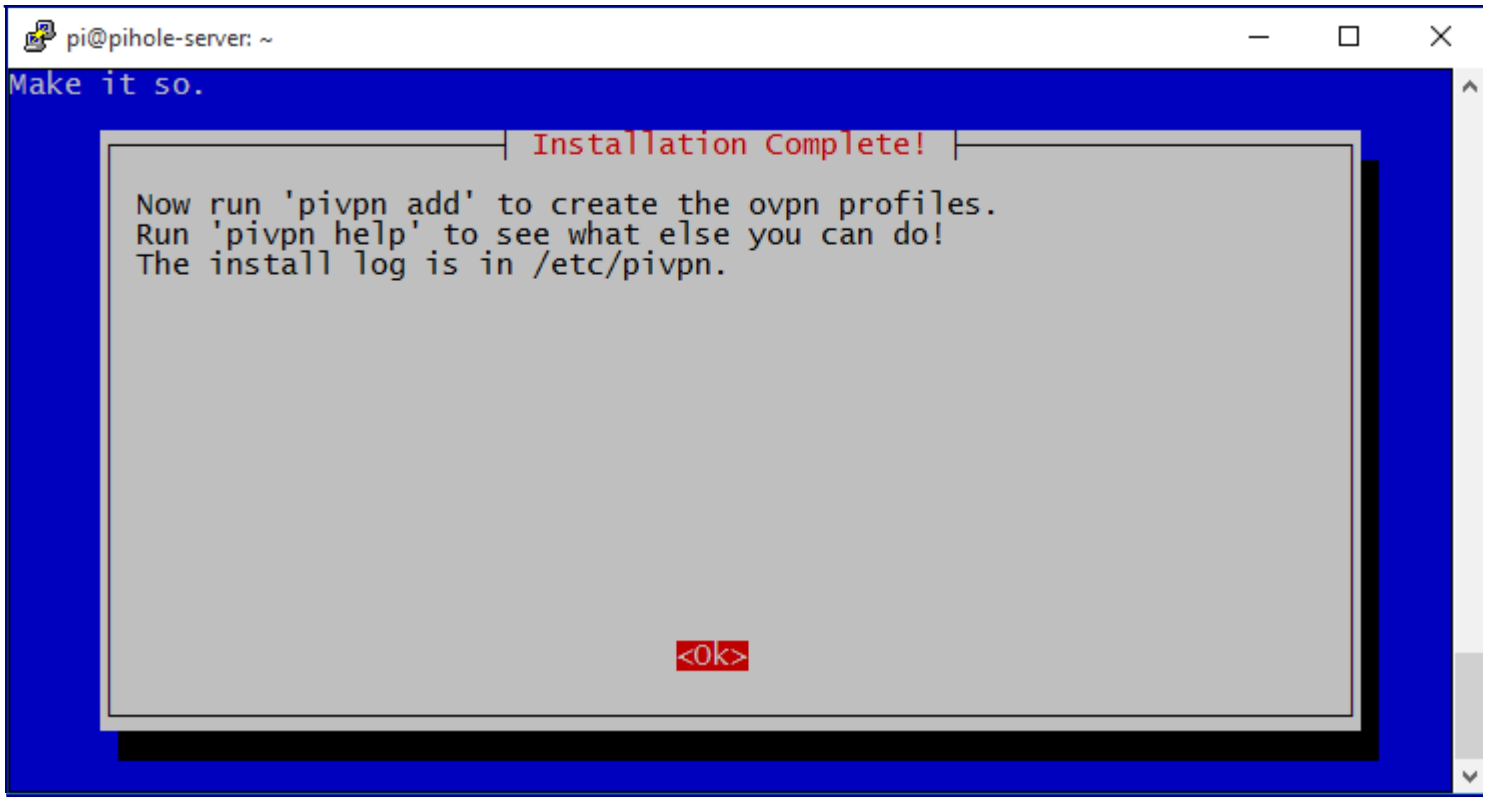


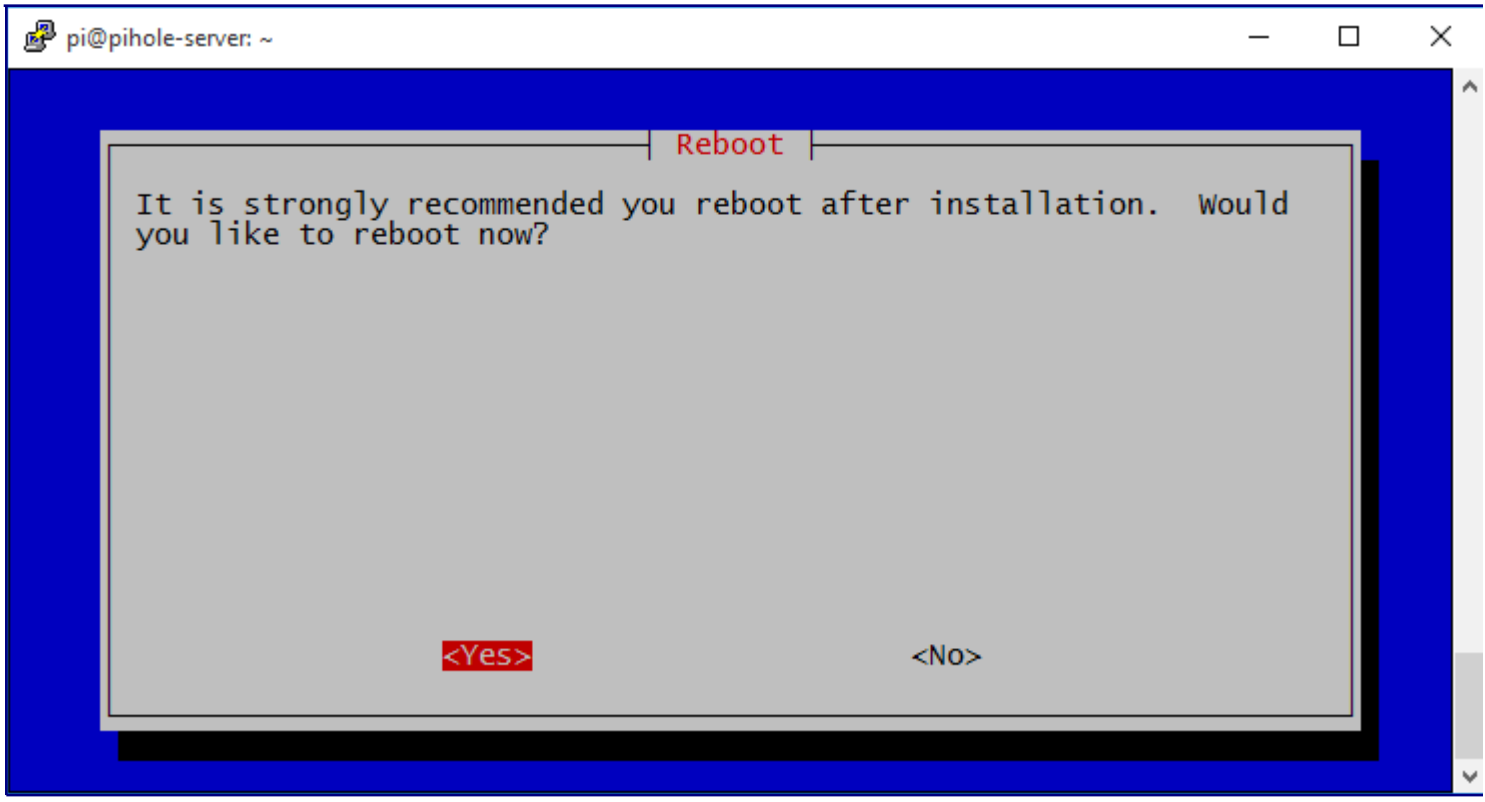
Next, you'll be asked to select the DNS provider you'd like to use for your VPN. This can be important if the reason you're looking to have a VPN is for privacy. The DNS provider converts URL's into IP Addresses and lets your computer know where to go on the internet. Many DNS providers log this information and can build a data-set about you. If you don't know which DNS provider to choose

simply use Google’s DNS provider.



