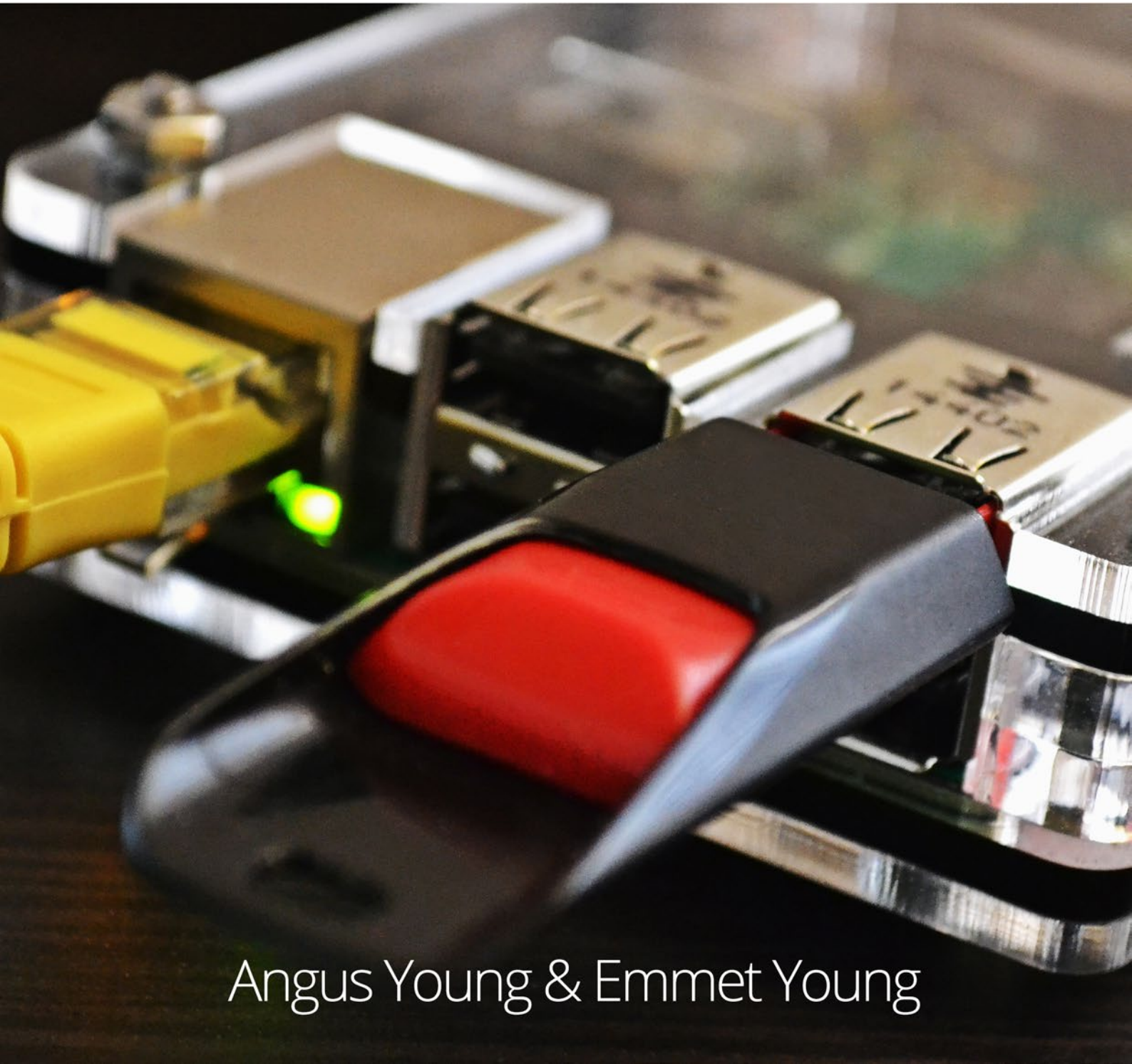
# Pi My Life Up's **Projects** For the Raspberry Pi



Angus Young & Emmet Young

PiMyLifeUp

Copyright © 2018 Pi My Life Up, All Rights Reserved

No parts of this book may be reproduced in any form or by any electronic means, including information storage and retrieval systems without permission from the copyright holder.

For permission contact Pi My Life Up at:

support@pimylifeup.com

The tutorials in this book are for general information purpose only.

While we strive to keep our information up to date, we do not make any warranties about the completeness, reliability, and accuracy of this information.

Any action you take upon the information on within this book is strictly at your own risk.

We will not be liable for any losses and damages in connection with the use of the information within this book

Raspberry Pi is a trademark of the Raspberry Pi Foundation, http://raspberrypi.org

All trademarks are the propety of their resperctive owners.

# Contents


## Gaming Projects 6

Retro Game Emulator 8
Steam Box 16
DOSBox 22
Setting up Parsec 28
Setting up Xbox Controllers 36
Setting up PlayStation Controllers 42
Setting up a Wiimote on the Raspberry Pi 54

## Networking Projects 62

Spoofing MAC Address 64
Setting up and using a VPN 70
VPN Server 76
Wireless Access Point 90
VPN Access Point 96
Wi-Fi Bridge 104
Wi-Fi Extender 110
TOR Access Point 116
Captive Portal 120
Network-Wide Adblocker 124
Build your own NAS 136

## Multimedia Projects 142

Kodi Media Center 144
Headless Music Player 152
Plex Server 158
AirPlay Receiver 164
Pirate Radio 168
Volumio 172
Viewing RTMP Streams 180
Kiosk Mode With Chromium 184
Raspberry Pi Chromecast 192

## Server Projects 196

Wireless Print Server 198
Setting up a Mail Server 208
Basic GIT Server 224
Installing Gitea 228
GITLab Server 236
Mumble Server 244
Setting up a Minecraft Server 248
Minecraft Pocket Edition Server 256
Terraria Server 260

Installing Apache 264
Setting up NGINX 268
Setting up Caddy 274
Setting up Lighttpd 280
Setting up MYSQL 284
Installing PHPMyAdmin 288
Setting up Wordpress 292
Let's Encrypt 296
Installing OwnCloud 302
Installing NextCloud 312
IRC Server 322
Webmin 328

## Video & Photography Projects 332

Setting up a Webcam Server 334
Networked Security Camera 340
Digital Photo Frame 350
Time Lapse 354

## Data Processing Projects 360

Build your own Amazon Echo 362
Build your own Google Assistant 376
Snowboy Hotword Detection 392
FlightAware ADSB Receiver 402
Flightradar24 ADSB Receiver 408
BOINC Client 416
Programming your own Twitch Mod Bot 422
Programming your own Twitter Bot 430
Telegram 438
Controlling your Raspberry Pi with Telegram 444
WhatsApp 450
Internet Speed Monitor 456
Installing OctoPrint 464

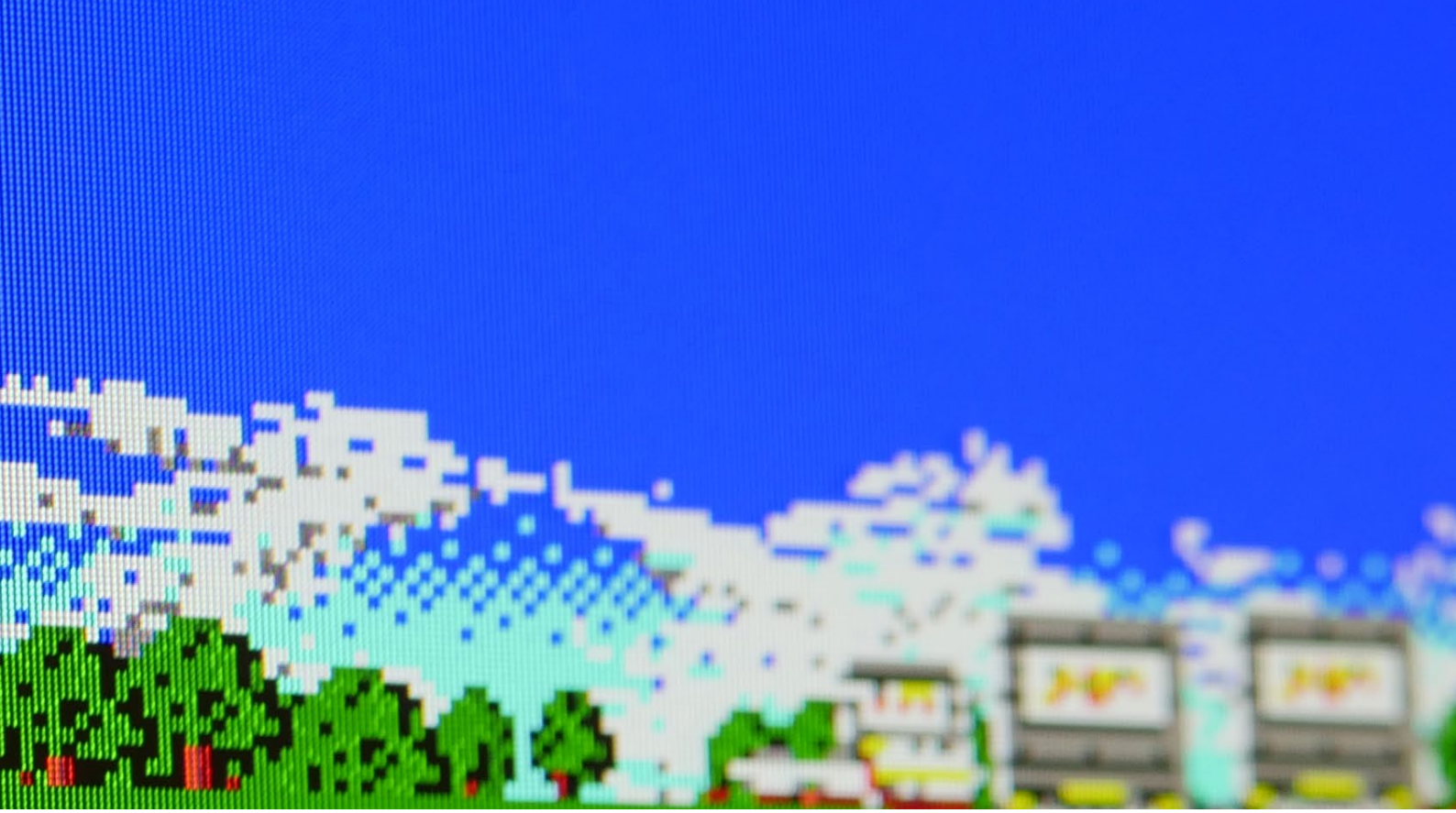
## File Management Projects 476

Torrent Box 478
Setup your own eBook Library 482

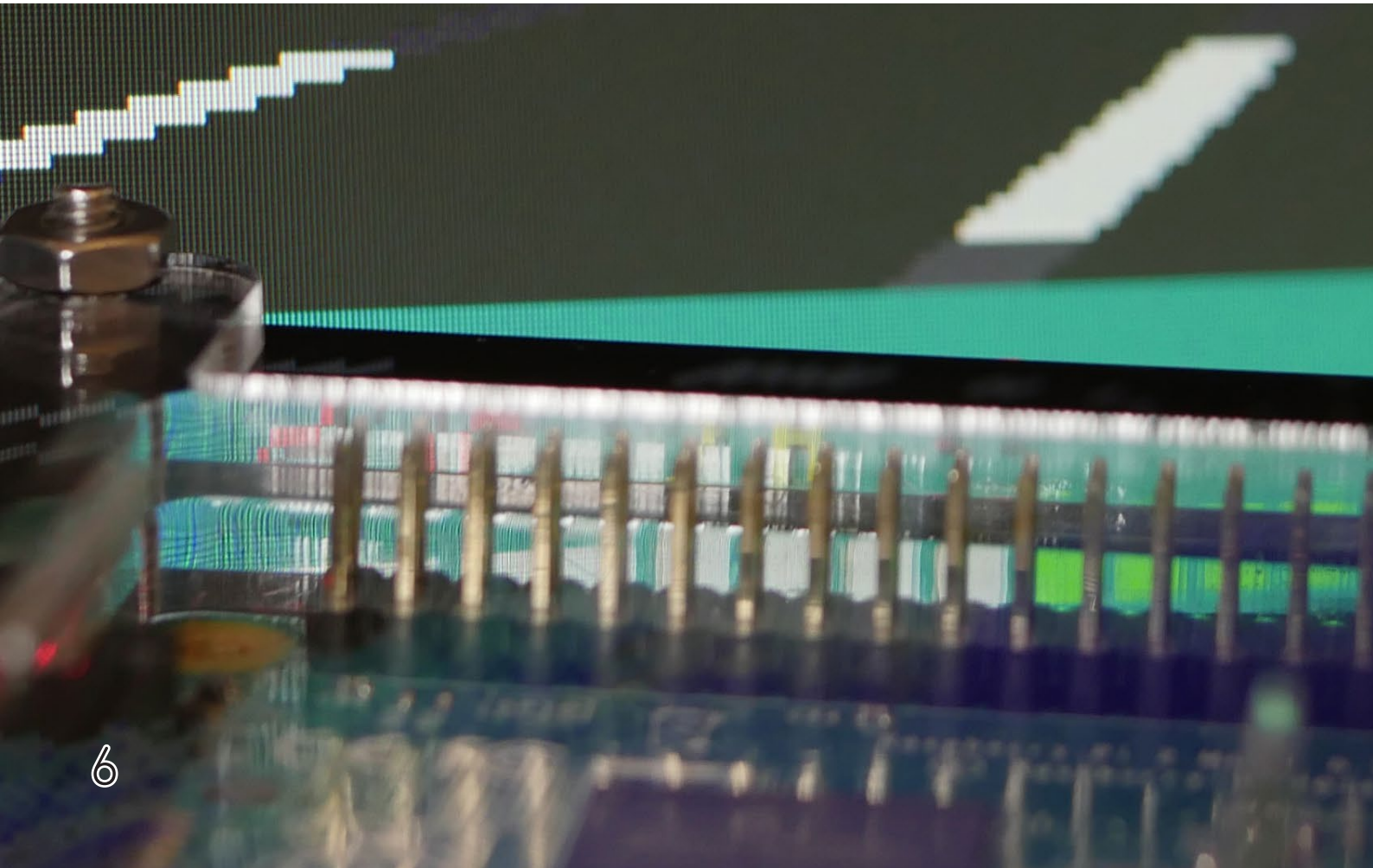
## References 490

Linux Cheat Sheet 491
Raspberry Pi GPIO Pins 492
Resistor Colour Guide 493
Voltage Divider Equation 494

PiMyLifeUp.com



# Gaming
## Projects



# Retro Game Emulator

## Project Description

A Raspberry Pi Retro Game Emulator can provide you with hundreds of hours of fun and be remembering those good times playing those great retro games. Thanks' to the hard work of a group of programmers, the Raspberry Pi is now capable of running a large variety of retro games.

There are two ways you can go about doing setting up RetroPie. The first is to download and install the RetroPie SD Card image. The other is to install RetroPie on an existing installation of Raspbian.

We will run through both methods of installing RetroPie, though as a rule of thumb we recommend people to install RetroPie from the image, so you are running a clean and correctly configured version of Raspbian.

## Equipment

**Required**

- Raspberry Pi
- SD Card (8 GB+ Recommended)
- HDMI Cable
- USB Keyboard
- USB Mouse

**Optional**

- Retro USB Controller / Joystick
- Network Connection

# Step 1 - Formatting your SD Card

First off you should do when installing a new image to an SD card is format it correctly.

The following steps will show you the best way to format your SD Card for the Raspberry Pi:

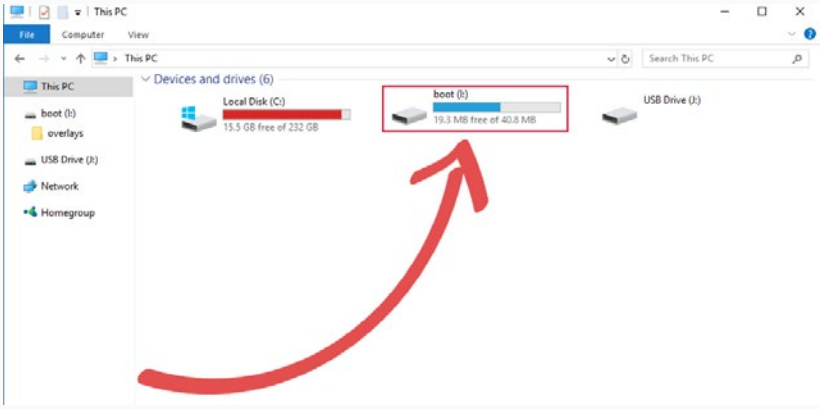
**1.** To begin you need a tool to format your SD Card. You can obtain one of the most robust ones from here: **https://www.sdcard.org/downloads/formatter_4/**

Once you are on the website will find a download link for the SD Formatter.

**2.** Download and install the SD Card Formatter from the SD Card association.

**3.** With the SD Formatter installed to your computer, insert your SD card and check the drive letter that is allocated to it, e.g., **G:/**.

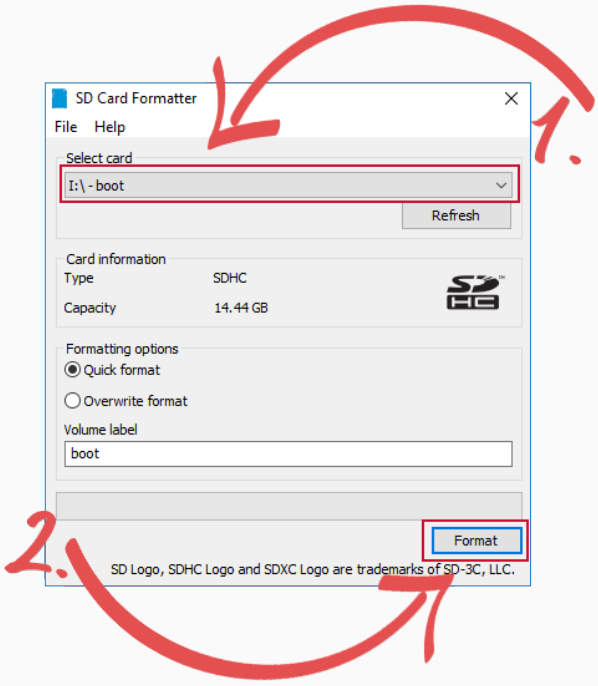
Make sure that you have the correct drive letter otherwise you could accidentally format the wrong device.




**4.** With your SD Card now plugged into your computer, launch the SD Formatter software that we installed earlier.

First set the drive letter to the correct one **(e.g., G:/)**.

Once you have everything set correctly, press the "**FORMAT**" button.



# Step 2 - Installing RetroPie from Scratch

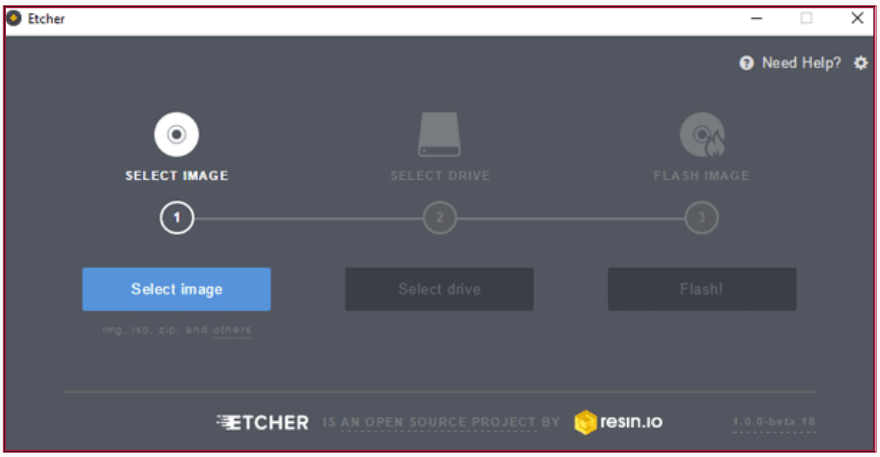
**1.** First head over to the RetroPie website at **https://retropie.org.uk/download/** and download the latest zip file.

Once downloaded, unzip the file so, you are left with a **.img** file.

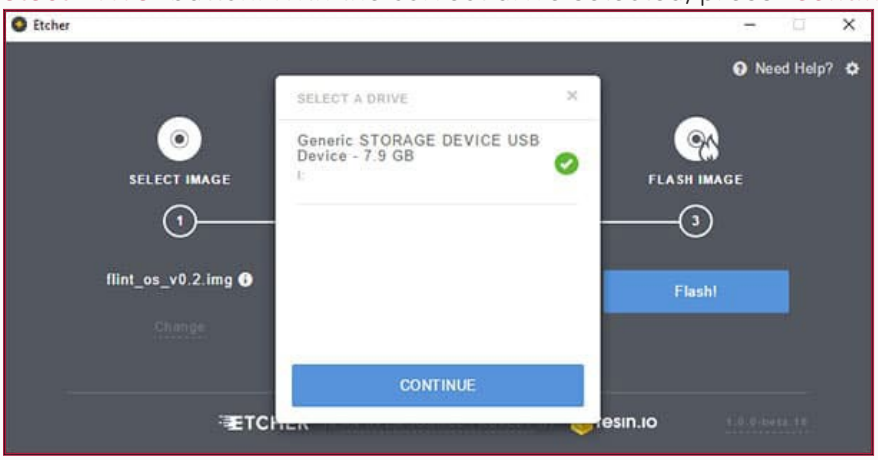
**2.** Next, you will need some tools to be able to write the image to the SD card.

For this we will use a tool called Etcher, you can download at https://etcher.io/.

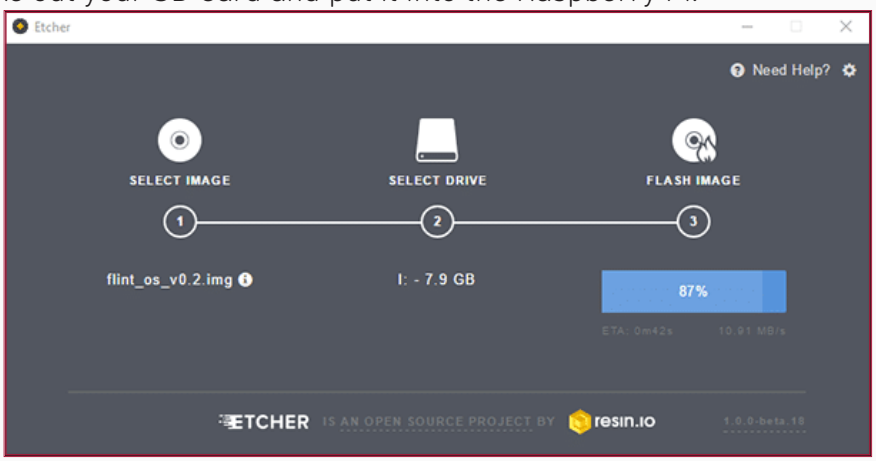
**3.** Open Etcher and press the "**Select Image**" button. Browse to the image you downloaded in **Step 1** and select it.




**4.** Next click the "**Select Drive**" button. With the correct drive selected, press "**Continue**".




**5.** Finally, click the "**Flash!**" button to begin the flashing process. This can take some time. Once completed, take out your SD Card and put it into the Raspberry Pi.



# Installing Raspberry RetroPie from scratch

**1.** This tutorial is for setting up RetroPie on a pre-existing operating system, such as a pre-existing installation of RetroPie.

To get started with this, we need to ensure there's a minimum of 2GB free disk space on our primary partition. We can do this by using the following command, and paying attention to **/dev/root**.

```
df-h
```

**2.** Before we get too far ahead of ourselves, we should also make sure we update the package list and run an upgrade on all pre-existing packages.

We can ensure everything is up date to by issuing the following command in the terminal:

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

**3.** After that we can install the two packages we will need to run the RetroPie setup script, use the following command to install both git and dialog software.

```
sudo apt-get install -y git dialog
```

**4.** Now we can retrieve the latest version of the RetroPie setup script easily, enter the following commands to get it to download the latest version:

```
cd
git clone --depth=1 https://github.com/RetroPie/RetroPie-Setup.git
```

**5.** We can now cd to the Retropie setup folder and make the script executable, so we can run it, once we have done that we just run the script.

```
cd RetroPie-Setup
chmod +x retropie_setup.sh
sudo ./retropie_setup.sh
```

**6.** This should now display the RetroPie setup script, this will have a variety of options, but the one we want to pay attention to is "**Basic Install**."

Select this option by using the arrow keys to navigate to it and using the enter button to select it.

**7.** Next, you will be presented with a screen asking you to confirm whether you want to install the "**Core**" and "**Main**" branches of RetroPie, select "**Yes**" to this option to begin the installation.

**8.** When completed  it will take you back to the menu, this time select "**Configuration/tools**".

**9.** Within here, go to "**autostart**" and select this menu.

**10.** Here select, "**Start Emulation Station at boot**," this will make the frontend boot when the Raspberry Pi powers on.

**11.** It is now safe to quit out of the configuration tool. You can do this quickly by pressing **Ctrl + C** or press escape and select the exit option.

**12.** Once this has completed, you will now be able to copy over roms for RetroPie.

To launch the frontend of RetroPie we need to type the following command into terminal, this command will launch the EmulationStation frontend.

```
emulationstation
```

# Booting up and configuring RetroPie

**1.** If you are starting with a fresh SD Card with a clean install of the RetroPie image, then insert the SD Card into the Raspberry Pi.

Also, connect a mouse, keyboard, network cable and of course the power supply.

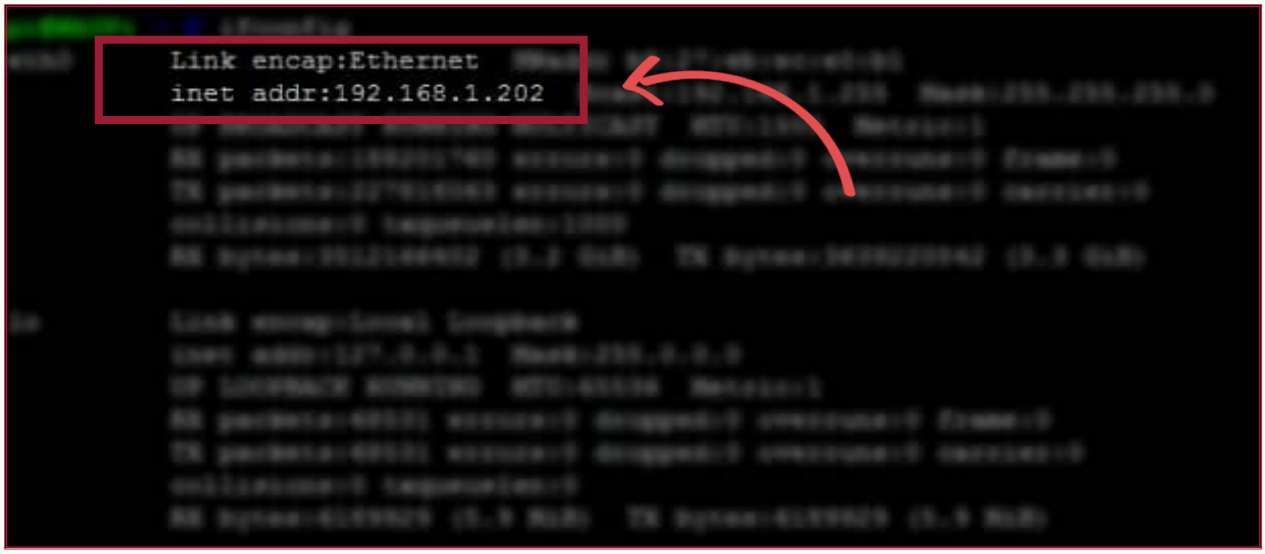
If you have just installed RetroPie onto a pre-existing Raspbian installation, then we recommend re-booting your Raspberry Pi with the following command:

```
sudo reboot
```

**2.** The Raspberry Pi will now boot up and launch into the RetroPie frontend.

**3.** Within the RetroPie frontend, press **F4**, this will take you into the command line.

**4.** Type **ifconfig** and write down the numbers and dots next to inet addr. (You may need it later)




**6.** Now if you haven't done so already, we will now enable SSH, expand the file system to cover the whole SD Card, and set an overclock to achieve the best possible performance out of the RetroPie emulators.

To get into the configuration tool, type in the following command:

```
sudo raspi-config
```

**7.** Firstly, select the "**Expand File System**," this option will allow you to expand the file system. If you are running a NOOBS installation, you will not have to worry about this.

**8.** Now, go to **Advanced Options** -> **SSH** and select **Yes** to enabling it, this will be important for managing your Raspberry Pi.

**9.** Next, we need to go to **Overlock**, here select either **Medium** or **Turbo**.
Read our "**Getting Started with the Raspberry Pi**" book to learn more about Overcloking.

Please note that the Raspberry Pi 3 is not currently supported by the **raspi-config** tool.

**10.** After you have applied these changes select **Finish** and then select **Yes** to reboot.

## Adding ROMS for the RetroPie emulators.

ROM is short for read-only memory and is the format in which you will find pretty much all the classic games in. ROMs can be easily found on the internet and since there many different sources it's best just to google the game you wish to download followed by ROM, e.g. ("Doom ROM").

There are three different primary methods for transferring ROMS over to your RetroPie, this being USB, SFTP, and SAMBA. We will explore all three different methods below.

## Utilizing a USB Stick

**1.** Before we get started, we need to make sure that the USB service is enabled on retropie. If it's not, then this tutorial will fail to work.

Let's get started by going to the terminal. You can get to this from RetroPie by pressing F4 and pressing any key.

**2.** Now that we are in the terminal, we need to get to the RetroPie setup script. We can launch it up by typing in the following:

```
cd RetroPie-Setup
sudo ./retropie_setup.sh
```

**3.** In here go to "**Configuration/tools**", within this menu, you will want to go to near the bottom of the list and look for the option labeled "**usbromservice**" select this option.

**4.** This option will now install the usbmount daemon. Before it finishes up, you will be asked whether you want to enable it, when prompted select "**Enable USB Rom Service**".

**5.** Once complete, quit out of the setup tool by hitting **ESC** and selecting **exit**, or using **Ctrl + C**.

**6.** Now reboot your Raspberry Pi with the following command:

```
sudo reboot
```

**7.** First, ensure that your USB drive is formatted to **FAT32**. You can check this on Windows by right-clicking on the drive, clicking "**Properties**" and look at the text next to "**File System:**".

**8.** On the USB, create a directory called **retropie**, the software will detect this folder when plugged in and set it up for copying over ROMS.

**9.** Now plug that USB into the Raspberry Pi, give this a few minutes as the RetroPie software will be setting it up in preparation for copying over ROMS.

If your USB has a flashing light, wait until it has stopped flashing before you pull it out. If it doesn't have a light, then wait a few minutes for the job to complete.

**10.** Now take out the USB from the Raspberry Pi and plug it back into the computer.

**11.** Add the ROMS to their respective folders on the USB. These folders will be in the **retropie/roms** folder. (Eg. **retropie/roms/snes**)

**12.** Once you have finished copying your ROMs to the USB, plug it back into the Raspberry Pi. The RetroPie software will immediately start copying these files off, don't take out your USB for some time.

**13.** Refresh emulationstation by pressing F4, or choosing quit from the start menu

# Copying over SFTP

For SFTP you will need to utilize a program such as WinSCP to connect to the Raspberry Pi if you are running Mac use something like Cyberduck

**1.** On Windows, go to https://winscp.net/eng/download.php and download the latest version of WinSCP. WinSCP is the tool that will interact with the Raspberry Pi and allow us to copy over files directly to the Raspberry Pi.

**2.** Once downloaded, launch up WinSCP. It will immediately ask you for new login details. You will need to enter the following details into the correct fields.

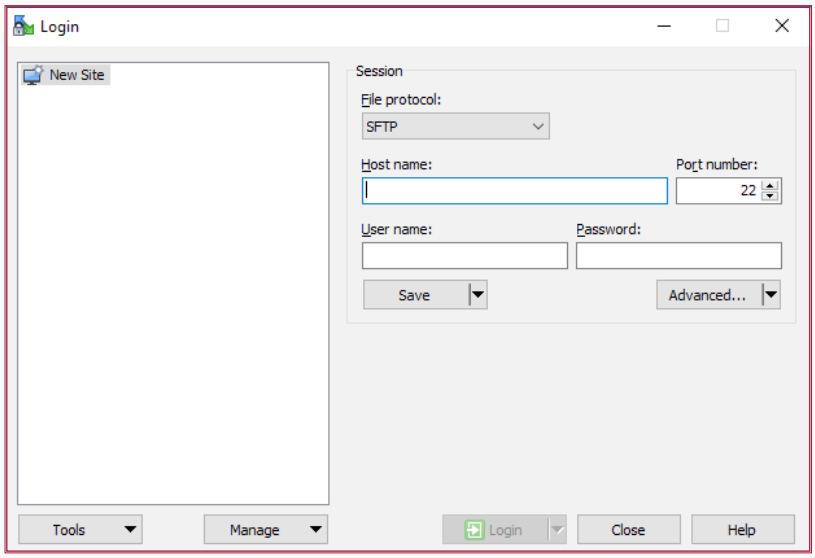
**File Protocol:** SFTP

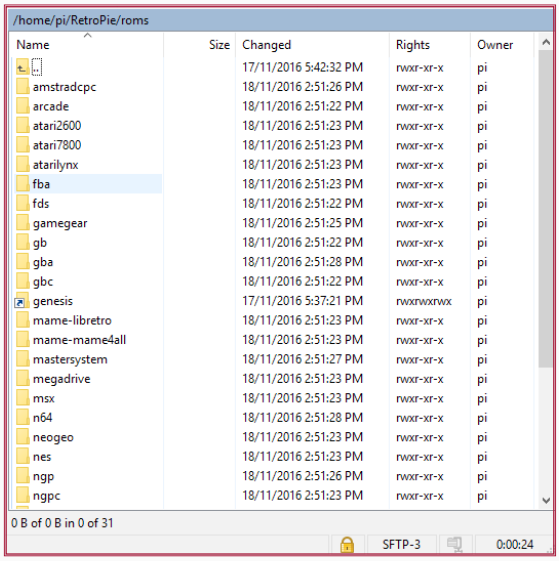
**IP address:** To find the IP address of your RetroPie, go into RetroPie options from the main menu, and select the last option "**Show IP address**".

**Port Number:** 22 (default)

**Username:** pi (default)

**Password:** raspberry (default)




**3.** Once successfully connected, you should pay attention to the right-hand side screen. Locate the folder named "**RetroPie**", double-click on that, once within that folder locate the folder named "**Roms**" and double-click again to enter that folder.

You should now be sitting in the directory where all your roms will be stored. The file directory displayed at the top should be something like **/home/pi/RetroPie/roms**

**4.** Once in the right folder, all you simply need to do is drop the files into the relevant folder for your console, such as for a SNES game you would drag and drop the file onto the folder named **snes**.

**Note:** If you would like to copy over more files instead of just roms you may first need to enable the root password by typing the following command into the terminal: **sudo passwd root**.

This will allow you to set up a password for the root user.

# Copying over Samba Network shares

A clean installation of RetroPie also has "Samba" pre-installed and enabled. Samba is the interface that allows Linux and Mac-based devices to connect with Microsoft's shared network devices interface.

This allows you to access the files on your Raspberry Pi over the network without needing to connect with something like WinSCP.

**1.** If you are running a clean install from the RetroPie image, then you can skip ahead in this tutorial. Otherwise, if you have installed RetroPie to a pre-existing Raspbian installation, you will need to go through a few extra steps to setup this.

To get started, enter the following commands to launch the RetroPie setup script.

```
cd RetroPie-Setup
sudo ./retropie_setup.sh
```

**2.** Once the setup script is loaded, go to "**Configuration/tools**" by using the **arrow keys** and pressing **enter**.

**3.** Within this menu, you need to search for the option labeled "**samba**", once you have selected this option, press enter to select it.

**4.** Selecting this option will install the packages required to setup samba, once that has completed, you will be met with another screen.

On this screen, select "**Install RetroPie Samba shares**". This option will set samba up to share the RetroPie related folders and allow network access to them.

**5.** Once done, you can now safely quit out of the RetroPie software. There are two ways you can do this, one is to hit **Ctrl + C**, and the other is to hit **ESC** and select "**exit**".

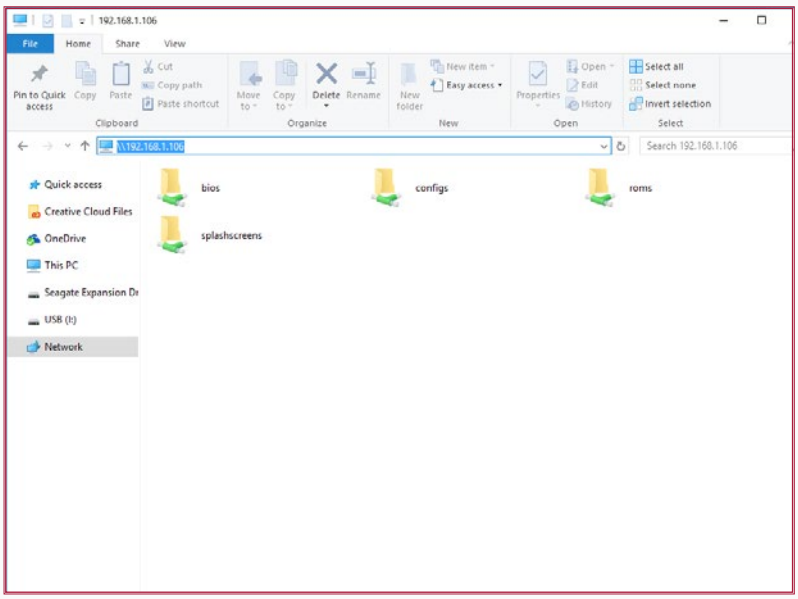
**6.** Now back on your computer. On windows open a file explorer window, and in the address bar type in the following:
**Note:** Make sure you swap out the IP address with your own.

```
\\192.168.1.106
```

**7.** There is a chance it may ask for your login details for your Raspberry Pi. Just enter your password and username. If you are still using the default user, that will be the following:
**Username:** pi
**Password:** raspberry

**8.** Once in, you can now copy any file you want over to your Raspberry Pi. For copying over roms, you will want to go into "**roms**" and copy the file into the folder of the console it belongs with.

