Pi3 Home Server – Ubuntu-MD April 22, 2017

Overview

Introduction

This demonstration will cover setting up a Raspberry Pi3 with Raspian-lite Debian Jessie operating system (OS) to be used as a home based server. The server will act as a network file server using Samba and an OpenVPN server for remote access from other devices. It will be designed to run without a monitor (headless) and with no keyboard or mouse attached after the initial configuration. Additional storage via an attached usb hard drive of at least 1 tb in size is highly recommended. The OS will run on a 32 gb class 10 sd card. We will also hard wire it to our network and not use a wireless connection.

We will be using the command line in a terminal for the initial setup and configuration. This is because most servers do not have a windowed desktop GUI due to the resources that would be needed to support it. Once it is configured, I will introduce you to Webmin which is a web based interface that you can use to manage the server.

Setting Up the Pi3

To prepare the microSD card for use you first must download the OS using this link, https://www.raspberrypi.org/downloads/raspbian/

On Ubuntu, you can use usb startup disk creator program or the command line to write to the microSD card. You need an adapter to insert it in your computer and once inserted run lsblk in the terminal to identify what your system has labeled it (i.e /dev/sdbx, /dev/sdcx). I would suggest that you remove it run the lsblk command again to ensure you don't see it then reinsert it.

```
      sudo dd if=raspian-lite.img of=/dev/sdc.. status=progress bs=4M && sync
Next you want to insert the microSD card into your Pi3, connect a usb keyboard,
ethernet cable and connect the hdmi cable from the Pi3 to your monitor. Lastly,
connect the power cable to the Pi3 which will start it.
When it starts, you will only see a terminal prompt after the initial programs
load. The default user is pi and the default password is raspberry. Login with
these credentials then immediately change the pi password using the command,
sudo passwd pi
```

You can also add a new user with the command

    sudo adduser (user name) #without the ( )

when completed give it sudo privileges with

```
sudo adduser (user name) sudo
```

We need to update the system using the following:

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

Next you want to edit the rasbian-lite configuration,

```
sudo raspi-config
```

You can also change the default pi user password from here and you can change the hostname (raspberrypi is the default) so it will be visible on your network.
Turn on ssh server from the Interfaces section in raspi-config so you can run the system headless without a monitor. You may need to adjust the ssh configuration file sudo nano /etc/ssh/sshd_config and change

PermitRootLogin from No to Yes.

Set up static ip interface for your device:
route -n to determine your gateway and write it download
Edit file sudo nano /etc/dhcpcd.conf and add at bottom of file (crtl O to save when done)

```
interface eth0
static ip_address={DESIRED IP ADDRESS}/24
static routers={GATEWAY ADDRESS}
static domain_name_servers={GATEWAY ADDRESS}
```

sudo reboot # to reboot server and apply the changes
Test remote access from a terminal on another network device with the command:

ssh user name@{IP_address} (You make be asked to confirm the connection and then be prompted for your password. Once entered you will be in a terminal on the Pi.

Adding Hard Drive

Attach the usb hard drive and reboot the Pi (sudo reboot)
When it comes back up, enter the command:

    sudo lsblk (this should show all your drives to include the microSD (mmcblk0x, /dev/sd{a,b,c}

It can be formatted as ext4 or ntfs
sudo parted
print all (to see all the devices on your system)

```
select /dev/sda
print
mklabel gpt
print
mkpart primary 0GB 100%
q
sudo mkfs.ext4 /dev/sda1 (if drive wasn’t formatted as ext4 when primary partion
was created)
sudo fdisk -l
sudo mkdir /data (make dir for mounting drive)
sudo mount /dev/sda1 /data (mount the drive)
sudo chgrp -R users /data (adding group permissions to users)
sudo chmod -R g+w /data (allow anyone from the user group to write to the drive)
ls -l /dev/disk/by-uuid/ (to determine id of the external drive, /dev/sda1)
sudo nano /etc/fstab (open the file and add a line at the bottom so the drive
mounts at boot up)
      UUID=e79c0ae1-49cb-4835-a13f-7fdd7ba88ecd [tab] /data [tab] ext4 [tab]
      auto,x-systemd.automount [tab] 0 [tab] 2 (replace with your drives uuid)
```

sudo reboot
login and enter df -h in terminal and you should see your drive listed mounted to /data

Samba

Samba allows your server to share files with Windows, Linux and other devices on your local network. All users must be added to Samba via smbpasswd command-line before they can access their files. You can also have public directories that will allow everyone access.
Install it: sudo apt-get install samba **samba-common-bin** (check if already installed; whereis samba)
Setup your shares starting with a public directory: mkdir /data/public (may require sudo)
Change folder ownership: sudo chown -R nobody:nogroup /data/public
Open Samba config file: sudo nano /etc/samba/smb.conf

First you want to enable WINS support so that your other computers can see your server name, uncomment WINS support = no and change it to yes.