That’s it! You will get the following screens telling you to run the ‘pivpn add’ command as well as rebooting to make sure all the configuration files are applied. Go ahead and reboot.


pi@pihole-server: ~
Make it so.
Installation Complete!
Now run 'pivpn add' to create the ovpn profiles.
Run 'pivpn help' to see what else you can do!
The install log is in /etc/pivpn.
<Ok>


pi@pihole-server: ~
Reboot
It is strongly recommended you reboot after installation. Would you like to reboot now?
<Yes>
<No>

After the reboot go ahead and run the following command to upgrade and install all our packages. After doing that reboot one more time to make sure everything is applied:

```
sudo apt-get upgrade
```

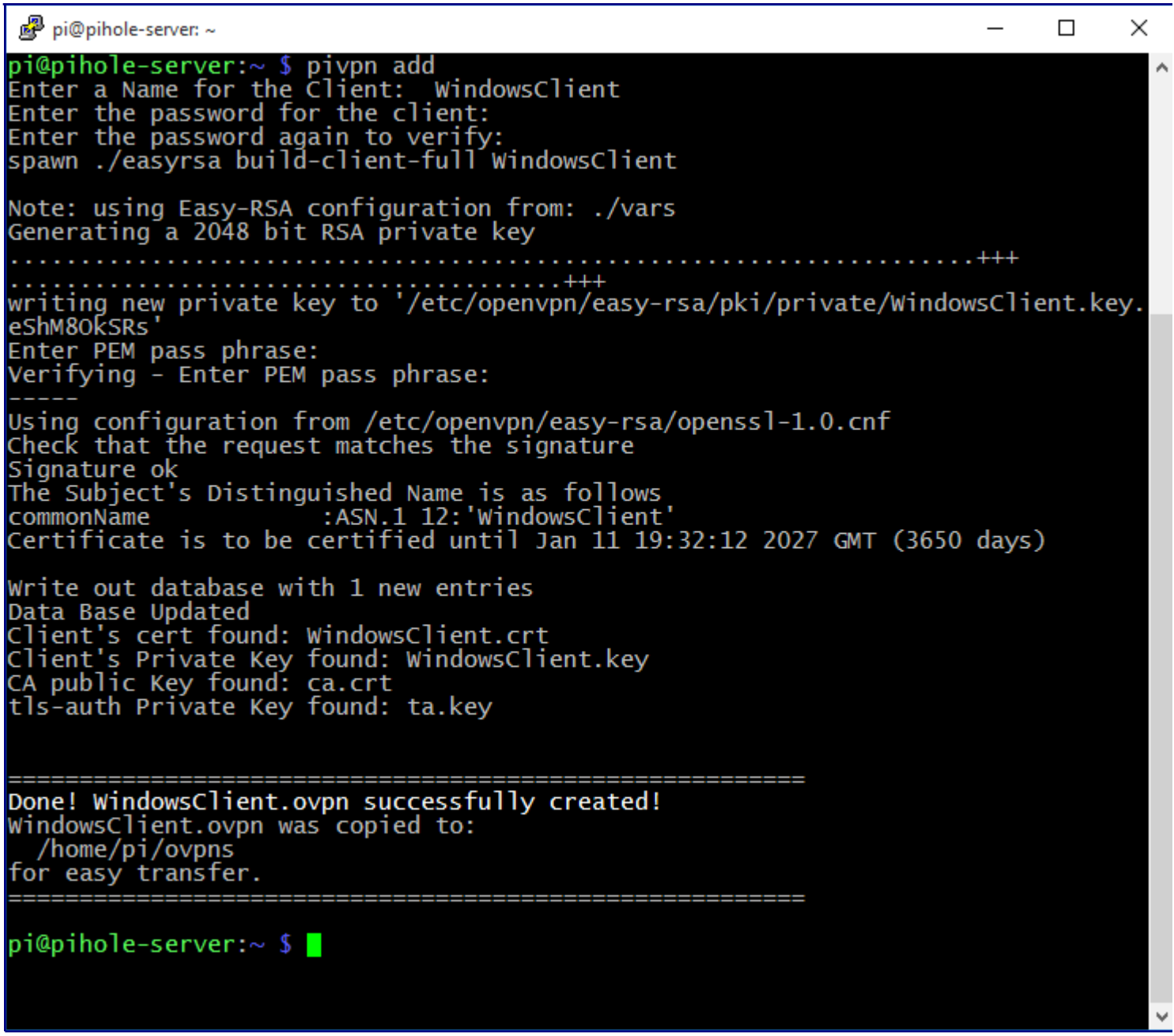
```
1 sudo apt-get upgrade
```

Create your OpenVPN Client File

Once you have rebooted your Raspberry Pi again, run the ‘pivpn add’ command to create a .ovpn file which we will need to transfer to our clients. This file contains a generated key that is used for logging in to our server. You can use this file for every device or you can generate new .ovpn files with the ‘pivpn add’ command.

```
pivpn add
```

```
1 pivpn add
```



When creating the .ovpn file, you will be asked for a pass phrase. This pass phrase will need to be entered each time you use your VPN client to connect to your Raspberry Pi VPN server. I suggest you use a strong and long pass phrase since the client .ovpn encrpytion file and the pass phrase are your only weaknesses for someone hacking your Raspberry Pi VPN Server. Keep your configuration/encryption file safe.

OpenVPN Clients

There are many OpenVPN clients to choose from. I use the official OpenVPN software for my Windows computer and my Android phone. I don’t own a Mac or an IPhone so I can not recoomend anything on that end.

The OpenVPN client for Android can be found here. You can download the official client for Windows from the OpenVPN website here.

Options for Transferring your .OVPN file to your OpenVPN Client

You will need to transfer the .ovpn file your created in the previous step to your client. The client is device which you will be using to connect to your Raspberry Pi VPN server. Your computer or phone can both be clients.