For instance, a **SNES** game will go in the folder called **snes.**



# Steam Box

## Project Description

In this tutorial, we will be showing you how to set up a Raspberry Pi so you can stream PC games from your computer straight to the Raspberry Pi. It is an excellent way to get games to another screen or TV in your house without spending a significant amount of money on extra equipment.

This tutorial is perfect for anyone who loves video games but wants an easy way to be able to play games from your computer on your Raspberry Pi. With this you will be able to play most modern games, if your host computer can handle it, also you will need a decent network connection between the computer and the Raspberry Pi.

One of the most significant downsides to this tutorial is that you will **need an NVIDIA graphics card** in your computer. Unfortunately, there isn't an AMD equivalent at the moment.

We will be using a software package called **moonlight** on the Pi. This package will connect to the **Nvidia Geforce Experience** app running on a PC. So, if you have not currently installed that application onto your computer, you should do that now.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection (Ethernet Reccomended)
- Computer with a NVIDIA GPU
- HDMI Cable

**Optional**
- USB Keyboard
- USB Mouse
- Raspberry Pi Case
- USB Game Controller

## Setting up Raspberry Pi Steam Box Software

**1.** To begin you will need to either SSH into your Raspberry Pi or load up the terminal

**2.** Now where in the terminal, we will need to add the moonlight repository to our **sources.list** file for the package manager.

We can do this by first utilizing the follow command to edit the file:

```
sudo nano /etc/apt/sources.list
```

**3.** We will need to add the following line to the bottom of the file. This line will tell the package manager to search the moonlight repository for packages.

**NOTE:** If you are still using Raspbian **jessie** or Raspbian **wheezy** make sure you replace **stretch** with either name. Otherwise, it will fail to acquire the correct packages.

We reccommend however that you just update your copy of Raspbian to stretch as it features numerous improvements

```
deb http://archive.itimmer.nl/raspbian/moonlight stretch main
```

**4.** Save the file by pressing **Ctrl + X** and then pressing **Y** and hitting **Enter**.

**5.** Now to get our newly added package repository to function properly on our Raspbian operating system we will need to fetch its GPG key.

For those who do not know what a GPG key is, it is basically a special key that allows decrypting the files provided from the repository. It is used as a way of helping guarentee privacy.

We can grab this key and install it by running the following two commands on the Raspberry Pi's terminal.

```
wget http://archive.itimmer.nl/itimmer.gpg
sudo apt-key add itimmer.gpg
```

**5.** With that done, we need to get our package manager to update its package list.

We do that by entering the following command into the terminal on the Raspberry Pi:

```
sudo apt-get update
```

**6.** Now we can install the application that makes everything work with the following command:

```
sudo apt-get install moonlight-embedded
```

That's all you need to do to install moonlight onto your Raspberry Pi. Now that we have installed the Moonlight package we are now ready to connect the Raspberry Pi up to a PC.

Like installing Moonlight, this is a super easy and we will take you through all the required the steps over the next couple of pages.

# Connecting Raspberry Pi Moonlight with the PC

To begin, you will need to make sure you have a Nvidia GeForce graphics card. Moonlight relies on a fewfeatures of the NVIDIA GPU's drivers to function.

If you don't have Geforce experience application installed, then you can go ahead and download it from NVIDIA's website here: http://www.geforce.com/geforce-experience

**1.** To begin, you will need to know the local IP Address of the gaming computer that you want the Raspberry Pi to stream from.

There are two ways of doing this. One is checking your routers web interface. The other is to make use of the command prompt on Windows.

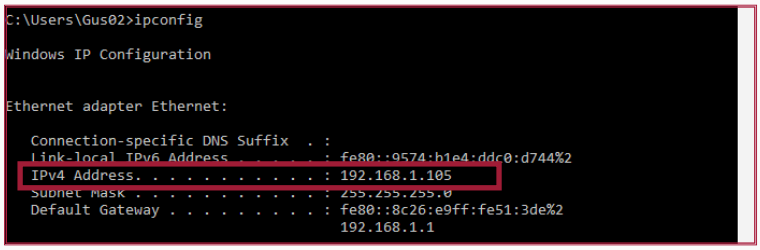
**2.** A quick and easy way of getting up the Command prompt is to press **Windows Key + R** and type in **cmd** and hit **enter**.

**3.** Within the command prompt, type in the following command:

```
ipconfig
```

Write down or copy the IP Address in the field next to IPv4 Address.

IP Address

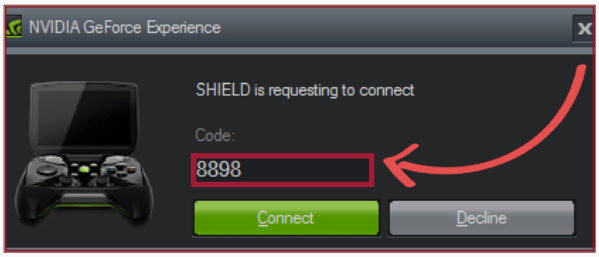


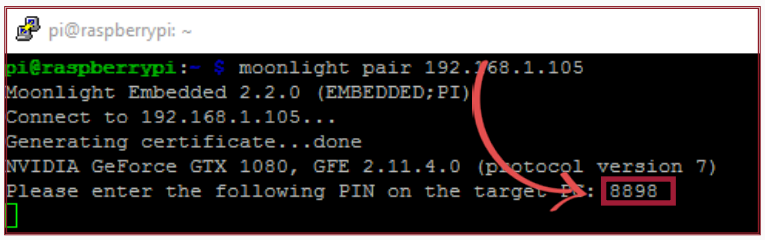

**4.** On the Raspberry Pi, we will need to enter the following command now to get Moonlight to pair with the gaming computer.

Make sure you replace **192.168.1.105** with your computers local IP address.

```
moonlight pair 192.168.1.105
```

**5.** This command will generate a 4-digit pin that you will need to enter on your computer in the pop up saying "**SHIELD is requesting to connect.**"




**6.** Once done It's all good to go now, to begin streaming enter the following command onto your Raspberry Pi.Again make sure you replace **192.168.1.105** with own IP address.

**7.** You can quit out of the moonlight streaming session by pressing **Ctrl+Alt+Shift+Q** at any time. Of course, this means that you are required to have a keyboard connected to your Raspberry Pi. You can also end the streaming session from the host computer itself.

**8.** There are additional configuration options that you might be interested in changing. We will run through them and the various side effects you can achieve by changing the values.

# Extra configuration options.

To help improve the performance of the Raspberry Pi's streaming you can tinker with the settings.

When yougo to launch moonlight streaming on the Pi, there are several additional options that you're able to add to the command line.

If you want to add options, they go after **moonlight stream**, and **before the IP Address**.
For example: **moonlight stream [OPTIONS] x.x.x.x**

Below is a list of almost all the options that you're able to add:

```
-720 Use 1280x720 resolution (default)

-1080 Use 1920x1080 resolution

-width <width> Horizontal resolution (default 1280)

-height <height> Vertical resolution (default 720)

-30fps Use 30fps

-60fps Use 60fps (default)

-bitrate <bitrate> Specify the bit-rate in Kbps

-packetsize <size> Specify the maximum packet size in bytes

-app <app> Name of app to stream

-nosops Don't allow GFE to modify game settings

-localaudio Play audio locally

-keydir <directory>\tLoad encryption keys from directory
```

As you can see above, there are quite a few options to pick from. You will probably find the FPS and resolution are the two you will find handy the most.

Reducing the FPS and the resolution will help lower the amount of data that needs to be sent over the network. Reducing these will help games streaming over slow local networks.

Here is an example of utilizing the settings, this command would reduce the FPS to only 30 and output in 1080p:

```
moonlight stream -30fps -1080 192.168.1.105
```

If you want to have moonlight launch a certain application such as steam by default, then you can add it as a parameter for -app such as we have shown below:

```
moonlight stream -app steam 192.168.1.105
```

These are just a couple of examples of what you can do with moonlights extra options. There are also other additional options that you're able to add that we have chosen not to include here.
You can find the full list at Moonlights wiki page at:
https://github.com/irtimmer/moonlight-embedded/wiki/Usage

## Troubleshooting & Limitations

Throughout testing this tutorial, we have found that some games just refuse to stream at all. We are not entirely sure why this is the case, but we think it could have something to do with running multiple monitors. It's also possible the GameStream driver just currently does not support some games.

The new Doom 2016 game, for example, runs perfectly fine on my computer but refuses to work when streamed to the Raspberry Pi.

Games can vary quite differently in the quality of the streaming, for example during testing, games such as Arma there was quite a significant difference between what was being displayed on the computer and what was being displayed to the Raspberry Pi.

However, a game like Ori and the blind forest there was little to no delay between the computer and the Raspberry Pi.

As mentioned on the previous page, it is possible to try and improve the streaming by fiddling with the various options they give you access to. Reducing the frames per second, for example, should ensure that the Raspberry Pi receives the data faster.

We found moonlight would sometimes crash but, this was only between changing games and didn't occur commonly enough for it to become any nuisance.

You can simply relaunch the stream with one simple command in the terminal so crashes aren't too bad to recover from. (If it's not mid game)

## Bad quality or slow stream

If you are using your Raspberry Pi over wireless, you may run into issues such as having a bad quality or lagging stream. There are two ways to fix this. One is to buy better wireless equipment.

A dual-band wireless router with 802.11ac capabilities and an equivalent adapter for the Raspberry Pi and computer should be enough to allow a fast and stable enough connection.

The alternative is to use an Ethernet cable. Connections over Ethernet tend to be much more stable and faster than your typical wireless setup.

There will also be far less chance of packet loss over an Ethernet cable.

The only real downside to using an Ethernet cable is having a cable going across your room.

# DOSBox

## Project Description

In this Raspberry Pi tutorial, I will be taking you through all the steps to set up the popular MS-DOS emulator called DOSBox. We will also explore how to play some MS-DOS games through DOSBox.

Setting up DOSBox is an easy and straightforward process that will quickly let you play the hundreds of great old DOS games that have been released as shareware, demos or purely as freeware.

The software runs decently on the Raspberry Pi despite being initially designed for x86 hardware. Be aware though that some demanding games will struggle on the Raspberry Pi.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection
- USB Keybaord
- USB Mouse

**Optional**
- Raspberry Pi Case
- External Hard drive or USB Drive
- Game Controller

# Installing Raspberry Pi DOS Box

**1.** To get started we first need to load into the terminal on the Raspberry Pi. You can do this on the Raspberry Pi itself or over SSH.

**2.** Now we need to make sure the Raspberry Pi is up to date by running the following two commands:

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

**3.** With that done, all we need to do install DOSBox with the following command:

```
sudo apt-get install dosbox
```

**4.** With dosbox now installed to our Raspberry Pi we need to now make a directory where our games will be stored with the following command:

~ refers to our home directory for the current user, this should be **/home/pi** with the default user.

```
mkdir ~/dos-games
```

**5.** If you want to edit the config file, you can do it with the following command. Keep in mind the version number may have changed since this tutorial was written.

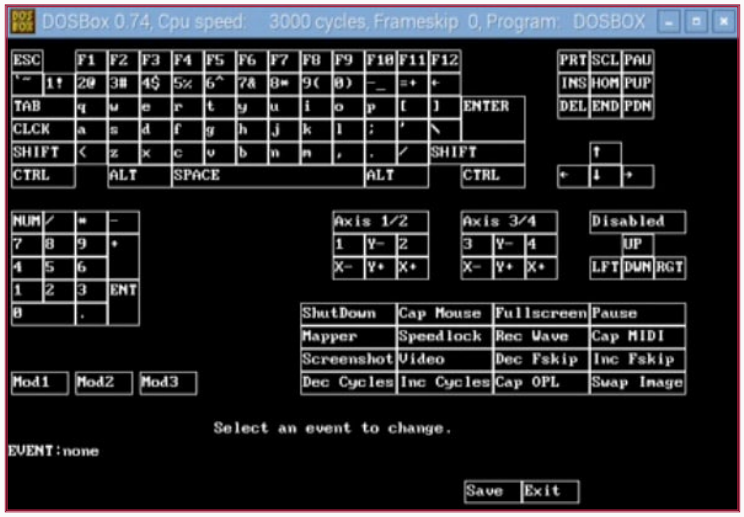
```
nano ~/dosbox/dosbox-0.74.conf
```

**6.** In here you're able to change quite a few different settings including the screen size, mouse sensitivity and so much more.

There are too many options to go over for this tutorial so if you want more information to be sure to check out the documentation page available at https://www.dosbox.com/wiki/Dosbox.conf

**7.** Next thing you may want to do is setup mapping for a controller. To do this, you will need to launch DOSBox by typing **dosbox** into the terminal.

Once started just press **CTRL+F1**. You can also use the following to bring up the key mapper as soon as you launch DOSBox.




**Please Note:** For some controllers, DOSbox will automatically detect and apply settings for the controller. I know this happened to my classic Xbox 360 controller. It's also important to note controller compatibility varies from game to game so it may work better in some than others.

## Installing Games for DOS Box

Installing games is a straight forward process it basically involves downloading and extracting them. There is two ways to do this but the easiest way is to just use the GUI (Graphical User Interface) to move the files to the correct place.

To begin go to the browser of your Pi and head over to https://abandonia.com as they probably have the largest collection of DOS games currently available for download.

Here you can navigate and find a game that is available to run on DOSBox. In my example I download a game called Outrun.

A classic racing game by Sega where your goal is simple. Reach the end of the track before time runs out. You can crash as many times as possible but it will slow you down severely. This is certainly worth checking out if you love old school racing games.

Now to move it into the dos-games folder you can either run a line in the terminal or use the GUI.

## Copying games over Terminal

Now in the terminal, you can just run a single line to move and unzip the zip file we downloaded.

In this example, I am using outrun so make sure you replace this with the same name as the game that you are using.

To move and unzip the file, just enter the following command into the terminal on the Raspberry Pi.

```
unzip ~/Downloads/OutRun.zip -d ~/dos-games/
```

## Copying games using the GUI

Using the GUI is straightforward, but I will just cover it anyway for anyone who is entirely new to the Pi.

1. In the default desktop, just go to file manager (the filing cabinet icon) in the top navigation bar.

2. Once this window opens, go to downloads in the left column and, you will see the zip file in the right column.

3. Now just right click on the zip file and go either cut or copy.

4. Now go to the dos-games folder and paste it in here.

5. Just right-click on it and extract it.

6. Now it's ready to be run and played.

# Playing Games through DOS Box

Now for the best part of the tutorial and that is launching Raspberry Pi DOSBox and playing the games you have just downloaded.

**1.** Once you have downloaded and unzipped the game running, it is easy. To do this, you will need to open DOSBox.

You can open DOSBox by entering the following into the terminal on your Raspberry Pi.

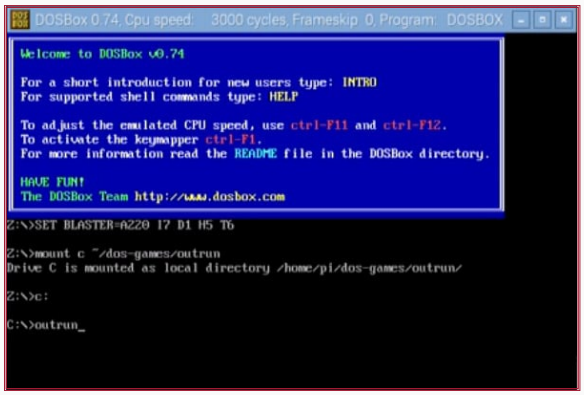
```
dosbox
```

**2.** Next, we need to mount the folder of where our game is installed. In this example, we will attach the directory that contains our game to the letter **c**.

```
mount c ~/dos-games/outrun
```

**3.** Now we need to change into the C mounted directory we just created. You can do this by just entering the letter followed by the semicolon into the DOSBox window.

```
C:
```




**4.** Now run the exe by just typing the name of the game into the terminal.

For example, Outrun you just simply need to enter **outrun** as the exe is called **outrun.exe**

```
outrun
```

**5.** Once that is entered the game will launch, and you should be able to play it like normal.

**6.** Now some games you will find will run too fast and others too slow.

These speed issues can be corrected by changing the cycle speed by pressing **Ctrl+F11** to decrease cycles and **CTRL+F12** to increase cycles.

There are also a few other key bindings that you may want to know. These can be found on the official DOSBox website here: http://www.dosbox.com/wiki/Mapper

## So What kind of games can you play?

As we mentioned earlier, there are hundreds of games you're able to play. If you remember the good old DOS operating system, then you will probably remember the classics that came from it.

Just to name some of the most popular games:

- Doom (1993)
- Oregon Trail (1990)
- Prince of Persia (1990)
- Quake (1996)
- Lemmings (1991)
- Tetris (1984)
- Anuse (1995)
- Liero (1998)
- 4D Sports Boxing (1991)
- Arkanoid (1986)
- Lode Runner (1983)
- Gauntlet (1985)
- Leisure Suit Larry (1987)
- Day of the Tentacle (1993)

It's important to know what while most DOS games have been declared as abandonware, freeware or shareware there are still some that you're meant to pay for.

## Troubleshooting

Like every tutorial, you are bound to run into some problems. Here are the most common issues we came across when we were testing this tutorial.

## Performance

While most games will run fine, you will come across a few that the Pi may appear to struggle a bit on. Your performance, of course, will vary depending on the version of the Raspberry Pi you're using. I found overclocking and tinkering the amount of memory reserved for the GPU helped with performance. You can find these settings within the raspi-config setting.

## Game Wont Load

Unfortunately, if a game won't load it's likely it there is nothing you can do about it. However, some games require that you run a setup file first. Check the game directory to see if there is a setup file and if there is simply run setup within the DOSBox terminal (It may also be called install).

## Screen Size

If you want a larger screen, then you will need to tinker with the DOSBox settings. Unfortunately, most old games were designed for low resolutions and will likely look terrible if forced to a full resolution.

# Setting up Parsec

## Project Description

Parsec is a game streaming software designed to offer a simple setup and low latency connection. Parsec allows you to be able to connect to your gaming pc or a cloud gaming pc from anywhere in the world.

This software allows you to gain access to your gaming PC from anywhere and control it using your input controls whether this is a keyboard, mouse or a joystick device. Meaning its a perfect solution for the Raspberry Pi as it allows you to run any game you want and not require to utilize an NVIDIA GPU like the Moonlight game stream client.

Even better is that Parsec also allows you to invite other friends to connect at the same time, allowing you to play local multiplayer games online.

Setting up parsec on your Raspberry Pi is a dead simple process and just takes a few minutes to get up and running correctly.

## Equipment

**Required**

- Raspberry Pi
- SD Card (8 GB+ Recommended)
- HDMI Cable
- Network Connection
- USB Mouse
- USB Keyboard

# Step 1 - Installing Parsec on your PC

Before we get started, we first must setup Parsec on your PC. Parsec will act as a server on which the Raspberry Pi will communicate with. This software woorks on Windows, Mac and Ubuntu.

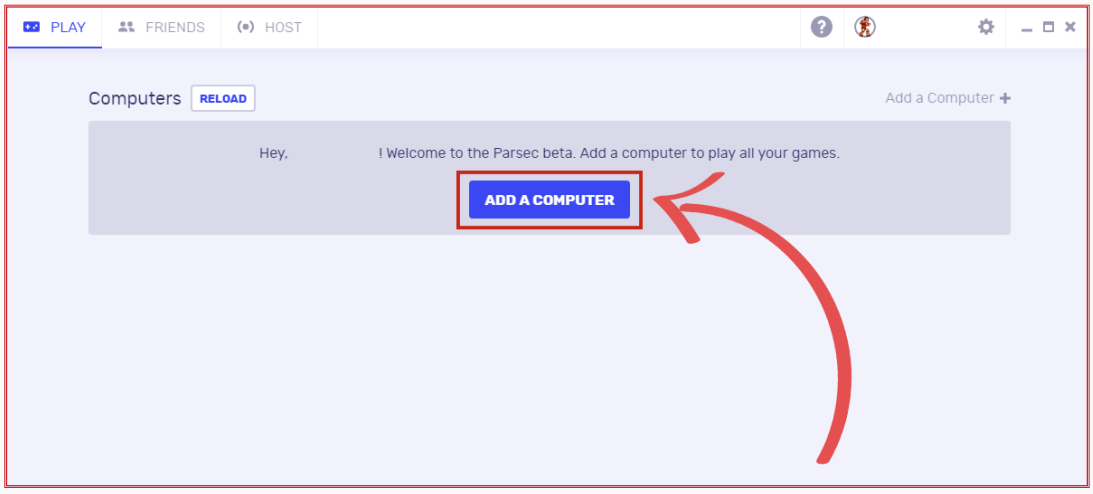
**1.** To get started go to Parsec's website and download the software. You can find Parsecs download page by going to the following URL: https://parsecgaming.com/downloads.

**2.** While the software downloads we recommend you use this time to sign up to Parsec, you will need an account to use the software. You can signup to Parsec by going to the following web address: https://parsecgaming.com/signup

**3.** Once you have downloaded Parsec and installed it, run it. You will initially be asked to log in to your Parsec account. Once logged in you will be greeted with a screen like below.

If you see a prompt asking if you would like to host from your current computer, make sure you click the "**Enable Hosting**" button, this will save you having to go into settings

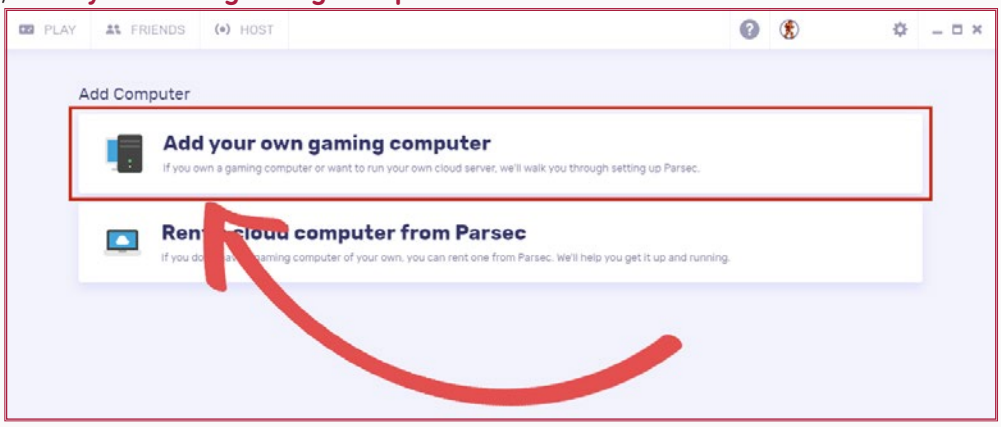
Afterward, click the "**Add a computer**" button like shown below.




**4.** You will now be introduced with two options, "**Add your own gaming computer**" or "**Rent a cloud computer from Parsec**".

In this tutorial, we will just be adding your own, but if you lack a computer powerful enough to play games or would prefer an always-on solution you can check out Parsecs renting solutions.

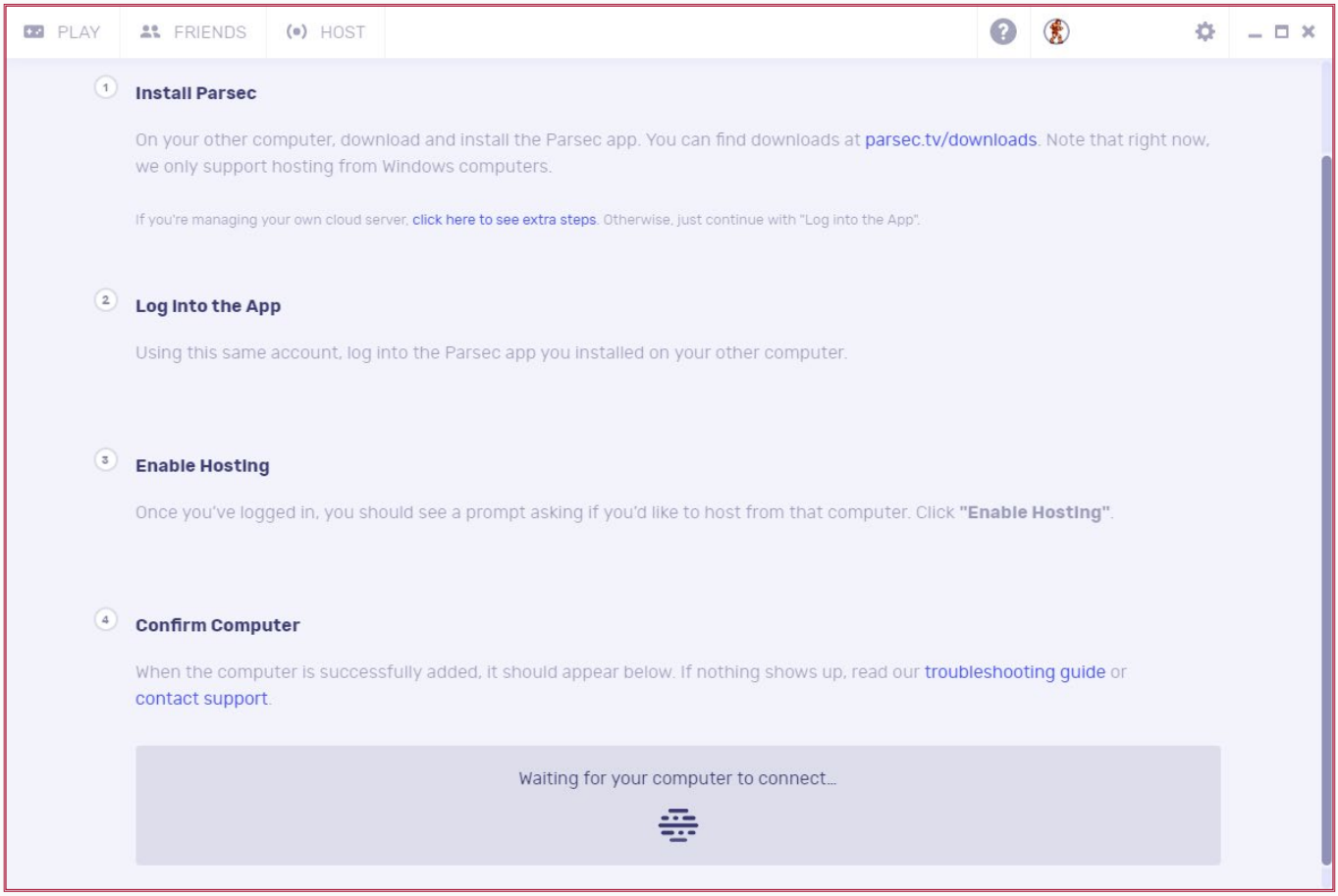
To continue click, "**Add your own gaming computer**".



# Step 1 - Installing Parsec on your PC

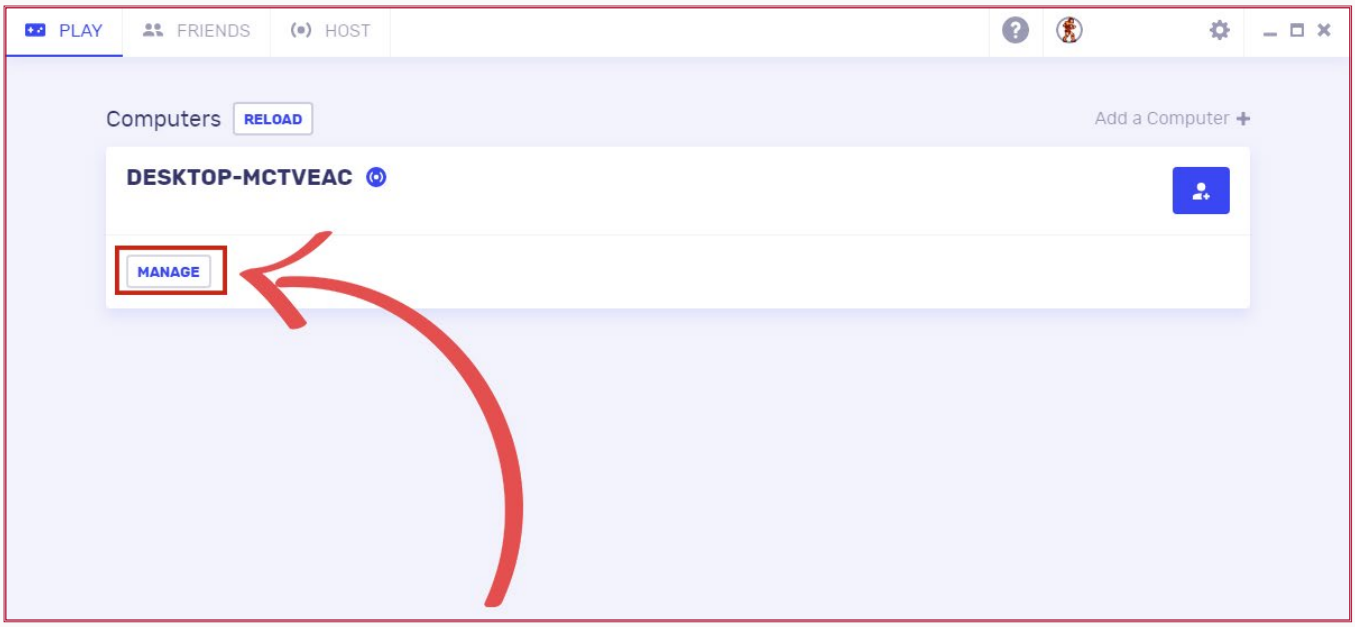
**5.** The next screen just runs you through all the steps required to adding your computer to Parsec as a possible host.

Just follow the prompts if your computer does not show up in the "**Confirm Computer**" section.




**6.** Now that you have finally added your computer to Parsec we can now grab the ID.

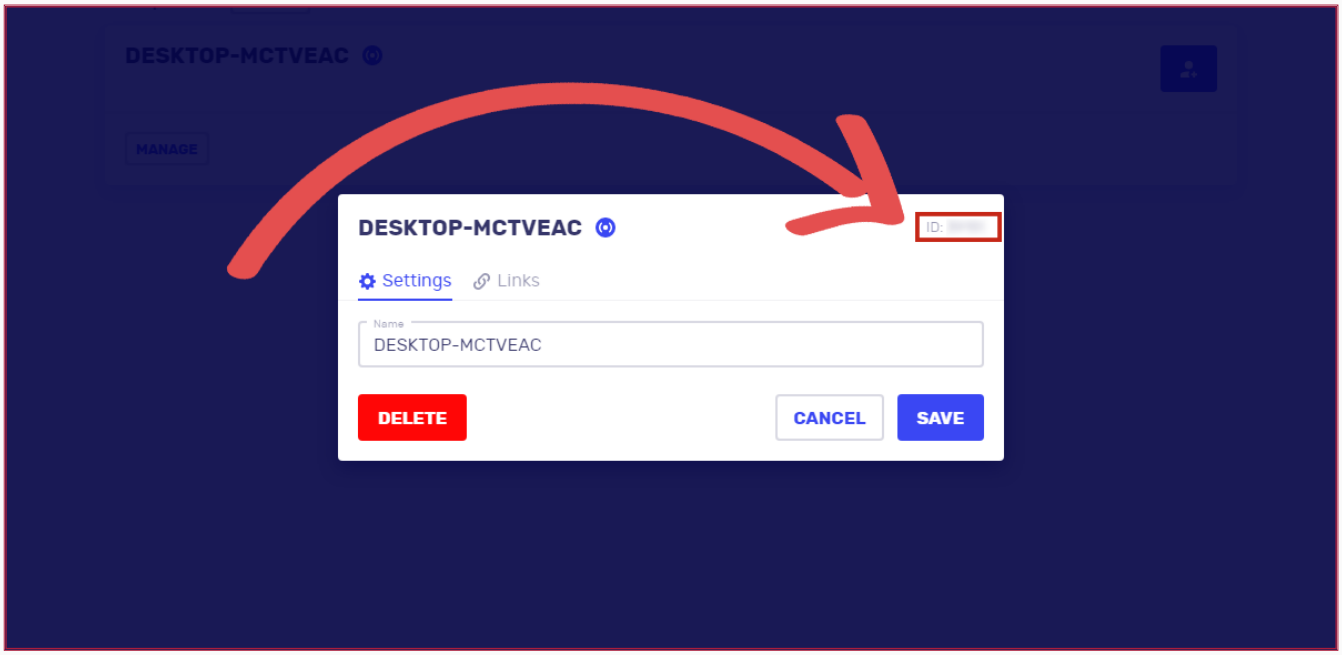
Go back to the "**Play**" tab and click the "**Manage**" button on your computer like that we have shown below.



## Step 1 - Installing Parsec on your PC

**7.** On this screen, you will want to copy down the number located next to ID as we have highlighted below.

We need this to connect your Raspberry Pi to your computer and allow it to stream between each other.




Now that we have setup Parsec on our computer and have grabbed the ID that Parsec has given our computer we can proceed onto the next steps of setting Parsec up on our Raspberry Pi.

## Step 2 - Preparing your Raspberry Pi for Parsec

**1.** Now that we have setup Parsec on our computer we need to begin setting it up on your Raspberry Pi.

To start the process of this SSH into your Raspberry Pi and run the following two commands.

These two commands will ensure that our Raspbian installation is entirely up to date before we continue.

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

**2.** With our Raspberry Pi now up to date there are a few things we have to do before we get installing.

These will help ensure that the Raspberry Pi can handle the video stream from Parsec by increasing the amount of memory that is dedicated to the GPU.

Run the following command on your Raspberry Pi to open up the Raspberry Pi Configuration tool.

```
sudo raspi-config
```

**3.** Now that we have loaded up the Raspberry Pi Configuration tool we will want to go down and select "**7. Advanced Options**".

**4.** Once we are in the "**Advanced Options**" menu, use the arrow keys to select "**A3 Memory Split**" and press Enter.

**5.** Finally, in here we want to change the memory split from the default to **128**.

This setting will ensure that Parsec has access to all the GPU memory that it requires.

**6.** Once you have set the new memory split to **128**, select "**<OK>**" and press **Enter**.

**7.** You can now safely quit out of the Raspberry Configuration Tool by pressing Enter.

**8.** Before our new memory split comes into effect, we are required to reboot our Raspberry Pi. We can do this by running the following command on it.

```
sudo reboot
```

**9.** Once the reboot has finished, we can finally move onto the process of actually installing Parsec.

Don't worry it only gets easier from here on and you will soon have Parsec up and running on your Raspberry Pi!

## Step 3 - Setting up Parsec on your Raspberry Pi

**1.** Finally, we can proceed onto the steps of installing Parsec on your Raspberry Pi. To do this, we will need to download a package from Parsecs cloud.

We can just do this by running the following commands on our Raspberry Pi.

```
cd ~
wget https://s3.amazonaws.com/parsec-build/package/parsec-rpi.deb
```

**2.** Now that we have downloaded the package to our Raspberry Pi we can go ahead and install it.

We can do this by using the package manager, run the following commands to tell the package manager to extract and install the **parsec-rpi.deb** file we downloaded to our Raspberry Pi.

```
cd ~
sudo dpkg -i parsec-rpi.deb
```

**3.** With Parsec finally installed we can go ahead and test to make sure everything is working correctly by running the following command.

This command will allow us to set up our user for Parsec on our Raspberry Pi. This command won't, however, initiate a connection with your server yet.

```
parsec
```

**4.** You will now be asked to enter the email address of the account you created during Step 1 and also the password you set.

Your information is needed since it will need to use your account to authenticate and allow the connection.

You can use a different account, but this means it will have to be manually approved by the host computer.

You can add that user as a friend and allow them to be accepted and given control of your computer automatically. However use this functionality wisely as that person will gain full control over your computer as it accepts all mouse and keyboard inputs.

**5.** If you have chosen to use the same account you used for your computer, then you will now be presented with a list of available hosts.

If your computer is the only host you have set up, then you should be able just type in **1** and press **Enter**.

If no option for selecting what host to connect to appears then you can utilize the following command to connect to your server.

Make sure you replace **YOUR_SERVER_ID** with the server ID you got earlier in Step 1 of this tutorial.

```
parsec server_id=YOUR_SERVER_ID
```

## Step 3 - Setting up Parsec on your Raspberry Pi

**6.** You will now be connected to your computer from your Raspberry Pi. You should now see your screen appear on whatever device your Raspberry Pi is connected to. You will notice that you have full control over everything and should also be receiving sound from the connection.

Parsec allows you to utilize the computer as if you were there yourself. Parsec also handles inputs incredibly well which means you can play all sorts of games over the connection with as little input lag as possible.

You will quickly find that Parsec is a perfect solution if you also want to play local co-op only games as it splits controllers and keyboards into separate input devices meaning each one can be a new player.

**7.** Once you have had a bit of a play around in Parsec, you can disconnect by clicking the little blue floating icon and clicking "**Disconnect**".

**8.** You have now successfully setup Parsec on your Raspberry Pi.

However, there are a few things that we can do to ensure that you have the best experience possible with it. Follow our next section to learn how to setup several items such as fixing up mouse input.

We also have tutorials that detail getting a Wiimote, Xbox Controller and a PlayStation controllers

## Improving the mouse polling rate on the Raspberry Pi

One thing you may have noticed is that the tracking speed of the mouse on the Raspberry Pi can feel very slow and sluggish.

The reason for this is that the default polling speed for mice is set by default to 62.5hz or 62.5 times per second. It is set lower as it makes it easier on the CPU, but this can lead to poor performance in games.

Most people would prefer a much higher polling speed such as about 125hz, or a call every 8ms.

Luckily we can change the rate at which the Raspberry Pi polls the mouse, to do this follow the next few steps.

**1.** We can achieve this by first editing the cmdline.txt file located within the boot folder. Editing this file allows us to perform certain things and set specific variables before the Raspberry Pi starts up.

Run the following command to edit the file on the Raspberry Pi.

```
sudo nano /boot/cmdline.txt
```

**2.** Within this file you should see something like below, the line is likely much longer then what we showed below.

```
dwc_otg.lpm_enable=0 console=serial0,115200 console=tty1 etc...
```

**3.** To this line add the following to the end of it, this is the line that will set the mouse polling rate.

This line will set the Raspberry Pi to poll the mouse every 8ms.

```
usbhid.mousepoll=8
```

This should make the line look something like the following:

```
dwc_otg.lpm_enable=0 console=serial0,115200 console=tty1 usbhid.mousepoll=8 etc...
```

**4.** Once you have added the additional line, you can now save the file by pressing **Ctrl + X** then pressing **Y** and then **Enter** to save the file.

