Creating a Raspberry Pi Web server

Contents

Introduction

The Raspberry Pi is a good choice for a web server that will not receive a high amount of traffic, such as a testing server or small intranet. It does not generate much heat or make any noise and it only uses around 5 watts of power; costing approximately £3.50 a year when running 24/7.

Note: You can overclock this device, but I wouldn't recommend it. With a web server setup like this you're not going to see any real gain, but you'll be reducing the life of the CPU and creating instability.

Instructions assume that it is physically connected to a home router with internet connection.

Raspbian Linux

This install is based on the **Debian** Raspberry Pi (Raspbian) image from the **Raspberry Pi download page**. **eLinux.org** details how to install the image using **Win32DisIkmager.exe**.

Default login credentials

You will often be prompted to login after initial configuration. The default credentials for the Raspberry Pi are:

```
Username: pi

Password: raspberry
```

The command line and text editor

Configuration is done at the command line. It has many advantages, including the ability to remotely manage and install the server. It also means that the computer can spend more of its time serving up web pages and less processor-time drawing a GUI, which is after all the whole point of a webserver.

SUDO

Throughout the install you will see many commands prefixed with sudo. The sudo command allows the user to issue a command as the superuser. Without using the sudo command many of the commands would fail stating insufficient permissions. This is a security feature that protects the system from other users, but also limits the amount of damage that a user can do by mistake (although if prefixed with the sudo command it will not help against accidents).

Text editors

There are two command line text editors. The nano editor is the easiest for new users (so that's what I've referred to in this document), but I do recommend learning the **vi text editor** as it is a useful tool

that is installed on all Linux systems. If you are familiar with vi then replace nano with vi for the rest of this guide.

Performance tuning the OS

The Raspberry Pi has 256MB (512MB for later versions) of RAM. This RAM is shared between the graphics and main system memory. By default 64MB is allocated to graphics. This is overkill if you don't plan to run the graphical interface. This can be changed using the `memory_split` option in `raspi-config`.

Using `raspi-config`; you would also want to:

- `expand_rootfs` to make use of the extra space if your SD card is larger than 2GB
- update `change_locale` to only `en_GB.UTF-8 UTF-8` using spacebar to check or uncheck an option and set as `Default`
- `change_timezone` to `Europe` then `London`
- change `boot_behaviour` to disable booting to desktop

Select `Finish` to end configuration and reboot.

Networking setup

The next step is to give the Raspberry Pi a static IP address. By default the Raspberry Pi will request a dynamic IP address which is issued by your router as required. This however may change in future which would make it hard to connect to. Instead, we provide it with an address that doesn't change such as `192.168.1.#`, where `#` is an unused number. Note that this address can be used on the local network, but not on the Internet - later we'll look at providing access through your router/firewall from the Internet.

First find out what DHCP address has been allocated by using the `ifconfig` command - see the extract below:

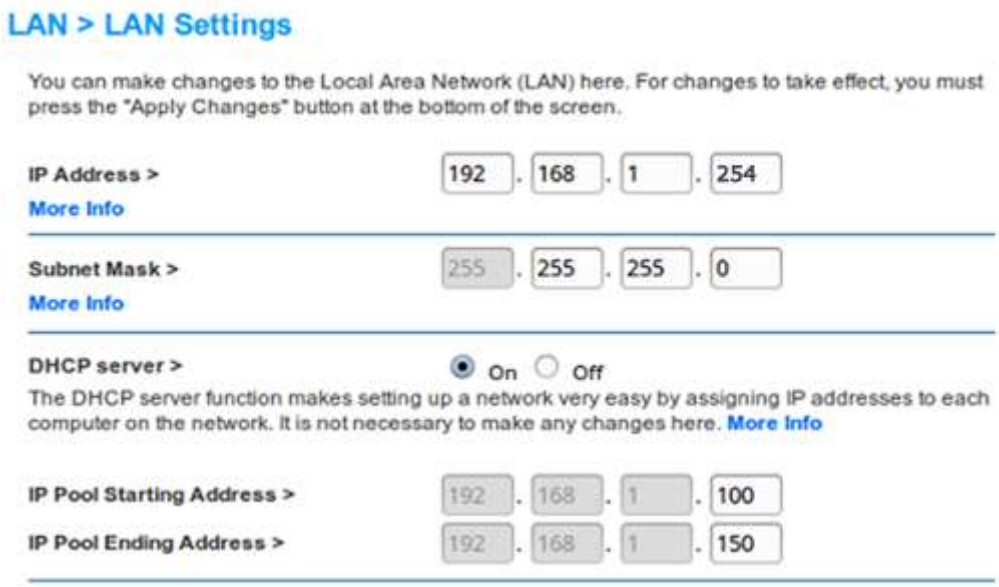
```
...

eth0      Link encap:Ethernet  HWaddr b8:27:eb:68:9a:b9

          inet addr:192.168.1.68  Bcast:192.168.1.255
Mask:255.255.255.0

...
```