If your client will be a PC or Mac computer then the easiest way to transfer your .ovpn file will be over FTP. You can download a FTP client like FileZilla to connect to your VPN server and transfer the .ovpn file. Once you transfer it you will need to import this file into your VPN client.

if your client is a phone like and Android or an IPhone you have two options. You can either email the .ovpn file or you can transfer it using an SD card. If you email the file remember to delete from your email since you want to keep this file a secret. If this file gets compromised then the only thing that's stopping your Raspberry Pi VPN server from getting hacked is your pass phrase, that is why you need a strong pass phrase as well.

Port Forwarding on your Router

The final step you will want to do is to forward your Raspberry Pi's VPN port on your router. The default port you need to forward will be 1194 unless you changed this port in the PiVPN setup. Google "port forwarding" and your router name to find out how to do this for your own router.

Conclusion

With PiVPN setting up OpenVPN on the Raspberry Pi couldn't have been easier. Having your own VPN server on the Raspberry Pi will definitely improve your privacy and online security when you are away from home. Setting up your own VPN server only takes a few minutes and the step by step guide created by PiVPN is great.

The one thing I can not stress enough is locking down your Raspberry Pi because you will be exposing your Pi to the wider internet with the port forwarding. This may increase the attacks to your network and I recommend reading some basic security steps you can do to improve the security on your Raspberry Pi and your network.

How to install a web server on the Raspberry Pi (Apache + PHP + MySQL)

After creating your SD card, and after starting your Raspberry Pi for the first time, there are strong chances that you want to use as a web server.

Why a Raspberry Pi as a web server ?

But **why use a Raspberry as a web server**, rather than using services providers specialized in web hosting?