**5.** For the changes to now take effect you are required to restart your Raspberry Pi. You can do this by running the following command on your Raspberry Pi.

```
sudo reboot
```

# Setting up Xbox Controllers

## Project Description

If you have ever tried to use an Xbox controller with the Raspberry Pi, you will find very quickly that they do not work correctly right out of the box. In fact to get them working you will be required to install a special driver.

With the newer Xbox One controllers that feature the Bluetooth functionality, you will also find that they will need extra work on top of the driver installation to get them to run. Namely, they are not properly supported by some of the Bluetooth functionality that is switched on by default.

This guide will show you how to get your Xbox Controllers up and running on the Raspberry Pi while also walking you through how to get the newer Bluetooth enabled controllers to pair successfully on the Pi.

## Equipment

**Required**

- Raspberry Pi (Raspberry Pi 3 Reccomended)
- SD Card (8 GB+ Recommended)
- Xbox Controller (Xbox 360, Xbox One, Xbox One Bluetooth)

**Optional**

- Bluetooth adapter (if not using a Pi 3)

## Setting up for Xbox controllers and Pairing Bluetooth Controllers

There are three different ways you can connect your Xbox One and Xbox 360 controllers to the Raspberry Pi. The two easiest ways of connecting your controllers are via USB cable or using the official wireless adapters.

### Xbox Controller USB Wireless Adapter

The Xbox USB Wireless adapter is one of the easiest ways to connect your controllers while retaining their wireless functionality.

This option is also required if you want to use the wireless Xbox 360 controllers or utilize the generation one Xbox One controller wirelessly.

Please note that Xbox 360 controllers and Xbox One controllers use different USB adapters.

### Bluetooth

Using Bluetooth is only a viable option for the newer Xbox One controllers that were introduced when the Xbox One S was released. Using Bluetooth requires a few different steps to get working correctly.

If you would like to know how to connect your Xbox One Bluetooth controller to your Raspberry Pi, then proceed down to the next section.

### USB Cable

A USB Cable is the easiest way to setup any Xbox One controllers. However, if you use a 360 controller you will only be able to use the wired ones this way since the port available on that controller is purely for charging.

Just plug a Micro USB cable into the Xbox One controller and the USB-A end into the Raspberry Pi and you should be ready to go.

## Setting up the Raspberry Pi for Xbox Controllers

**1.** Before we get started with anything in this tutorial, we should make sure our Raspberry Pi is entirely up to date by running the following two commands within the terminal.

These commands will make sure we have the latest available software.

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

**2.** While we don't necessarily need to install xboxdrv anymore since the Xpad driver is now included in Raspbian's kernel, we will install it for any programs that might be built to utilize xboxdrv.

It also allows us to do a few things within the user space by using xboxdrv. A userland driver means that it works just like an application and is not built into the core system.

To install the userland driver that allows us to interact with the Xbox One and Xbox 360 controllers type the following command into the terminal on your Raspberry Pi.

```
sudo apt-get install xboxdrv
```

## Pairing Bluetooth Xbox Controllers to the Raspberry Pi

**1.** Before you try to pair your Bluetooth Xbox Controller with the Raspberry Pi ensure you have installed the appropriate driver like mentioned in our previous section.

Once you have that installed the driver, we can then proceed with the process of getting the controller to pair.

Begin by running the following command on your Raspberry Pi.

This command disables the Enhanced Re-Transmission Mode (ERTM) of the Bluetooth module, with it, enabled the Xbox Controller won't pair correctly.

```
sudo bash -c echo 1 > /sys/module/bluetooth/parameters/disable_ertm
```

**2.** Now before this change takes effect, we are required to restart the Raspberry Pi.

Run the following command on your Raspberry Pi to restart it.

```
sudo reboot
```

**3.** With the Raspberry Pi now booted we can proceed with pairing our Bluetooth Xbox One Controller with the Raspberry Pi.

To be able to pair our Xbox One controller we will need to start up the Bluetooth tools by running the following command on our Raspberry Pi.

```
sudo bluetoothctl
```

**4.** Within the Bluetooth tool, we can now run a few different commands that will allow us to find and pair our Bluetooth Xbox One controller.

We begin by switching the agent on by typing in the following two lines.

```
agent on
default-agent
```

**5.** With that done we can now begin a scan for devices, simply type in the following command to start the scanning process.

```
scan on
```

Once the scanning process has begun, you should begin to see any Bluetooth device that is discoverable pop-up within the terminal.

## Pairing Bluetooth Xbox Controllers to the Raspberry Pi

**6.** When you press the sync button located on the front of your Xbox One Controller you should see a new entry appear on your Raspberry Pi's command line, such as below.

Take note of the MAC address of your controller. The MAC Address is the hexadecimal number that is separated by colons "**:**".

```
[NEW] Device B8:27:EB:A4:59:08 Wireless Controller
```

**7.** With the MAC Address of your Bluetooth Xbox One Controller, we can now make a connection with it by using the Raspberry Pi's Bluetooth stack.

Just type in the following command, making sure that you replace "**YOUR_MAC_ADDRESS**" with the one that appeared when you started the syncing process on your controller in **step 6**.

In our case, this was B8:27:EB:A4:59:08

```
connect YOUR_MAC_ADDRESS
```

**8.** Now once you have had a successful connection with the controller, you should see something like below appear in your command line on the Raspberry Pi.

```
Attempting to connect to B8:27:EB:A4:59:08
[CHG] Device B8:27:EB:A4:59:08 Modalias: usb:v054Cp0268d0100
[CHG] Device B8:27:EB:A4:59:08 UUIDs:
    00001124-0000-1000-8000-00805f9b34fb
    00001200-0000-1000-8000-00805f9b34fb
```

**9.** Once we have made a successful connection with the Xbox One controller, we need to add it to our trusted devices list. Doing this will allow the Xbox One controller to reconnect to the Bluetooth on our Raspberry Pi automatically.

To do this, we need to run the following command on our Raspberry Pi. Make sure that you replace "**YOUR_MAC_ADDRESS**" with the same MAC Address that you used to connect to the device in **step 8**.

```
trust YOUR_MAC_ADDRESS
```

## Setting up for Xbox controllers and Pairing Bluetooth Controllers

**10.** Now that you have paired your Bluetooth Xbox One Controller you can quit out of the bluetoothctl software by either typing in "**quit**" or pressing **Ctrl + D**.

**11.** Lets now install the joystick toolset. This toolset will allow us to check to see whether our Xbox One controller is working correctly.

Run the following command on your Raspberry Pi to install it.

```
sudo apt-get install joystick
```

**12.** With the joystick toolset now installed to our Raspberry Pi, we can proceed to test whether our controller is working correctly. Run the following command to test the joystick set to js0.

```
jstest /dev/input/js0
```

You should see a screen with a bunch of numbers, moving and pressing buttons should change the text on the screen. If anything changes then your controller should be fully set up and ready to go.

## Controlling your Raspberry Pi with an Xbox controller

**1.** If you want to control your Raspberry Pi by using an Xbox Controller, you can just use the following command in terminal.

This command allows the Xbox Controller to interact with the computer and will enable it to imitate the mouse.

```
sudo xboxdrv --detach-kernel-driver --silent --mouse
```

Once you have run that command, you should now be able to control your mouse on your Raspberry Pi with your Xbox controller.

The following buttons are bound to specific mouse actions.

- A: Left click
- B: Right click
- X: Middle mouse click
- Y: Enter
- Left Joystick: Mouse movement
- Right Joystick: Scroll wheel
- D-Pad: Arrow keys
- Start: Forward
- Back: Back
- LB: Page up
- RB: Page down

# Setting up PlayStation Controllers

## Project Description

If you have ever tried to utilize a PlayStation 3 or PlayStation 4 controller with the Raspberry Pi, you may have found it's not very simple to do. Our guide will walk you through the process of setting up and connecting both PlayStation 3 controllers and PlayStation 4 controllers.

For PS3 controllers we show you how to install the six-pair software and utilize it to set up your PS3 controller so it can connect to whatever Bluetooth device you are using on your Raspberry Pi.

For PS4 controllers we walk you through the process of pairing it with the Raspberry Pi's Bluetooth and also offer alternative solutions if your controller doesn't work with the standard Bluetooth stack.v

## Equipment

**Required**

- Raspberry Pi (Reccomended to use a Raspberry Pi 3)
- SD Card (8 GB+ Recommended)
- PlayStation Controller (PS3 Controller, PS4 Controller)

**Optional**

- Bluetooth adapter (required if not using Pi 3)

## Preparing your PS3 Controller to connect to the Raspberry Pi

Setting up a PlayStation 3 controller on the Raspberry Pi isn't an entirely straightforward process. Especially if you intend on using it wirelessly, we will walk you through all the steps to get it up and running in no time.

If you want to use your PS3 Controller wired, you should be able to use it just as a plug and play device. However, we recommend you go to our section on compiling and installing sixad. Sixad is a piece of software that ensures that all of the PS3 Controllers features are supported.

Before you begin this section of the tutorial make sure that you have a USB mini cable handy as you will require this to pair the controller with the Raspberry Pi. The reason for this is due to the way the PS3 Controller is set up to talk with Bluetooth controllers.

**1.** Before we get started with installing anything on our Raspberry Pi, we must first run the following two commands.

These commands will ensure that your Raspberry Pi is up to date.

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

**2.** To begin we first need to install a package, this is so we can compile the sixpair software.

This package being the libusb-dev, this is for programs to interact with USB devices.

```
sudo apt-get install libusb-dev
```

**3.** With our required package now installed we can go ahead and download the sixpair code and make a folder for it.

```
mkdir ~/sixpair
cd ~/sixpair
wget http://www.pabr.org/sixlinux/sixpair.c
```

**4.** Now with the sixpair code downloaded to our Raspberry Pi, we need to compile it, to do this we can just run the following command.

This command will tell the GCC compiler to compile the code.

```
gcc -o sixpair sixpair.c -lusb
```

**5.** With Sixpair now compiled on our Raspberry Pi, we need to plug our PS3 Controller into the Raspberry Pi using the USB mini cable.

**6.** Once the controller has been plugged in, we can run sixpair by running the command below. Sixpair will re-configure the controller so that it will talk with out Bluetooth device.

```
sudo ~/sixpair/sixpair
```

**7.** If the sixpair software has successfully reconfigured the PS3 Controller to talk with your Raspberry Pi's Bluetooth dongle, then you should see something like below appear on your command line.

```
Current Bluetooth master: b8:27:eb:a4:59:08
Setting master bd_addr to b8:27:eb:a4:59:08
```

## Setting up a PlayStation 3 controller on the Raspberry Pi

**8.** Now at this point, you are required to unplug the PS3 controller from the Raspberry Pi, now the changes have been made we should no longer need to plug your controller in by USB unless your Bluetooth adapter changes.

You now have two options, these being compiling and installing sixad to handle the connection between the PS3 controller or just using the bluetoothctl tool to make a connection to the PS3 Controller.

Personally, we recommend you try using Bluetooth as it is the most compatible with other Bluetooth devices.

Continue to the next section for how to pair your PS3 controller by using Bluetooth. Otherwise, skip to the section after for how to use SIXAD to pair the controller.

## Using bluetoothctl to connect your PS3 Controller

**1.** With the PS3 Controller now ready to be connected with our Raspberry Pi we can move onto actually pairing it with the Raspberry Pi.

To do this, we need to first start up the Bluetooth configuration tool by running the following command on the Raspberry Pi.

```
sudo bluetoothctl
```

**2.** Now that we have launched up the Bluetooth configuration tool, we need to switch the agent on and set it to utilize the default-agent. Just type in the following two commands to do this.

```
agent on
default-agent
```

**3.** Once we have enabled the agent we now want to actually scan for new Bluetooth devices, running the following command will tell the tool to start scanning.

```
scan on
```

You will notice immediately that any Bluetooth device in the area will start showing up in your command line. Ignore these for now as we only want to pay attention to the PS3 controller itself.

**4.** Now is the time to press down the PlayStation button on the controller, so it begins to talk to the Raspberry Pi. You should soon start to see entries appear in the command line.

**5.** Once you see something like the following appear, make sure you copy down the MAC address that is displayed. It is the sequence of text that is separated with ':' symbols.

```
[NEW] Device B8:27:EB:A4:59:08 B8-27-EB-A4-59-08
[CHG] Device B8:27:EB:A4:59:08 Connected: no
[DEL] Device B8:27:EB:A4:59:08 B8-27-EB-A4-59-08
```

Now that you have a MAC address presented to you, you need to copy it down.
The MAC address in our example, is "**B8:27:EB:A4:59:08**".

## Using bluetoothctl to connect your PS3 Controller

**6.** With the MAC address handy you want to use that in the next command. With this following command, we are trying to get a successful connection and a UUID returned by the device.

Please note you may have to run this command a few times before it is successful.

Make sure that you replace "**YOUR_MAC_ADDRESS**" with the one you found in **step 7**.

```
connect YOUR_MAC_ADDRESS
```

Please keep an eye on your PlayStation 3 controller, if it stops trying to connect then make sure you press the PS button again.

**7.** You will know a successful connection has been made when you see the text as we have shown below appears on the command line. With the **UUIDs** available we can proceed to the next step.

```
Attempting to connect to B8:27:EB:A4:59:08
[CHG] Device B8:27:EB:A4:59:08 Modalias: usb:v054Cp0268d0100
[CHG] Device B8:27:EB:A4:59:08 UUIDs:
    00001124-0000-1000-8000-00805f9b34fb
    00001200-0000-1000-8000-00805f9b34fb
```

**8.** Now that you have the MAC address we need to add this to our Bluetooth trusted list. What this does is allow the controller to connect to the Raspberry Pi on its own.

To do this, we need to just type in the following command to our Raspberry Pi.

Make sure that you replace "**YOUR_MAC_ADDRESS**" with the one you found in **step 7**.

```
trust YOUR_MAC_ADDRESS
```

**9.** Once your PS3 controller has been successfully added to the trusted list, you should see something like below appear on the command line.

```
[CHG] Device B8:27:EB:A4:59:08 Trusted: yes
Changing B8:27:EB:A4:59:08 trust succeeded
```

**10.** With your PS3 controller now successfully connected to your Raspberry Pi, we can now quit out of the Bluetooth configuration tool. You can use the command below or just press **Ctrl + D**.

```
quit
```

**11.** Now you can restart the Raspberry Pi. We do this so we can test to make sure everything is working as intended.

Run the following command to reboot the Raspberry Pi.

```
sudo reboot
```

**12.** With the Raspberry Pi now restarted, press the PS button on your controller to test the connection between it and the Raspberry Pi.

The LED's on the controller should briefly flash as it connects, afterward however if everything has worked correctly only one LED should remain lit on the controller.

# Using SIXAD to conenct your PS3 Controller

**1.** Before we get started with compiling and setting up SIXAD on our Raspberry Pi, we must first install a few packages that we require.

The essential packages here are **libusb-dev** and **libbluetooth-dev** as they contain the code that we need to compile the sixad software.

**Libusb-dev** contains code that allows us to interact with USB devices that are plugged into the computer through code.

**Libbluetooth-dev** provides the code required to be able to interact with the operating systems Bluetooth stack, in sixad's case this is partly used to detect when a PS3 device is connected to the Raspberry Pi.

```
sudo apt-get install git libbluetooth-dev checkinstall libusb-dev
sudo apt-get install joystick pkg-config
```

**2.** Now that we have all the packages that we could need let's go ahead and clone the SIXAD repository from Retropie.

The reason we use Retropie's fork is that it has a few improvements and is tested on the Raspberry Pi.

Run the following two lines on your Raspberry Pi to clone the repository.

```
cd ~
git clone https://github.com/RetroPie/sixad.git
```

**3.** With the repository now cloned to our Raspberry Pi, we can now move onto actually compiling it.

Compiling SIXAD is simple to do and just requires us to run the following two commands.

The first command moves us into the folder, the second line runs the makefile and compiles the program.

```
cd ~/sixad
make
```

**4.** Now with the sixad software now compiled there are a couple more things to be done before we can use it.

The first thing that we need to do is to make a folder that sixad will use to keep track of all its profile configuration files.

Run the following command on your Raspberry Pi to create a folder for it. Using the **-p** flag we can create all parent directories as well.

```
sudo mkdir -p /var/lib/sixad/profiles
```

# Using SIXAD to conenct your PS3 Controller

**5.** Finally lets run the **checkinstall** package that we installed in **step 1** of this section. Basically what this tool does is check the current install then proceeds to run a "make install" if it is required.

At specific points, you will be asked a few things, ignore these and just press Enter to continue until the installation is complete.

Once the installation is complete, you should see the following appear in your command line.

```
**********************************************************************
 Done. The new package has been installed and saved to
 /home/pi/sixad/sixad_20171230-1_armhf.deb
 You can remove it from your system anytime using:
     dpkg -r sixad
**********************************************************************
```

**6.** Now that we have finished installing sixad we can finally run it by typing the following command in on our Raspberry Pi.

```
sudo sixad --start
```

If everything is working as intended, you will now be asked to press the PS button on your controller. Do this now to connect your controller to your Raspberry Pi.

**7.** Now that we know that sixad is up and running we want to make sure that it starts at boot so we can always connect our PS 3 controller.

To do this, we can run the following command to add sixad to the boot sequence of the Raspberry Pi.

```
sudo update-rc.d sixad defaults
```

## Using SIXAD to conenct your PS3 Controller

There are a few different ways you can utilize your PlayStation 4 controller with your Raspberry Pi. Some of these are simpler to deal with than others.

### Sony USB Wireless Adapter

The Sony USB Wireless Adapter is probably the easiest solution to getting your PS4 controller paired with the Raspberry Pi.

Especially since the dongle should be able to work out of the box and pair with any official PS4 Controllers.

### Bluetooth

Bluetooth is a bit more complicated than just using the official Sony USB Wireless adapter and requires you to have a Raspberry Pi 3, Raspberry Pi Zero W, or a USB Bluetooth adapter.

However, this method does not work correctly for every PS4 controller, and you may have to end up utilizing a userspace controller driver called ds4drv to be able to interact with it.

Please note that if you previously set up the SIXAD tool, you will have to uninstall it to use your PS4 controller with the Raspberry Pi.

### USB Cable

Using a USB Cable is the easiest way to utilize your PS4 Controller, just plug it in with a Micro USB cable like you would when first connecting your controller to the PS4 itself.

Using the USB connection, the PS4 controller should just work straight off the bat.

## Pairing your PS4 Controller via Bluetooth

### Pairing the controller the normal way

Please note that if you previously followed our guide on connecting a PS3 controller to your Raspberry Pi via Bluetooth, you will have to follow the next couple of steps.

Otherwise, skip to **step 2** of this section.

**1.** The reason for having to remove SIXAD is that it seems to conflict with the PlayStation 4 controllers and does not allow them to pair and function correctly.

To uninstall the sixad package that we compiled and installed in our PS3 section we need to just run the following command on the Raspberry Pi.

```
sudo dpkg -r sixad
```

**2.** Now before we get to set up everything lets make sure our Raspberry Pi is entirely up to date by running the following two commands on it.

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

# Pairing your PS4 Controller via Bluetooth

**3.** Now let's begin the process of pairing our PS4 controller with our Raspberry Pi. For this guide, we will be using the Bluetooth command line tool.

Run the following command on your Raspberry Pi to start it up.

```
sudo bluetoothctl
```

**4.** With the bluetoothctl tool now up we need to run the following two commands in it, this we enable the agent and set the agent to the default.

```
agent on
default-agent
```

**5.** Now that we have enabled the agent we can finally start scanning for devices. Switching scan on is crucial as it will show up our controller when we tell it to start syncing.

```
scan on
```

**6.** Now is the time to press both your "Share" button and PS button on your controller at the same time like shown in the diagram below. Hold these till the light on your controller starts flashing.

**7.** You should see something like below appear on your command line, once you see this make a note of the MAC address. The MAC address is the hexadecimal number separated by the '**:**' symbol.

```
[NEW] Device 00:01:6C:B4:06:7E Wireless Controller
```

In this case, the MAC Address is "**00:01:6C:B4:06:7E**".

**8.** While your controller is still flashing, type in the following command. Make sure that you replace "**YOUR_MAC_ADDRESS**" with the MAC address that you got in **step 8**. This command makes a connection with your PS4 controller.

If your controller stops flashing just press the "**Share**" and "**PS**" buttons again.

```
connect YOUR_MAC_ADDRESS
```

## Pairing your PS4 Controller via Bluetooth

**9.** Upon a successful connection you should see the following appear in your command line.

```
Attempting to connect to 00:01:6C:B4:06:7E
[CHG] Device 00:01:6C:B4:06:7E Connected: yes
[CHG] Device 00:01:6C:B4:06:7E UUIDs: 00001124-0000-1000-8000-00805f9b34fb
[CHG] Device 00:01:6C:B4:06:7E UUIDs: 00001200-0000-1000-8000-00805f9b34fb
[CHG] Device 00:01:6C:B4:06:7E ServicesResolved: yes
[CHG] Device 00:01:6C:B4:06:7E Paired: yes
Connection successful
```

**10.** Now we need to add our MAC address to the trusted list so the PS4 controller can automatically connect to the Raspberry Pi.

```
trust YOUR_MAC_ADDRESS
```

**11.** With the PS4 controller now trusted by the Raspberry Pi's Bluetooth device, we can now quit out of the tool by pressing **Ctrl + D** or typing in "**quit**".

We can now proceed on to testing whether the controller is working.

**12.** Let's now install the joystick package, while this comes with some installations of Raspbian we will run the install command just in case.

This package includes the applications we need to be able to test whether our controller is working correctly.

Run the following command on your Raspberry Pi to install the software.

```
sudo apt-get install joystick
```

**13.** With the joystick package now installed to your Raspberry Pi you can now run the following command, this will test js0 and receive the values back from it.

```
sudo jstest /dev/input/js0
```

**14.** Now if values change when you move and press buttons on your controller, then you are ready to go with your PS4 controller, and you can finish the tutorial here.

However, if your controller refuses to work, you may have to try the alternative way and utilize the userspace driver, follow our guide in the next section.

## Using ds4drv to connect your PS4 Controller

Only utilize this option if standard Bluetooth doesn't work at all. This method makes use of a userspace driver that talks with the controller and allow it to be understood by the Raspberry Pi.

You will need to of already paired your PS4 controller with your Raspberry Pi before you can continue with the following steps.

**1.** To begin, we need to install the packages we require for ds4drv.

To install the necessary python3 components run the following command on your Raspberry Pi.

```
sudo apt-get install python3-dev python3-pip
```

**2.** With the required python packages now installed to the Raspberry Pi, we can use pip to grab ds-4drv and the python libraries necessary for it.

Run the following command to use pip3 to install it.

```
sudo pip3 install ds4drv
```

**3.** Lets now setup the Raspberry Pi so non-root users can gain access to the joystick device that ds-4drv creates.

To be able to give non-root users access we need to run the following three commands on your Raspberry Pi.

These three commands downloads and applies the ruleset from a github repository then applies them.

```
sudo wget https://raw.githubusercontent.com/chrippa/ds4drv/master/udev/50-ds4drv.rules -O /etc/udev/rules.d/50-ds4drv.rules
sudo udevadm control --reload-rules
sudo udevadm trigger
```

**4.** Now while your PlayStation 4 controller is still paired to the Raspberry Pi, we can run the following command to see if ds4drv can read and interact with your controller.

```
sudo ds4drv --hidraw --led 000008
```

**5.** Once you see text like below appear on your command line, that means that ds4drv can see and talk with your PS4 controller that is paired with the Raspberry Pi.

```
[info][controller 1] Connected to Bluetooth Controller (00:01:6C:B4:06:7E hidraw5)
[info][hidraw] Scanning for devices
[info][controller 1] Battery: Fully charged
```

Now quit out of ds4drv by pressing **Ctrl + C**.

**6.** We need to modify the bootup script now, so it automatically launches ds4drv at startup.

To begin doing this run the following command on the Raspberry Pi.

```
sudo nano /etc/rc.local
```

## Using ds4drv to connect your PS4 Controller

**7.** Now we need to add the following line to this file. This line will launch ds4drv when the Raspberry Pi boots up.

### Find

```
exit 0
```

### Add Above

```
/usr/local/bin/ds4drv --hidraw --led 000008 &
```

Once you have made the changes, you can save the file by pressing **Ctrl + X** then **Y** then hitting **Enter**.

**8.** You can now restart the Raspberry Pi to test whether it is starting up correctly. Do this by running the following command.

```
sudo reboot
```

**9.** Once your Raspberry Pi has rebooted, you can now use the following command to test whether everything is working. If the numbers change then everything is working as it should.

```
sudo jstest /dev/input/js0
```

# Setting up a Wiimote on the Raspberry Pi

## Project Description

Utilizing Wiimote controllers on the Raspberry Pi can be painful to deal with due to its handling of the Bluetooth stack. While they run over standard Bluetooth, their actual functionality is not picked up as a conventional joystick device like Xbox controllers and PlayStation controllers.

Instead to support the full functionality of Wiimote controllers in a way that most applications can support we must utilize two pieces of software.

These being cwiid and wminput. One acts as a library that interprets the Wiimotes actions into joystick actions, the other being somewhat of a driver that creates an input device that applications can utilize easily.

In this guide, we will walk you through the process of setting up a Wiimote on your Raspberry Pi and show you how to setup both cwiid and wminput.

## Equipment

**Required**

- Raspberry Pi (Reccomended to use a Raspberry Pi 3)
- SD Card (8 GB+ Recommended)
- Wiimote Controller

**Optional**

- Bluetooth adapter (required if not using Pi 3)

# Setting up a Wiimote on the Raspberry Pi

Sadly like most wireless controllers, Wiimotes do not work correctly with the Raspberry Pi straight out of the box.

While Wiimote controllers use Bluetooth most of their functionality is not picked up correctly under standard interfaces.

To circumvent this, we can utilize two different pieces of software.

We use these as they are widely accepted as the straightforward solution and is used by popular projects such as retropie.

**1.** Now before we start on our first method of utilizing a Wiimote with the Raspberry Pi, we must first ensure that everything is entirely up to date.

Do this by running the following commands on the Raspberry Pi.

```
sudo apt-get update
sudo apt-get upgrade -y
sudo apt-get dist-upgrade -y
sudo rpi-update
```

**2.** Now that we have updated our Raspberry Pi lets go ahead and install some of the packages that we will need for this tutorial.

**Python-Cwiid** is the python interface to libcwiid. Libcwiid is a library that is designed to act as a bridge between the Wiimote controller and the Linux operating system and interprets the controller's actions into something that can be understood programmatically.

**Wminput** is an essential part of this tutorial, it acts as an event/joystick/mouse driver for the Wiimote, allowing the operating system to understand it as one of its controller devices.

Wminput helps enable it to be compatible with any program that has support for joystick devices.

```
sudo apt-get install python-cwiid
sudo apt-get install wminput
```

Please note that if you are using a **Raspberry Pi 2 or earlier**, you may be required to install the Bluetooth packages, to do this we need to run the following command on our Raspberry Pi.

```
sudo apt-get install bluetooth
```

**3.** With the packages we require now installed to our Raspberry Pi we need to now set up the udev rules to allow non-root users access to our uinput devices.

**Uinput** is the interface that wminput creates our Wiimote on.

We can create these Wiimote rules by running the following command.

```
sudo nano /etc/udev/rules.d/wiimote.rules
```

## Setting up a Wiimote on the Raspberry Pi

**4.** Type in the following line into the file. This line will be read when the system boots up.

```
KERNEL=="uinput", MODE="0666"
```

**5.** Now let's save the file by pressing **Ctrl + X**, then **Press Y** and finally press **Enter**.

**6.** Let's now restart the udev service, we do this so that it will load in the new rules file we wrote in the previous step.

To do this, we just need to run the following command on our Raspberry Pi.

```
sudo service udev restart
```

**7.** Now before we proceed too far, we should make sure that the Bluetooth service is up and running, to do this we can utilize the following command to check its status.

```
sudo systemctl status bluetooth
```

If you see the message "**Active: active (running)**" appear in the command line then you can proceed on to **step 8** of this tutorial as that line tells us that that everything should be working as intended.

If you see anything else appear in the command line next to active such as the following two messages: "**Active: active (exited)**", " **Active: inactive (dead)**"

To try and solve this we can attempt to restart the bluetooth service by running the following command in our Raspberry Pi's terminal.

In some cases this will start the Bluetooth service back up but in others there may be actual errors stopping it from working correctly.

```
sudo systemctl start bluetooth
```

**8.** Now that we know that the bluetooth service is up and running lets proceed to finishing up our set-up of wminput by creating a config file.

This config file will associate all the wiimote buttons with a particular action/event within the Linux kernel's joystick driver.

Lets create the config file we need by running the following command, for our guide we will by naming this "**mywminput**".

```
sudo nano /home/pi/mywminput
```

## Setting up a Wiimote on the Raspberry Pi

**9.** Within this file enter the following lines, this gives a basic setup for the classic controller, Wiimote and nunchuck controller

```
# Classic-Controller
Classic.Dpad.X = ABS_X
Classic.Dpad.Y = ABS_Y
Classic.LStick.X = ABS_HAT0X
Classic.LStick.Y = ABS_HAT0Y
Classic.RStick.X = ABS_HAT1X
Classic.RStick.Y = ABS_HAT1Y
Classic.A = BTN_A
Classic.B = BTN_B
Classic.X = BTN_X
Classic.Y = BTN_Y
Classic.Minus = BTN_SELECT
Classic.Plus  = BTN_START
Classic.Home  = BTN_MODE
Classic.L  = BTN_TL
Classic.R  = BTN_TR
Classic.ZL = BTN_TL2
Classic.ZR = BTN_TR2

# WiiMote
Wiimote.A = BTN_A
Wiimote.B  = BTN_B
Wiimote.Dpad.X = ABS_Y
Wiimote.Dpad.Y = -ABS_X
Wiimote.Minus = BTN_SELECT
Wiimote.Plus = BTN_START
Wiimote.Home = BTN_MODE
Wiimote.1 = BTN_X
Wiimote.2 = BTN_Y

# Nunchuk
Nunchuk.C = BTN_C
Nunchuk.Z = BTN_Z
```

# Setting up a Wiimote on the Raspberry Pi

**10.** This next step is completely optional, if you don't want any of the Wiimote LED's to turn on you can skip to **step 11** of this tutorial.

Also if you want to learn up how to better configure your wiimote's configuration file you can checkout these two helpful doccuments, the first is a list of wiimote buttons and axis that you can utilize in the configuration file, you can find these here: https://github.com/abstrakraft/cwiid/blob/master/doc/wminput.list.

The other is a list of possible buttons and axis's that you can assign to your wiimote's button and axis as listed in the link above, you can find this list here: https://github.com/abstrakraft/cwiid/blob/master/doc/wminput.list.

If you want to set which LED's will be enabled when the connection to your Wiimote has been created then you can set them by using one of these following lines.

Simply add whatever one you want to the bottom of the mywminput configuration file

```
Plugin.led.Led1 = 1
#Plugin.led.Led2 = 1
#Plugin.led.Led3 = 1
#Plugin.led.Led4 = 1
```

**11.** Let's now save our wminput config file. Press **Ctrl + X** then **Y** then finally press **Enter**.

**12.** Now we can proceed to grabbing all our Wiimote's MAC addresses, we will need these to initiate the connection with our Raspberry Pi's bluetooth device.

We can utilize the following command to search for any bluetooth devices around.

```
hcitool scan
```

Once you have started the tool press the **1 + 2 buttons** at the same time on your Wiimote controller to start the discovery process.

If you are unsure what buttons we mean then check out the diagram shown below.

**13.** You should see something like below appear in your command line, this will be the MAC address of your Wiimote controller followed by an identifier.

Write down the MAC address of each of the controllers you intend on using.

```
Scanning ...
    00:24:1E:63:E0:AD      Nintendo RVL-CNT-01
```

## Setting up a Wiimote on the Raspberry Pi

**14.** Now let's proceed with setting up a script to simplify connecting the controllers to the Raspberry Pi. We will write a script so we can get it to initialize at startup.

We will explain what the bash script does section by section to give you an idea of how everything works.

To begin let's create a file in our home directory to store this bash script by running the following command on the Raspberry Pi.

```
nano /home/pi/connectwiimote.sh
```

**15.** Now add the following lines to our **connectwiimote.sh** file. We will try to explain everything as we go along. Begin by entering these first three lines of code.

```
#!/bin/bash
sleep 1 # Wait until Bluetooth services are fully initialized
hcitool dev | grep hci >/dev/null
```

The top line acts as an identifier, so the operating system knows to use **/bin/bash** to run the file.

The second line tells the script to sleep for one second, we do this to make sure that the Bluetooth services are fully initiated before we try to use them.

The third line utilizes the **hcitool** to see whether there are any Bluetooth adapters available to utilize, this will stop the running of the wminput tool if there are no Bluetooth adapters to utilize.

```
if test $? -eq 0 ; then
        wminput -d -c  /home/pi/mywminput YOUR_MAC_ADDRESS &
else
```

The first line just checks to see whether our previous hcitool line has given us a result that we need to know that the Bluetooth adapter is available.

Our next line is one of the most critical parts of the file. This line initializes wminput to begin searching for your Wiimote controller.

You can have multiple of these lines just make sure you replace **YOUR_MAC_ADDRESS** in each occurrence with the MAC Address of a Wiimote controller that you want to connect.

The third line is a simple else statement that follows our final three lines of code.

```
        echo "Bluetooth adapter not present!"
        exit 1
fi
```

These final three lines of code are incredibly simple. The first line just prints "**Bluetooth adapter not present!**" to the command line. This message occurs when the hcitool doesn't see the Bluetooth adapter.

The second line just stops the script from executing any further.

Finally, the last line of our code just marks the end of our if, else statements.

# Setting up a Wiimote on the Raspberry Pi

**16.** Once you have finished writing in all the lines of code, you should end up with something like what we have shown below.

Remember to replace your MAC address in place of **YOUR_MAC_ADDRESS**.

```
#!/bin/bash
sleep 1 # Wait until Bluetooth services are fully initialized
hcitool dev | grep hci >/dev/null
if test $? -eq 0 ; then
        wminput -d -c  /home/pi/mywminput YOUR_MAC_ADDRESS &
else
        echo "Blue-tooth adapter not present!"
        exit 1
fi
```

**17.** Once you are sure, the code is correct, save the file by pressing **Ctrl + X** then **Y** and then finally **Enter**.

**18.** Now with our script created let's go ahead and make it executable by using chmod to give it **775** permissions. This line will allow the owner and group to have full **read**/**write**/**execute** permissions, but only grants everyone else the ability to run the script.

Let's do this by running the following line on our Raspberry Pi's command line.

```
sudo chmod 775 /home/pi/connectwiimote.sh
```

**19.** Now, lets move onto the testing stage of our script. First, we must first enable the uinput module.

We can do this by utilizing modprobe by running the following command on our Raspberry Pi.

```
sudo modprobe uinput
```

**20.** With the uinput module now enabled we can finally run our script to test if it's actually working. To do this just run the following command.

```
/home/pi/connectwiimote.sh
```

While the script is running press the **1 + 2** buttons on your Wiimote controllers, this will initialize the connection.

They should flash briefly before either turning off entirely or changing to the settings you set in **step 9** of this tutorial.

## Setting up a Wiimote on the Raspberry Pi

**21.** Now that we know the script is working let's proceed onto the next steps, and that is making the script and the uinput module startup at boot so your controllers can connect at any time.

We will begin by editing the **/etc/modules** file to load up uinput at boot up. Run the following command to start editing the file.

```
sudo nano /etc/modules
```

**22.** To this file add the following line to the bottom of the file. This line will make sure uinput is loaded at boot up time.

```
uinput
```

**23.** You can now save the file by pressing **Ctrl + X** then **Y** and then finally pressing **Enter**.

**24.** Finally, let's edit the **rc.local** file, we will utilize this to execute our **connectwiimote.sh** script at startup.

Let's begin by editing the file by utilizing the following command.

```
sudo nano /etc/rc.local
```

**25.** Now in this file add the following line.

### Find

```
exit 0
```

### Add Above

```
sh /home/pi/connectwiimote.sh
```

**26.** You can now save the file by pressing **Ctrl + X** then **Y** and then finally pressing **Enter**.

**27.** We must now restart the Raspberry Pi to test to see whether our script is being executed correctly. To do this run the following command on your Raspberry Pi.

```
sudo reboot
```

**28.** Now if you press the **1 + 2 buttons** on your Wiimote controller, it should make a successful connection automatically. If it does, you have completed the process os syncing your Wiimote to the Raspberry Pi.

**29.** Now we should test the controller to make sure it is, in fact, working correctly we make use of the jstest tool. To begin using this lets first install the joystick package, while this is usually included with the full installation of Raspbian, it's not included in the lite edition.