This is saying that the Ethernet port 0 - has an IP address of 192.168.1.68. You will also need to find out what address your router is, using the `route` command - see the extract below:

```
Kernel IP routing table

Destination     Gateway         Genmask         Flags Metric Ref
Use Iface

default         192.168.1.254   0.0.0.0         UG    0      0
0 eth0

192.168.1.0     *               255.255.255.0   U     0      0
0 eth0
```

This shows that the router IP address or Gateway is 192.168.1.254 and all traffic is sent via that router.

At this point you will also need to check what address range is being issued by the router. This depends upon the individual router. In my case, my router can be reached by pointing a web browser to the IP address of the router 192.168.1.254. The LAN settings are shown below:



Any DHCP requests will be given entries between 192.168.1.100 and 192.168.1.150; you can change the range of the DHCP addresses if required. I have used 192.168.1.20 for this server.

To change to static IP address:

```
cd /etc/network
```

```
sudo nano interfaces
```

Replace the line `iface eth0 inet dhcp` with

```
iface eth0 inet static

address 192.168.1.20

netmask 255.255.255.0

gateway 192.168.1.254
```

You should also take a look at the file `/etc/resolv.conf` and check it has a name-server entry; probably pointing at your default gateway:

```
nameserver 192.168.1.254
```

Alternatively, you could point directly at your ISPs DNS servers. Whilst you can dynamically reload the network interface I suggest a reboot at this stage to make sure that the configuration is correct.

```
sudo reboot
```

After logging in, check using `ifconfig` to confirm that we have a static IP address:

```
eth0      Link encap:Ethernet  HWaddr b8:27:eb:68:9a:b9

          inet addr:192.168.1.20  Bcast:192.168.1.255
Mask:255.255.255.0
```

Enabling SSH and SFTP

SSH (Secure Shell) is a network protocol that allows you to login and control the computer through the command line remotely. As the name suggests, it is secure as it encrypts communication across the network so that others cannot see your password etc. The SSH server is installed on the default image and is started by default. You can enable/disable the SSH server using `sudo raspi-config`.

You can now connect to the Raspberry Pi remotely (on the same network) via SSH. If you have another Linux computer on the network then from a terminal run `ssh 192.168.1.20`

This will login with the same username. If you want to use a different username then prefix that before the IP address with a `@` sign. For example `ssh user1@192.168.1.20`

If you want to connect from a Windows machine then there are several options, but I suggest the open source software **Putty**.

If you're switching between multiple OS's and/or devices, it may sometimes be necessary to clear the keys associated with an SSH device with a command like this:

```
ssh-keygen -R 192.168.0.101
```

You can also securely transfer files to and from your Raspberry Pi using any file transfer program that supports Secure File Transfer Protocol (SFTP), such as **WinSCP**.

Headless

Now that we have a basic system configured, it's time to start removing unnecessary bloat. But first, let's see how much flash space we're using and how much is available with the `df -h` command. The first couple of lines of the output should look something like this on a 4GB SD card:

```
Filesystem      Size  Used Avail Use% Mounted on

rootfs          3.6G  1.4G  2.1G  41% /
```

This shows us that a large percentage of the flash space is already used, leaving very little free space. We're going to change that by removing the GUI, developer tools, and anything else we can find that isn't required on a headless device. Each of the following commands will free up a significant amount of space, while the `autoremove` command at the end will free up all configuration data associated with the removed programs.

```
sudo rm -rf /opt ~/python_games

sudo apt-get --yes purge `dpkg --get-selections | grep "^lx" | sed
s/install//`

sudo apt-get --yes purge `dpkg --get-selections | grep xserver | sed
s/install//`
```

```
sudo apt-get --yes purge xarchiver xauth xdg-utils xfonts-encodings
xfonts-utils xinit xkb-data xpdf xz-utils
```

```
sudo apt-get --yes purge `dpkg --get-selections | grep -v deinstall
| grep x11 | sed s/install//`

sudo apt-get --yes purge x11-xserver-utils libfontenc1 aspell-en
libaspell15 desktop-file-utils

sudo apt-get --yes purge fontconfig fontconfig-config fonts-droid
fonts-freefont-ttf gsfonts

sudo apt-get --yes purge gnome-accessibility-themes icelib menu-xdg
omxplayer penguinspuzzle Plymouth

sudo apt-get --yes purge `dpkg --get-selections | grep "\-dev" | sed
s/install//`

sudo apt-get --yes purge `dpkg --get-selections | grep -v deinstall
| grep python | sed s/install//`

sudo apt-get --yes purge `dpkg --get-selections | grep -v deinstall
| grep sound | sed s/install//`

sudo apt-get --yes purge `dpkg --get-selections | grep -v deinstall
| grep alsa | sed s/install//`

sudo apt-get --yes purge `dpkg --get-selections | grep -v deinstall
| grep qt | sed s/install//`

sudo apt-get --yes purge gcc-4.5-base:armhf gcc-4.6 gcc-4.6-
base:armhf

sudo apt-get --yes purge debian-reference libraspberrypi-doc
raspberrypi-artwork gnome-themes-standard-data

sudo apt-get --yes --purge autoremove
```