You can also change your network Workgroup name to something else but that is not necessary.
Scroll down to just before the Networking section and add: guest account = nobody
Scroll down to the end of the file and add:

```
[public]
  path = /data/public
  public = yes
   writeable = yes
```

Save the file, Ctrl o and Ctrl x
Restart the service: sudo systemctl restart smbd.service nmbd.service
The above setup gives everyone who has access to your network read and write access to all files on your public share. If you want to give each registered user on your server access to their home directory or add a private directory only accessible by certain users you can do that as follows:
Scroll down to the Homes section in the sudo nano /etc/samba/smb.conf filenames
Change read only to no
Crtl o and Ctrl x to save and close the file.
Then for every user on the server, add them to the Samba password file, sudo smbpasswd {user}
Next create a new share directory like /data/accounts (make sure to create it on the attached drive first, mkdir /data/accounts and give it the correct permissions)

```
[accounts]
comment = Accounts data directory
path = /data/accounts
valid users = ron raj joe
public = no
writable = yes
```

Then restart samba, sudo systemctl restart smbd.service nmbd.service

OpenVPN

To be able to access your home network through the VPN, you will need to access your ISP home router and forward udp/tcp port 1194 to your Pi3 home server static ip. You will also need a dynamic IP resolution service like www.no-ip.org.
Need to install the service,

    sudo apt-get install openvpn openssl easy-rsa

You then want to copy the sample config file;

    sudo cp -r /usr/share/easy-rsa /etc/openvpn/easy-rsa

Open the vars file;

    sudo nano /etc/openvpn/easy-rsa/vars
    Find the export-rsa and set it as: export EASY_RSA="/etc/openvpn/easy-rsa"
    Verify that export KEY_SIZE is 2048
    Move to the bottom and change the defaults to match your site:

```
        export KEY_COUNTRY="US"
        export KEY_PROVINCE="MD"
        export KEY_CITY="Baltimore"
        export KEY_ORG="Home"
        export KEY_EMAIL="yourname@…"
        export KEY_OU="Pi3-Server-Name"
```

Save and close the file.
Now you need to create root certificates for your server;

```
    sudo su
    cd /etc/openvpn/easy-rsa
    source ./vars
    ./clean-all
    ./build-ca
```

(only critical field is common name where you want to use the name for the user you want to give access so you can identify their keys.)
When that finished, enter the following command, substituting the name of your server, and accepting the defaults again. You'll get a couple extra questions this time. Make sure the challenge password and company name are left blank, and accept any other defaults.

```
./build-key-server pi3home
```

Answer yes to the "Sign the certificate?" and "commit?" prompts.

Next, we'll build the Diffie-Hellman parameters file (this takes some time and the screen fills with dots).

```
./build-dh

openvpn --genkey --secret keys/ta.key
```

Configure the OpenVPN Server

Now it's finally time to edit the OpenVPN configuration and tie up the loose ends.

```
nano /etc/openvpn/server.conf
```

You'll notice that the editor is totally blank. That's because this file doesn't exist yet. Paste in the following text, substituting your own values for the highlighted values. You'll need your Raspberry Pi's IP address, the IP address of your router, and the name you used above when calling build-key-server.