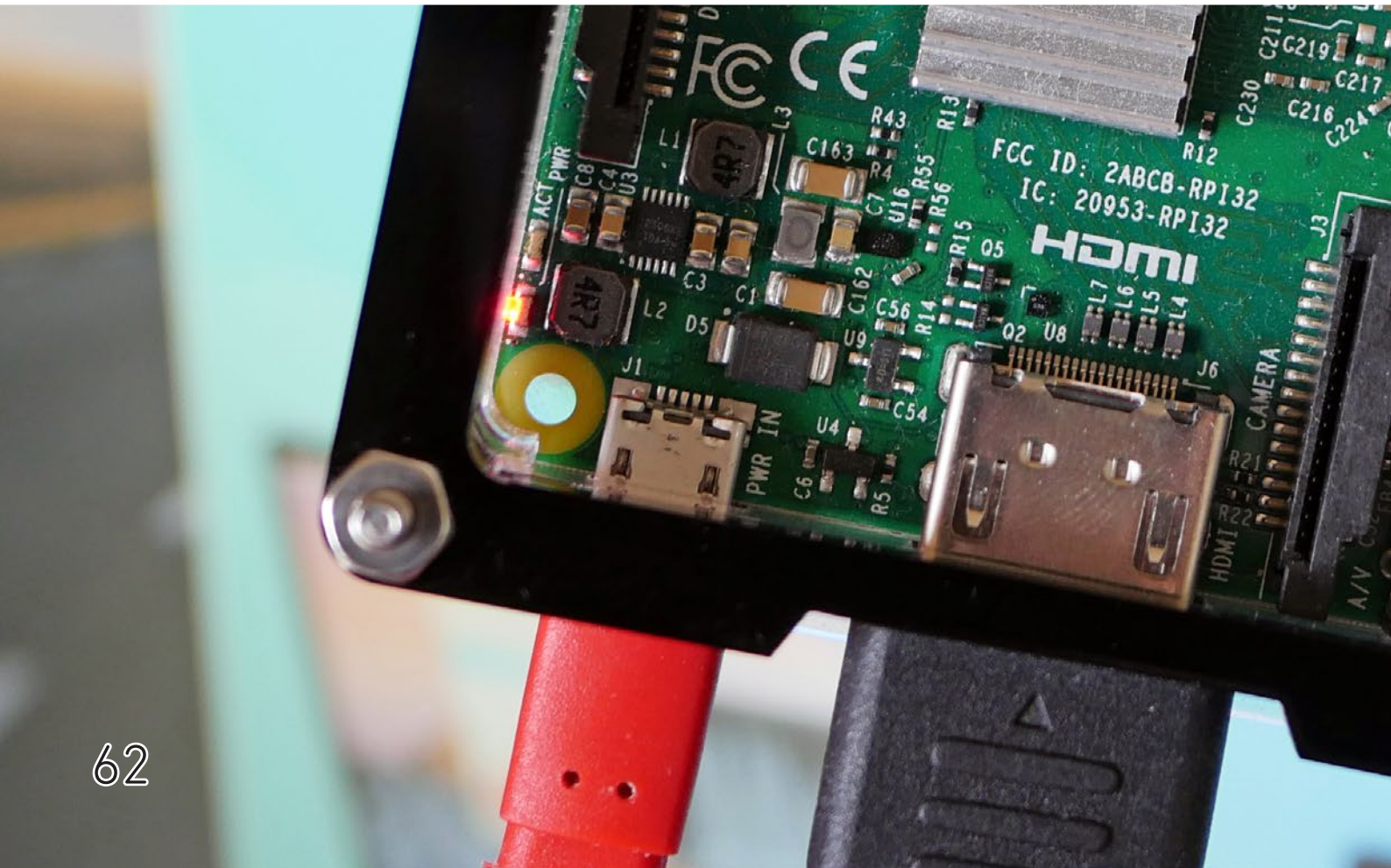
Install it on your Raspberry Pi by running the following command on it.

```
sudo apt-get install joystick
```

**30.** Now lets test our new joystick device by running the following command, when in the jstest tool you should see values change upon doing certain actions on your Wiimote.

```
sudo jstest /dev/input/js0
```

# Networking
## Projects



Wi-Pi™
FCC ID:
OYRCOMFASTBB

# Spoofing MAC Address

## Project Description

In this tutorial, we will be showing you how you can spoof the MAC Address of your Raspberry Pi's network interfaces, either setting it to a known different MAC address or randomly generating it on the fly.

This tutorial is split into two different sections. The first section shows you how you can easily spoof the Raspberry Pi's inbuilt ethernet ports MAC address by modifying the cmdline.txt file.

The second section shows you how you can take this a step further utilizing the **macchanger** tool to change the MAC address of any of your network interfaces.

Spoofing your MAC address can be handy for numerous different reasons, one of these might be to try and retain your device's privacy so you can't just be tracked based off your MAC Address.

Another reason might be that you want to use the Raspberry Pi as a direct replacement for an older Pi and want it to retain the same MAC Address.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

**Optional**
- Raspberry Pi Case
- USB Keyboard
- USB Mouse

## Spoofing the Raspberry Pi's In-built Ethernet MAC Address

**1.** Spoofing your Raspberry Pi's Ethernet's MAC address is an incredibly simple task, especially in comparison to doing this for a WiFi connection.

Spoofing your Raspberry Pi's ethernet MAC Address involves utilizing an override that you can set through the **cmdline.txt** file. This change will prevent the system from just using the default address.

Now there are two different ways that you can modify the **cmdline.txt** file. One is to take out your SD Card from your Raspberry Pi and plug it into your computer. The other is to edit it on the Pi itself.

We recommend taking out the SD Card when changing something like the Raspberry Pi's MAC Address.

If you have inserted your SD Card into your computer you should find the **cmdline.txt** file on the root of your SD Card, open it in your favorite plain text editor.

However, if you decide to edit this file on the Raspberry Pi, you can utilize the following command to begin editing it.

```
sudo nano /boot/cmdline.txt
```

**2.** Within this file, you will find one single line of text with a dozen different options set in it.

To the very end of the line, we want to add the following text, make sure you replace the example MAC Address (**XX:XX:XX:XX:XX:XX**) with your desired one before continuing.

This text is what will tell the system what MAC address to utilize for the ethernet port.

```
smsc95xx.macaddr=XX:XX:XX:XX:XX:XX
```

**3.** Once you have made the change to the file, save it. If you are using nano, you can save the file by pressing **CTRL + X** then **Y** and finally **ENTER**.

You will now need to either plug back in your SD Card into your Raspberry Pi or if you edited the file directly you will now need to reboot the device.

If you are editing your Raspberry Pi's MAC Address directly on the device, then reboot with the following command.

```
sudo reboot
```

**4.** You can now verify whether you have successfully spoofed your Raspberry Pi's MAC Address by running the following command on it. This command will bring up all the information about your ethernet network adapter.

```
sudo ifconfig eth0
```

**5.** If everything has now worked as it should have you should now be able to find your custom, spoofed MAC address displayed on the screen.

For example, if you just used **XX:XX:XX:XX:XX:XX** you should see a line similar to what we have displayed below.

```
ether XX:XX:XX:XX:XX:XX  txqueuelen 1000  (Ethernet)
```

# Spoofing the Raspberry Pi's Network MAC Address's

**1.** As always when first doing a tutorial, we should first ensure that our Raspberry Pi's operating system is entirely up to date by running the following two commands.

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

**2.** In addition to updating all the available packages, we need to go ahead and upgrade the Raspberry Pi's firmware. We do this as some models had issues with the MAC addresses being modified that was fixed in a firmware update.

To update the Raspberry Pi's firmware run the following command, make sure your Raspberry Pi does not lose power while this runs.

```
sudo rpi-update
```

**3.** To get the Raspberry Pi to utilize this updated firmware, we need to now restart the Raspberry Pi by running the following command.

```
sudo reboot
```

**4.** The easiest way to spoof the MAC address for other network devices such as your WiFi is to utilize a program called macchanger. This program handles all the grunt work with modifying and spoofing the MAC address of network devices.

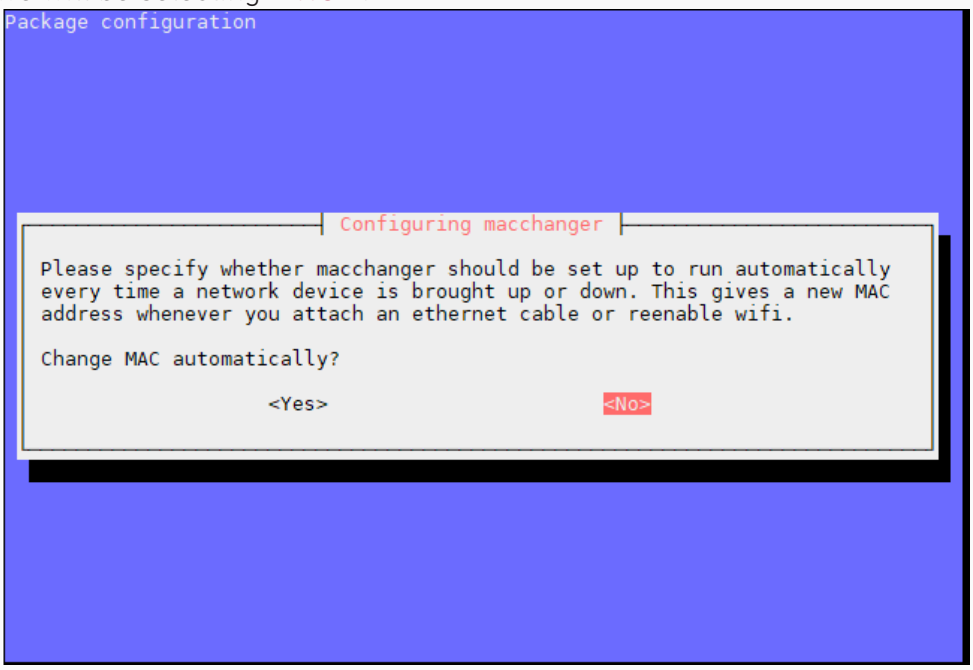
We can install this program to our Raspberry Pi easily by running the following command.

```
sudo apt-get install macchanger
```

**5.** During the installation process of macchanger, you will be asked if you want it to automatically generate new MAC addresses every time a network device is brought up

This, of course, will mean you will not be able to predict what the MAC address will be, but if this is the behavior you want, select "**<YES>**".

For this tutorial, we will be selecting "**<NO>**".



# Spoofing the Raspberry Pi's Network MAC Address

**6.** With macchanger now installed to our Raspberry Pi, we can proceed on to utilizing it.

To start let's first get a list of all currently up network interfaces. We can do that by utilizing the ifconfig program with the following command.

```
ifconfig
```

**7.** From that list of network interfaces, pick the one you intend on spoofing the MAC address for.

Whichever it is make sure you are not currently connected over that for SSH as this will temporarily bring that device offline.

For our tutorial, we will be modifying the MAC address of our **wlan1** device, replace **wlan1** in the next command with the network interface you want to change.

```
sudo ifconfig wlan1 down
```

**8.** Now that we have brought down the interface we can go ahead and utilize the macchanger software to modify its MAC address.

There are two ways to do this. We can either specify the MAC address that we want to utilize or tell the software to generate a random one for it.

Follow **step 8a** if you want to generate a random spoofed MAC address, otherwise follow **step 8b** to see how you can specify your own MAC address.

**8a.** We will assign a random new MAC address to our network interface by using the **-r** argument of macchanger. (There is also the **-e** argument that gives it a somewhat random MAC address by generating it based on the same vendor name)

```
sudo macchanger -r wlan1
```

**8b.** If you want to specify your own MAC Address, you can utilize the **-m** argument alongside your MAC address. Make sure you replace **XX:XX:XX:XX:XX:XX** with the mac address you want to use.

```
sudo macchanger -m XX:XX:XX:XX:XX:XX wlan1
```

**9.** Using either one of these commands should give you a result as we have shown below. With it specifying the current MAC address and the new one that you either just generated or specified.

```
pi@raspberrypi:~ $ sudo macchanger -r wlan1
Current MAC:   00:c1:41:07:11:74 (unknown)
Permanent MAC: 00:c1:41:07:11:74 (unknown)
New MAC:       1e:c7:73:41:5f:40 (unknown)
```

**10.** Now that we have learned how we can spoof the MAC address for any of our network interfaces there is one last thing we need to do, and that is to make these changes take effect on the system boots up.

To do this, we need to create a file within the **/etc/network/if-up.d** folder by running the following command. The files within this directory are called when a network interface is brought online.

```
sudo nano /etc/network/if-up.d/macchange
```

## Spoofing the Raspberry Pi's WiFi MAC Address

**11.** To this file, we need to add the following lines of scrip, when you come to the macchanger line make sure that you change this to reflect what you want it to do.

For instance to use **-r** if you intend on just randomizing the MAC Address or utilizing **-m** if you want to specify your own.

You can change as many MAC addresses from within this script as the file is only called

```
#!/bin/sh

if [ "$IFACE" = lo ]; then
        exit 0
fi

/usr/bin/macchanger -m 02:c9:91:54:f6:06 wlan1
```

**12.** Finally, save the file by pressing **CTRL + X** then **Y** and finally **ENTER**.

**13.** Let's test to ensure everything is working as it should be restarting the Raspberry Pi with the following command.

```
sudo reboot
```

**14.** Once your Raspberry Pi has finished rebooting you will be able to check your network interfaces current MAC address by running the following command.

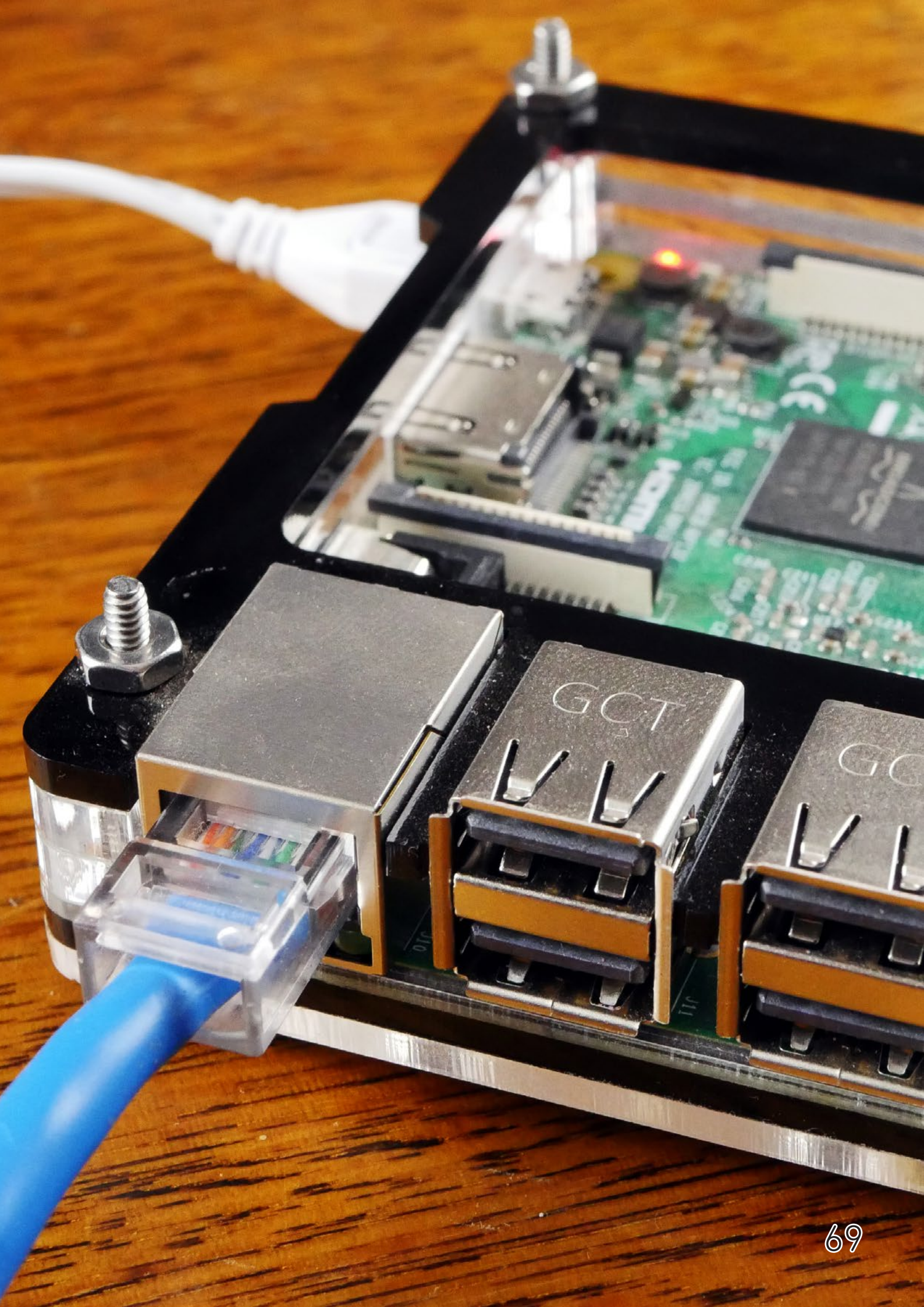
Replace **wlan1** with the network interface that you are utilizing.

```
macchanger wlan1
```

**15.** If everything is working as it should be, you should now see that your "**Current MAC**" differs from your "**Permanent MAC**" as we have showcased below.

```
Current MAC:   02:c9:91:54:f6:06 (unknown)
Permanent MAC: 00:c1:41:07:11:76 (unknown)
```

Hopefully, at this point, you now have your Raspberry Pi automatically spoofing the MAC address for your desired network interface on startup.


GCT

# Setting up and using a VPN

## Project Description

In this tutorial, we will be showing you how you can connect your Raspberry Pi to a VPN to help anonymize your internet activity.

We will be showing you how you can utilize the OpenVPN software, alongside setting it up using two popular VPN services, namely NordVPN and VyprVPN.

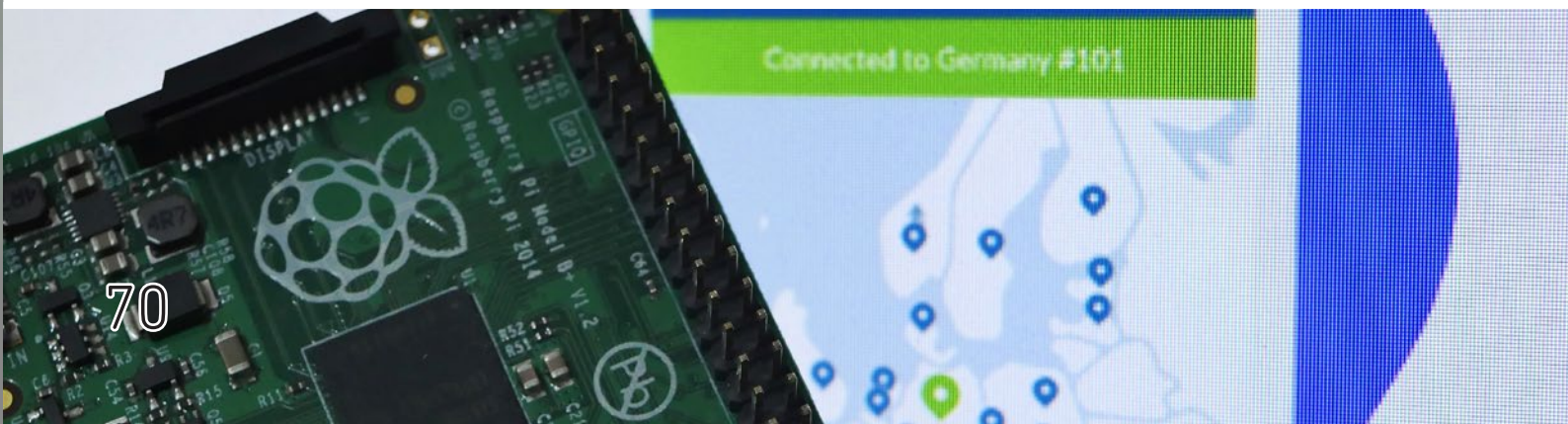
Alongside teaching you how to set up a VPN connection using NordVPN or VyprVPN, we will also show you how to get OpenVPN to start at boot and also show you how to try and prevent DNS leaks by utilizing a different DNS service to the default.

While this tutorial shows you how to set up a VPN connection on your Raspberry Pi, you can also follow our VPN Access Point tutorial to take this a step further and allow your Raspberry Pi's WiFi to provide an easy to connect to VPN connection.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

**Optional**
- Raspberry Pi Case
- USB Keyboard
- USB Mouse



## Preparing your Raspberry Pi for the VPN

**1.** Setting up your Raspberry Pi to access a VPN is a very simple task, to begin however we must first update the operating system to ensure all of its packages are up to date.

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

**2.** Once your Raspberry Pi has completely updated we can proceed on with installing the software that we will be utilizing.

The package that we will be installing is the OpenVPN apckage, this is the software that will do all the grunt work, such as creating the VPN tunnel and handling all the encryption requirements of the VPN.

Install OpenVPN to your Raspberry Pi by running the following command.

```
sudo apt-get install openvpn
```

**3.** Now that we have install the OpenVPN software we will move into its configuration directory. This directory is where we will be storing the .ovpn files that we acquire shortly in the tutorial.

Move to OpenVPN's configuration directory by running the following command on the Raspberry Pi.

```
cd /etc/openvpn
```

**4.** Our tutorial will now split off into two seperate sections temporarily, one section will show you how to setup NordVPN as your VPN provider and the other will show you how to use VyprVPN as your VPN provide.

Before proceeding either sign up to NordVPN, VyprVPN or find your own providers OpenVPN files.

If you are using **NordVPN** move to the section titled "**Setting up NordVPN**".
If you are using **VyprVPN** move to the section titled "**Setting up VyprVPN**".

## Setting up NordVPN

**1.** Now that we are in our OpenVPN folder lets go ahead and download NordVPN's OVPN files by running the following command.

```
sudo wget https://nordvpn.com/api/files/zip
```

**2.** With our NordVPN's OVPN files now on our Raspberry Pi, we need to unzip them by using the following command.

```
sudo unzip zip
```

**3.** Now to connect to NordVPN's VPN network we need to enter the following command.

```
sudo openvpn file_name
```

You will need to replace **file_name** with the file of the location you wish to connect to. You can use the command **ls -l /etc/openvpn/** to see all the possible servers to connect to.

For example, to connect to an Australian server we would utilize the following command.

```
sudo openvpn /etc/openvpn/au151.nordvpn.com.tcp443.ovpn
```

## Setting up NordVPN

**4.** Upon attempting to make a connection to the NordVPN servers you will be asked to enter your credentials. These will be the username and password to your NordVPN account.

Once you have done that you will be connected to your specified VPN server, you can test this by going to the NordVPN website and looking at the banner at the top.

**5.** If you wish to for any reason, disconnect from your VPN you can either just press **CTRL + C** if you are still in the same terminal session, or you can use the following command to kill all sessions of the OpenVPN software.

```
sudo killall openvpn
```

## Setting up VyprVPN

**1.** Now that we are in our OpenVPN folder lets go ahead and download VyprVPN's .OVPN files to our folder by running the following command

```
sudo wget -O vyprvpn.zip \
https://support.goldenfrog.com/hc/article_attachments/360008728172/GF_OpenVPN_10142016.zip
```

**2.** With our VyprVPN zip file now downloaded to our Raspberry Pi, we need to actually extract it. We can easily do this on our Raspberry Pi by running the following command.

```
sudo unzip zip
```

**3.** Now let's move all the files that we need to the base OpenVPN directory to make it easier to deal with the **.ovpn** files.

We can do this by running the following two commands.

```
sudo mv /etc/openvpn/OpenVPN256/* /etc/openvpn/
sudo rm -r /etc/openvpn/OpenVPN256
```

**4.** Now to connect to VyprVPN's VPN network we need to use the following command.

```
sudo openvpn file_name
```

Make sure that you replace **file_name** with the location of where you wish to connect.

For example, If we wanted to connect to **Canada** for example, then we will use **Canada.ovpn**. You can view all available the locations by using **ls -l /etc/openvpn** command.

Below is an example of connecting to Canada utilizing the **Canada.ovpn** file.

```
sudo openvpn /etc/openvpn/Canada.ovpn
```

**5.** You will now be asked for your credentials, enter your VyprVPN username and password to complete the connection.

You can test your connection by going to ipleak.net and checking to see if your IP now differs.

**6.** If for some reason you want to disconnect from the VPN you can just use CTRL + C to kill the current running program in the terminal. Otherwise utilize the command below.

```
sudo killall openvpn
```

## Getting your VPN to start at boot.

**1.** To get the VPN to start up at boot we will first need to write a file to store both the username and password of your VPN provider. We will store this file in our openvpn folder.

Begin writing this file by running the following command.

```
sudo openvpn file_name
```

**2.** Now add your username and password seperated by a new line as we have shown below.

```
username
password
```

**3.** Once done save the file by pressing **CTRL + X** then **Y** and lastly **ENTER**.

**4.** Now we need to copy the ovpn file that you intend on utilizing and rename it, for our example we will be using one from NordVPN but you can do this for any **.ovpn** file.

The key part here is to change the file ending in **.ovpn** to **.conf**.

```
sudo cp /etc/openvpn/au151.nordvpn.com.tcp443.ovpn /etc/openvpn/au151.conf
```

**5.** Once we have copied the file we need to go ahead and modify it. You can do this by utilizing the nano text editor and the following command. Make sure you replace the filename with your own.

```
sudo nano /etc/openvpn/au151.conf
```

**6.** In here you need to find and replace the following line.

**Find**

```
auth-user-pass
```

**Replace With**

```
auth-user-pass auth.txt
```

**7.** Once you have made the change, save the file by pressing **CTRL + X** then **Y** and finally **ENTER**.

**8.** We now need to modify OpenVPN's configuration file so that it automatically starts up and utilizses the **.ovpn** file we modified.

Open up the configuration file with nano by using the following command.

```
sudo nano /etc/default/openvpn
```

**9.** Now find and replace the following line, make sure you use your filename, minus its extension

**Find**

```
auth-user-pass
```

**Replace With**

```
auth-user-pass auth.txt
```

**10.** Save the file by pressing **CTRL + X** then **Y** and finally **ENTER**.

**11.** Now reboot your Raspberry Pi by entering the following command, the VPN should start up after the reboot.

```
sudo reboot
```

## Preventing DNS Leaks

You probably also want to make sure that your DNS isn't leaking your location so you will need to adjust dhcpcd configuration.

It's a simple fix as we will use the Cloudflare public DNS rather than your internet service providers DNS. You of course can set this to utilize any DNS provider that you choose.

**1.** Begin by opening up the dhcpcd configuration file by using the following command on your Raspberry Pi.

```
sudo nano /etc/dhcpcd.conf
```

**2.** Within this file find and replace the following line.

**Find**

```
#static domain_name_servers=192.168.0.1
```

**Replace With**

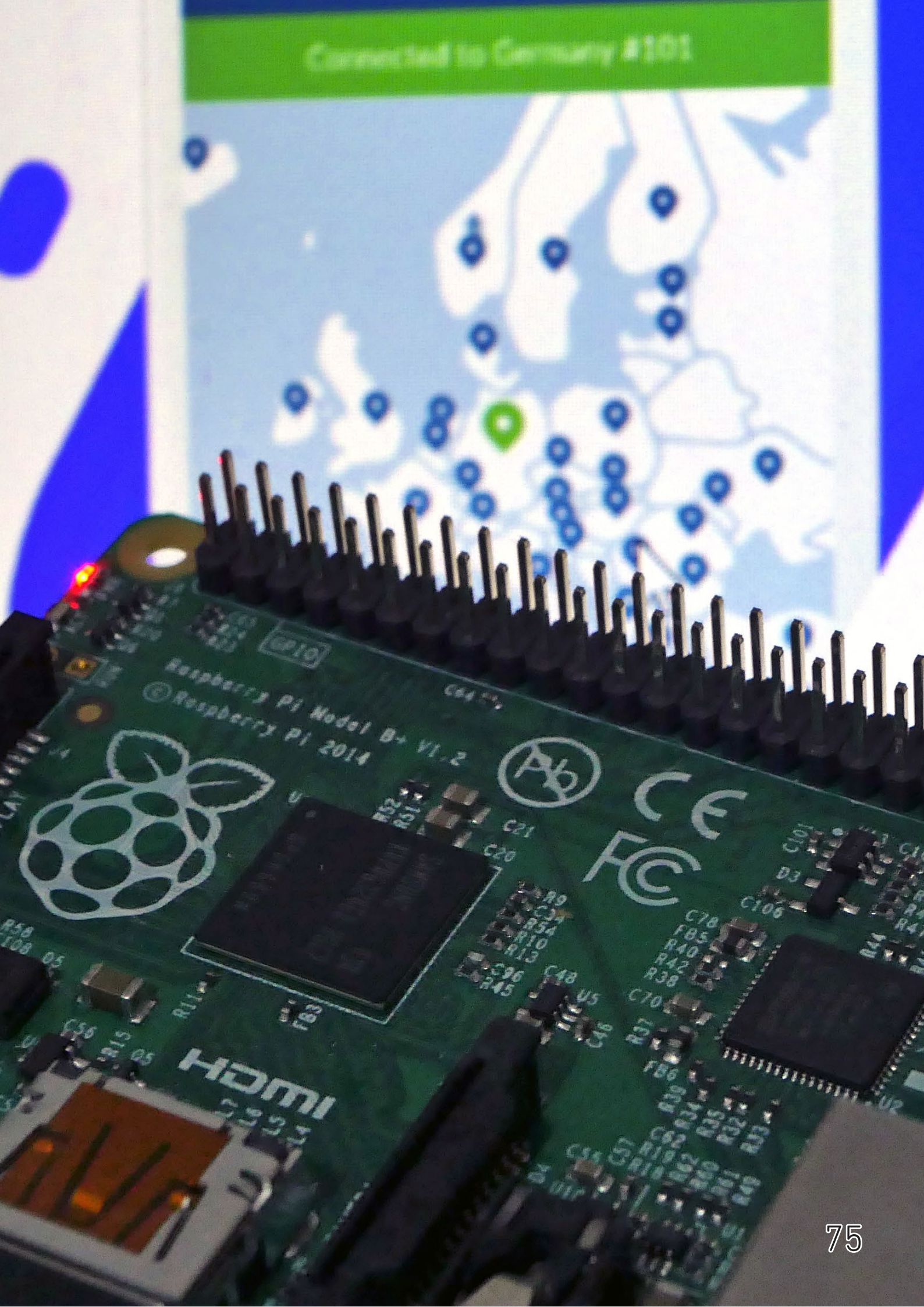
```
static domain_name_servers=1.1.1.1
```

**3.** Once you have made the change you can save the file by pressing **CTRL + X** then **Y** and finally **ENTER**.

**4.** Now let's restart the Raspberry Pi so that it loads in our changes to the dhcpcd service.

```
sudo reboot
```

**5.** You can now check to see if your DNS is leaking by utilizing the website ipleak.net.


Connected to Germany #101
GPIO
Raspberry Pi Model B+ V1.2
© Raspberry Pi 2014
HDMI

# VPN Server

## Project Description

In this tutorial, I will be looking at how to set up a Raspberry Pi VPN server using the OpenVPN software and go into setting up various things you must do to ensure that your connection is as secure as possible by setting up multiple encryption keys.

Setting up a VPN can be a bit of a long process, but it is a relatively simple tutorial to follow, and shouldn't require any interaction once it has been configured.

A Raspberry Pi VPN is a cheap way of setting up a VPN that can stay online 24/7 without consuming a significant amount of power.

VPN's are an incredibly useful network tool that can allow you to gain access to encrypted and secure internet traffic even when you are utilizing public Wi-Fi.

As an bonus, you can also use it to allow you to connect to your computer and your home network, allowing your other devices to act as if it was on the local network of the VPN Server.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

**Optional**
- Raspberry Pi Case
- USB Keyboard
- USB Mouse

## Getting prepared for your VPN server

Before we get started with setting up our Raspberry Pi VPN server, there are a few things we must go over to ensure that you are ready to use it.

For starters make sure you truly need a VPN before you start setting this up, as it can act as a gateway into your home network.

If you do intend on using a VPN make sure all computers on your home network are secure, and that you aren't sharing anything in your network you wouldn't mind someone gaining access to.

## Preparing your VPN Servers IP Address

Secondly decide whether you are going to make use of a static IP address or a dynamic IP address, setting up a VPN for a static IP address is a rather simple process and requires no extra work.

However, if you want to utilize a dynamic IP address, you must make use of a dynamic DNS service.

If you choose to go down the dynamic DNS service route, then you should decide whether you want to make use of your domain, or a free one.

If you want to make use of your domain name, then you can use a service like **CloudFlare**, if you want to make use of a free subdomain then a service such as **no-ip.org** will be useful for you

We have a tutorial earlier on in the book that explains how to set up a dynamic DNS service and how to set up the software on your Raspberry Pi so your IP will always be left up to date on the service.

This tutorial is located on **page 54** in our "**Getting Started with the Raspberry Pi**" section.

Remember the domain name you set up for either Cloudflare or no-ip.org as you will need this later on in the tutorial.

## Portforward for your Raspberry Pi VPN

The third important thing that you will need to get done before you start setting up your Raspberry Pi is to port forward for the OpenVPN software.

The default port you need to forward is **1194**. However we recommend port forwarding a different port and using that instead to try and avoid open port scans on your home network. Remember the port you set as you will need this later on in the tutorial. The protocol you will have to make use of for this port is **UDP**.

You can follow our guide on port forwarding for your Raspberry Pi on **page 50** in our "**Getting started with the Raspberry Pi**" section of this book.

## Installing the VPN Server

**1.** Setting up a Raspberry Pi VPN Server can be quite a complicated process, normally you would have to install the software, generate the encryption keys, add the port to the firewall, set the Pi to keep a static IP address and much more.

Luckily for us there is a much easier way to setup a Raspberry Pi VPN server thanks to an install script called PiVPN, this handles all the grunt work for setting up a VPN and reduces the potential for making mistakes.

Before we get started, we should first change the password of the default pi user, this is to ensure if someone managed to gain access to your VPN they wouldn't be able to access your Raspberry Pi easily.

```
sudo passwd
```

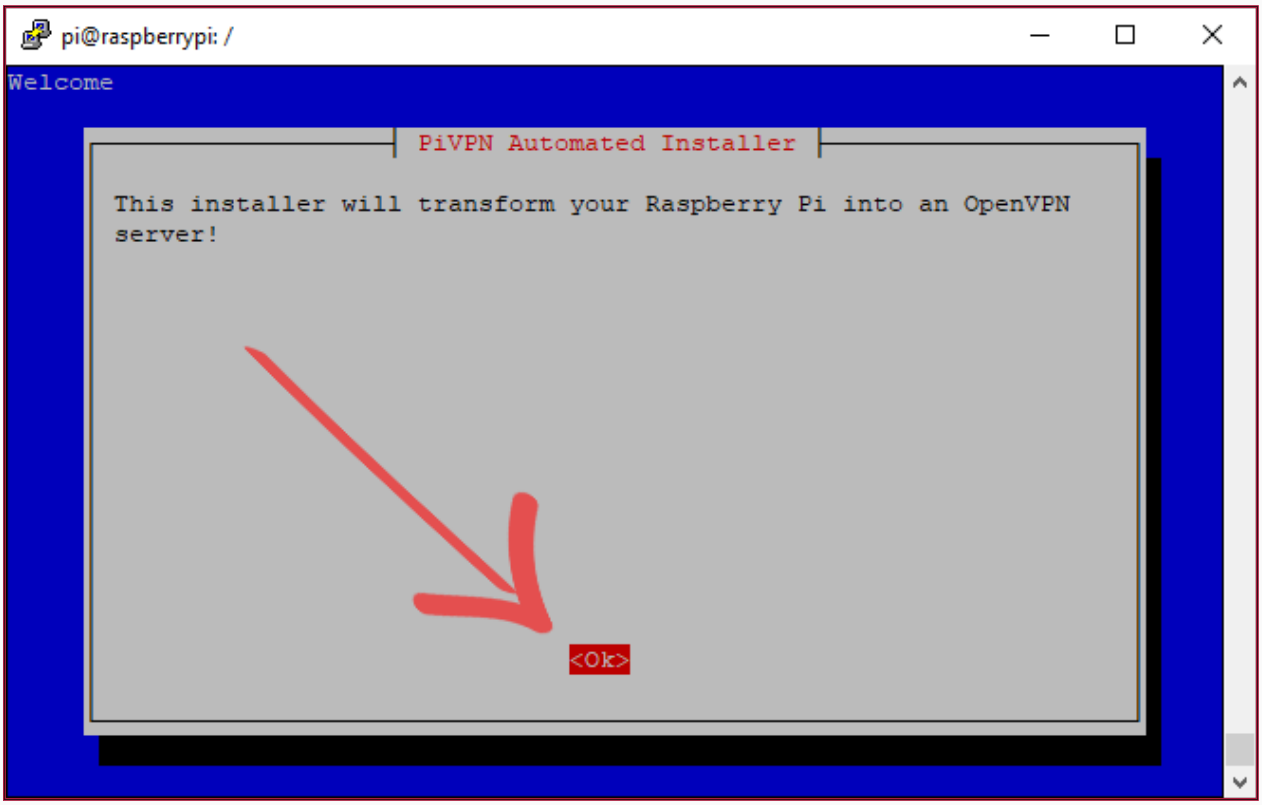
**2.** With the password changed we can begin the process of setting up our VPN server on the Raspberry Pi. We can begin this process by running the command below, this command downloads the install script from PiVPN's github page and runs it.

Normally running a script straight from a URL is a poor idea, as it can be an easy way for someone to gain access to your Raspberry Pi and do some serious damage.

However this is a trusted source that we have verified, if you want to check out the code yourself, just go to https://install.pivpn.io.

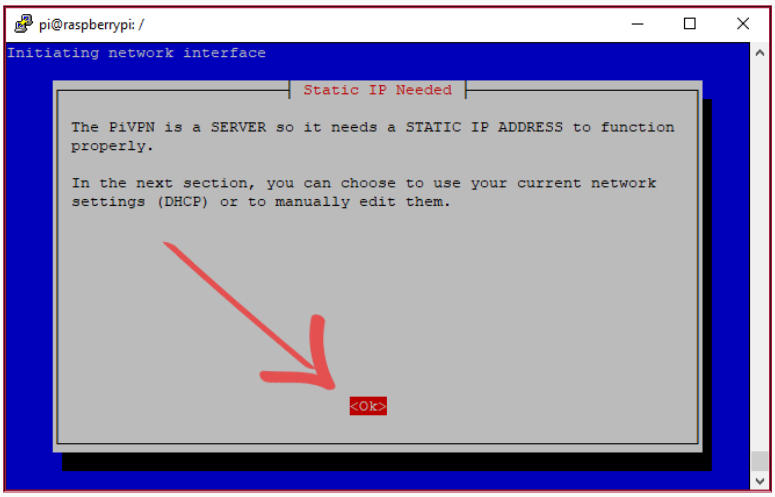
```
curl -L https://install.pivpn.io | bash
```

**3.** Once you have ran that above command you should be met with the following screen. To proceed to the next screen you just need to press enter.