Security

It is important to make the Raspberry Pi a little more secure. You would want to get the firmware and software up-to-date. The image includes a default username and password, which once connected to the internet; would allow anyone to login and have free roam of the device.

Update firmware and software

The first thing you'll want to do is get Debian up to date:

```
sudo apt-get update

sudo apt-get upgrade
```

Next we're going to do a firmware upgrade, we need to install GIT: `sudo apt-get install git-core`

We then download the firmware update:

```
sudo wget https://raw.github.com/Hexxeh/rpi-update/master/rpi-update -O /usr/bin/rpi-update && sudo chmod +x /usr/bin/rpi-update
```

...and run it: `sudo rpi-update`

After the firmware update completes, you should see something like the following:

```
Raspberry Pi firmware updater by Hexxeh, enhanced by AndrewS

Performing self-update

ARM/GPU split is now defined in /boot/config.txt using the gpu_mem option!

We're running for the first time

Setting up firmware (this will take a few minutes)

Using HardFP libraries

If no errors appeared, your firmware was successfully setup

A reboot is needed to activate the new firmware
```

If you didn't see any errors, reboot the device: `sudo shutdown -r now`

Changing the password

To change the password for the pi user after logging in, enter `passwd` at the command line and follow the prompts.

Add a user

I have used tariq as the username, and this typically would be your name. The following commands will add a new user and change their password.

```
sudo useradd -m tariq

sudo passwd tariq
```

User groups and privileges

The new user will need to be added to certain groups to allow the same privileges that the pi user has. You can add the new user to the groups using the `usermod` command or you can edit the file directly. I have edited the file so that you can see the file content. It's also arguably a little quicker as you can make multiple changes. Please be aware that when editing files like these a mistake can result in not being able to login.

```
sudo nano /etc/group
```

Go through the file adding, `tariq` to the end of all of the groups that pi is in. For example: `adm:x:4:pi,tariq`

The most important is the admin entry as without that the user will not be able to run `sudo` and hence perform any system administration. Of course if you want to add a different user and don't want to give them admin access then you don't need to make any updates to the `/etc/group` file.

Type `exit` to logout and now login under the new username to check that it is working correctly.

Shell

By default the shell for the new user is the bourne shell. The bash shell is an improvement on that allowing the use of the arrow keys on the command line and auto-completion.

To set the default shell for your new account (when logged in under that account) use: `chsh -s /bin/bash`

You could now remove the pi username if it is no longer required: `sudo userdel pi`

Cleaning up

While `rpi-update` is very handy and does a lot of work for us, it downloads a lot of stuff from the Internet and doesn't clean up after itself at all. So here are some simple commands to do that clean up:

```
sudo rm -rf /opt /boot.bak /lib/modules.bak /lib/modules/3.2.27-
cutdown+/ /lib/modules/3.2.27+ /root/.rpi-firmware

sudo apt-get --yes purge binutils git-core

sudo apt-get --yes --purge autoremove
```

Even though we now have a clean, updated, headless system, there are still some things we can do to prepare the system for taking a snapshot/image. Specifically, we can install the `sfill` utility, remove the apt-get cache, clear (fill with 0′s) the empty file system space, clear the swap file, delete the log files, and shut down the device. Here are those commands:

```
sudo apt-get --yes install secure-delete

sudo apt-get --purge autoremove

sudo apt-get clean

sudo sfill -f -ll -z /

sudo swapoff -a

sudo dd if=/dev/zero of=/var/swap bs=1M count=100

sudo swapon -a

sudo rm `find /var/log -type f`

sudo shutdown -h now
```

Creating your own image

Now is a good time to take an image of your setup.

If you ever need to rebuild your server then this is a good checkpoint having completed all necessary configuration, optimisation, updates and clean up. Remember to keep firmware and software up to date.

I used **Win32DiskImager** as used earlier to write the image to the SD card. **TNET Raspberry Pi** has a very good tutorial on how to do this.