First, from an economic point of view, you should know that **web hosting services are not free** and that you have to pay every month / year. Unlike the Raspberry **who just need to a connection**.
In addition, by choosing Raspberry, **you have the possibility to modify your services like you want** (examples: the size of the disk, the hosting of Database, etc.), which is generally **not the case with specialized hosts** , Which often sell shared hosting with **low configuration capacity**.
However, to support more users, you should use a Raspberry Pi 3 (the Pi 3 can be found here), the Raspberry Pi with 1 GB of RAM, rather than the Raspberry type B + (512 MB of RAM)

The question that now arises is, **how to make a web server on Raspeberry Pi** ?Installation du serveur Apache avec Raspbian

What is Apache ?

First, we will install Apache, which **is the web server as such**.
When we speak of a web server, we often think about the machine, but **this term also refers to the software** that allows the machine to analyze user requests (in http form), and to return the file corresponding to the request (Or an error if the file isn't found, or the query incorrectly formulated).
As part of Apache, **it's software that we talk about**.

At the moment, **Apache is the most used web server**, with about 60% market share. Apache has its own license, used by many other projects. In addition, the massive use of Apache (which has become the standard for web servers), coupled with its high popularity, has led to a tremendous abundance of documentation, courses, and other books dealing with its use, and his security, like this book.

Whether it is for the Raspberry Pi and Raspbian, or for a more general-purpose machine, **Apache is therefore a safe choice**, and the skills you will be able to acquire on the subject will always be useful.

Apache installation

Before installing the server, make sure we have an up-to-date machine. To do this **we must have administrator rights**, either because of the sudo command.

```
sudo apt update
```

```
sudo apt upgrade
sudo apt update
```

Once the Raspberry Pi is up to date, we will install the Apache server.

```
sudo apt install apache2
```

By the way, we'll take advantage of it to give rights to the apache file that you can easily manage your sites. To do this, run the following commands:

```
sudo chown -R pi:www-data /var/www/html/
sudo chmod -R 770 /var/www/html/
```

Check if Apache is working

Once the installation completed, we can **test that Apache is working properly** by going to the Raspberry address.
To do this, it's necessary to try to access to the Raspberry from port 80 (this port not being opened from the outside, it will have to do since the Raspberry itself). Do not worry, it's very easy. Simply open the Raspberry web browser, and go to "http://127.0.0.1". You should then get a page with a message like "It works! "And plenty of other text.
If you do not already have a GUI on your Raspbian, or you use SSH to connect to your Raspberry, you can use the following command:

```
wget -O check_apache.html http://127.0.0.1
```

This command will **save the HTML code of the page in the file "check_apache.html"** in the current directory.
So you only have to read the file with the command

```
cat ./check_apache.html
```

If you see marked at a location in the code "It works! " is that Apache is working.

Apache uses the directory "/var/www/html" as the root for your site. This means that when you call your Raspberry on port 80 (http), Apache looks for the file in "/var/www/html".
For example, if you call the address "http://127.0.0.1/example", Apache will look for the "example" file in the "/var/www/html" directory.
To add new files, sites, etc., you will need **to add them to this directory.**

You can now use your Raspberry to make a site in **HTML, CSS and JavaScript**, internally.
However, you may want to quickly **allow interactions between the site and the user**. For example, to allow the user to register, etc. For this, you are going to need PHP.

PHP installation on your Raspberry Pi

What is PHP ?

First of all, you should know that **PHP is an interpreted language**. And as in the case of servers, the acronym PHP **can have several meanings**. In fact, when we talk about PHP, we can talk about **either the language or the interpreter**.
Here, when we talk about installing PHP, it means that **we will install the interpreter**, in order to use the language.

PHP (the language this time) is mainly used to make a site dynamic, that is to say that the user sends information to the server which returns the **modified results according to this information**.
Conversely, a static site **doesn't adapt to information provided by a user**. It's saved as a file once for all, and will always deliver the same content.

PHP is free, and maintained by the PHP Foundation, as well as Zend Enterprise, and various other companies (it should be noted that Zend is also the author of the famous Zend PHP framework, widely used and recognized in the world of " business).

It's one of the most widely used programming languages, and it is even the most used for web programming, with about 79% market share.