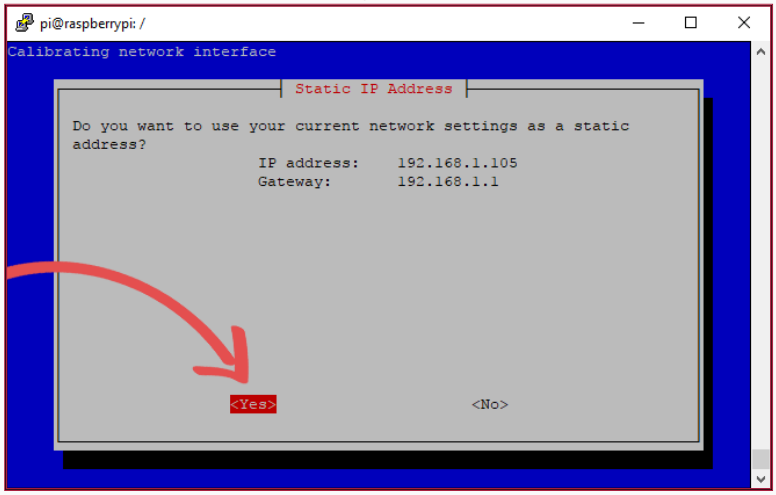


# Installing the VPN Server - Continued

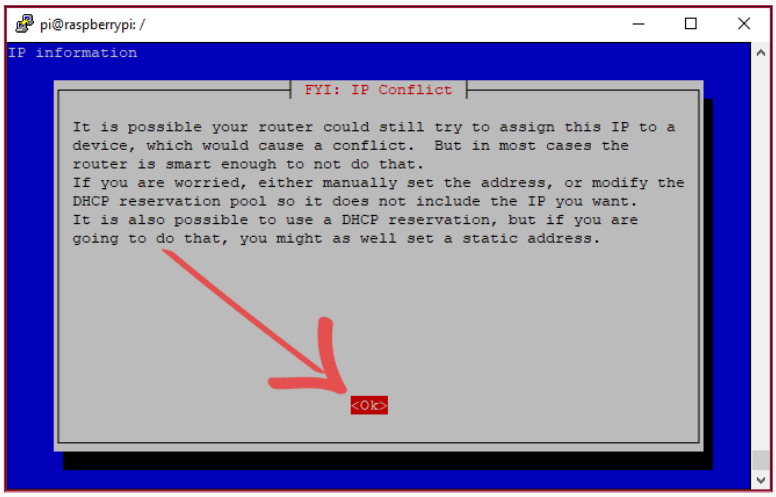
**4.** The next screen basically explains that it will need to setup a static ip address for the Raspberry Pi, this is so that when the Raspberry Pi is restarted for any reason, it will try and utilize the same IP address again.




**5.** On here we will just be selecting **<Yes>** to using the current network settings as a static local IP Address. If you are unhappy with the current settings, then select **<No>**.

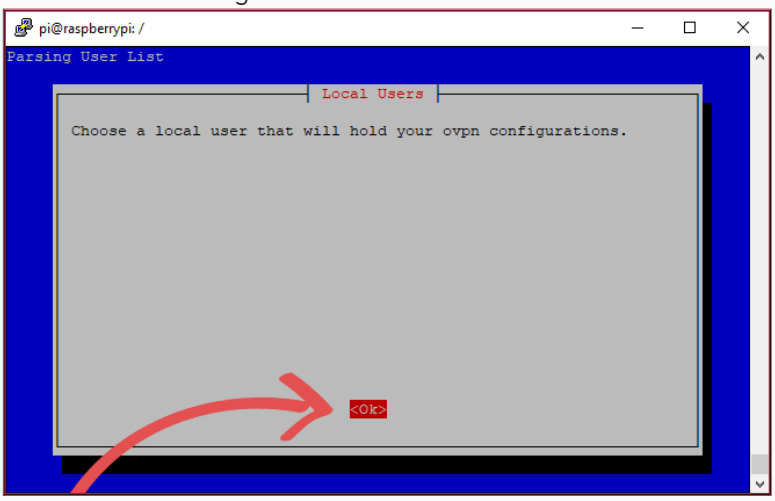



**6.** The warning you are presented with after this basically warns you that there is a chance your router will assign the IP address to another device. Luckily most routers also let you set the IP address to set within its interface as well. For the most part you should be able to just ignore it. So just select **<Ok>** and press enter.

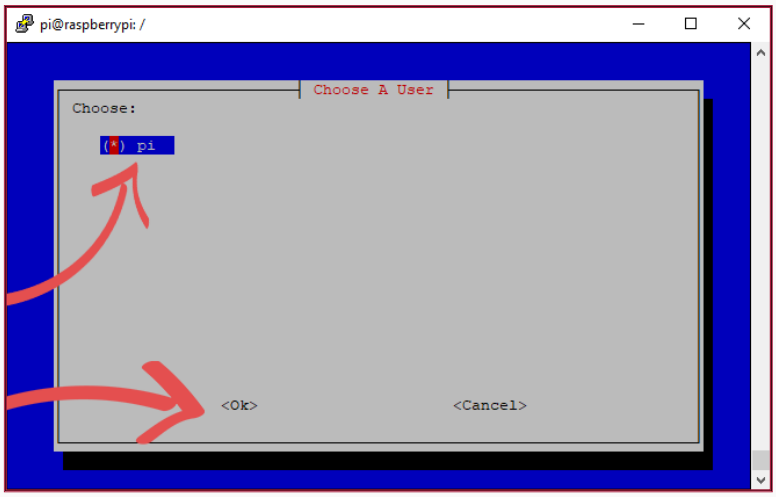


# Installing the VPN Server - Continued

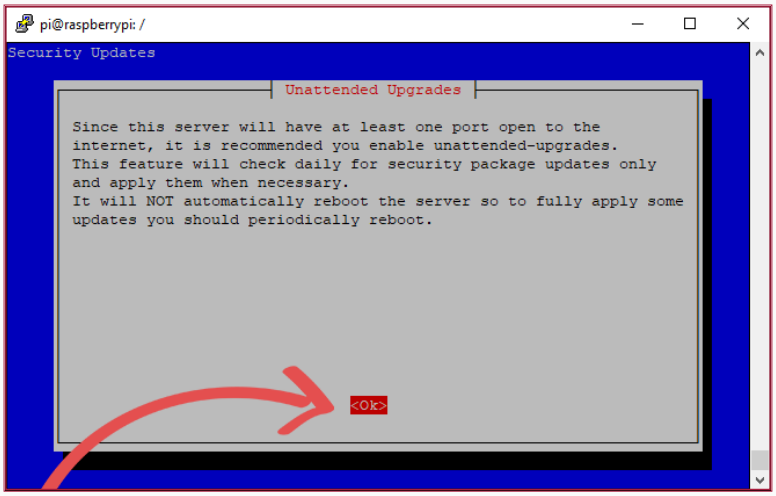
**7.** The next screen explains that we will need to set a local user that the OVPN configurations will be created for. You can just select **<Ok>** and go to the next screen.




**8.** Here we will be presented with a list of users that we can choose. In this tutorial, we will be making use of the **pi** user. Once you are happy with the user you have selected, press **enter**.

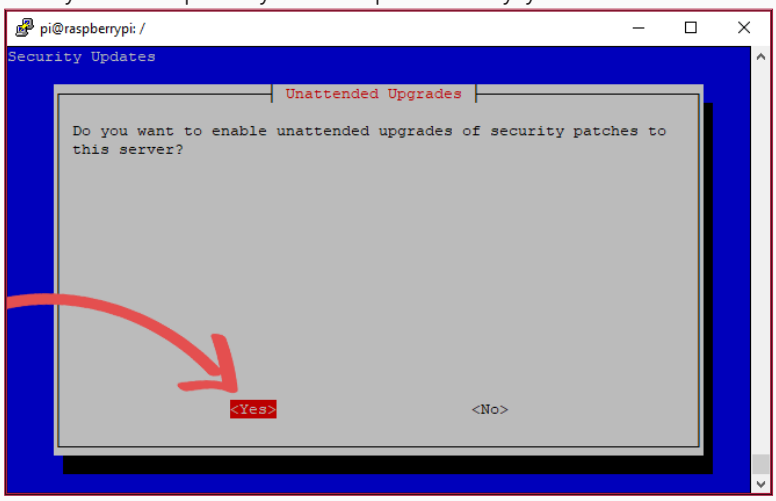



**9.** Now you will be presented with an explanation of Unattended Upgrades. This feature makes Raspbian automatically download security package updates to your Raspberry Pi daily.
This feature will help secure your Raspberry Pi which is incredibly important since we will be opening a port in the router. Select **<Ok>** to continue.

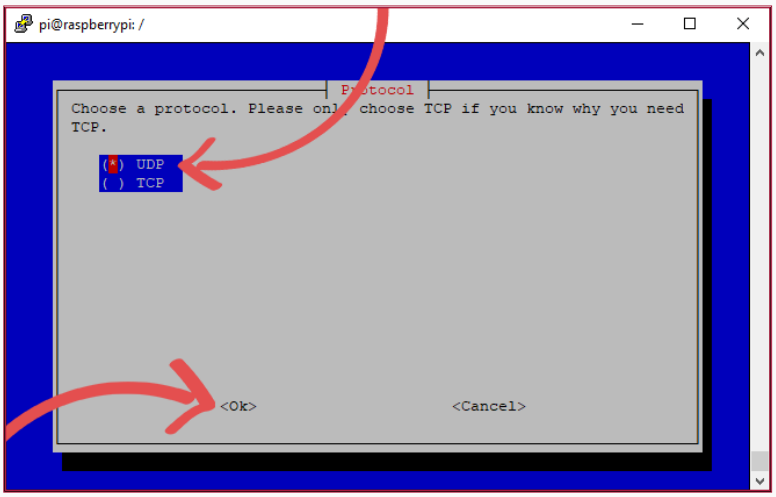


# Installing the VPN Server - Continued

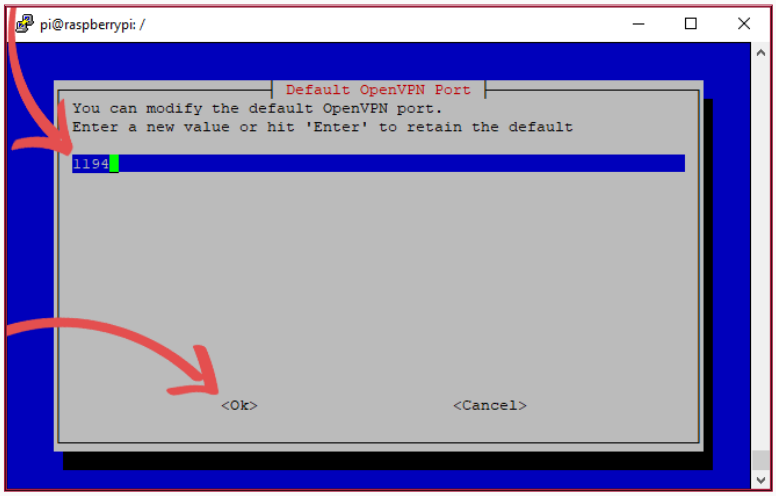
**10.** On this screen, we highly recommend selecting **<Yes>**. Leaving this feature switched off can pose a significant security risk to your Raspberry Pi and potentially your home network.




**11.** Now we will be asked to set the protocol that OpenVPN will run through. We will be making use of UDP, only select TCP if you know why you need it. Press **enter** when you are happy with your choice.

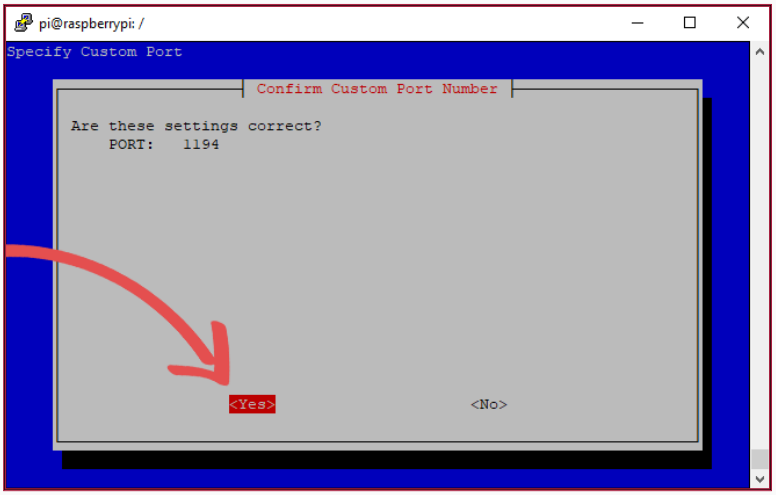



**12.** Now we will be selecting the port OpenVPN will operate through, while you can just press **Enter** on your keyboard to retain the default port of **1194**, we do recommend changing this.
The reason to change it is that if someone does a default port scan on your IP address, it will be much harder for them to know you have a VPN up and running.



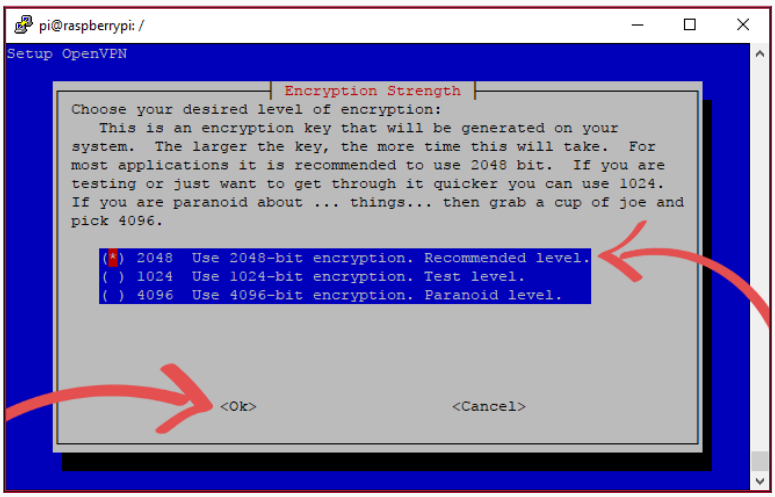
# Installing the VPN Server - Continued

**13.** Below is the confirmation screen for the port number you set, if you are happy with the port, you have chosen then select **<Yes>** to continue.

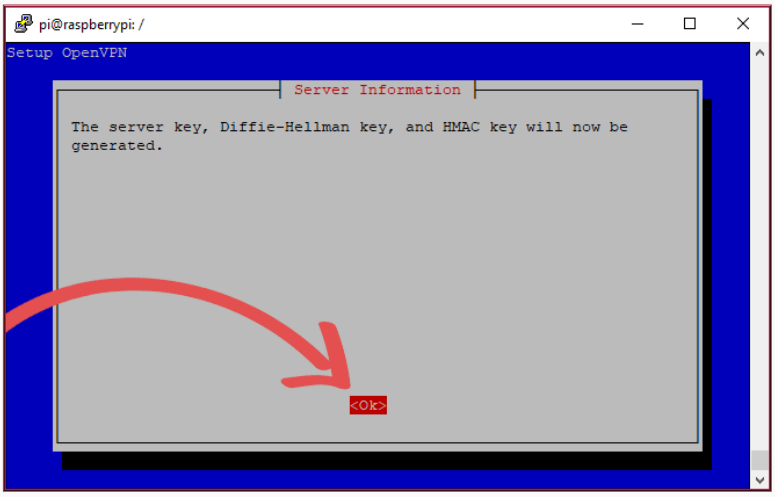



**14.** Now we must choose the encryption key size. We recommend using **2048-bit** encryption as it currently offers excellent protection without sacrificing speed.

If you are anxious about the encryption of your connection, then you can make use of the **4096-bit** encryption key. However, this will take some series time and will slow the overall connection.



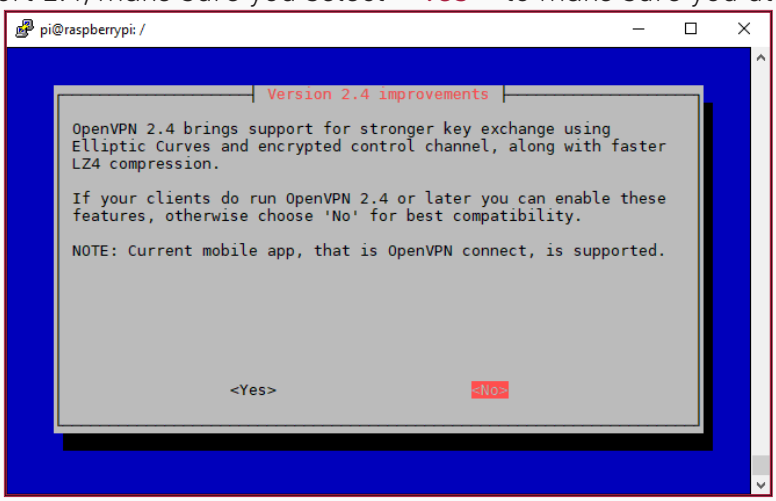

**15.** The next screen tells us what the PiVPN script is about to do, expect this process to take some time, it can take anywhere from a couple of minutes to an hour. Select **<Ok>** to proceed.



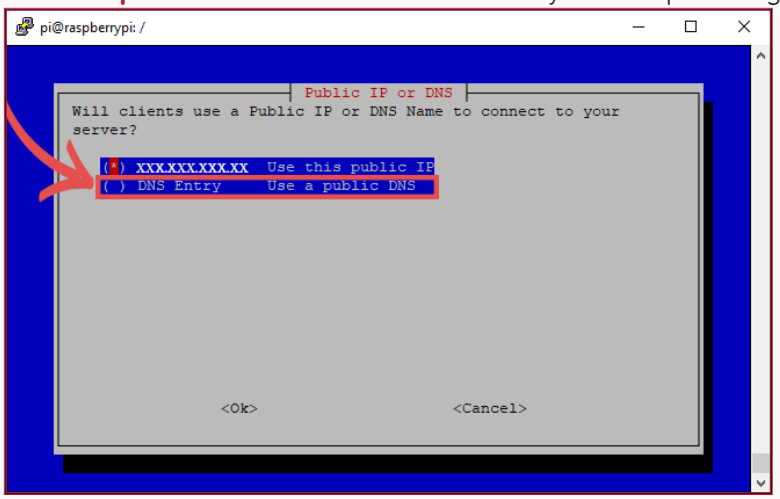
# Installing the VPN Server - Continued

**16.** You will now be asked if you want to utilize the 2.4 version of the OpenVPN software. If you are unsure if your client software supports 2.4 then select "**<No>**".

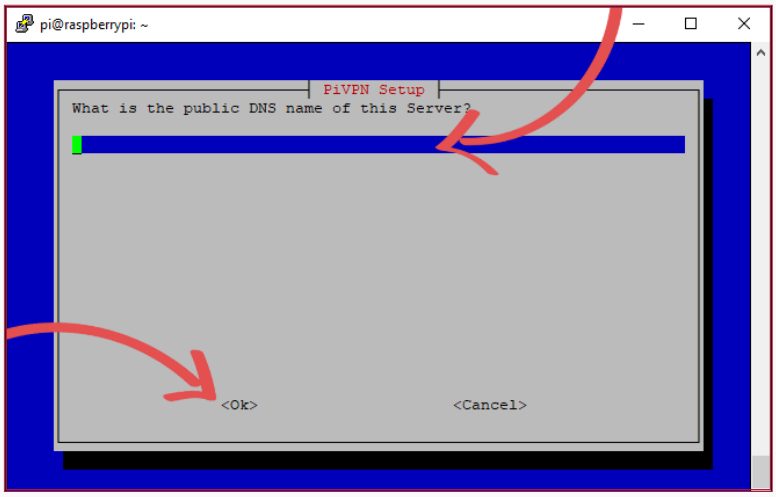
If your client does support 2.4, make sure you select "**<Yes>**" to make sure you utilize all of its benefits.




**17.** Now we need to decide whether we want to make use of our public IP Address or utilize a Dynamic IP service such as **no-ip.org**. If you have a dynamic IP address, then use the **arrow keys** to navigate up and down, and use the **space bar** to select the DNS Entry before pressing **Enter**.



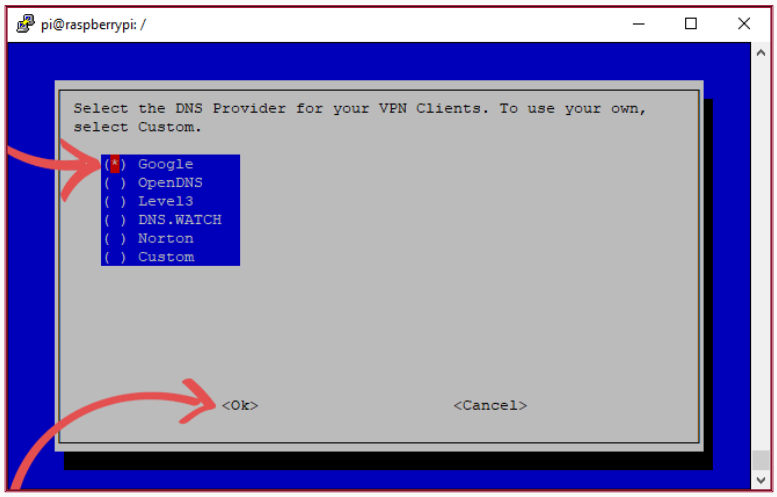

**18.** Now, set your DNS name here, this can be either a **xxxxx.no-ip.org** address or a domain name of your choosing that points to your IP address. You should have set this up before completing this tutorial.



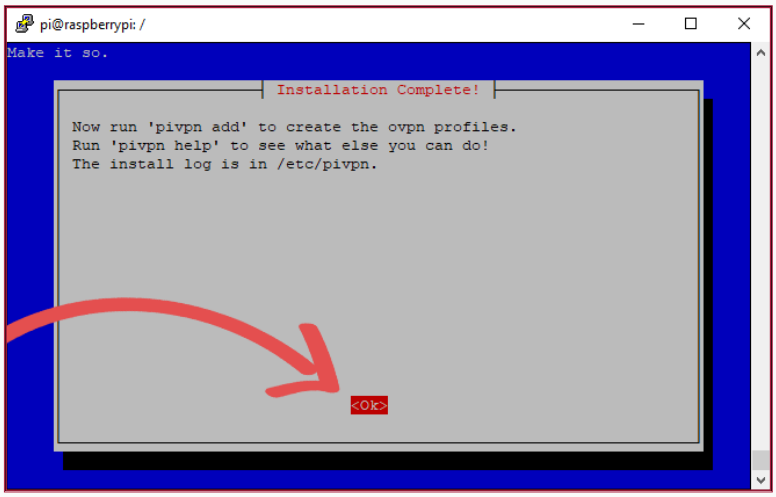
# Installing the VPN Server - Continued

**19.** Now we must select a DNS provider. A DNS provider is what resolves a URL into an IP address.

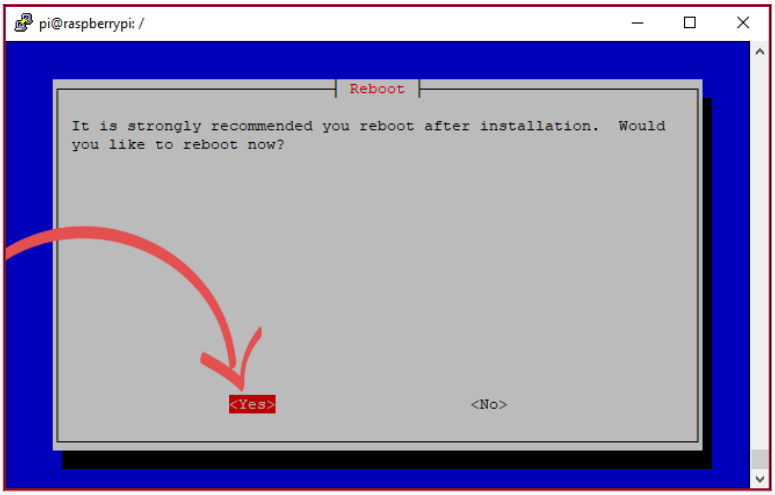
For the sake of simplicity, we will be just making use of Googles public DNS servers. However, they are known for recording the data that passes through them.




**20.** Now you have completed the installation of your Raspberry Pi VPN. While there is still a couple more things you will need to complete to allow connections, you are now about 90% done.




**21.** Now we will be greeted by a screen asking for us to reboot the Raspberry Pi, just select <Yes> to the next two screens as its crucial you reboot.



## Setting up your first OpenVPN user

**1.** Usually setting up a user for Open VPN would be a painful process as you would have to generate the individual certificates for the user, luckily we can do this with one single command thanks to PiVPN.

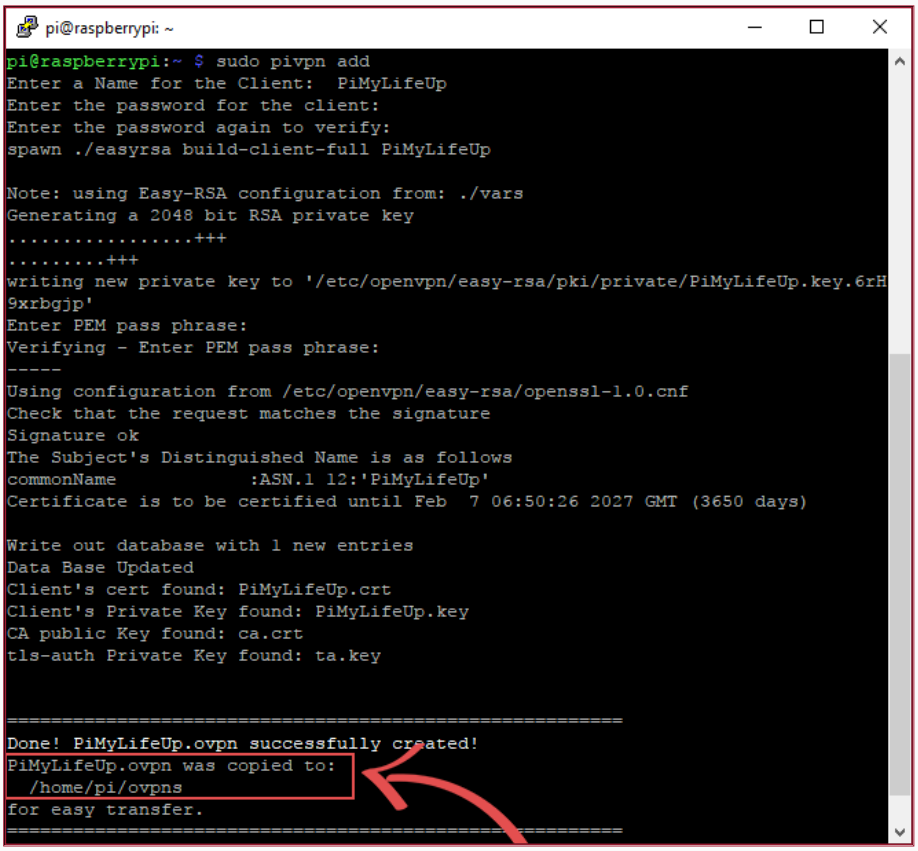
To begin adding the user, run the following command:

```
sudo pivpn add
```

On this screen, you will need to enter a name for the client. This name will act as an identifier so you can differentiate between different clients.

It will also ask you to set a password for the client. It is essential to make this something secure and not easy to guess as this will secure the encryption key.

So, if someone can guess the password, easily it severely reduces the security of your VPN.




Once you press enter to these, the PiVPN script will tell Easy-RSA to generate the 2048-bit RSA private key for the client, and then store the file into **/home/pi/ovpns**.

**/home/pi/ovpns** is the folder we will have to gain access to in the next few steps so we can copy the generated file to our devices.

Make sure you keep these files safe as they are your only way of accessing your VPN.

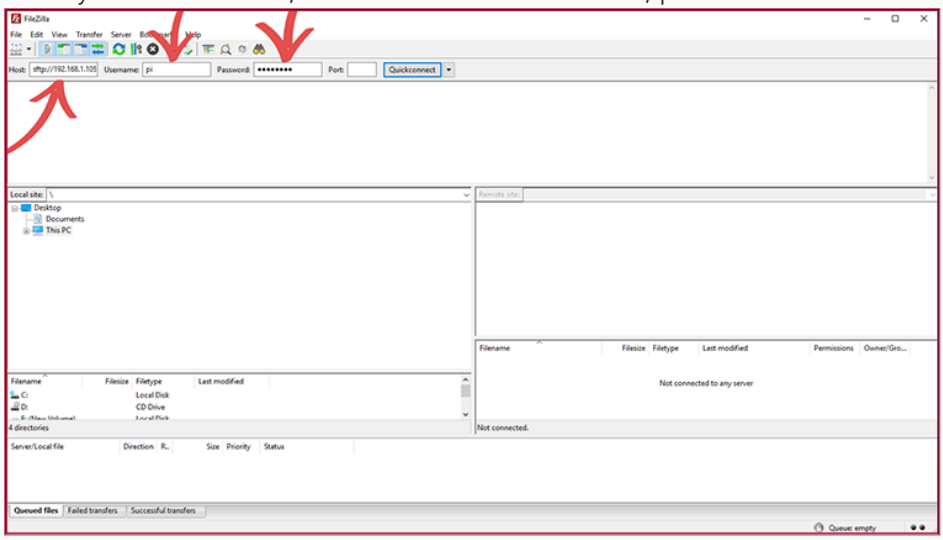
# Setting up your first OpenVPN user - Continued

**2.** Now our new "client" has been set up for OpenVPN with our passphrase we need to get it to the device that we intend on connecting from. The easiest way to do this is to make use of SFTP from within your home network.

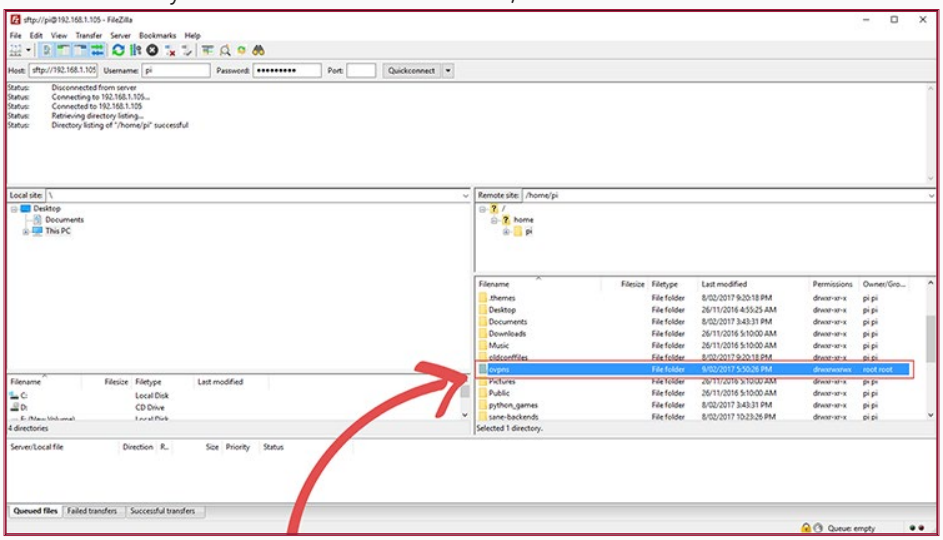
Make sure you have a program such as FileZilla (That you can find here: https://filezilla-project.org/) that can handle SFTP connections installed before continuing with this tutorial.

To get started lets login to our Raspberry Pi over SFTP. Remember to type **sftp://** in front of your Raspberry Pi's IP address. If you don't have you Pi's local address use the command **hostname -I** in the terminal.

Once you have entered your IP address, Username and Password, press the Quickconnect button.



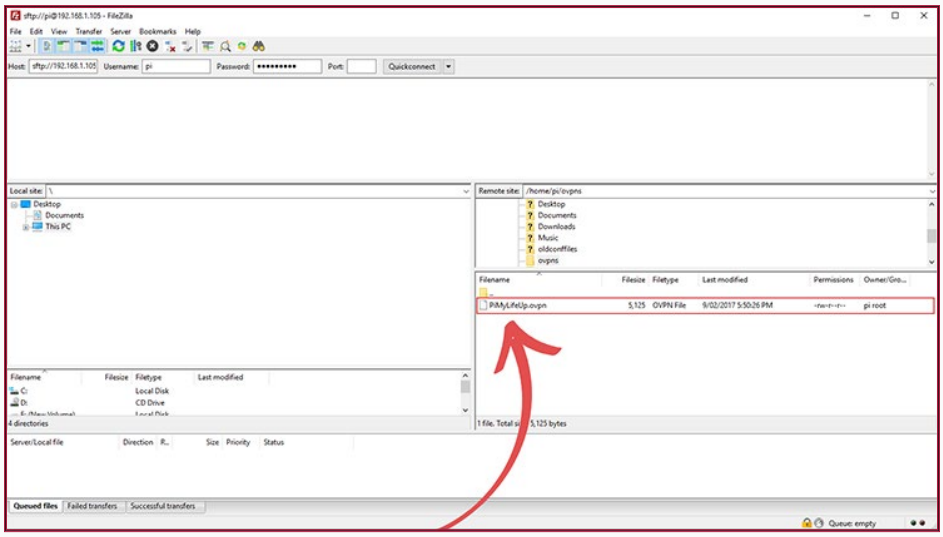
**3.** Once you have successfully logged in, we need to look for the **ovpns** folder, as this is where the file we need will be located. Once you have found the folder, **double-click** on it.



# Setting up your first OpenVPN user - Continued

**4.** Now, all we need to do is drag the **.ovpn** file you want to somewhere safe on your computer. This file contains the data that we will need to connect to the VPN so keep this file safe.

It is also the only way someone could potentially gain access to your VPN, so keeping the passphrase and the file secure is incredibly important. If someone gains access to these, they could cause some harm to your network.

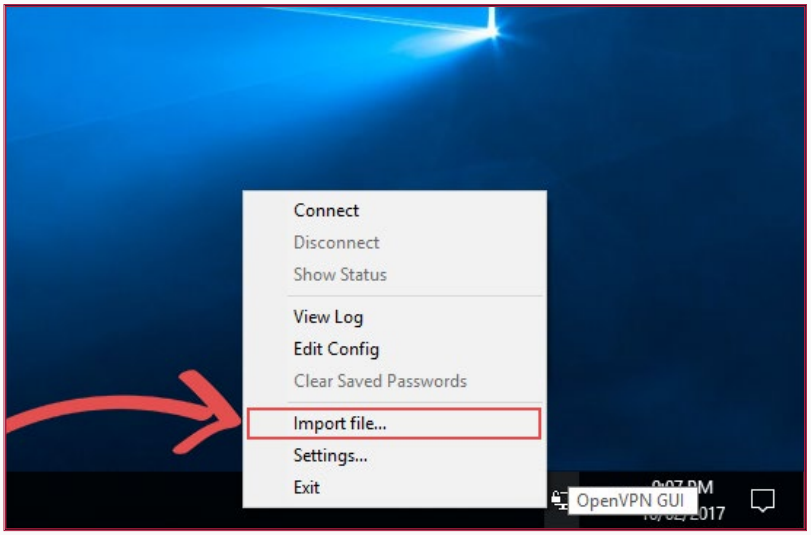


**5.** Now we have the **.opvn** file on our device we can use this to make a connection to our VPN.

The **.opvn** file stores everything we need to make a secure connection. It contains the web address to connect to, and all the encryption data it needs. The only thing it does not contain is your passphrase, so you will need to enter this when you connect to the VPN.

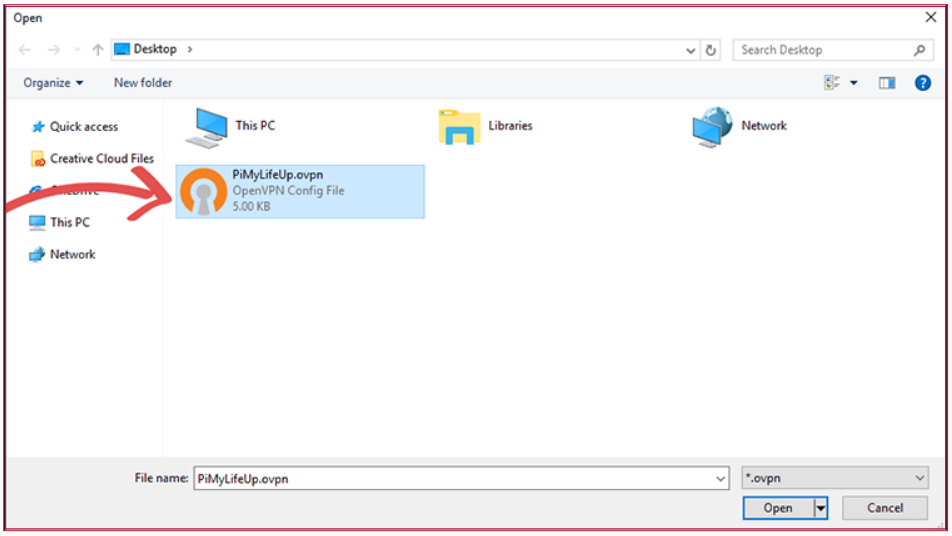
The client we are going to use is the official OpenVPN client. You can obtain this from there official website at https://openvpn.net/index.php/download/community-downloads.html

Download and install this client, on the initial run it will automatically minimize to the taskbar, right click on the icon, then select "**Import file...**"

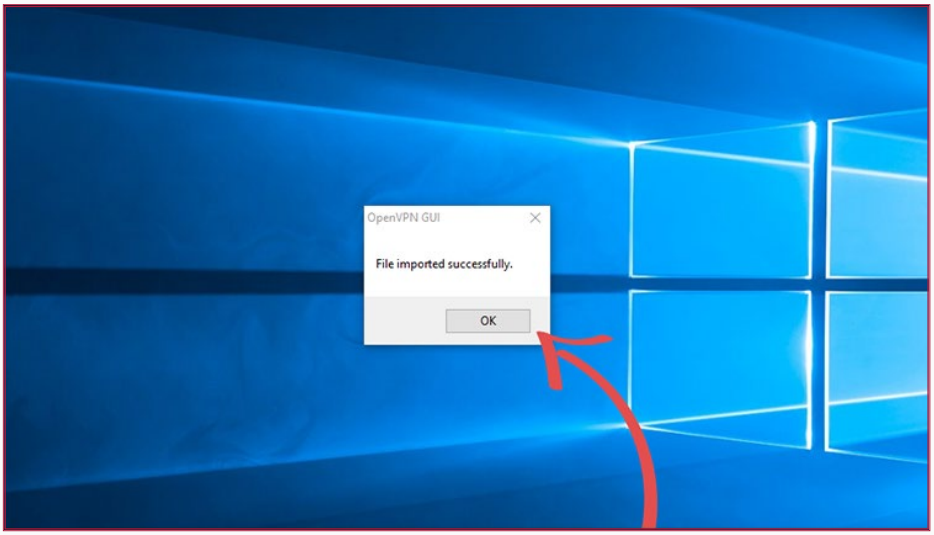


# Setting up your first OpenVPN user - Continued

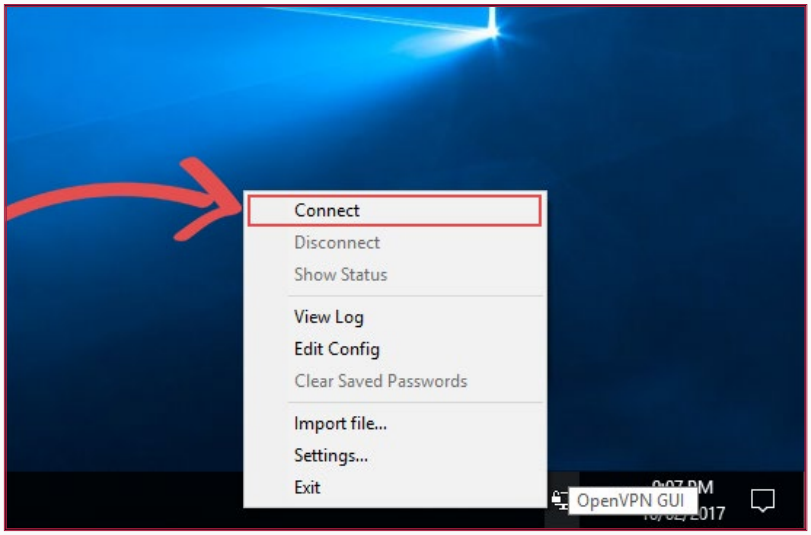
**6.** You will be presented with a File Explorer screen, in here go to where you saved the **.opvn** file from earlier. Once you have found it, **double-click** the file to import into the OpenVPN client.