Note: I've also highlighted the protocol (udp) and port (1194). You can change these to something else if needed.

```
local 192.168.1.XXX # YOUR PI'S IP ADDRESS
dev tun
proto udp
port 1194
ca /etc/openvpn/easy-rsa/keys/ca.crt
cert /etc/openvpn/easy-rsa/keys/RPHS.crt
key /etc/openvpn/easy-rsa/keys/RPHS.key
dh /etc/openvpn/easy-rsa/keys/dh2048.pem
server 10.8.0.0 255.255.255.0
# server and remote endpoints
ifconfig 10.8.0.1 10.8.0.2
# Add route to Client routing table for the OpenVPN Server
push "route 10.8.0.1 255.255.255.255"
# Add route to Client routing table for the OpenVPN Subnet
push "route 10.8.0.0 255.255.255.0"
# your local subnet
push "route 192.168.1.0 255.255.255.0" # YOUR PI'S IP SUBNET
# Set primary domain name server address to the SOHO Router
# If your router does not do DNS, you can use Google DNS 8.8.8.8
push "dhcp-option DNS 192.168.1.1" # YOUR ROUTER'S IP ADDRESS
# Override the Client default gateway by using 0.0.0.0/1 and
# 128.0.0.0/1 rather than 0.0.0.0/0. This has the benefit of
# overriding but not wiping out the original default gateway.
push "redirect-gateway def1"
client-to-client
duplicate-cn
keepalive 10 120
tls-auth /etc/openvpn/easy-rsa/keys/ta.key 0
cipher AES-128-CBC
comp-lzo
user nobody
group nogroup
persist-key
persist-tun
status /var/log/openvpn-status.log 20
log /var/log/openvpn.log
verb 1
```

Exit nano, saving your changes (ctrl-x, y, enter) Next, you need to allow the Raspberry Pi to forward IP traffic, which it does not do by default.

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

Find the line that says “Uncomment the next line to enable packet forwarding for IPv4”, and uncomment the line immediately after it.

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

Exit Nano, saving your changes (ctrl-x,y,enter), and force a reload of the settings.

```
sysctl -p
```

Configure Firewall

The Raspberry Pi has its own firewall, which must be configured to allow the VPN traffic through. Create a script file to automate the opening of the appropriate ports.

```
nano /etc/firewall-openvpn-rules.sh
```

Copy in the following text, substituting your own Raspberry PI’s IP address where highlighted.

```
#!/bin/sh

iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -o eth0 -j SNAT --to-source
192.168.1.XXX
```

Change the permissions on the file you just created so that it can be executed, and assign ownership to the root user.

```
chmod 700 /etc/firewall-openvpn-rules.sh
chown root /etc/firewall-openvpn-rules.sh
```

This script file needs to run every time the Raspberry Pi boots up in order to do us any good. Edit the /etc/network/interfaces file.

```
nano /etc/network/interfaces
```

Find the line that configures the wired ethernet port. If you are running your server wirelessly, then you’ll need to adjust accordingly. Insert a new line, indented underneath so that the result looks like this:

```
...
iface eth0 inet dhcp
    pre-up /etc/firewall-openvpn-rules.sh
```

```
...
```

This will ensure that the firewall rules are applied to that network interface even before it has started up.

One last thing, and this is new with the Jessie release. As commenter Craig pointed out in the previous version of this post, we need to stop OpenVPN from starting TOO soon. Edit the OpenVPN service file.

```
sudo nano /lib/systemd/system/openvpn@.service
```

Add the followling line at the end of the first section (Unit).

```
After=multi-user.target
```

Finally, reboot in order to apply the changes, and everything should be up and running.

```
sudo reboot
```