Again, all the skills you can acquire, on the language, or on the installation and configuration of the interpreter, will always be useful. So we can only advise you to learn the PHP, which is really a wonderful language and too often underestimated.

How to install PHP

We will again use the administrator to install PHP with the command line.

```
sudo apt install php php-mbstring
```

Control if PHP is working

To know if PHP is working properly, it's not very complicated, and **the method is quite similar to the one used for Apache**.

You will first **delete the file "index.html"** in the directory "/var/www/html".

```
sudo rm /var/www/html/index.html
```

Then create an "index.php" file in this directory, with this command line

```
echo "<?php phpinfo ();?>" > /var/www/html/index.php
```

From there, **the operation is the same as for the Apache check**. You try to access your page, and you should have a result close to this image (if you do not have an interface, use the same method as before, and look for the words "PHP Version").

PHP Version 5.4.4-14+deb7u7	
System	Linux ajaniserveur 3.2.0-4-amd64 #1 SMP Debian 3.2.54-2 x86_64
Build Date	Dec 12 2013 08:42:50
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-gd.ini, /etc/php5/apache2/conf.d/20-mysql.ini, /etc/php5/apache2/conf.d/20-mysqli.ini, /etc/php5/apache2/conf.d/20-pdo_mysql.ini
PHP API	20100412
PHP Extension	20100525
Zend Extension	220100525
Zend Extension Build	API220100525,NTS

Table generated by the phpinfo command on a raspberry.

A MySQL database for your server

A DBMS what's it ? Why MySQL ?

Now that we have set up PHP, you will probably **want to store information** for use in your sites. For this purpose, databases are most often used.
We will therefore set up a DBMS (Database Management System), namely MySQL.

MySQL is a free, powerful, massively used DBMS (about 56% market share of free DBMS). Here again, MySQL is so essential to development, whatever the language, that you must absolutely learn and master it, with this book for example.

How to install MySQL

To do this, we will install mysql-server **and** php-mysql (which will serve as a link between php and mysql)

```
sudo apt install mysql-server php-mysql
```

Verify that MySQL is working correctly

To check the operation of MySQL, this time **we will only use the command line**. To do this, we will simply connect via the command:

```
sudo mysql --user=root
```

We will no delete the default mysql root user and create a new mysql root user, because the default one can only be used with Linux root account, and so not available for the webserver and php scripts.

To do so, once your connect to MySQL, simply run thoses commands (replace `password` with the password you want) :

```
DROP USER 'root'@'localhost';
CREATE USER 'root'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost'
```

So you now have **a web server, connected to PHP and MySQL**. That's all it takes.

(On your nexts connections, you will be able to connect to mysql without using sudo, with the command `mysql --user=root --password=yourmysqlpassword`).

Add PHPMyAdmin

The installation of PHPMyAdmin is absolutly not necessary. **In this installation, we will not take care about any special security settings !**

The PHPMyAdmin installation is pretty quick and easy, we simply have to use the packet manager with this command :

```
sudo apt install phpmyadmin
```

PHPMyAdmin installation program will ask you few question. About the `dbconfig-common` part, choose to not use it (as we have already configure our database). About the server to configure PHPMyAdmin for, choose Apache. And the root password is the one you set for MySQL.

Check that PHPMyAdmin is working properly

To check that PHPMyAdmin works, you will simply try to access it, using the address of your Raspberry followed by /phpmyadmin. For example, locally it will be `http://127.0.0.1/phpmyadmin`

If you still get an error, it could be because PHPMyAdmin has moved to another directory. In this case, try the command

```
sudo ln -s /usr/share/phpmyadmin /var/www/html/phpmyadmin
```

Now, we can access to PHPMyAdmin from Raspberry Pi's browser, with the url : `http://127.0.0.1/phpmyadmin`

Making a server accessible from the web

Your web server is ready. However, **you probably can not access it from the internet**. Indeed, it would be necessary for that your modem to redirects the requests to your Raspberry, the good ports. To put these redirections in place, and even get a URL, you should look to DynDNS and port forwarding !