**7.** You should now be presented with a dialog telling you the file has been successfully imported into OpenVPN. Just press the **OK** button to proceed.



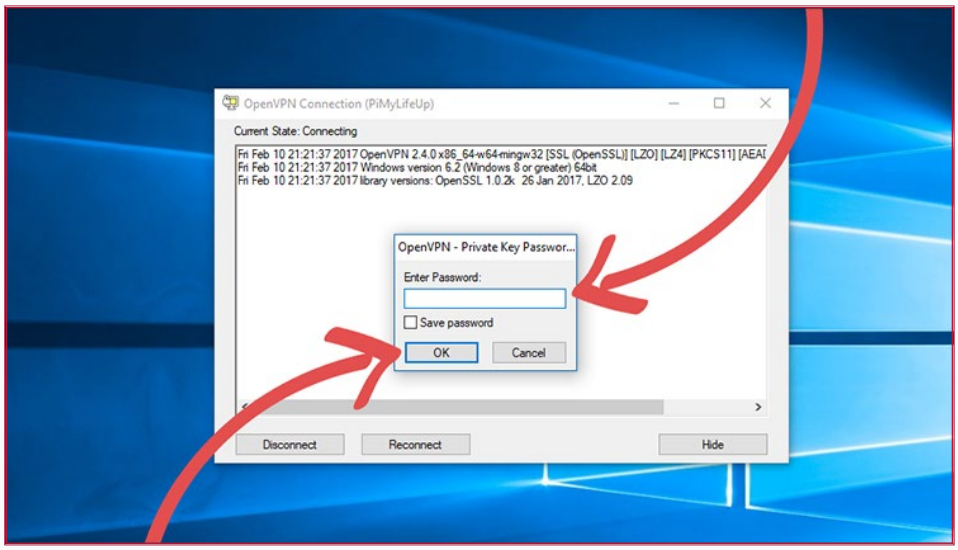

**8.** Now right click the OpenVPN client icon in the taskbar again, this time click "**Connect**".



## Setting up your first OpenVPN user - Continued

**9.** Now the OpenVPN client will attempt to read the data located in the .opvn file. Since we have a passphrase set, it will now ask for you to enter the passphrase you set earlier on in this tutorial.

Once you are certain, you have entered the correct passphrase, press the "**OK**" Button.




**10.** The OpenVPN client will now attempt to connect to your Raspberry Pi's VPN. If the OpenVPN icon turns to a solid green, then it means that you have successfully connected to your VPN.

However, if it turns yellow and fails to turn green after 60 seconds that means something is causing the connection.

In most cases the connection failing will be caused by a port forwarding issue, my router, for instance, has numerous problems with port forwarding. It is easiest to google your routers model number to try and find help on any issues you may face with port forwarding.

If you are using a dynamic DNS service, then make sure that the service is being correctly updated with your latest IP address. If your IP address has changed, but the DNS setting hasn't then it will cause the connection to fail.

Hopefully, by now you will have a fully functional Raspberry Pi VPN that you can successfully connect into.

## Uninstalling the VPN

**1.** If for some reason you want to remove your VPN from your Raspberry Pi you can simply utilize the following command on it.

This command will utilize the pivpn software to uninstall the VPN tunnel.

```
sudo pivpn uninstall
```

# Wireless Access Point

## Project Description

In this tutorial, we will be showing you how to set up a wireless access point, and how to configure the multiple packages that allow users to connect to your access point as if it was a router itself.

You will need to keep in mind that a Wi-Fi dongle most likely won't be able to handle as much traffic as a normal router. Meaning you should avoid allowing too many connections to the device to stop it from becoming too overburdened and slow.

While you can use any Wi-Fi dongle that supports being enabled as an access point our tutorial will directly focus on how to set this up for the Raspberry Pi 3's Wi-Fi Module. If you are using an earlier model of the Raspberry Pi you can purchase a Wi-Fi adapter that supports being utilized as an access point from numerous websites, make sure you research before buying to make sure it works on the Raspberry Pi.

This tutorial can be combined well with our VPN Access Point tutorial. You can find the tutorial on how to set that up located directly after this tutorial. It will show how to set up an OpenVPN client and redirect all traffic to that client.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Ethernet Network Connection
- WiFi (USB or built-in module on the Pi 3)

**Optional**
- Raspberry Pi Case

# Setting up the Wireless Access Point

To set up our Raspberry Pi Wireless Access Point, we will be making the use of two packages.

These two packages are **hostapd** and **dnsmasq**. **hostapd** is the package that allows us to utilize a Wi-Fi device as an access point, in our case, we will be using this to turn the Raspberry Pi 3's Wi-Fi into our access point.

**dnsmasq** acts as both a DHCP and DNS server so that we can assign IP addresses and process DNS requests through our Raspberry Pi itself.

Luckily **dnsmasq** is easy to set up and configure. It also has the advantage that it is somewhat light-weight in comparison to **isc-dhcp-server** and **bind9** packages.

Remember for this you will need to be utilizing an ethernet network connection and not your Wi-Fi connection.

**1.** Before we get started installing and setting up our packages, we will first run an update on the Raspberry Pi by running the following two commands.

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

**2.** With that done we can now install our two packages, run the following command to install **hostapd** and **dnsmasq**.

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

**3.** Now that we have the packages installed we don't want them running yet as we haven't configured them correctly.

Stop the packages from running by utilizing the following two commands in the terminal. These commands will tell the system manager to stop the dnsmasq and hostapd services.

```
sudo systemctl stop hostapd
sudo systemctl stop dnsmasq
```

**4.** With hostapd and dnsmasq now stopped we will want to modify our dhcpd configuration so that we can take control of the wlan0 interface.

With this file, we will be setting ourselves a static IP Address as well as telling it to not to make use of the **wpa_supplicant** file so we can configure it purely as an access point to our device.

Run the following command on your Raspberry Pi to begin modifying the dhcpcd.conf file.

```
sudo nano /etc/dhcpcd.conf
```

**5.** Go to the bottom of the file and add the following configuration lines.

```
interface wlan0
   static ip_address=192.168.220.1/24
   nohook wpa_supplicant
```

Once you have entered those lines we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

# Setting up the Wireless Access Point - Continued

**6.** Now we need to restart our dhcpd service so it will load in all our configuration changes. To do this we just need to run the following command.

```
sudo systemctl restart dhcpcd
```

**7.** Next, we need to adjust our hostapd configuration, to do this we need to begin editing the config file with the following command:

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

**8.** In this file, we need to write out the following lines. These set up how we want to interact with the WLAN device. The only real lines you need to worry about in this file is the **ssid=** line and the **wpa_passphrase=** line.

As a general rule of thumb, you should try and make your WPA Passphrase longer than 6 characters to help keep your connection secure.

**NOTE:** If you are doing this tutorial with a different Wi-Fi device then the inbuilt Pi 3 one, you may have to also change the **driver=** line to the best driver for your device, Google will be your friend for working out what the best driver to use is.

```
interface=wlan0
driver=nl80211

hw_mode=g
channel=6
ieee80211n=1
wmm_enabled=0
macaddr_acl=0
ignore_broadcast_ssid=0

auth_algs=1
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP

# This is the name of the network
ssid=Pi3-AP
# The network passphrase
wpa_passphrase=pimylifeup
```

Remember to change **wpa_passphrase** to your password, make sure you set it to something secure so random people can't just connect to your Wi-Fi access point.
Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

**9.** With that done we now have our hostapd configuration file, but before it can be used, we need to edit two files. These files are what hostapd will read to find our new configuration file.

To begin editing the first of these two files run the following command:

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

# Setting up the Wireless Access Point - Continued

**10.** In this file, we need to find the following line and replace it.

**Find:**

```
#DAEMON_CONF=""
```

**Replace with:**

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

Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

**11.** Now we need to edit the second configuration file. This file is located within the **init.d** folder. We can edit the file with the following command:

```
sudo nano /etc/init.d/hostapd
```

**12.** In this file, we need to find the following line and replace it:

**Find:**

```
DAEMON_CONF=
```

**Replace with:**

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

Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

**13.** With hostapd now set up, we need to move onto setting up **dnsmasq**. Before we begin editing its configuration file we will rename the current one as we don't need any of its current configurations.

We can do this with the following command on our Raspberry Pi.

```
sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
```

**14.** Now that the original configuration file has been renamed we can begin by creating our own new configuration file.

We will create and edit the new file with the following command:

```
sudo nano /etc/dnsmasq.conf
```

**15.** To this file add the following lines.

These lines tell the **dnsmasq** service how to handle all the connections coming through and what interface it should be handling them for.

```
interface=wlan0      # Use interface wlan0
server=1.1.1.1       # Use Cloudflare DNS
dhcp-range=192.168.220.50,192.168.220.150,12h # IP range and lease time
```

Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

**16.** Next, we need to configure our Raspberry Pi so that it will forward all traffic from our **wlan0** connection over to our ethernet connection.

First, we must enable it through the **sysctl.conf** configuration file, so let's begin editing it with the following command:

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

# Setting up the Wireless Access Point - Continued

**17.** Within this file, you need to find the following line and remove the **#** from the beginning of the line.

**Find:**

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

**Replace with:**

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

Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

**18.** Now since we are impatient and don't want to wait for it to enable on next boot we can run the following command to activate it immediately:

```
sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

**19.** With IPv4 Forwarding now enabled we can configure a NAT between our **wlan0** interface and our **eth0** interface.

Enabling IPv4 Forwarding will forward all traffic from our access point over to our ethernet connection.

Run the following commands to add our new rules to the iptable:

```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

**20.** The iptable is flushed on every boot of the Raspberry Pi so we will need to save our new rules somewhere, so they are loaded back in on every boot.

To save our new set of rules run the following command:

```
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

**21.** Now with our new rules safely saved somewhere we need to make this file be loaded back in on every reboot. The most straightforward way to handle this is to modify the rc.local file.

Run the following command to begin editing the file:

```
sudo nano /etc/rc.local
```

**22.** Now we are in this file. We need to add the line below. Make sure this line appears above **exit 0**.

This line reads the settings out of our **iptables.ipv4.nat** file and loads them into the iptables.

**Find:**

```
exit 0
```

Add **ABOVE**:

```
iptables-restore < /etc/iptables.ipv4.nat
```

Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

## Setting up the Wireless Access Point - Continued

**23.** Finally, all we need to do is start the two services and enable them in systemctl. Run the following two commands:

```
sudo service hostapd start
sudo service dnsmasq start
```

**24.** Now you should finally have a fully operational Raspberry Pi Wireless Access Point.

You can ensure this is working by using any of your wireless devices and connecting to your new access point using the SSID and WPA Passphrase that was set earlier on in the tutorial.

To ensure everything will run smoothly its best to try rebooting now. Using this command will ensure that everything will successfully re-enable when the Raspberry Pi is started back up.

Run the following command to restart the Raspberry Pi:

```
sudo reboot
```

# VPN Access Point

## Project Description

In this project we will be exploring how to set up a VPN Access Point. This tutorial builds upon our previous Wireless Access Point tutorial but makes one significant change, and this is to route all the traffic through a VPN.

A VPN Access Point provides a quick and easy way of running any device you want through a VPN even if it doesn't support any VPN software. As all you need to do is connect that device to your Raspberry Pi wireless access point and all its traffic will be automatically routed through a VPN tunnel.

For our tutorial, we will be showing you how to set this VPN access point up by using either the NordVPN or VyprVPN services. While testing out various VPN providers for our tutorial, we found these to be some of the most reliable while retaining both good speed and good privacy policies.

You can if you choose, utilize your VPN service for this if you have any recommendations for other providers you can drop us a comment on our forums.

## Equipment

**Required**

- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Ethernet Network Connection
- WiFi (USB or built-in module on the Pi 3)

**Optional**

- Raspberry Pi Case

## Preparing OpenVPN for the access point

To set up our VPN Access Point, you will first have to of followed our Wireless Access Point tutorial, as this will set up your Raspberry Pi correctly for this tutorial.

For this tutorial, we will be showing you how you can utilize your VPN service to set up a VPN Access Point.

The two VPN's that we will touch on in this tutorial is VyprVPN and NordVPN. We chose these two as we found both of them to be reliable, speedy and good privacy policies.

For the VPN itself, we will be utilizing the OpenVPN software, the two VPN's that we will be touching on offers full support for the OpenVPN protocol and provides us with the .opvn files required to get it to run.

**1.** Before we get started with setting up our VPN Access Point, lets first make sure we have the latest packages by running the following two commands.

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

**2.** Now that we are entirely up to date we can install the OpenVPN software that we will rely on.

We can do that by running the following command in the terminal:

```
sudo apt-get install openvpn -y
```

**3.** Now let's jump to the **openvpn** directory where we will be storing all the stuff we need to get our VPN access point up and running.

Let's jump to the directory with the following command:

```
cd /etc/openvpn
```

**4.** Now that we have changed into the openvpn directory we need to create our authorization file to proceed.

This **auth.txt** file that we will be creating will keep your username and password for your VPN service.

Begin by creating the file by running the following command:

```
sudo nano /etc/openvpn/auth.txt
```

**5.** Now add your username and password to this file as shown in our example below. We will be setting OpenVPN up so that it reads from this file to login to your VPN service.

```
username
password
```

With your login details entered into the file, we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

## Getting the OVPN files for VyprVPN

**1.** Before we begin grabbing the files we need to set up VyprVPN we need first to make sure that we are still in the openvpn directory by running the following command.

Of course, before you can utilize these files you must first sign up to VyprVPN, which you can do by going to the following link:

```
cd /etc/openvpn
```

**2.** Now we are in the configuration directory we will now download our VyprVPN OVPN files. These are what we will load to make a connection to Vypr's network.

Run the following code to grab the zip file.

```
sudo wget -O vyprvpn.zip \
https://support.goldenfrog.com/hc/article_attachments/214728647/GF_OpenVPN_10142016.zip
```

**3.** With the zip file now on our Raspberry Pi, we need to extract it so we can access all the files.

We can easily extract this by running the following command in the terminal:

```
sudo unzip vyprvpn.zip
```

**4.** Now we can check out all the **.ovpn** files by changing into the newly generated directory, in here take note of the location you want to connect into as you will need to specify this in a later command.

```
cd OpenVPN256
```

**5.** Now use the following command to list all of the **.ovpn** files in here.

Take note of the filename for the location you want to use, for instance, if we wanted to connect to the San Francisco in the United States we would take note of **"USA - San Francisco.ovpn"**

```
ls
```

**6.** With your desired ovpn file, move it out of the **OpenVPN256** directory to the **openvpn** directory.

We do this as we will need to make some changes to the file and it will be easier for the daemon to autostart with it sitting in the openvpn directory.

For our example file, you can use a command similar to this to move it to the right directory. This command assumes you are still sitting in the vyprvpn directory.

It is essential that you also change the file type from **.ovpn** to **.conf**, this is required by OpenVPN to detect and load the file.

In addition to moving the file, we will also rename it to make it easier to reference.

```
sudo mv "USA - San Francisco.ovpn" "../USASanFran.conf"
```

**7.** With that done you can now move onto the section titled "**Setting up the VPN access point**", this section will walk you through the changes that you need to make to the .ovpn file and also show you how to get the VPN running at startup.

## Getting the OVPN files for NordVPN

**1.** Now that we have the **auth.txt** file created we will need to grab our required ovpn files from NordVPN.

Remember that before you can utilize these files, you will be required to sign up to NordVPN.

Before we do that, however, lets first make sure we are in the correct directory by running the following command on the terminal.

```
cd /etc/openvpn
```

**2.** Now to obtain the OpenVPN files for NordVPN you need to go to their ovpn webpage on their website that you can find here: https://nordvpn.com/ovpn/.

On this website find the specific server that you want to utilize, right-click the "**Download UDP**" button and copy the link by pressing "**Copy Link Address**".

Paste that URL into the following command in place of the URL that we used.

In our example, we will be making use of the au120 server.

```
sudo wget https://downloads.nordcdn.com/configs/files/ovpn_legacy/servers/au120.nordvpn.com.udp1194.ovpn
```

**3.** Now it is best to rename the file you just downloaded. Renaming the file will make it easier to utilize it later on in the tutorial.

You are also required to change the filetype from **.ovpn** to **.conf**. This change is necessary for OpenVPN to be able to detect and load the file in when we enable autostarting.

You can utilize the mv command to quickly rename the file as we have shown in our example below.

We shortened the **au120.nordvpn.com.udp1194.ovpn** filename to just **au120.conf**, much more straightforward to deal with but still gives enough information about what server it is for connecting to.

```
sudo mv au120.nordvpn.com.udp1194.ovpn au120.conf
```

**4.** Now that you have renamed the file you can move onto the section titled "**Setting up the VPN access point**".

This section will walk you through the process of actually utilizing OpenVPN as well as showing you how to route traffic through it.

## Setting up the VPN Access Point

**1.** Now that we have a .ovpn file ready to go we need to make some modifications to it.

Begin by running the following command, making sure you replace the filename in this command with your own.

```
sudo nano au120.conf
```

**2.** Now within this file, you need to search and make modifications to the following line. You can use **CTRL + W** to make finding this line easier.

This change will make it so that when the OpenVPN client opens up the file, it will know it needs to load your **auth.txt** file.

# Find

```
auth-user-pass
```

# Replace with

```
auth-user-pass auth.txt
```

**3.** Once you have modified the file, you can save it by pressing **CTRL + X** then **Y** and finally **ENTER**.

**4.** With that done we can now test to make sure that OpenVPN is making a successful connection to our VPN provider, in our example this will be NordVPN.

To do this, we will be running the following command, make sure to replace the **.conf** (**.ovpn**) file with the one you modified in the previous step. For our example, we will be using our **"au120.ovpn" file**

```
sudo openvpn --config "/etc/openvpn/au120.conf"
```

This command will output a fair bit of text as it makes the connection, the main text you will want to look for is something like what we have shown below.

If you get an output similar to this, it means that you have now made a successful connection to your VPN providers servers.

```
/sbin/ip route add 81.171.69.2/32 via 192.168.40.1
/sbin/ip route add 0.0.0.0/1 via 172.20.32.1
/sbin/ip route add 128.0.0.0/1 via 172.20.32.1
Initialization Sequence Completed
```

**4.** Now that we have test ran the OpenVPN client we can kill it and proceed onto setting it to launch at startup and changing up our firewall routing so that traffic is directed through the VPN tunnel.

To stop the currently running process press **CTRL + C**.

**5.** With our VPN now working correctly we need to utilize iptables to reroute the wlan0 connection through our tunnel, rather than through to the Ethernet as we did in our Wireless Access Point tutorial.

Before we do this though we will have to flush out our current iptables, we can do that by running the following three commands on the Raspberry Pi.

```
sudo iptables -F
sudo iptables -t nat -F
sudo iptables -X
```

## Setting up the VPN Access Point

**6.** Now let's finally install our new iptables by running the following command, this will tell it to forward all our wlan0 traffic over to the tunnel connection.

```
sudo iptables -t nat -A POSTROUTING -o tun0 -j MASQUERADE
```

**7.** With that done, lets now save our iptables to overwrite the old rules we set up in our Wireless Access Point tutorial. To do this run the following command on the Raspberry Pi.

```
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

**8.** As we now have the firewall set up and ready to go, we need to adjust the OpenVPN client's configuration file.

Begin editing the file by running the following command.

```
sudo nano /etc/default/openvpn
```

**9.** To this file find the following line and change it, so it is both uncommented and "all" is replaced with the name of the file you modified in Step 1 of this section.

You can see our example below.

# Find

```
#autostart="all"
```

# Replace with

```
autostart="au120"
```

**3.** Once you have made the required changes to the OpenVPN configuration file, you can save and exit by pressing **CTRL + X** then **Y** and finally **ENTER**.

**4.** You should now be able to see if everything is working by restarting your Raspberry Pi.

Upon startup, it should automatically make the connection to your VPN, establish the tunnel and then forward traffic from your wireless access point through the tunnel.

```
sudo reboot
```

The easiest way to verify that your VPN access point is working as intended is to check to see if your IP address differs from devices not connected through the VPN.

Alternatively, your VPN provider likely displays whether you are "**Protected**" at the top of their website, both our examples, NordVPN and VyprVPN do provide this feedback.

Hopefully, by now you should have a fully operational Wireless Access Point that will route all its traffic through our OpenVPN tunnel to VyprVPN's service.

Hopefully, you should be left with a stable and fast VPN connection and an easy way to switch to a more secure connection.

## Preventing DNS leaks

You might find that you get DNS leakage as you are likely still utilizing your ISP's DNS servers. Luckily a small tweak to a file can easily fix this issue on your Raspberry Pi.

The change that we will be making will force your Raspberry Pi's DNS to run via Cloudflare's public DNS (1.1.1.1) rather than the internet service provider's.

**1.** Let's begin by modifying the DHCPCD configuration file with Nano by running the following command within the terminal.

```
sudo nano /etc/dhcpcd.conf
```

**2.** Now that we are modifying the configuration file, go ahead and find and replace the following line, this line is what tells the dhcpcd client where to make DNS lookups.

If you have trouble finding the line you can utilize **CTRL + W** to search the file.

### Find

```
#static domain_name_servers=192.168.0.1
```

### Replace with

```
static domain_name_servers=1.1.1.1
```

**3.** You can now save and quit out of the the file by pressing **CTRL + X** then **Y** and finally **Enter**.

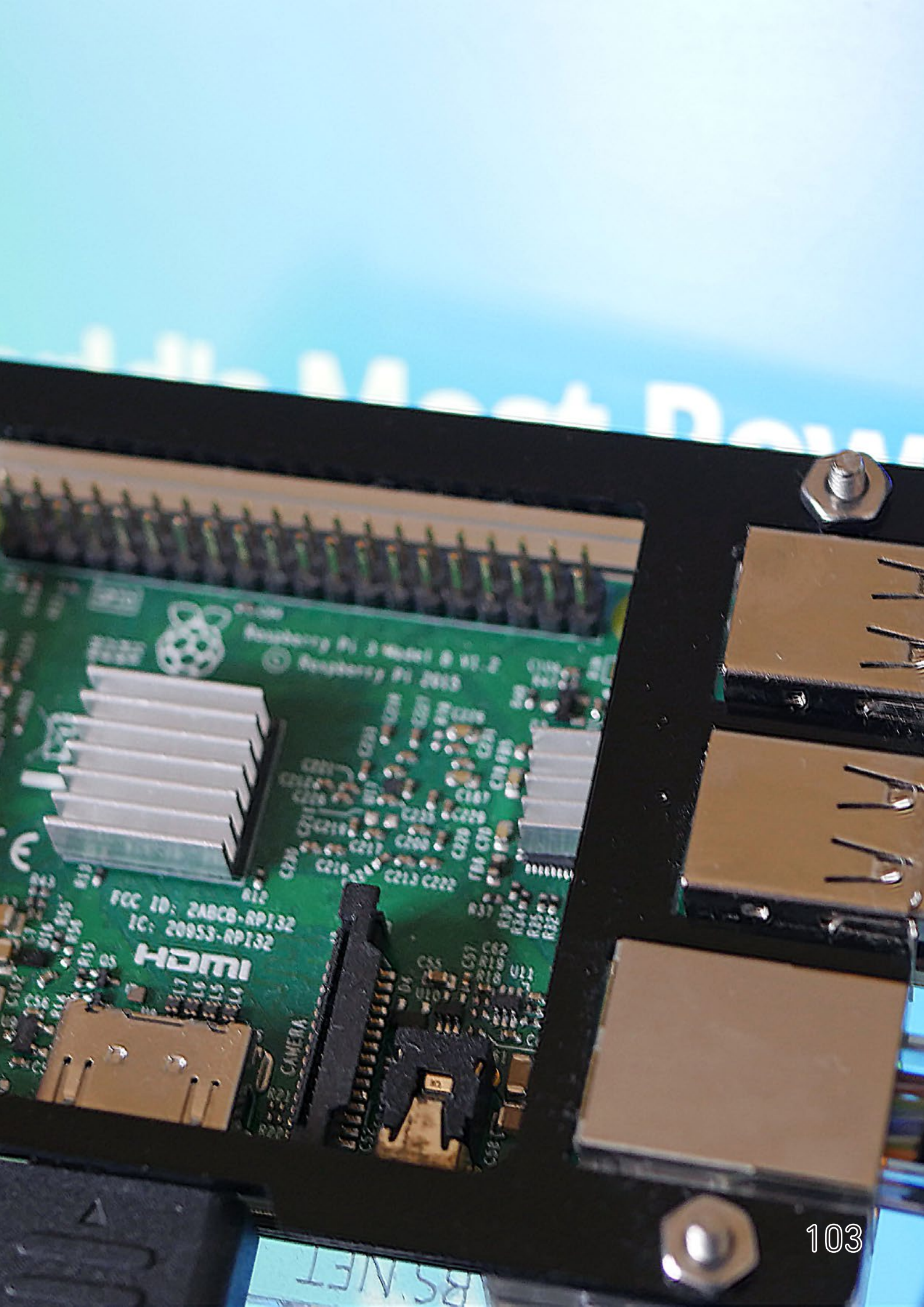
**4.** Now to load in our changes to the DHCPCD configuration file we need to go ahead and reboot the Raspberry Pi.

To restart your Raspberry Pi you can run the following command.

```
sudo reboot
```

**4.** Once you Raspberry Pi has powered back up you should go ahead and now check ipleaks.net to see whether your DNS is till leaking your actual IP address and compromising your identity.

If you are still facing a leak there is a chance that a web browsers WebRTC implementation could be causing that.


FCC ID: 2ABCB-RPI32
IC: 20953-RPI32
HDMI
CAMERA

# Wi-Fi Bridge

## Project Description

A Raspberry Pi WiFi bridge is one of the best ways of providing internet access to a device that only supports an ethernet connection.

In this tutorial, we will show you how to setup a Wi-Fi bridge, and how to setup dnsmasq correctly to allow any device to connect to your device without issues.

You will need to keep in mind that you will not see speeds as good as what you would with a direct connection to your router. As there is some overhead with the connection having to run through the Raspberry Pi.

Remember to do this tutorial you will need a Wi-Fi dongle or a Raspberry Pi 3 with the inbuilt Wi-Fi module.

This tutorial can be combined well with our VPN Access Point tutorial. You can find the tutorial on how to set that up located directly after this tutorial. It will show how to set up an OpenVPN client and redirect all traffic to that client.

Please note, however, this tutorial will require some slight changes, we will explain those changes needed at the end of this tutorial.

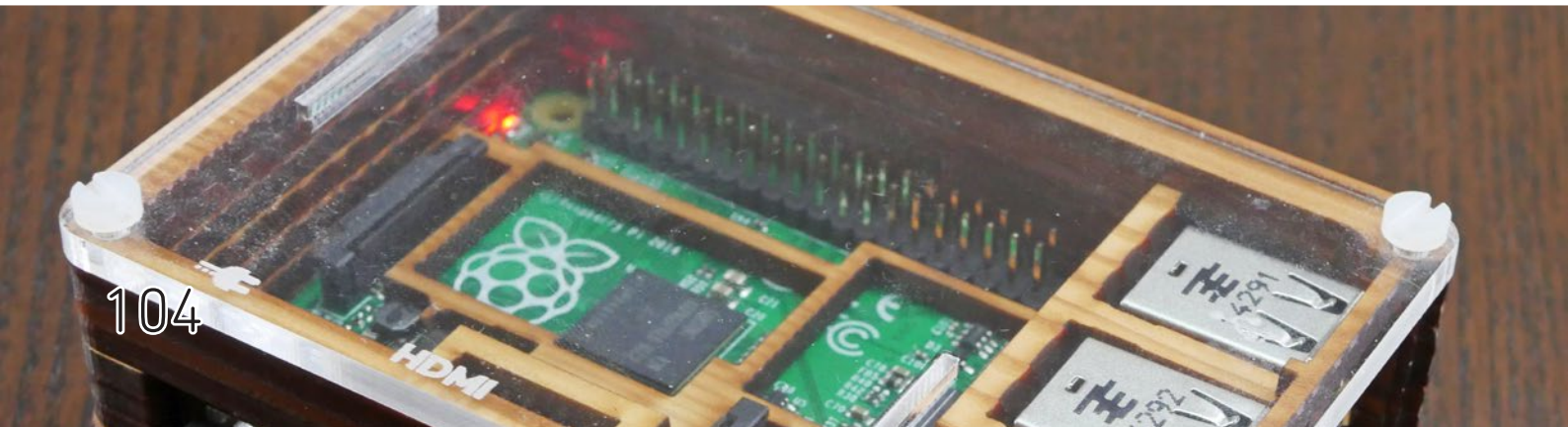
## Equipment

**Required**

- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Ethernet Network Connection
- WiFi (USB or built-in module on the Pi 3)

**Optional**

- Raspberry Pi Case



## Setting up the Wi-Fi Bridge

To setup our Raspberry Pi Wi-Fi Bridge we will be utilizing the dnsmasq package, this package handles most of the grunt work for this tutorial.

Dnsmasq is a package that acts as both a local DHCP server and a local DNS server.

We utilize this package so that we can assign IP addresses and process DNS requests through our Raspberry Pi itself and act as a router.

One of the bonuses to utilizing **dnsmasq** is that it is straightforward to configure while being somewhat lightweight in comparison to the **isc-dhcp-server** and **bind9** packages.

Remember you will need to have an active Wi-Fi router to connect to and an ethernet device you intend on bridging the Wi-Fi connection to.

**1.** Before we get started installing and setting up our packages, we will first run an update on the Raspberry Pi by entering the following two commands into the terminal.

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

**2.** With that done we can now install the only package we will be utilizing, run the following command to install **dnsmasq**.

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

**3.** Before we get too far ahead of ourselves, we should set up the **wlan0** connection that we plan on using.

If you have already setup your wireless connection, then you can skip ahead to **step 5**.

Otherwise, open up the wpa-supplicant file by running the following command:

```
sudo nano  /etc/wpa_supplicant/wpa_supplicant.conf
```

**4.** Within this file add the following, making sure you replace the **ssid** with the name of the network you want to connect to and replace the **psk** value with the password for that network.

```
network={
        ssid="networkname"
        psk="networkpassword"
}
```

**5.** With our wireless network now set up to correctly connect we can proceed with setting up our eth0 interface.

Editing this will allow us to force it to use a static IP address, not setting this up can cause several issues.

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

## Setting up the Wi-Fi Bridge

**6.** Within the interfaces file, we will need to modify the **eth0** section so, it looks something like what we have displayed below.

Doing this will set up the static up address and allow hotplugging on the **eth0** connection.

```
allow-hotplug eth0
iface eth0 inet static
   address 192.168.220.1
   netmask 255.255.255.0
   network 192.168.220.0
   broadcast 192.168.220.255
```

Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

**7.** Before we get started with modifying dnsmasq's configuration, we will first make a backup of the original configuration by running the following command.

```
sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
```

**8.** With the original configuration now backed up and moved out of the way, we can now move on and create our new configuration file by typing in the command below into the terminal.

```
sudo nano /etc/dnsmasq.conf
```

**9.** Now that we have our new file up we want to add the lines below. These lines tell the **dnsmasq** package how to handle DNS and DHCP traffic.

```
interface=eth0       # Use interface eth0
listen-address=192.168.220.1   # Specify the address to listen on
bind-interfaces      # Bind to the interface
server=8.8.8.8       # Use Google DNS
domain-needed        # Don't forward short names
bogus-priv           # Drop the non-routed address spaces.
dhcp-range=192.168.220.50,192.168.220.150,12h #IP range and lease time
```

Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

**10.** We now need to configure the Raspberry Pi 's firewall so that it will forward all traffic from our **eth0** connection over to our **wlan0** connection.

Before we do this, we must first, however, enable ipv4p IP Forwarding through the **sysctl.conf** configuration file, so let's begin editing it with the following command:

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

**11.** Within this file, you need to find the following line, and remove the **#** from the beginning of the line.

**Find:**

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

**Replace with:**

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

Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

## Setting up the Wi-Fi Bridge

**12.** Now since we don't want to have to wait until the next reboot before the configuration is loaded in, we can run the following command to enable it immediately.

```
sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

**13.** Now that IPv4 Forwarding is enabled we can reconfigure our firewall so that traffic is forward from our **eth0** interface over to our **wlan0** connection. Adding this means that anyone connecting to the ethernet will be able to utilize our **wlan0** internet connection.

Run the following commands to add our new rules to the iptable:

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

**14.** Of course, iptables are flushed on every boot of the Raspberry Pi so we will need to save our new rules somewhere, so they are loaded back in on every boot.

To save our new set of rules run the following command:

```
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

**15.** Now with our new rules safely saved somewhere we need to make this file be loaded back in on every reboot. The most straightforward way to handle this is to modify the rc.local file.

Run the following command to begin editing the file:

```
sudo nano /etc/rc.local
```

**16.** Now we are in this file. We need to add the line below. Make sure this line appears above **exit 0**.

This line reads the settings out of our **iptables.ipv4.nat** file and loads them into the iptables.

**Find:**

```
exit 0
```

Add **ABOVE**:

```
iptables-restore < /etc/iptables.ipv4.nat
```

Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

**17.** Finally, all we need to do is start our dnsmasq service. To do this, all you need to do is run the following command:

```
sudo service dnsmasq start
```

**18.** Now you should finally have a fully operational Raspberry Pi Bridge. You can ensure this is working by plugging any device into its ethernet port. The bridge should provide an internet connection to the device you plugged it into.

To ensure everything will run smoothly its best to try rebooting now. Rebooting will ensure that everything will successfully re-enable when the Raspberry Pi is started back up. Run the following command to reboot the Raspberry Pi:

```
sudo reboot
```

## Setting up the Wi-Fi Bridge with a VPN

This tutorial is fully compatible with our VPN access Point tutorial. However, there is one small change you will have to make in **step 13**, rather than using the commands showcased there, run the commands below.

The main change you will see here is that instead of redirecting the traffic from wlan0 through the tunnel we will be redirecting the traffic from our eth0 connection top the tunnel.

```
sudo iptables -t nat -A POSTROUTING -o tun0 -j MASQUERADE
sudo iptables -A FORWARD -i tun0 -o eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i eth0 -o tun0 -j ACCEPT
```

The rest of the VPN access point tutorial can be done without any other changes.

Wi-Pi™
FCC ID:
OYR-COMFAST88

# Wi-Fi Extender

## Project Description

A Raspberry Pi Wi-Fi Extender is a cheap and power efficient way of increasing the total range of your Wi-Fi Network. A Wi-Fi extender differs a bit from a Wi-Fi access point.

The main difference being that instead of getting its network connection from ethernet, it instead gets its network connection from a Wi-Fi adapter.
For this reason, to complete this tutorial, you will require two Wi-Fi adapters, one of these must be access point capable.

You will face a fair bit of degradation in the speed of your network connection when connecting to the Wi-Fi repeater. The main reason for this is that you must wait for the traffic to go over your initial Wi-Fi then be repeated to your access point.

This tutorial can be combined well with our VPN Access Point tutorial. You can find the tutorial on how to set that up located directly after this tutorial. It will show how to set up an OpenVPN client and redirect all traffic to that client.

Please note, however, this tutorial will require some slight changes, we will explain those changes needed at the end of this tutorial.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- WiFi Dongle * 2 (USB or built-in module on the Pi 3)

**Optional**
- Raspberry Pi Case

## Setting up the Wi-Fi Extender

To set up our Raspberry Pi Wi-Fi Extender, we will need to utilize the dnsmasq package. This package handles most of the grunt work for this tutorial as it acts as both our DNS and DHCP server for our connections.

We will also need to utilize the hostapd package. Hostapd is the package that will allow us to set up one of our Wi-Fi modules as an access point.

Remember for this tutorial you will need to have an active Wi-Fi router to connect to and an ethernet device you intend on bridging the Wi-Fi connection to.

**1.** Before we get started installing and setting up our packages, we will first run an update on the Raspberry Pi by entering the following two commands into the terminal.

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

**2.** With that done we can now install the only package we will be utilizing, run the following command to install **dnsmasq** and **hostapd**.

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

**3.** Before we get too far ahead of ourselves, we should set up the wlan0 connection that we plan on using. If you have already setup your wireless connection, then you can skip ahead to **step 5**.