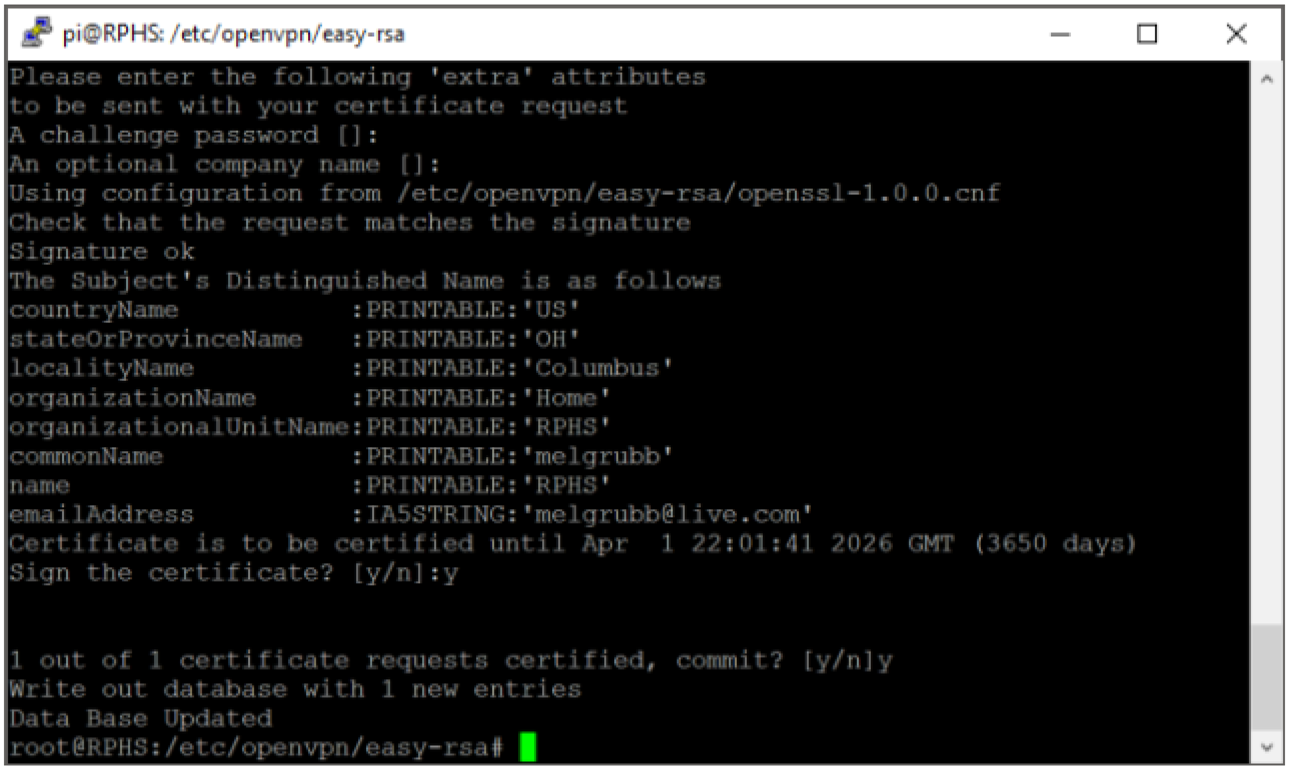
Generate keys

You may be able to just take your bank's word for it that they are who they say they are, but VPN servers like the one we're building want proof of the *client's* identity as well. They won't let just anyone in. You need to give a key to each device or user you want to allow to connect to the VPN server. You have a decision to make at this point. You could generate a unique key for each individual *device* that you want to connect via VPN, or you could take a shortcut and generate a key for each *user*. The difference is whether you expect to need to connect more than one device at the same time. If you don't need to connect more than one device per user at the same time, generate a key named for the user. If you think users will need more than one device connected at the same time, I'd suggest naming the key after the device. Whichever you decide, generate a key like this, substituting the name of the user or computer that will use this key to connect:

```
sudo su
cd /etc/openvpn/easy-rsa
source ./vars
./build-key-pass NAME
```

The PEM password is a password you'll need in order to use the resulting key file. Pick something nice and strong, but also something you won't forget. If you want to be really paranoid, you could randomly generate one and keep it in a password safe. The choice is yours.

Accept the remaining defaults again, again leaving the challenge password and company name blank. Sign and commit the certificate when prompted.



pi@RPHS: /etc/openvpn/easy-rsa

```
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Using configuration from /etc/openvpn/easy-rsa/openssl-1.0.0.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'US'
stateOrProvinceName   :PRINTABLE:'OH'
localityName          :PRINTABLE:'Columbus'
organizationName      :PRINTABLE:'Home'
organizationalUnitName:PRINTABLE:'RPHS'
commonName            :PRINTABLE:'melgrubb'
name                  :PRINTABLE:'RPHS'
emailAddress          :IA5STRING:'melgrubb@live.com'
Certificate is to be certified until Apr  1 22:01:41 2026 GMT (3650 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
root@RPHS:/etc/openvpn/easy-rsa#
```

Almost done. Next we need to convert the user/computer's key into a format usable by OpenVPN.

```
cd keys
openssl rsa -in NAME.key -des3 -out NAME.3des.key
```

Use the same password as you did before. You'll have to enter it three times. Technically, the first time is a different password, but how are you supposed to keep them straight?