How to Install a LAMP Server on Debian 9 Stretch Linux

Introduction

The LAMP server is the cornerstone of Linux web hosting. In the early days of dynamic web content LAMP was what won Linux the crown in the web space, and it still is responsible for powering a very large portion of the Internet's sites.

If you're looking to set up a LAMP stack to host your website, it'd be hard to find a better option to build it on than Debian Stretch. Debian is, after all, well known for its stability, security, and massive package repositories, and Stretch is certainly no exception.

MariaDB(MySQL)

To get started, install and setup the database portion of the stack, MariaDB. Traditionally, the "M" in LAMP stands for MySQL. However, MariaDB is a drop-in replacement that isn't controlled by Oracle, so it tends to be a better option.

To install MaridaDB on Stretch, just use `apt` to install the packages.

```
# apt install mariadb-client mariadb-server
```

During the install process, you will be prompted to create a root password for MariaDB. Make sure to choose something as secure as possible, since it will determine, in part, the security of your databases.

Now that the MariaDB server is installed, you can log in as your root user and set up a regular user and a database.

```
mysql -u root -p
```

MariaDB will then prompt you for the root password that you just set up.

Creating a database is fairly simple. Just run the following.

```
CREATE DATABASE newdb;
```

You need to create a regular user now to use the database. It is an absolutely terrible idea to use the root user for anything other than managing MariaDB as a whole.

```
CREATE USER 'username'@'localhost' IDENTIFIED BY 'userpassword';
```

That command creates a regular user that can sign in locally and set that user's password.

In order for that user to be able to use the database that you just created, you need to grant them privileges on it. Since this is a general purpose user for managing everything on this database, it will be given all privileges.

```
GRANT ALL PRIVILEGES ON newdb.* to 'username'@'localhost';
```

Once that's done, flush all privileges from the console and exit.

```
FLUSH PRIVILEGES;
quit
```

That's all for the database. Certainly, you can customize any portion of this as you need.

PHP

The next step in getting the LAMP server set up is installing PHP. In the LAMP stack, PHP powers the web content and interacts with the database. To install PHP on Debian Stretch, run the following line.

```
# apt install php7.0 php7.0-mysql
```

That's really all that you need. PHP is now ready to use.

Apache

The Apache web server is extremely powerful and can be extremely easy to set up or ridiculously difficult, depending how in-depth you wan to go. Because this is just a simple guide, it's going to follow the quickest path for getting a basic server set up.

So, install both the Apache server and the module for PHP support.

```
# apt install apache2 libapache2-mod-php7.0
```

Testing Your Server

By default, Apache will server the contents of `/var/www/html` and will look first for a file called `index.php` or `index.html`. Create that file, and place the following line of code in it.

```
<?php phpinfo(); ?>
```

Open up your browser and type in `localhost` in your address bar. If you aren't doing this locally, type your domain name or IP. You should see a long table containing information about your PHP install. At this point, your sever is officially working.

If you want an easy way to manage your database through a graphical web interface, you can install an application called, phpmyadmin. It allows you to manage your database using PHP through your LAMP server. To install it on Stretch, just pull it with `apt`.

```
# apt install phpmyadmin
```

Once the package installs, you can navigate in your browser to `localhost/phpmyadmin` You will be greeted with a login screen that will accept your database credentials and finally, an interface to work with your database.

Closing Thoughts

Your LAMP server is now ready to go. Of course, there are tons of other options, and if you plan to use this as a public facing server, you may want to look into more security options for Apache. That said, this LAMP server can run everything from your custom PHP application to popular solutions like WordPress and even development frameworks like Laravel.

MotionEye

Installation

Calin Crisan edited this page · 17 revisions

Requirements

- a machine running a recent Linux distro
- python 2.7
- tornado 3.1+
- jinja2
- PIL or pillow
- curl, libcurl & pycurl
- motion (optional)
- ffmpeg (optional)
- v4l-utils (optional)

The Motion Daemon

The `motion` daemon itself is optional, but needed in most cases. Install it (along with `ffmpeg` and `v4l-utils`) unless you configure a machine that will only act as a hub for other motionEye-based cameras.