Use the command `free -m` to display memory usage. Memory total should be 512 minus allocated graphics memory.

Uninstalling packages

At some point, you may want to remove a package that you have installed.

To remove a package, for example Apache2, along with all configuration files:

```
sudo apt-get --purge autoremove apache2
```

Change your hostname

We are going to change the hostname from the default 'raspberrypi'. It's not essential to change the hostname, but it is something I have done, in case I ever have more than one Raspberry Pi. Your hostname should be something unique. Some people name their servers after planets, philosophers, or animals. Note that the system's hostname has no relationship to websites or email services hosted on it, aside from providing a name for the system itself. Your hostname should not be 'www' or anything too generic.

First run the hostname command and pass it the new host name: `sudo hostname tariqkhan.rpi`

Second, you have to modify the `/etc/hostname` file and replace the name that's already there with the new name:

```
sudo nano /etc/hostname
```

Replace `raspberrypi` with the domain from which your Raspberry Pi will run from, save then exit.

And third, you have to modify the hosts file: `sudo nano /etc/hosts`

Add the lines (the first line may already be present but commented out with a # symbol):

```
127.0.0.1 localhost

192.168.1.20 localhost
```

If you wish, also add the following line to the hosts file: `192.168.1.20 tariqkhan.rpi`

In the hosts file, `127.0.0.1` is the standard IP address and `localhost` the standard TC/IP hostname for local reference to the current physical machine.

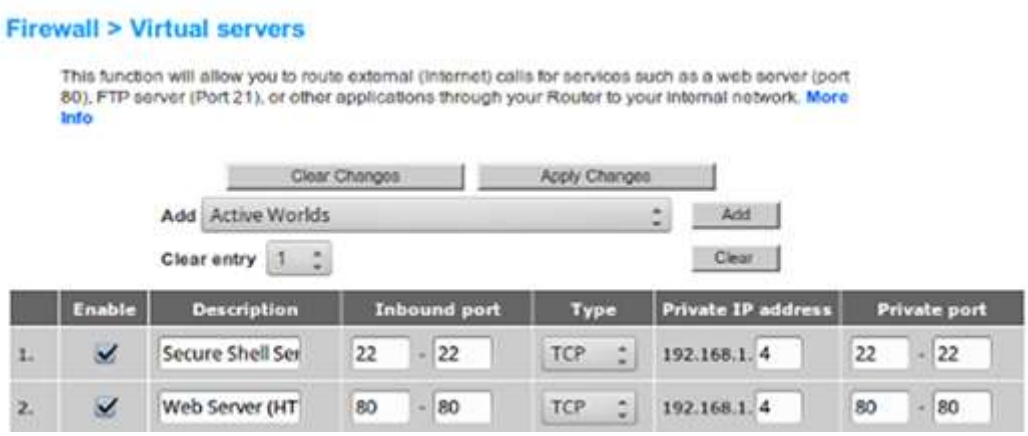
Now restart your Raspberry Pi for changes to take effect: `sudo reboot`

Find out the FQDN (Fully Qualified Domain Name) to ensure your RPi has saved the new hostname:

```
hostname
```

Making the server available on the Internet

Next we are going to configure the router to allow web traffic through its firewall to the Raspberry Pi.



As a home user, the IP address used on your local network is a private address range that will not work over the Internet. Instead your ISP will provide a single dynamic IP address which is used by the router. To allow traffic to flow from the Internet to your Raspberry Pi; your RPi IP address needs to be made to look as though it is from the router. This is a process called Network Address Translation (NAT) or port forwarding.

The ports that need to be allowed through are port 80 (http) and if you would like to be able to login to the computer from the Internet then port 22 (SSH). To do this you will need to consult the instructions on your router.

Domain Name Server

The final stage is to have a Domain Name Server (DNS) entry point at your router's IP address. Perfect if you have a static IP, not too bad if you have sticky IP, but not very good if you have a dynamic IP address. A solution would be to register for a dynamic DNS service, such as **http://www.noip.com/**.

You can use an online Dig tool **http://www.geektools.com/digtool.php** to check DNS record settings.

Domain name

If you have your own domain name like I do, you will find that as you no longer require server space, your domain name registrar will not provide you with as many features as you may have had prior to no longer having any server space.

Things like creating sub domains will become difficult and almost definitely limited.

I have repointed the Name Servers for all my domains to a free service called **EntryDNS**. EntryDNS provide you with all the services you could ever need to manage your domain names and for FREE. I highly recommend you sign yourself up for an account.

Summary

You now have a Raspberry Pi web server that is accessible from the Internet using a domain name. Your server is free of bloat and ready for installing software required to serve web pages.

I am presuming this is something you would like to do so here is another article: **Creating a Raspberry Pi LAMP server**.