Generating client keys

Connecting a VPN client to a remote server takes a bit of configuration, too. The OpenVPN client has to know where the server is, and it has to have a copy of the keys we generated earlier. All of this configuration gets wrapped up into a file with a .ovpn extension. You can create these by hand if you like, but Eric Jodoin, the author of the original SANS.org article was kind enough to write a script to do it for us. Create the script file.

```
nano /etc/openvpn/easy-rsa/keys/MakeOVPN.sh
```

This is a new file, so it will be totally blank. Paste in the following:

```
#!/bin/bash

# Default Variable Declarations
```

```
DEFAULT="Default.txt"
FILEEXT=".ovpn"
CRT=".crt"
KEY=".3des.key"
CA="ca.crt"
TA="ta.key"

#Ask for a Client name
echo "Please enter an existing Client Name:"
read NAME

#1st Verify that client's Public Key Exists
if [ ! -f $NAME$CRT ]; then
echo "[ERROR]: Client Public Key Certificate not found: $NAME$CRT"
exit
fi
echo "Client's cert found: $NAME$CR"

#Then, verify that there is a private key for that client
if [ ! -f $NAME$KEY ]; then
echo "[ERROR]: Client 3des Private Key not found: $NAME$KEY"
exit
fi
echo "Client's Private Key found: $NAME$KEY"

#Confirm the CA public key exists
if [ ! -f $CA ]; then
echo "[ERROR]: CA Public Key not found: $CA"
exit
fi
echo "CA public Key found: $CA"

#Confirm the tls-auth ta key file exists
if [ ! -f $TA ]; then
echo "[ERROR]: tls-auth Key not found: $TA"
exit
fi
echo "tls-auth Private Key found: $TA"

#Ready to make a new .opvn file - Start by populating with the default file
cat $DEFAULT > $NAME$FILEEXT

#Now, append the CA Public Cert
echo "<ca>" >> $NAME$FILEEXT
cat $CA >> $NAME$FILEEXT
echo "</ca>" >> $NAME$FILEEXT

#Next append the client Public Cert
echo "<cert>" >> $NAME$FILEEXT
cat $NAME$CRT | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' >>
$NAME$FILEEXT
echo "</cert>" >> $NAME$FILEEXT

#Then, append the client Private Key
```

```
echo "<key>" >> $NAME$FILEEXT
cat $NAME$KEY >> $NAME$FILEEXT
echo "</key>" >> $NAME$FILEEXT

#Finally, append the TA Private Key
echo "<tls-auth>" >> $NAME$FILEEXT
cat $TA >> $NAME$FILEEXT
echo "</tls-auth>" >> $NAME$FILEEXT

echo "Done! $NAME$FILEEXT Successfully Created."
```

Exit Nano, saving your changes (ctrl-x,y,enter) Once again, because this is a script, permissions will have to be altered to allow it to run. Make sure you match the casing of the script name here.

```
chmod 700 /etc/openvpn/easy-rsa/keys/MakeOVPN.sh
```

Create the Default.txt file to hold the default values the script will use. The casing isn't important, but it must match what was specified at the top of the script file. I'm keeping the capitalized "D" just to keep it the same as anyone else who followed Eric's instructions.

```
nano /etc/openvpn/easy-rsa/keys/Default.txt
```

Paste in the following, substituting your public IP address for the highlighted text. If you don't have a static public IP address, you can use a dynamic name from a service like DynDNS or no-ip here as well. The "1194" is the standard port number OpenVPN uses, adjust as needed to match your network configuration.

Note: Once again, I've highlighted the protocol (udp) and port (1194), which you can change if needed. To determine your server public ip go to http://ipchicken.com.

```
client
dev tun
proto udp
remote YOUR_PUBLIC_IP_ADDRESS 1194
resolv-retry infinite
nobind
persist-key
persist-tun
mute-replay-warnings
ns-cert-type server
key-direction 1
cipher AES-128-CBC
comp-lzo
verb 1
mute 20
```

Exit Nano, saving your changes (ctrl-x,y,enter) Execute the script to create a .ovpn file. Remember to use the user or device name you chose earlier when creating the client key.

```
cd /etc/openvpn/easy-rsa/keys
./MakeOVPN.sh
```

The result is a `NAME.ovpn` file in the `/etc/openvpn/easy-rsa/keys` folder on the Raspberry Pi. That's great, but we need the key on the client machine. You can copy the file using a secure copy program like WinSCP, copy it to a flash drive and move it by hand, or any other number of ways to move a file around. Since this is my own private home server, I'm going to put the file on the data share, at least temporarily. Once the key is installed and working on the client, I'll delete it from the server.

```
cp /etc/openvpn/easy-rsa/keys/NAME.ovpn /mnt/data/
```

Keys like this aren't something you should leave lying around. On the other hand, you should probably have a backup of them somewhere. If you put them on a flash drive, go put it in a safe or something. Don't let anyone get a hold of your keys, or they have a free pass into your home network, and you may not even notice it. You can always go back and generate new keys, delete the compromised ones, and continue on, of course.