Install Instructions

motionEye releases are uploaded to PyPI, so you can use the `pip` (or `pip2`) command to install it as well as (some of) its dependencies. Following are detailed instructions for some common distributions.

note 1: The given commands normally need to be run as root; type them in a root shell or use `sudo` before each command.

note 2: On systems where Python3 is the default Python interpreter, you should use the `pip2` command instead of `pip`.

note 3: If you are configuring a motionEye system that will only act as a hub for other motionEye-based cameras (i.e. no locally connected cameras and no IP cameras), you can skip installing `motion`, `ffmpeg` and `v4l-utils`.

Choose one of the following specific install instructions. When you're done, you may want to come back here and read on to find out how to access the frontend or how to update your motionEye.

- Install On Debian
- Install On Ubuntu

- Install On Raspbian
- Install On Fedora
- Install On Arch
- Install On Other Distributions
- Manual Download And Installation

Accessing The Frontend

After having successfully followed the installation instructions, the motionEye server should be running on your system and listening on port 8765. Fire up your favorite web browser and visit the following URL (replacing `[your_ip]` with... well, your system's IP address):

```
http://[your_ip]:8765/
```

Use `admin` with empty password when prompted for credentials. For further details on how to configure motionEye, see Configuration.

Upgrading

Upgrading should be as easy as running the following command (as root):

```
pip install --upgrade motioneye
```

To upgrade to a specific version (say 0.27.1), use:

```
pip install --upgrade motioneye==0.27.1
```

If you have manually downloaded and installed motionEye, the `pip` command above won't work and you'll need to repeat the manual installation procedure, preserving the configuration directory.

Install On Raspbian

Calin Crisan edited this page · 29 revisions

Before Proceeding

- Read the general Installation page first.
- These instructions apply only to an up-to-date Raspbian Stretch.
- All commands require *root*; use `sudo` before each command or become root using `sudo -i`.
- If you want to use the CSI camera module for the Raspberry PI, make sure you have enabled it in `raspi-config`.

Instructions

1. Install `ffmpeg` and `v4l-utils`:

   ```
   apt-get install ffmpeg v4l-utils
   ```

   note: `v4l-utils` appears to be preinstalled on Raspbian systems

2. Install `libmariadbclient18` and `libpq5` required by `motion`:

   ```
   apt-get install libmariadbclient18 libpq5
   ```

3. Install `motion`:

   ```
   wget https://github.com/Motion-Project/motion/releases/download/release-4.1.1/pi_stretch_motion_4.1.1-1_armhf.deb
   dpkg -i pi_stretch_motion_4.1.1-1_armhf.deb
   ```

   note: Raspbian Stretch comes with `motion` version 4.0; it is however recommended that you install version 4.1

4. Install the dependencies from the repositories:

   ```
   apt-get install python-pip python-dev libssl-dev libcurl4-openssl-dev libjpeg-dev libz-dev
   ```

5. Install `motioneye`, which will automatically pull Python dependencies (`tornado`, `jinja2`, `pillow` and `pycurl`):

   ```
   pip install motioneye
   ```

6. Prepare the configuration directory:

   ```
   mkdir -p /etc/motioneye
   cp /usr/local/share/motioneye/extra/motioneye.conf.sample /etc/motioneye/motioneye.conf
   ```

7. Prepare the media directory:

   ```
   mkdir -p /var/lib/motioneye
   ```

8. Add an init script, configure it to run at startup and start the `motionEye` server:

   ```
   cp /usr/local/share/motioneye/extra/motioneye.systemd-unit-local /etc/systemd/system/motioneye.service
   systemctl daemon-reload
   systemctl enable motioneye
   systemctl start motioneye
   ```

9. To upgrade to the newest version of motionEye, just issue:

   ```
   pip install motioneye --upgrade
   systemctl restart motioneye
   ```