Port forwarding

Before you'll be able to connect to your home network from outside, you'll need to set up your router to forward all traffic on port 1194 to the Raspberry Pi. I can't tell you how to configure the firewall on your router at home because I don't know what kind of router you have. An excellent resource that may have information specifically for your router is http://portforward.com/. You can also search FIOS and Xfinity (Comcast) for specific port forwarding guides for their home routers.

Client configuration

You can use the OpenVPN client for Windows, but the instructions should be similar for other platforms. You can download open-source clients for Windows, and source tarballs for other systems from here.

Note: Don't try to download client software from the links on the front page of the OpenVPN site or you'll just end up with "SecureTunnel", a paid-subscription-based system that lets you do exactly what you're already set up to do on your own. Get the .ovpn file that you generated on the Raspberry Pi over to the computer you're going to connect from, and put it in the OpenVPN `config` folder. For Windows users, this should be `C:\Program Files\OpenVPN\config`.

Connecting the client

You’ll need to be somewhere other than on your own network for this next part. Otherwise you’re seriously crossing the streams, shutting down the containment grid, etc. Disconnect from your home network and tether yourself to a phone or something before continuing. Run the OpenVPN GUI application. It should have created a shortcut in your start menu for Windows 7 users, or on your app list for Windows 8 users. Run it, and it should pick up on the .ovpn file and open a connection. Right-click on the notification area icon (for Windows users), and select “Connect”.

You’ll be prompted for the password you created earlier, and if everything is configured correctly, the OpenVPN icon should turn green, and you’ll be effectively connecting to the outside world as part of your home network. There are, of course, many issues you could run into when using a VPN. Most of them are explained pretty well on the HowTo page of the OpenVPN site. One of the more vexing problems is that of disambiguating IP addresses between your home network, and the network you are connected to. See “Numbering Private Subnets” for more information.

Uncomplicated Firewall (UFW)

Install: sudo apt-get install ufw
Add rules: sudo ufw allow ssh, http
Check status: sudo ufw status
Enable: sudo ufw enable
Disable: sudo ufw disable

Accessing Your Server by Name Over the Local Home Network

You can use avahi-daemon to access your server locally by name vs it’s ip address. Install it with:

```
sudo apt-get install avahi-daemon
```

Pi-hole (Suggested by Chuck Fran)

Network based ad blocker. Install;

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

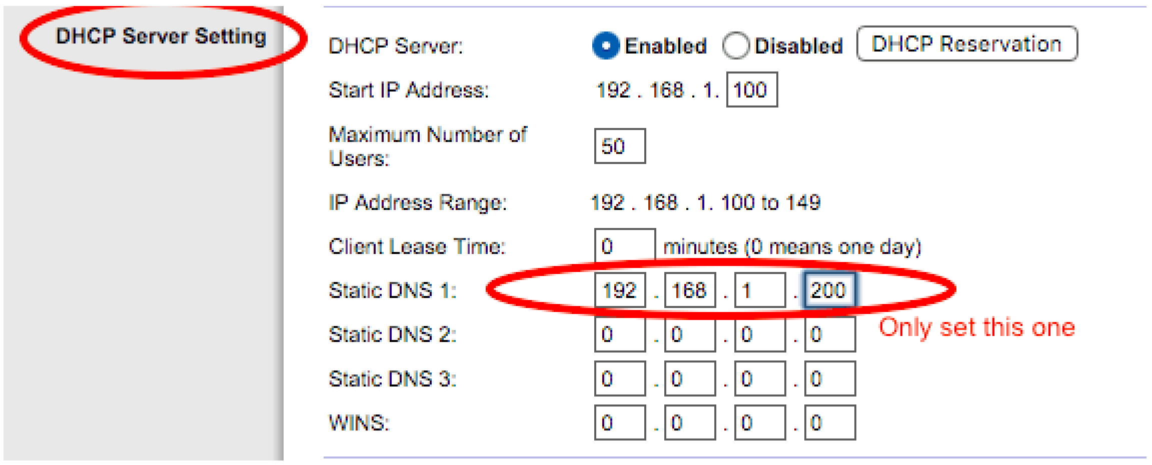
or
Slightly safer install
wget -O basic-install.sh https://install.pi-hole.net
bash basic-install.sh
Once installed have either your router or individual devices use pi-hole as your dns server:

Two Methods

Preferred: Set Your DNS Server In Your Router's Settings

This is the fastest way to get all of your devices using Pi-hole. If you set this configuration via your router's DHCP options, *any* device that connects to your network will immediately begin blocking ads.

Make sure you adjust this setting under your **LAN settings** and *not* the WAN.



DHCP Server Setting

DHCP Server:	[✓ Enabled] Disabled DHCP Reservation
Start IP Address:	192 . 168 . 1. 100
Maximum Number of Users:	50
IP Address Range:	192 . 168 . 1. 100 to 149
Client Lease Time:	0 minutes (0 means one day)
Static DNS 1:	192 . 168 . 1 . 200
Static DNS 2:	0 . 0 . 0 . 0
Static DNS 3:	0 . 0 . 0 . 0
WINS:	0 . 0 . 0 . 0

Only set this one

Please note that your Pi-hole should be the *only* DNS server set here as Pi-hole already delivers the other upstream servers. If you set another server in your router, it's possible your ad blocking will be negatively affected.

Manual Method

You can manually configure each device to use Pi-hole as their DNS server. You just need the IP address of your Pi-hole and then follow the instructions below for your operating system.

macOS

1. Click Apple > System Preferences > Network
2. Highlight the connection for which you want to configure DNS
3. Click Advanced
4. Select the DNS tab
5. Click + to replace any listed addresses with, or add, your Pi's IP addresses at the top of the list:
6. Click Apply > OK
7. Repeat the procedure for additional network connections you want to change.

Linux

In most modern Linux distributions, DNS settings are configured through Network Manager.

1. Click System > Preferences > Network Connections
2. Select the connection for which you want to configure
3. Click Edit
4. Select the IPv4 Settings or IPv6 Settings tab
5. If the selected method is Automatic (DHCP), open the dropdown and select Automatic (DHCP) addresses only instead. If the method is set to something else, do not change it.
6. In the DNS servers field, enter your Pi's IP addresses
7. Click Apply to save the change
8. Repeat the procedure for additional network connections you want to change.
9. If your distribution doesn't use Network Manager, your DNS settings are specified in `/etc/resolv.conf`.

Windows

DNS settings are specified in the TCP/IP Properties window for the selected network connection.

1. Go to the Control Panel
2. Click Network and Internet > Network and Sharing Center > Change adapter settings
3. Select the connection for which you want to configure
4. Right-click Local Area Connection > Properties
5. Select the Networking tab
6. Select Internet Protocol Version 4 (TCP/IPv4) or Internet Protocol Version 6 (TCP/IPv6)
7. Click Properties

8. Click Advanced
9. Select the DNS tab
10. Click OK
11. Select Use the following DNS server addresses
12. Replace those addresses with the IP addresses of your Pi
13. Restart the connection you selected in step 3
14. Repeat the procedure for additional network connections you want to change.

Webmin

This is a web based server administration tool. It makes administering and adding programs easier. We can manage our drives, Samba, firewall and other programs. Unfortunately, you can not manage OpenVPN from webmin.

To install LAMP (Linux, Apache2, MySQL and PHP5) for possible web hosting, we will do that from the command line since MySQL hangs when trying to install it via Webmin (because of the root password entry prompts).

```
     sudo apt-get update
     sudo apt-get install apache2 (if PiHole is installed you will
     need to either change lighttpd server to another port from 80)
     sudo install mysql-server
     mysql_secure_installation
     sudo apt-get install php5 php-pear php5-mysql
     sudo systemctl restart apache2.service
```

References
https://melgrubb.com/2016/12/11/rphs-v2-introduction/
https://www.pestmeester.nl/index.html#1.0
https://www.noip.com/
https://www.raspberrypi.org/