Otherwise, open up the wpa-supplicant file by running the following command:

```
sudo nano  /etc/wpa_supplicant/wpa_supplicant.conf
```

**4.** Within this file add the following, making sure you replace the ssid with the name of the network you want to connect to and replace the psk value with the password for that network.

```
network={
        ssid="networkname"
        psk="networkpassword"
}
```

**5.** With the packages now installed we will want to deny any interfaces from using our wlan1 connection, this will help ensure it is free for our access point utilize.

To do this, we need to edit our dhcpd configuration file with the following command:

```
sudo nano /etc/dhcpcd.conf
```

**6.** Within this file, we need to add the following line to the bottom.

However, if you have any interfaces there, then make sure you place it above those as it is needed to override them.

```
denyinterfaces wlan1
```

Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

## Setting up the Wi-Fi Extender

**7.** With our wireless network now set up to correctly connect we can proceed with setting up our eth0 interface. Editing this will force it to use a static IP address.

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

**8.** Within the interfaces, while we will need to modify the **wlan1**, section so it looks something like what we have displayed below, this will set up the static up address and allow hotplugging on the **wlan1** connection.

```
allow-hotplug wlan1
iface wlan1 inet static
  address 192.168.220.1
  netmask 255.255.255.0
  network 192.168.220.0
  broadcast 192.168.220.255
```

Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

**9.** Now we need to restart our dhcpd service so it will load in all our configuration changes, we also need to reload our **wlan1** interface to make sure it's loaded in our interface changes. Run the following two commands to reload both:

```
sudo service dhcpcd restart
sudo ifdown wlan1; sudo ifup wlan1
```

**10.** Next, we need to adjust our hostapd configuration, to do this we need to begin editing the config file with the following command:

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

**11.** In this file, we need to write out the following lines. These lines set up how we want to interact with the WLAN device. Remember to change the **passphrase** to something unique

**NOTE:** If you are doing this tutorial with a different Wi-Fi device then the inbuilt Pi 3 one, you may have to change the **driver=** line to the best driver for your device, Google will be your friend.

```
interface=wlan1
driver=nl80211

hw_mode=g
channel=6
ieee80211n=1
wmm_enabled=1
ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]
macaddr_acl=0
ignore_broadcast_ssid=0

auth_algs=1
wpa=2
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP

ssid=Pi3-Extender
wpa_passphrase=raspberry
```

Remember to change **wpa_passphrase** to your password, make sure you set it to something secure so random people can't just connect to your Wi-Fi access point.
Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

## Setting up the Wi-Fi Extender

**12.** With that done we should now have our hostapd configuration, but before it can be used, we need to edit two files. These files are what hostapd will read to find our new configuration file.

To begin editing the first of these two files run the following command:

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

**13.** In this file, we need to find the following line and replace it:

**Find:**

```
#DAEMON_CONF=""
```

**Replace with:**

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

Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

**14.** Now we need to edit the second configuration file. This file is located within the **init.d** folder. We can edit the file with the following command:

```
sudo nano /etc/init.d/hostapd
```

**15.** In this file, we need to find the following line and replace it:

**Find:**

```
DAEMON_CONF=
```

**Replace with:**

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

Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

**16.** With hostapd now setup we need to move onto setting up dnsmasq before we begin editing its configuration we will move the default one to a new location.

We can do this with the following command:

```
sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
```

**17.** Now that the original configuration file is moved out of the way we can begin by creating our new configuration file.

We will create and edit the new file with the following command:

```
sudo nano /etc/dnsmasq.conf
```

**18.** To this file add the following lines, these lines basically tell the **dnsmasq** service how to handle all the connections coming through.

```
interface=wlan1       # Use interface wlan0
listen-address=192.168.220.1   # Specify the address to listen on
bind-interfaces       # Bind to the interface
server=8.8.8.8        # Use Google DNS
domain-needed         # Don't forward short names
bogus-priv            # Drop the non-routed address spaces.
dhcp-range=192.168.220.50,192.168.220.150,12h # IP range and lease time
```

Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

## Setting up the Wi-Fi Extender

**19.** Next, we need to configure our Raspberry Pi so that it will forward all traffic from our **wlan1** connection over to our **wlan0** connection.

First, we must enable it through the **sysctl.conf** configuration file, so let's begin editing it with the following command:

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

**20.** Within this file you need to find the following line, and remove the **#** from the beginning of the line.

**Find:**

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

**Replace with:**

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

Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

**21.** Now since we are impatient and don't want to wait for it to enable on next boot we can run the following command to activate it immediately:

```
sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

**22.** With IPv4 Forwarding now enabled we can configure a NAT between our **wlan0** interface and our **wlan1** interface. These changes will forward all traffic from our access point over to our ethernet connection.

Run the following commands to add our new rules to the iptable:

```
sudo iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
sudo iptables -A FORWARD -i wlan0 -o wlan1 -m state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i wlan1 -o wlan0 -j ACCEPT
```

**23.** Of course, iptables are flushed on every boot of the Raspberry Pi so we will need to save our new rules somewhere, so they are loaded back in on every boot.

To save our new set of rules run the following command:

```
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

**24.** Now with our new rules safely saved somewhere we need to make this file be loaded back in on every reboot. The most straightforward way to handle this is to modify the **rc.local** file.

Run the following command to begin editing the file:

```
sudo nano /etc/rc.local
```

**25.** Now we are in this file. We need to add the line below. Make sure this line is above **exit 0**. This line reads the settings from the **iptables.ipv4.nat** file and loads them into the iptables.

Find:

```
exit 0
```

Add **ABOVE**:

```
iptables-restore < /etc/iptables.ipv4.nat
```

Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

## Setting up the Wi-Fi Extender - Continued

**26.** Finally, all we need to do is start the two services and enable them in systemctl.

To start these up on our Raspberry Pi we just need to use the following two commands.

```
sudo service hostapd start
sudo service dnsmasq start
```

**27.** Now you should finally have a fully operational Raspberry Pi Wireless Access Point. You can ensure this is working by using any of your wireless devices and connecting to your new access point using the SSID and WPA Passphrase that was set earlier on in the tutorial.

To ensure everything will run smoothly its best to try rebooting now. Rebooting will ensure that everything will successfully re-enable when the Raspberry Pi is started back up.

Run the following command to reboot the Raspberry Pi:

```
sudo reboot
```

## Setting up the Wi-Fi Extender with a VPN

This tutorial is fully compatible with our VPN access Point tutorial. However, there is one small change you will have to make in **step 13**, rather than using the commands showcased there, run the commands below.

The main change you will see here is that instead of redirecting the traffic from wlan0 through the tunnel we will be redirecting the traffic from our **wlan1** connection to the tunnel.

```
sudo iptables -t nat -A POSTROUTING -o tun0 -j MASQUERADE
sudo iptables -A FORWARD -i tun0 -o wlan1 -m state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i wlan1 -o tun0 -j ACCEPT
```

The rest of the VPN access point tutorial can be done without any other changes.

# TOR Access Point

## Project Description

In this tutorial, we will be teaching you how to set up a basic Raspberry Pi TOR Access Point. Before you get started with this tutorial, however, you must have already completed our Raspberry Pi Wireless Access Point tutorial as this tutorial is an extension of that.

Within this tutorial, we will show you how to download and install tor and set it up in a way that we can redirect our Wireless Access Point traffic through it. This will mean that any device that is connected to the Raspberry Pi's wireless access point will automatically have its traffic pushed through the tor network.

The Raspberry Pi provides a very cheap and power efficient way of setting up a TOR access point. It also has the bonus of being incredibly easy to move around, meaning you can take your TOR access point with you anywhere you go.

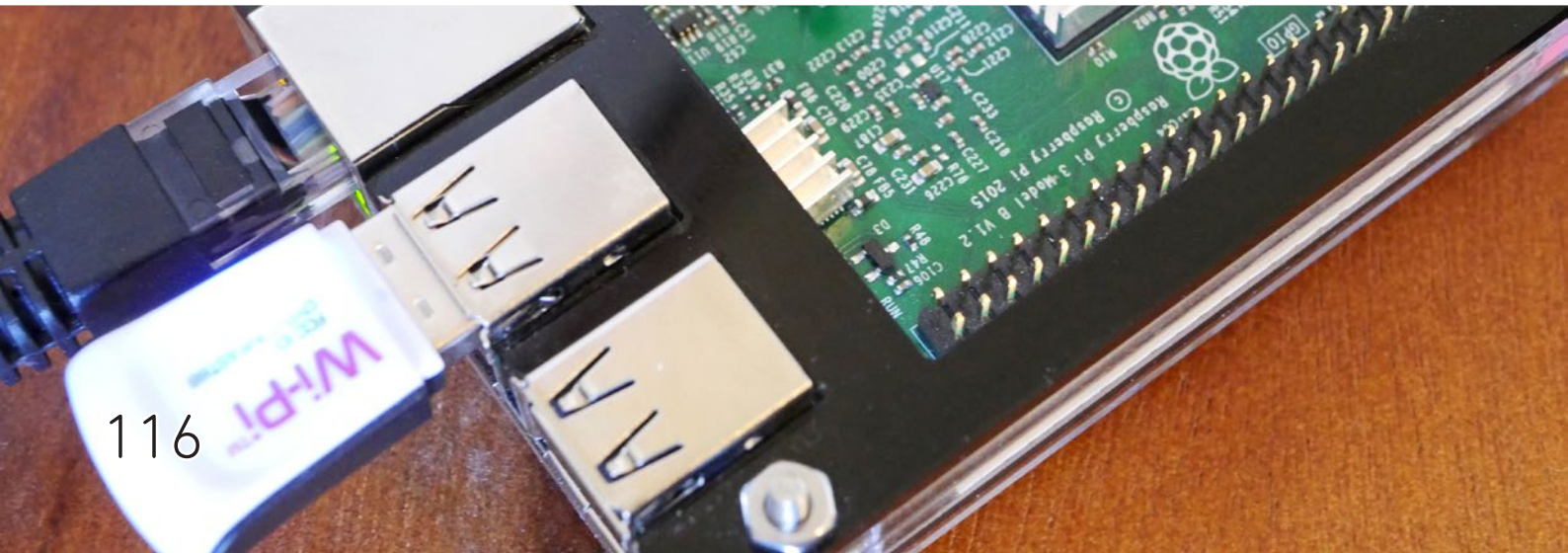
## Equipment

**Required**

- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Ethernet Network Connection
- WiFi (USB or built-in module on the Pi 3)

**Optional**

- Raspberry Pi Case



## Setting up the TOR Access Point

To set up our TOR Access Point you will first have to of followed our Wireless Access Point tutorial, as this will set up your Raspberry Pi perfectly for this tutorial. You can find this tutorial in the book, a few pages back.

**1.** We need first to make sure we are running up to date software before we set up our TOR Access Point. To do this, we can run the following two lines in the terminal:

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

**2.** With the Raspberry Pi now freshly updated we can get along with installing TOR itself, this is easily done by running the following command into terminal:

```
sudo apt-get install tor -y
```

**3.** Now that we have installed tor itself, lets begin by modifying its configuration, lets open up the file for this command:

```
sudo nano /etc/tor/torrc
```

**4.** To this file, add the following configuration just under the FAQ notice. These lines will configure TOR to run on port 9050 and port 53.

```
Log notice file /var/log/tor/notices.log
VirtualAddrNetwork 10.192.0.0/10
AutomapHostsSuffixes .onion,.exit
AutomapHostsOnResolve 1
TransPort 9040
TransListenAddress 192.168.220.1
DNSPort 53
DNSListenAddress 192.168.220.1
```

Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

**5.** With TOR now setup, we need to flush the ip tables, we can do this by running the following two commands:

```
sudo iptables -F
sudo iptables -t nat -F
```

**6.** With the IP Tables now flushed we can now install our new IP Tables. This will route all the traffic incoming from the wlan0 connection through to our TOR connection that is running through port 53.

```
sudo iptables -t nat -A PREROUTING -i wlan0 -p tcp --dport 22 -j REDIRECT --to-ports 22
sudo iptables -t nat -A PREROUTING -i wlan0 -p udp --dport 53 -j REDIRECT --to-ports 53
sudo iptables -t nat -A PREROUTING -i wlan0 -p tcp --syn -j REDIRECT --to-ports 9040
```

**7.** With our new iptables rules in place we will want to store this into the file we setup in our wirless access point, this will ensure the new IP Tables are loaded instead.

```
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

## Setting up the TOR Access Point

**9.** We can check to see if the log file has now been created and permissions set correctly by utilizing the following command:

```
ls -l /var/log/tor
```

**8.** Now lets create our log file, this will be handy for tracking problems. To do so run the following commands:

```
sudo touch /var/log/tor/notices.log
sudo chown debian-tor /var/log/tor/notices.log
sudo chmod 644 /var/log/tor/notices.log
```

**10.** Now we can finally fire up the tor service.

```
sudo service tor start
```

**11.** With the tor service started we can check the service is running by using the following command, if anything has gone wrong you will see a big FAIL notice appear.

```
sudo service tor status
```

**12.** Now finally, let's make the tor service start on boot, this will ensure that the traffic will always be routed through it. Do this with the following command:

```
sudo update-rc.d tor enable
```

https://www.torproject.org
Home
Anonymity Online
rotect your privacy. Defend yourself
gainst network surveillance
affic analysis.
Download
at is Tor?
free software and an open net
helps you defend against traff
ysis, a form of network surveil
atens personal freedom and
dential business activities an
ionships, and state security.
Learn more about T
Raspberry Pi

# Captive Portal

## Project Description

In this tutorial, we will be showing your how-to setup a Raspberry Pi Captive Portal by utilizing the nodogsplash software. You can check out the software we will be using by going to https://github.com/nodogsplash.

A Captive Portal is a screen that will be shown initially to anyone who connects to your Wi-Fi Access Point. Before they can begin utilizing the Wi-Fi connection, they will need to click a link or button.

You can display whatever you want to the captive portal, so it can be highly useful if you're going to lay out some ground rules for using your Wi-Fi access point before a user starts using it.

To complete this tutorial, you will first of had to of completed our Wi-Fi Access Point tutorial, this is also fully compatible with our VPN Access Point tutorial.

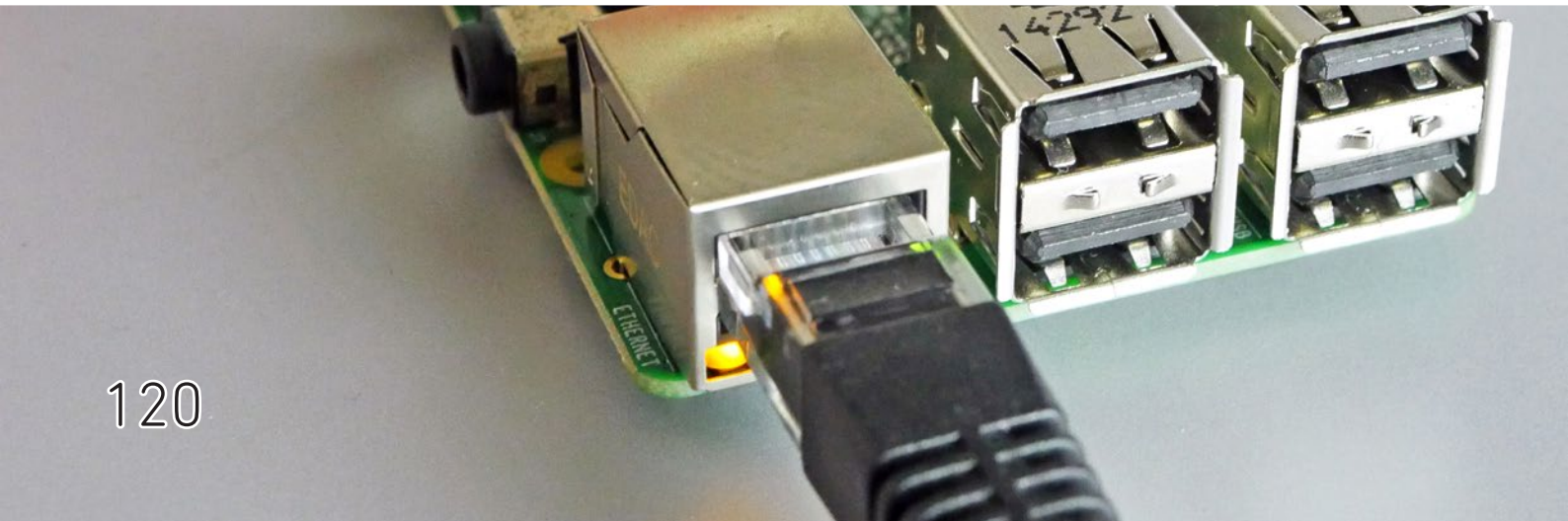
## Equipment

**Required**

- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Ethernet Network Connection
- WiFi (USB or built-in module on the Pi 3)

**Optional**

- Raspberry Pi Case



## Setting up the captive portal

Before we proceed with setting up our captive portal, you will of first of had to follow our Wireless Access Point tutorial. This tutorial will build on top of the foundations we set up in that tutorial and will show you how to reconfigure it to show a captive portal.

**1.** We need first ot make sure we are running up to date software before we set up our captive portal. To do this, we can run the following two lines in the terminal:

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

**2.** With the Raspberry Pi now freshly updated we can get along with setting up our captive software, for this tutorial we will be utilizing the software called **nodogsplash**, it's a somewhat lightweight.

Let's grab this software off their git by cloning its repository with the following couple of commands.

```
cd ~
git clone https://github.com/nodogsplash/nodogsplash.git
```

**3.** Now we have two different steps to do, if you are running **Raspbian Jessie or earlier** then follow **step 3a**, otherwise if you are running **Raspbian Stretch or later** follow **step 3b**.

**3a (Raspbian Jessie and earlier).** For this step, all we need to do is change the main branch of the repository we cloned to an earlier version of Nodogsplash.

The reason for this is that the main one requires a package that isn't available in Raspbian Jessie and earlier.

Run the following commands to switch the branch.

```
cd ~/nodogsplash
git checkout v1
```

**3b (Raspbian Strech and later).** For those who are running Raspbian Stretch, we can stay on the main branch.

However, we will need to install a package that Nodogsplash requires to compile.

Run the following command to install the package:

```
sudo apt-get install libmicrohttpd-dev
```

**4.** Once the software has finished being cloned and we have either changed branch or installed the package, we can now proceed with compiling and installing the software.

Run the following three commands to setup and install the software.

```
cd ~/nodogsplash
make
sudo make install
```

## Setting up the captive portal

**5.** With nodogsplash now installed to the system, we can now make some modifications to the configuration file.

Type in the following command to modify the configuration file:

```
sudo nano /etc/nodogsplash/nodogsplash.conf
```

**6.** To this file we need to add the following information, this tells what interface nodogsplash should show up on and what address it should be listening on.

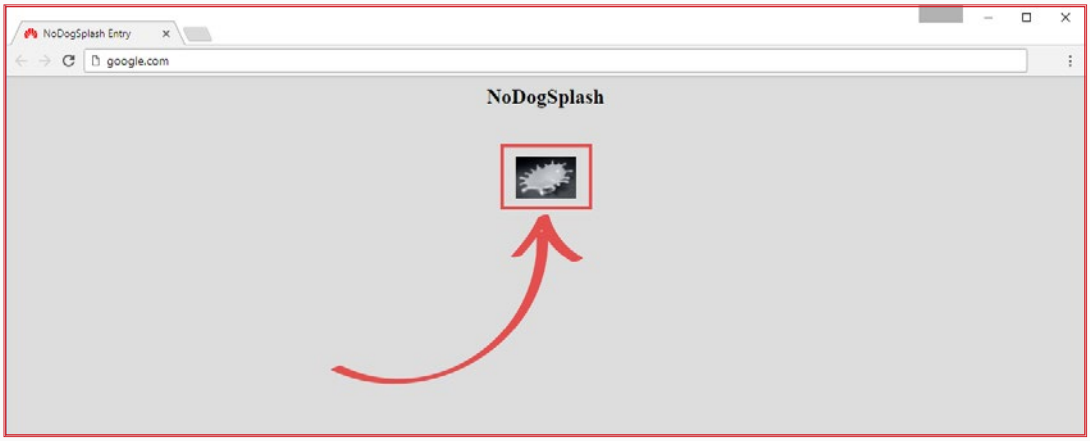
```
GatewayInterface wlan0
GatewayAddress 192.168.220.1
MaxClients 250
ClientIdleTimeout 480
```

Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

**7.** With our changes to Nodogsplash's configuration saved, we can start up the software. Starting the software is very simple to do, just run the following command in terminal to start it up.

```
sudo nodogsplash
```

**8.** Now if you connect to your hotspot, you should be greeted with the screen below. Clicking the splash image in the middle will allow you to browse the internet, but until then the captive screen will be continually shown.




**9.** Now that we have confirmed that the splash page is working, we should now make nodogsplash startup on launch. We can achieve this by modifying the **rc.local** file.

Run the following command to begin editing it:

```
sudo nano /etc/rc.local
```

**10.** Within this file we need to find **exit 0** and add nodogsplash above that line..

**Find:**

```
exit 0
```

**Add above:**

```
nodogsplash
```

Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

## Setting up the captive portal

**12.** You can also further customize Nodogsplash by modifying the configuration file that we added our Gateway Interface into. You can find documentation on this by going to their website at: http://nodogsplash.readthedocs.io/en/latest/customize.html

You can modify the configuration file by typing in the following command:

```
sudo nano /etc/nodogsplash/nodogsplash.conf
```

**11.** You should now hopefully have a fully operational wireless access point with a fully working captive portal. If you want to change the default portal, you can edit it by modifying the file located at: **/etc/nodogsplash/htdocs/splash.html**.

You can run the following command to begin modifying the file from within your terminal:

```
sudo nano /etc/nodogsplash/htdocs/splash.html
```

This file will explain the variables that are automatically available in the file, also take note of the **<a** link, as this is needed to authenticate the user.

# Network-Wide Adblocker

## Project Description

In this tutorial, we will be setting up a Raspberry Pi Network-wide ad blocker. A network-wide ad blocker works by acting as your DNS server.

What the network-wide ad blocker does is act as your DNS server. Your computers our routers will connect to it for DNS requests.

These DNS requests will then be filtered by the Raspberry Pi. Any DNS requests that match any that match a known ad host will be rejected meaning the connection will never meet your computer. Therefore the ad will never be loaded by the devices.

Network-wide ad blocking is a useful tool for stopping ads from appearing on devices that don't have any easy to use adblock tool. It's also a handy way to improve your network performance and bandwidth usage as the number of files downloaded during normal web browsing will be reduced.

It also has the added side effect of potentially improving the security and the privacy of your network, as some sites use relatively dodgy ad providers.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

**Optional**
- Raspberry Pi Case
- USB Keyboard
- USB Mouse

# Installing the Network-wide ad blocking software.

**1.** To set up our Network-wide Adblocker, we are going to utilize a piece of software called PiHole. PiHole is a software package that handles everything. It sets up the DNS server, handles all the DNS requests and filters them.

It also provides a fancy web frontend to the whole system, allowing you to see the statistics in real time and make changes to PiHoles configuration, including whitelisting or blacklisting specific URL's.

Another advantage to utilizing PiHole is that it has an easy to use installer. Meaning we don't have to mess around with trying to install and configure several different packages.

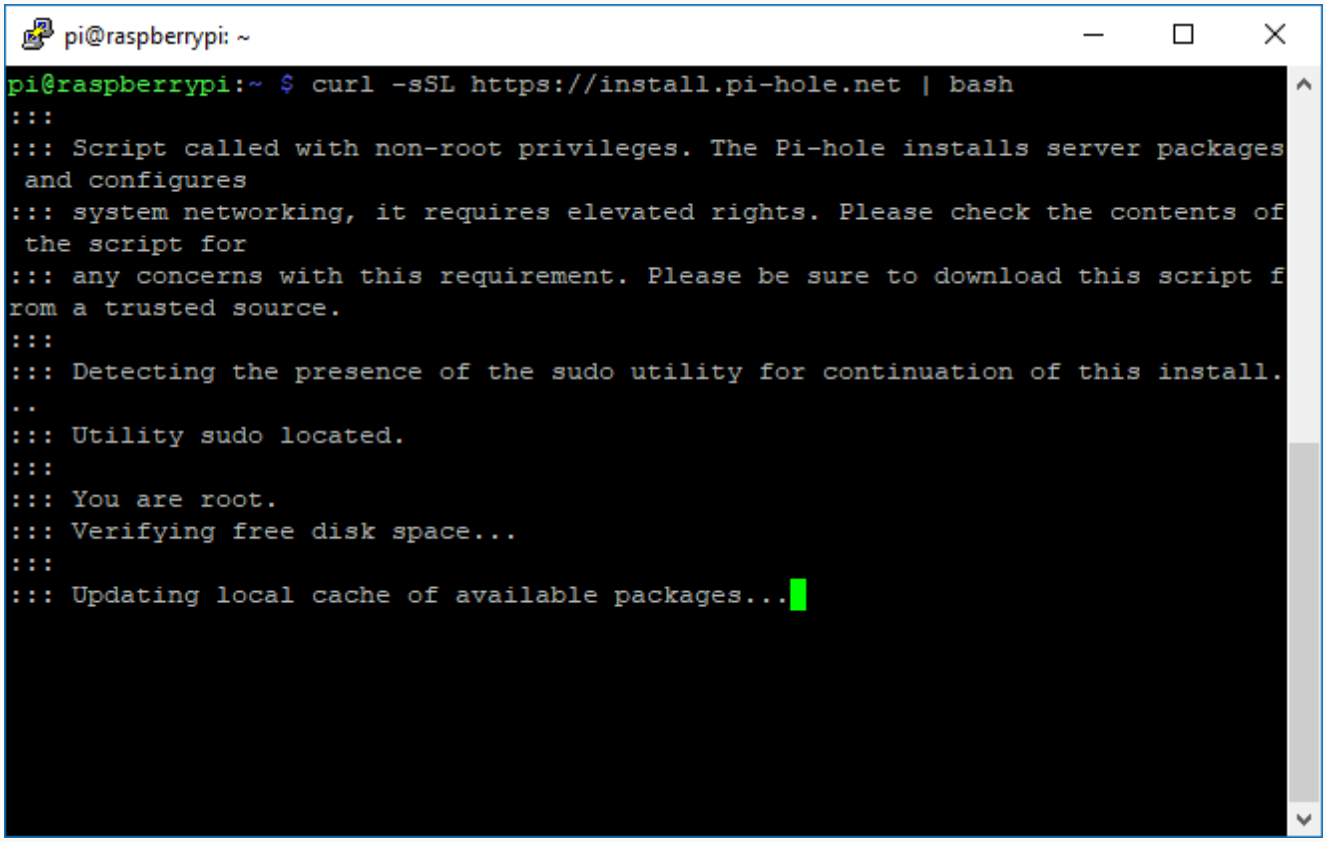
We can run this single command to download and run the install script.

While we know PiHole is safe, if you want to check the script that we will be running yourself, go to the following address: https://install.pi-hole.net in your favorite web browser.

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

**2.** Once you have run the command, it will download and run the script that begins the process of setting up your device for PiHole.

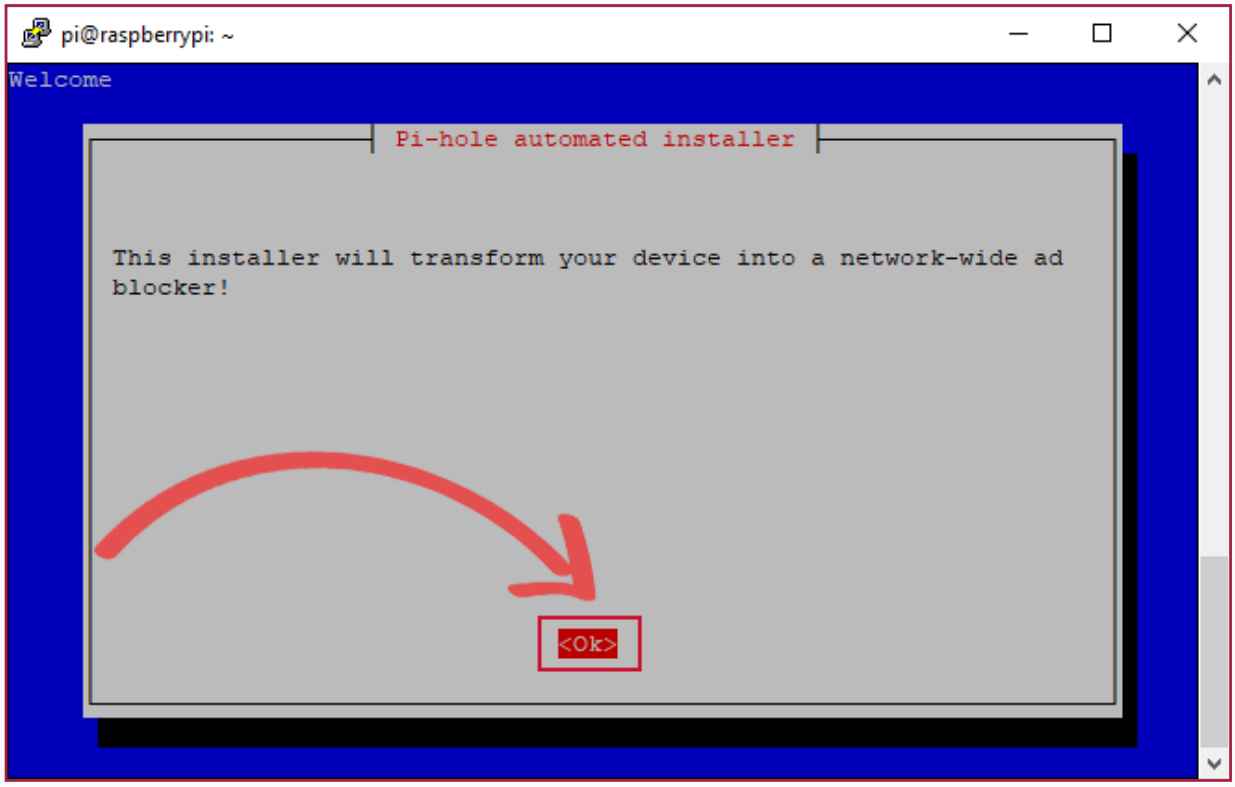
A few of the tasks it does is verify the amount of available free disk space, update the package list, and several other things.



# Installing the Network-wide ad blocking software

**2.** Once you have run this command, it will begin the setup of PiHole, after a few initial setup steps you will be greeted with the following install screen.

Just select **<Ok>** to the next few screens to proceed.

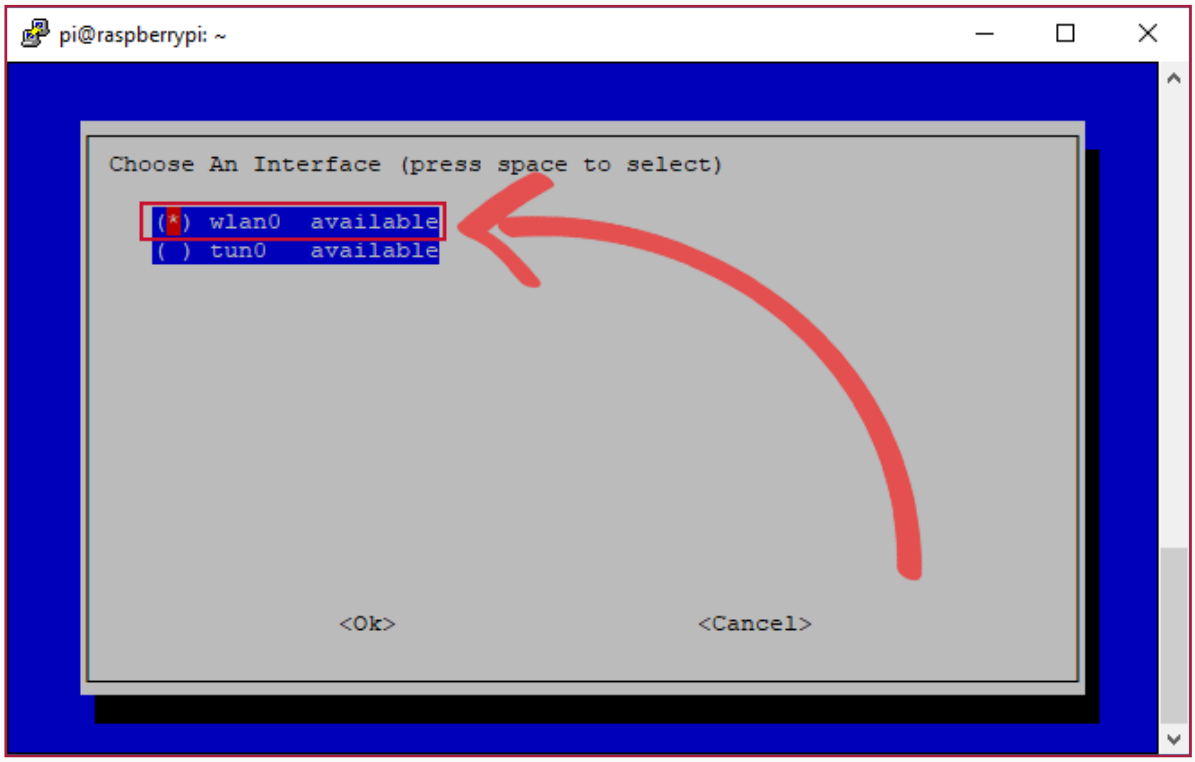



**3.** The next important screen to pay attention to is the one below. This screen is warning you that PiHole will need a static IP address to function correctly.

**4.** Now we need to select the interface we want PiHole to operate over.

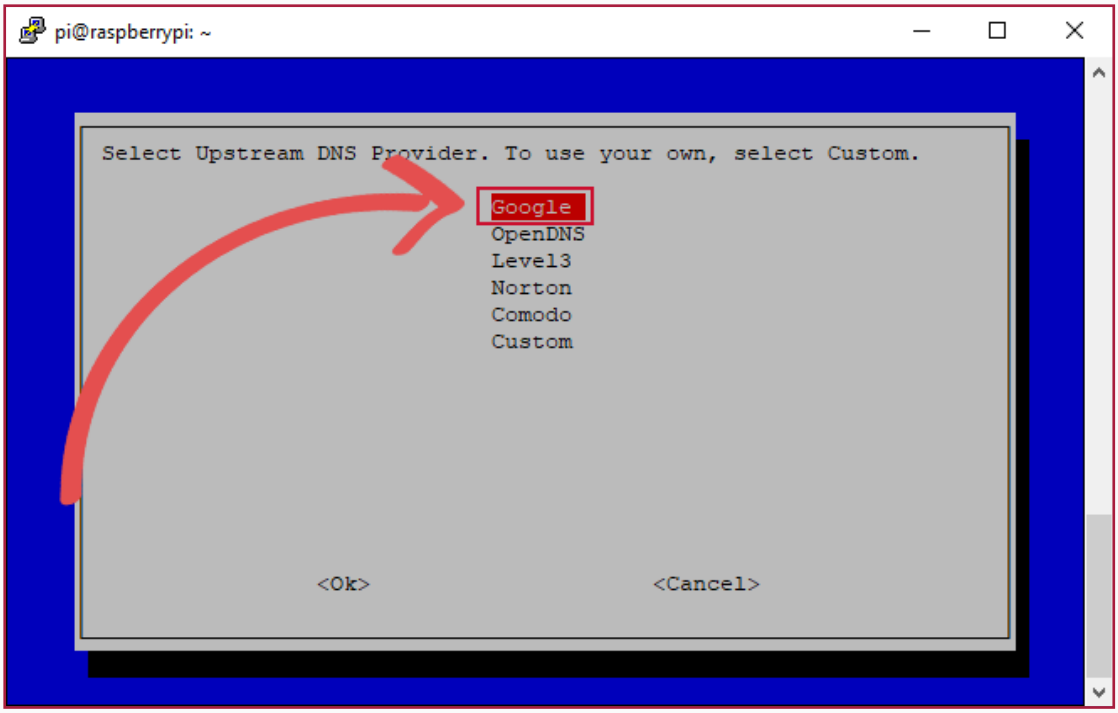
in our case we are going to select **wlan0** if you you're going to choose a different option, use the **arrow keys** to move to that option and press **space** to select it.

If you are happy with your selection press **enter** to proceed.



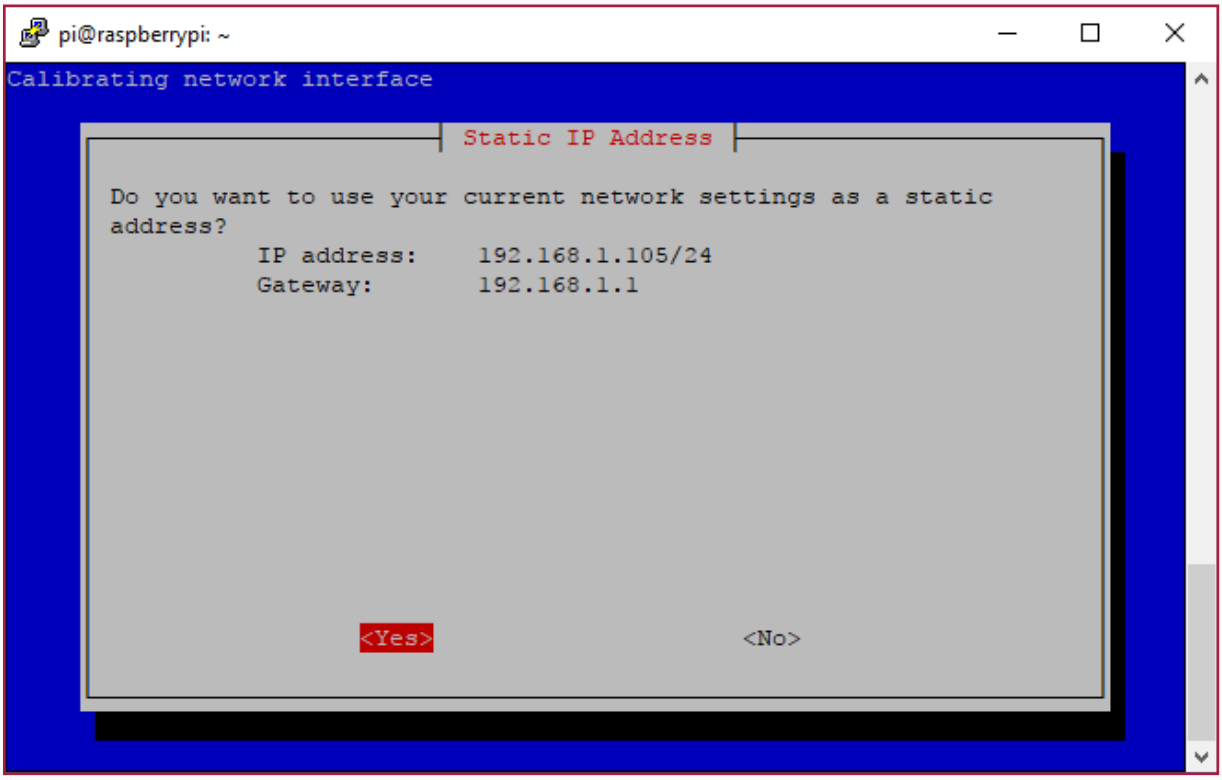

**5.** Now we have to select the Upstream DNS provider we want to use. We are going to just stick to using Google as it is one of the most reliable DNS providers.

Press **enter** when you are happy with your choice.



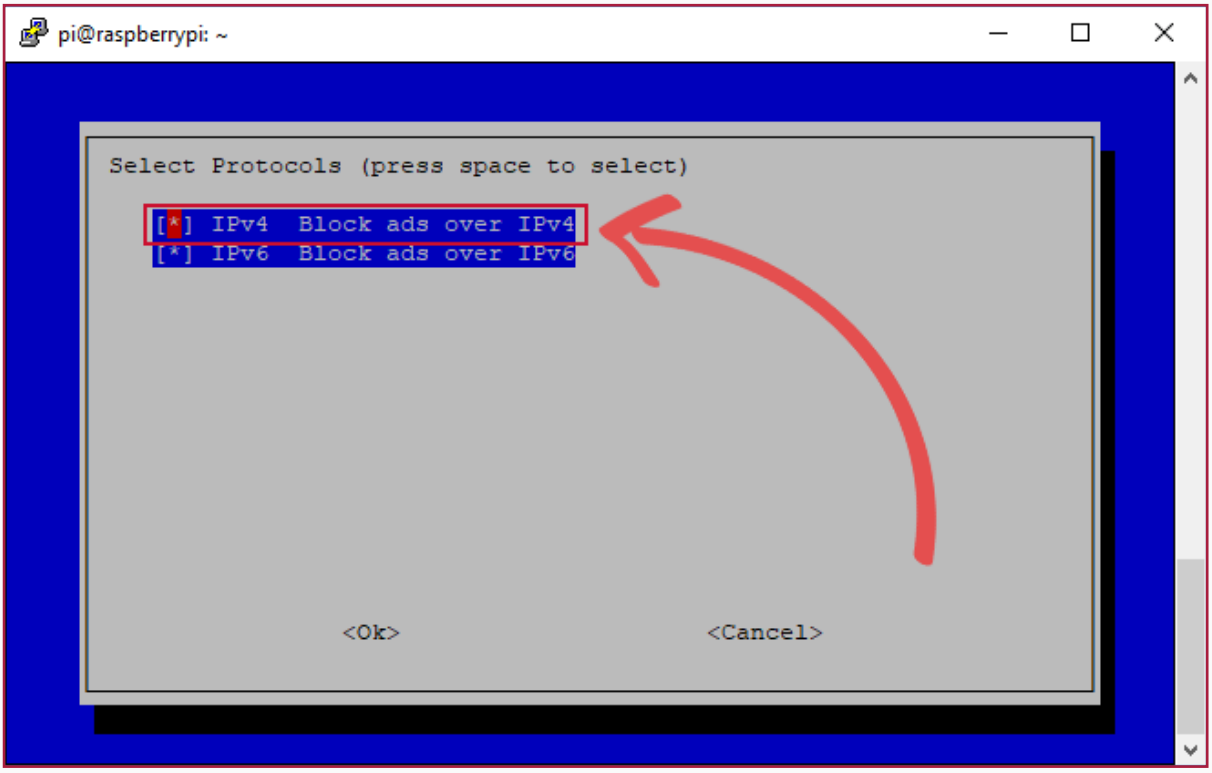
**6.** Now we have to select the protocols we want to utilize. It's usually best just to leave both options enabled.

Press **enter** to continue.




**7.** You will now be asked if you want to use your current networks settings as a static address.

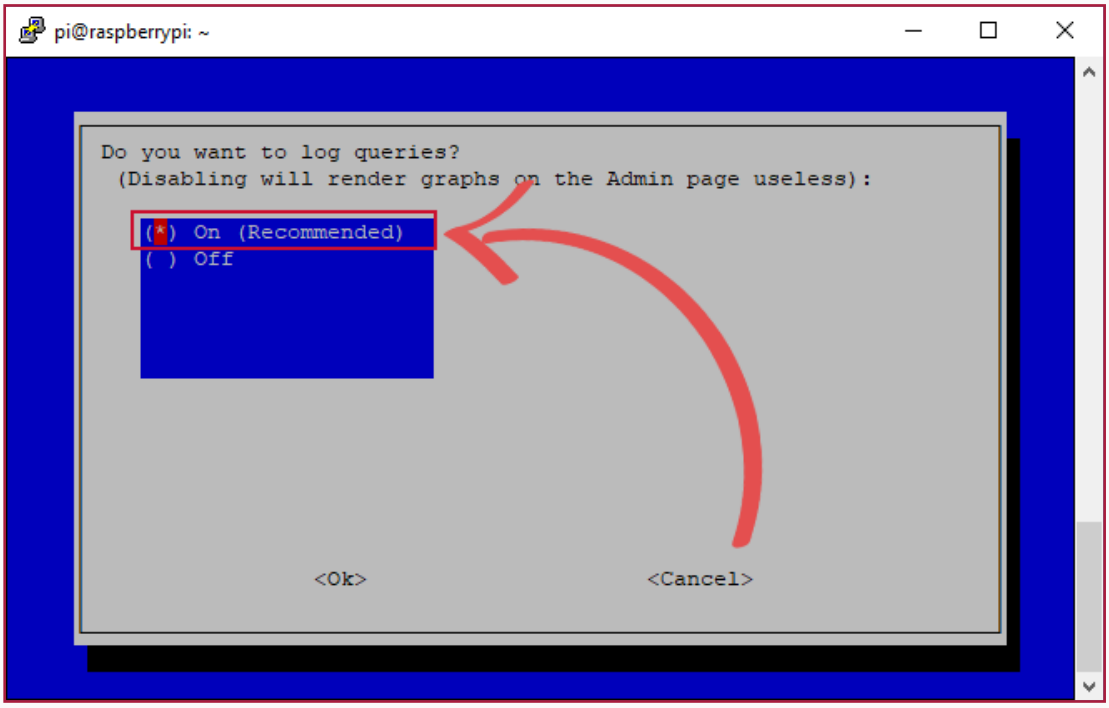
If you are happy with the information displayed then select **<yes>** to the next two screens continue.



**8.** Here you can disable the logging abilities of PiHole useless.

However, we recommend keeping it on as it allows you to keep track of everything that PiHole is doing and gives you some neat statistics.

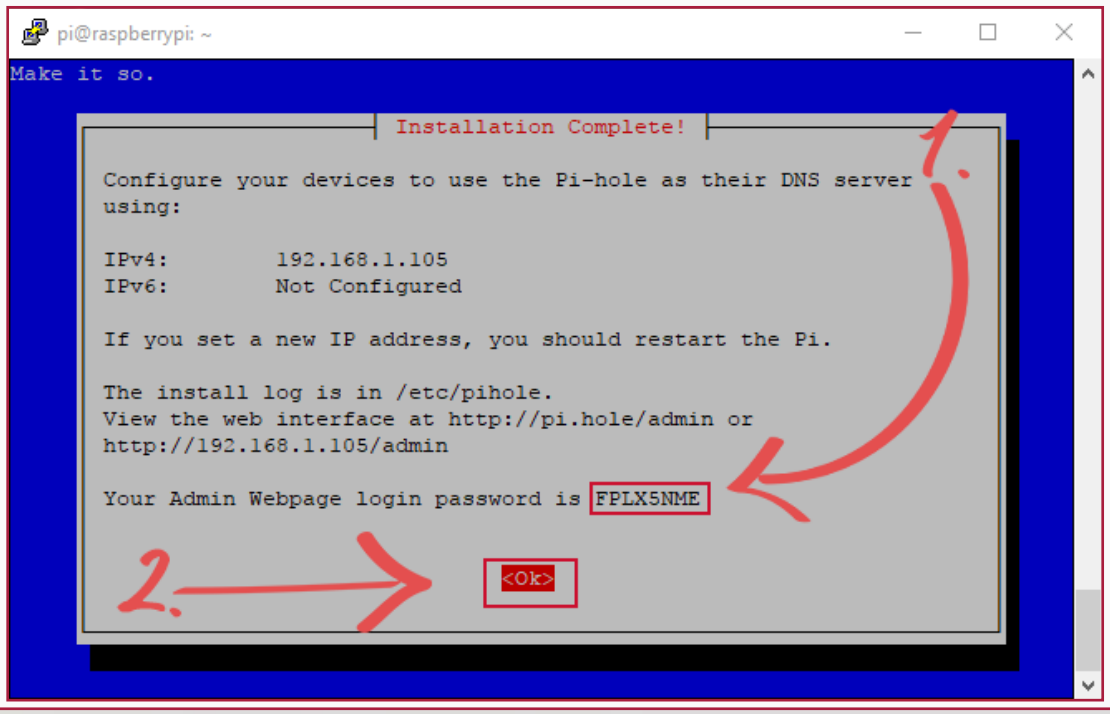
Press **enter** to continue.




**9.** That is the end of everything that needs configuring, after the installation screen you will be presented with this final screen.

Make sure you write down the password displayed on the screen (**1.**)

Once done press **enter** to continue (**2.**)



# Connecting your network to PiHole

There are two different methods for setting up the network-wide ad blocker on your network.

The first of these two options is the easiest and will extend the coverage of the network-wide ad blocker to all your devices. This option is to change your routers DNS settings to point towards the Raspberry Pi. We recommend this for setting up the Network-wide ad blocker.

The second is to set the DNS setting for each of your devices, this is more difficult to setup and something you must do every time you add a new device to your network, but can be useful if you only want specific devices running through the adblocker.

## Setting routers DNS settings

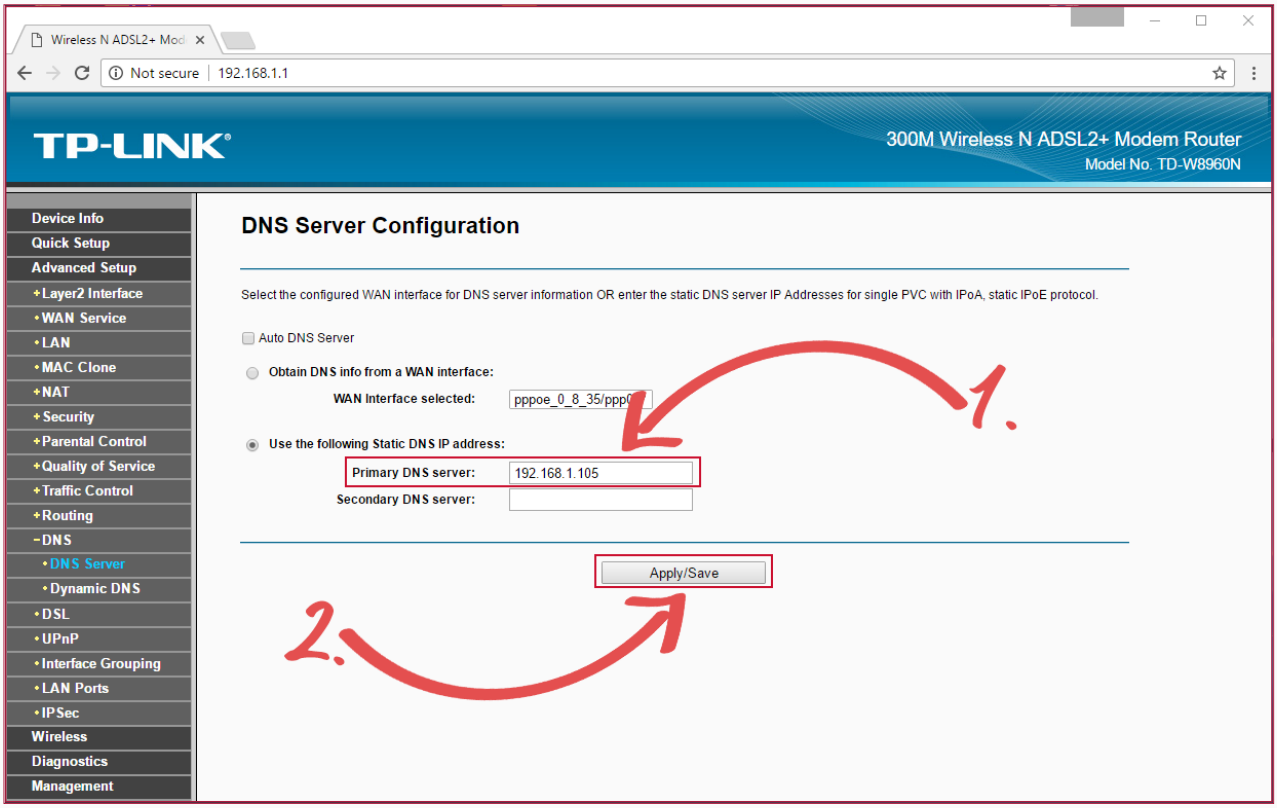
Changing the DNS settings is different for every router, but we will go through the steps we used for our router. First, we need to go to the routers admin page, must routers reside on the http://192.168.1.1 local IP address.

If you are asked for a username and password but are unsure of what it is, then try using **admin** for the username and **admin** for the password.

Once you are logged in, look for anything mentioning **DNS server**, or **DHCP server**. For my **TD-W8960N** router, these resided under "**Advanced Setup**" then under "**DNS**" then "**DNS Server**".

Within this screen, set the primary DNS server to the IP address of your Raspberry Pi, for instance, mine is **192.168.1.105**.

Make sure it is the only DNS server that is set, as it handles all upstream DNS services itself, and adding alternatives could break the functionality of the ad blocker.



## Setting devices DNS settings

# 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.

# 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 drop-down 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 address**
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**.

## Setting devices DNS settings

# Mac OS X

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**

# iPhone

1. From the iPhone's home screen, tap **Settings**.
2. Tap **Wi-Fi**. The available wireless networks in range of your iPhone should appear.
3. Find your wireless network in the list, and then **tap the arrow**.
4. Tap the DNS field.
5. Delete the current DNS servers, and enter your Raspberry Pi's IP Address.

# PS3

1. Go to **Settings**
2. Go to **Network Settings**
3. In here then select **Internet Connection Settings**
4. Now we need to choose **Custom**
5. Set all your options until you get to the screen displaying your IP address and DNS settings.
6. Set DNS Settings to **manual**.
7. Now set your primary DNS to the IP address of your Raspberry Pi.

# PS4

1. Go to **Settings**
2. Go to **Network**
3. Go to **Set up Internet connection**
4. Select **Wifi**/**LAN** depending on what you use to connect.
5. Select **Custom**
6. Set **DNS Settings** to **Manual**
7. Set the **Primary DNS** to the IP address of your Raspberry Pi, and make sure the secondary DNS is empty.

## Setting devices DNS settings

# Wii U

1. From the **Wii U Menu**, select "**System Settings**."
2. Select the **Internet** Icon
3. Tap **Connections** in the top right corner.
4. Select the connection you wish to configure.
5. Tap "**Change Settings**."
6. Select **Primary DNS Server**
7. Tap the arrow on the right and select "**DNS**."
8. Tap "**Don't Auto-Obtain**" and enter the DNS info as follows:
9. Tap "**Confirm**" to return to the manual settings screen.
10. To save these settings, select "**Save**" or press the **B Button**, and then select "**Save**."

# Xbox 360

1. In the **Settings** tab select **System**
2. Now select **Network Settings**
3. Choose the network you are currently using.
4. Select **Configure Network**
5. Highlight and select **DNS Settings**
6. Select **Primary DNS Server**
7. Select **Manual**.
8. Now enter the IP address of your Raspberry Pi, and then select **done**.
9. Repeat the procedure for additional network connections you want to change.

# Xbox One

1. Go to the **Dashboard**.
2. Press the **options** button on the controller.
3. Go to **Network**.
4. Go to **Advanced Settings**.
5. Select **Manual**.
6. Enter the **Primary DNS Server** as the IP address of your Raspberry Pi.
7. Press **B** to save the new settings
8. Restart the Xbox One.
9. Repeat the procedure for additional network connections you want to change.

# A quick look into the PiHole web interface

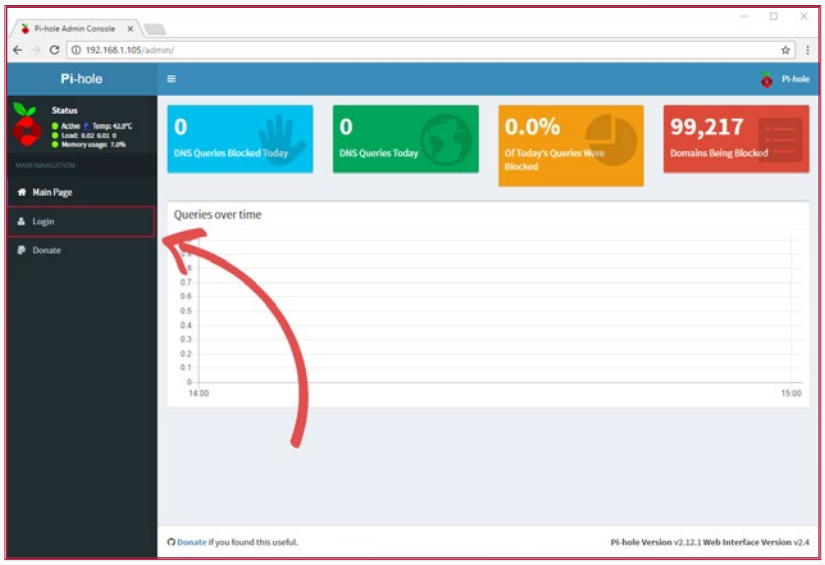
**1.** To get to PiHoles web interface, all we need to do is go to the following web address in your favorite web browser.

Make sure you swap out **192.168.1.105** with the IP address that was displayed in the last step of the previous section.

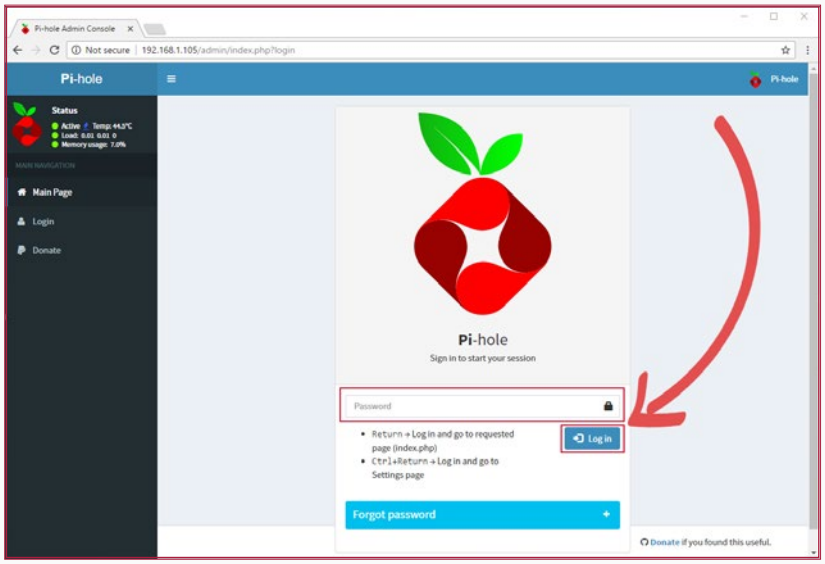
```
http://192.168.1.105/admin/
```

**2.** You will be greeted with the following screen, at the moment this will not show any real stats as we haven't hooked it up to anything.

For now, let's just go straight to the Login page, you can find this by clicking "**Login**" in the sidebar.




**3.** On here you need to enter the password that you got at the end of the installation of PiHole to login.

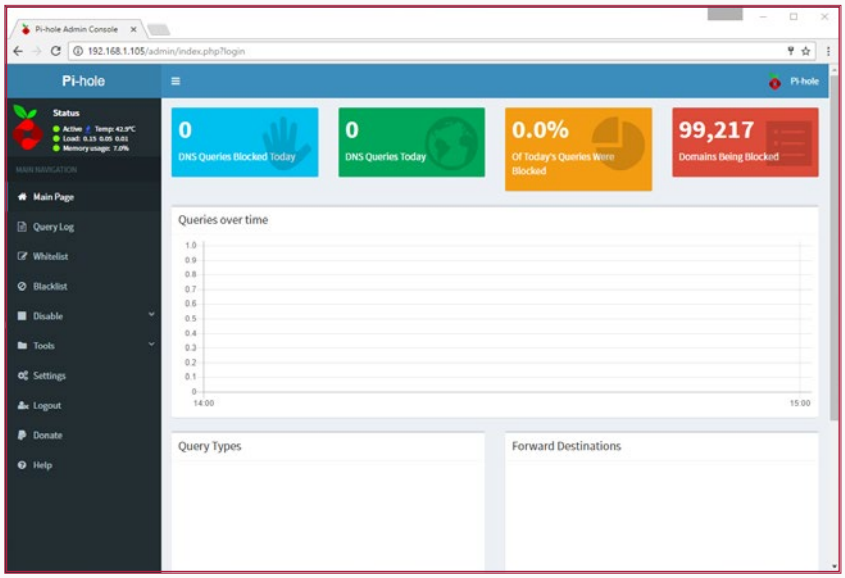



If you have lost the password, you can use the command shown below to set a new one, where new_password is the password you want to set.

```
pihole -a -p new_password
```

## A quick look into the PiHole web interface

**3.** You will now be presented with the admin dashboard. This dashboard is just like the normal one with a couple more graphs. Below we will go through all the available options.




• **Query Log -** The query log shows all the most recent queries that have been made to the DNS server. It is an easy way to find a recent address that may have been loaded and blacklist them.

• **Whitelist -** This screen allows you to add certain domain names to PiHoles whitelist. This whitelist means those websites will no longer be blocked by the DNS server. However, if that domain is blocked by a wildcard, then it will continue to be blocked.

• **Blacklist -** This screen allows you to add certain domain names into PiHoles blacklist, this means those websites will be blocked by the DNS server.

• **Disable -** These options allow you to temporarily or permanently disable the blocking functionality of PiHole, it will continue to function as a normal DNS server while disabled but will no longer block DNS requests.

• **Tools**

• **Update Lists** - This tool allows you to trigger an update of the ad blocking lists, it is useful for ensuring you are running the latest versions of the ad block lists.

• **Query Adlists** - This tool allows you to find out if a certain URL is being included in the ad blocking lists, it is useful for checking why a certain URL might be getting blocked.

• **Tail pihole.log** - This tool shows the last lines of the **pihole.log** file and continually updates live, this is useful for checking to see what PiHole is doing to requests.

• **Settings** - This section contains some configurable settings, this allows you to change the way your DNS works, what upstream DNS providers you want to use among several other options. You can even enable DNSSEC in the options.

For the average user, you will not need to change these options unless you made a mistake in the initial installation.

# Build your own NAS

## Project Description

A Raspberry Pi NAS or network attached storage is the perfect way to have files available to anyone within your local network

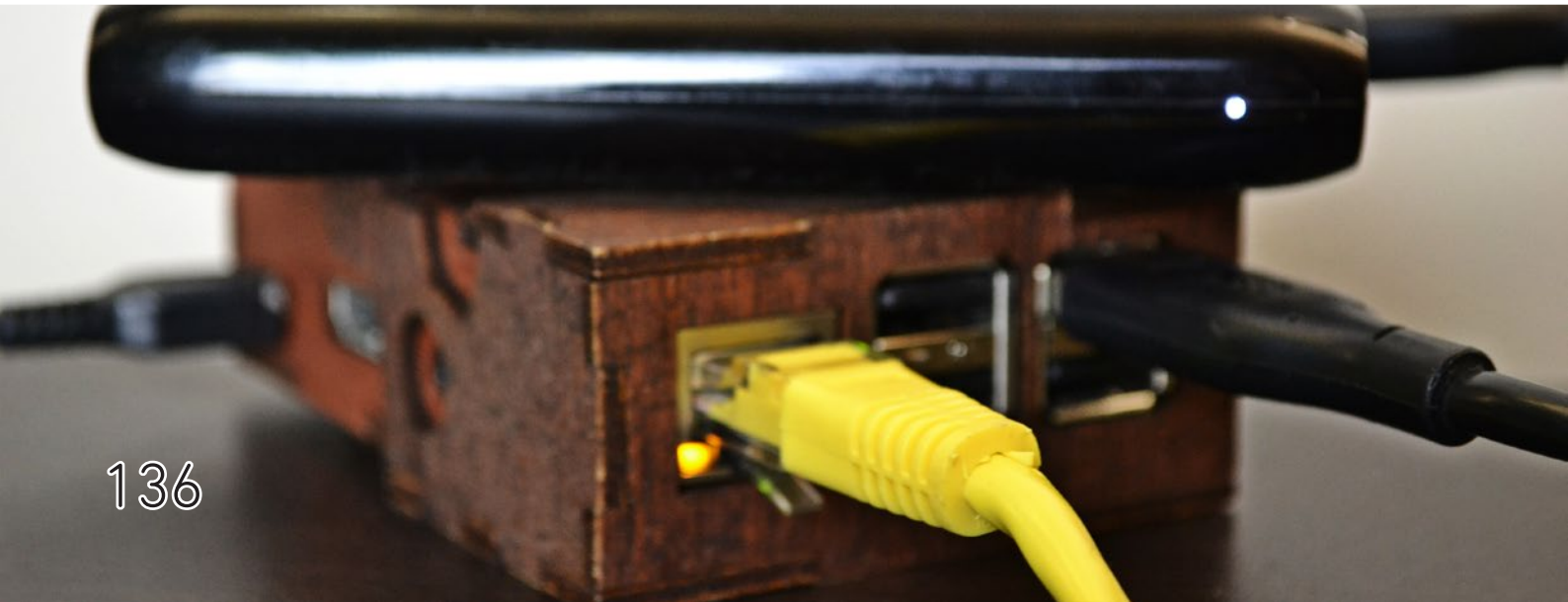
Once you have setup your first drive correctly you will find this process to be incredibly easy to repeat. You may come across some complications when first setting it up these problems are usually caused by having the permissions set incorrectly.

One other thing that I should mention is that if you're looking for blazing fast speeds then you're unlikely to get them with the Pi. This is due to having to use the USB ports (2.0) to connect a hard drive and the network interfaces. (100 mbps Ethernet or Wi-Fi)

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection (Ethernet Reccommended)

**Optional**
- Raspberry Pi Case
- External Hard drive or USB Drive
- USB Keybaord
- USB Mouse



## Mounting the Hard Drives

**1.** Bring up the Raspberry Pi terminal either by using SSH or loading the terminal on the Raspberry Pi.

**2.** First, let's bring Raspbian up to date by entering the following commands:

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

**3.** We will want to install NTFS support, so lets quickly install the package.

```
sudo apt-get install ntfs-3g
```

**4.** Next, we want to find our external hard drive to do this enter the following command.

```
sudo fdisk -l
```

**5.** There should be two lots of drives that pop up unless you have more than one drive connected.

The first would be the SD card that Raspbian is currently running on (Should be like **/dev/mcbblkop1**) ignore this one.

There should also be another one that looks like **/dev/sda1** this is the drive that we will use.

**6.** Before we mount the drives, we will need to create a directory to mount to.

To do this enter the following command. (Remember to fix up the command with the correct drive name)

```
sudo mkdir /media/NASHDD1
```

**7.** Now before we continue, we will need to create a user to log in as.

To do this enter the following command into the terminal on the Raspberry Pi.

Note: Change the word **pimylifeup** to change the user name.

```
sudo useradd pimylifeup -m -G users
sudo passwd pimylifeup
```

**8.** You will be prompted to enter a password twice, be sure to remember this password as you need it later.

**9.** Next, we need to get the gid and the uid, make sure you write these down as we will need them in the next step. Update pimylifeup with the user you created above.

To retrieve the gid enter the following into terminal:

```
id -g pimylifeup
```

To retrieve the uid enter the following into terminal:

```
id -u pimylifeup
```

Make sure to write down the values from both of those commands, we will need them shortly.

## Mounting the Hard Drives

**10.** Now we need to edit the fstab file so our Pi will automatically mount the drive on boot up and set the correct permissions.

To do this enter the following command on your Raspberry Pi:

```
sudo nano /etc/fstab
```

**11.** Add the following the line to the bottom of the file.

Changing **/dev/sda1** to whatever your hard drive is and changing both uid and gid as appropriate. (The code below is all one line)

```
/dev/sda1 /media/NASHDD1 auto nofail,
uid=enter_uid_here,gid=enter_gid_here,noatime 0 0
```

**12.** Reboot the Raspberry Pi using the following command, and the drives should automatically mount with the correct permissions as the Raspberry Pi starts up.

```
sudo reboot
```

## Setting up the Raspberry Pi Samba Server

In this part, we will download and install the samba package so we can have a Raspberry Pi samba server up and going.

If you want more information on the package and its documentation you can check out the Samba website at https://www.samba.org/samba/docs/.

**1.** First, we will need to install the samba package by entering the following command:

```
sudo apt-get install samba samba-common-bin
```

**2.** Just in case we make any mistakes along the way we should back up the samba config folder.

To do this enter the following command:

```
sudo cp /etc/samba/smb.conf /etc/samba/smb.conf.old
```

**3.** Now let's edit the config file by entering the following:

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

**4.** We need to make several files to the file.

First we need to remove the **#** from the **security = user** line, so it reads like:

```
security = user
```

## Setting up the Raspberry Pi Samba Server

In this part, we will download and install the samba package so we can have a Raspberry Pi samba server up and going. If you want more information on the package and its documentation you can check out the Samba website at https://www.samba.org/samba/docs/.

**1.** First, we will need to install the samba package by entering the following command:

```
sudo apt-get install samba samba-common-bin
```

**2.** Just in case we make any mistakes along the way we should back up the samba config folder.

To do this enter the following command:

```
sudo cp /etc/samba/smb.conf /etc/samba/smb.conf.old
```

**3.** Now let's edit the config file by entering the following:

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

**4.** We need to make several files to the file. First we need to remove the **#** from the **security = user** line, so it reads like:

```
security = user
```

**5.** Next, we will need to add our hard drive to the samba config file. Enter the following to add your hard drive as a share (Update the path if you have something different):

```
[NAS]
comment = NAS Folder
path = /media/NASHDD1
valid users = @users
force group = users
create mask = 0660
directory mask = 0771
read only = no
```

**6.** I'll quickly explain what each of the options shown above means so you can have a better understanding of how the Samba Server settings work.

- **[NAS]**: This is the name of the share (What you will see in the file explorer)
- **Comment**: This is the comment for the associated for share.
- **Path**: Path to the folder you wish to share
- **Valid User**: A list of users that can login to this share.
- **Force Group**: This specifies a UNIX group name that will be assigned to all users connecting to this share.
- **Directory Mask**: This creates a permission mask for all directories created on the drive.
- **Read Only**: This allows you to set the share to be read-only

**7.** Now restart the samba server by entering the following command:

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

**8.** Finally, you will need to connect the user to samba. We can do this by entering the following:

```
sudo smbpasswd -a pimylifeup
```

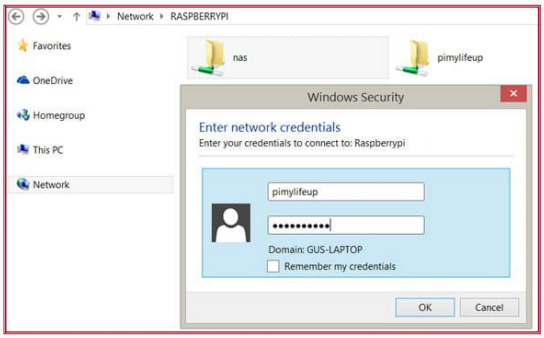
# Testing the Raspberry Pi NAS

Now we have it all setup we will need to test to make sure everything is configured properly. This is very easy to test so I will go through an example of both Windows and Mac.

## Windows

If you have a Windows PC, please do the following to access the Raspberry Pi NAS.

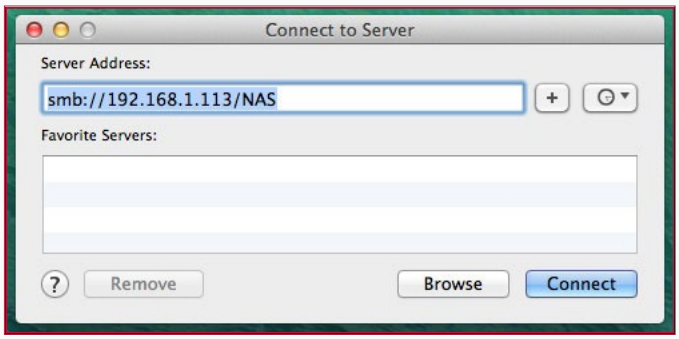
1. On a Windows PC open file explorer.
2. Click on network in the left-hand column.
3. In here open **Raspberry Pi**.
4. Double click on the folder.
5. Enter the credentials.
6. You should now be able to save, edit and delete files in this folder.




## Mac

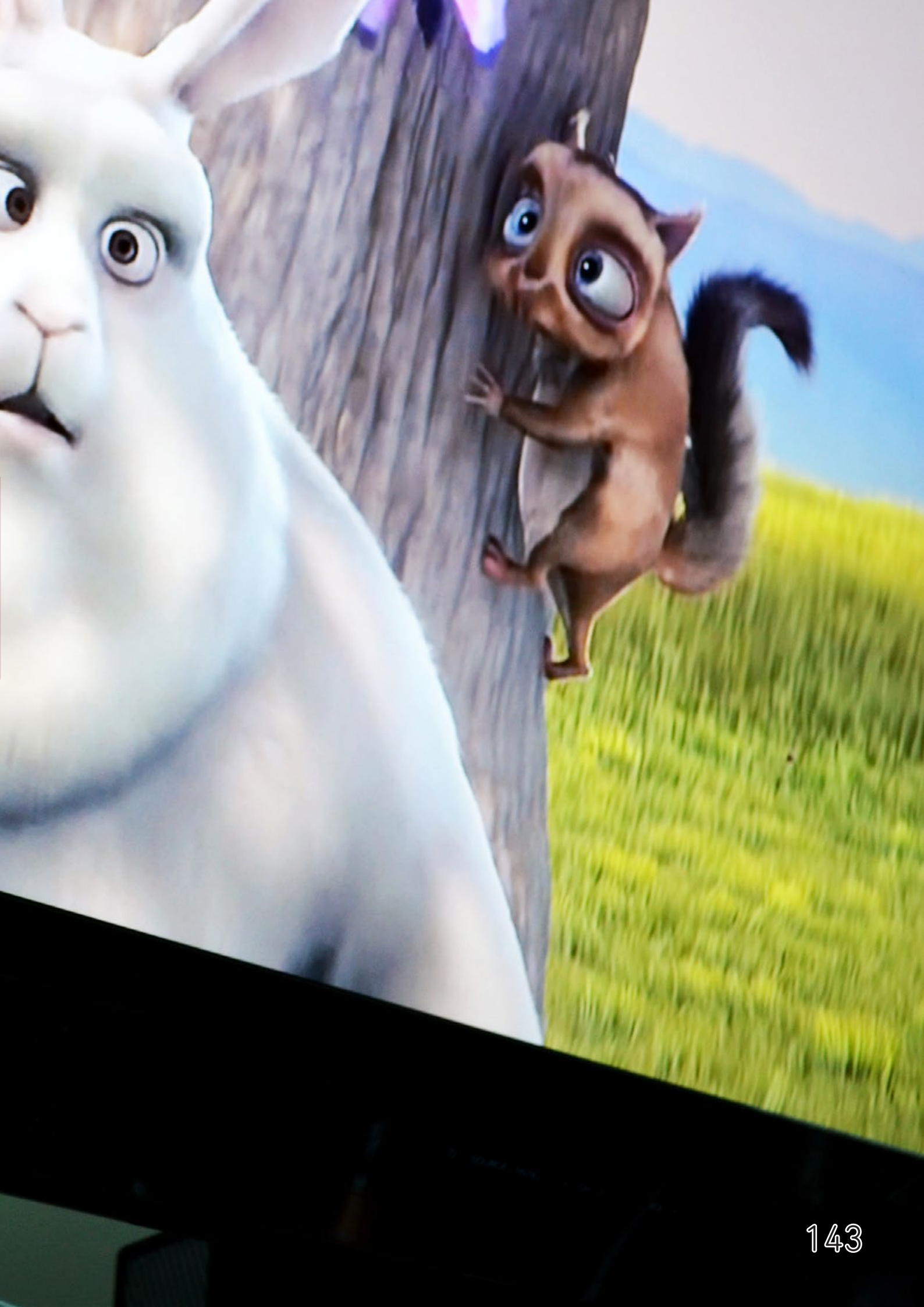
If you have a Mac, please do the following to gain access to the NAS.

1. Click on go in the top bar and navigate to connect to server.
2. In here add the IP address of the Pi or browse to the Raspberry Pi. (You will need to specify the folder)
3. Enter your credentials.
4. You should now be able to save, edit and delete files within this folder.



ADSL

# Multimedia Projects



# Kodi Media Center

## Project Description

Kodi is one of the best ways to turn your Raspberry Pi into a fully-fledged media center. Not only is it a rather feature-packed media center, but it has also gained a fair amount of development put into porting the Kodi to the Raspberry Pi.

There are various "flavors" of Kodi that are available for the Raspberry Pi. We will go into a couple of these in this tutorial.

The equipment we recommend you will you need for this project are listed below. You will probably only need a USB keyboard and mouse for the setup after this you can remotely control XBMC (Kodi) via a tablet and/or web browser.

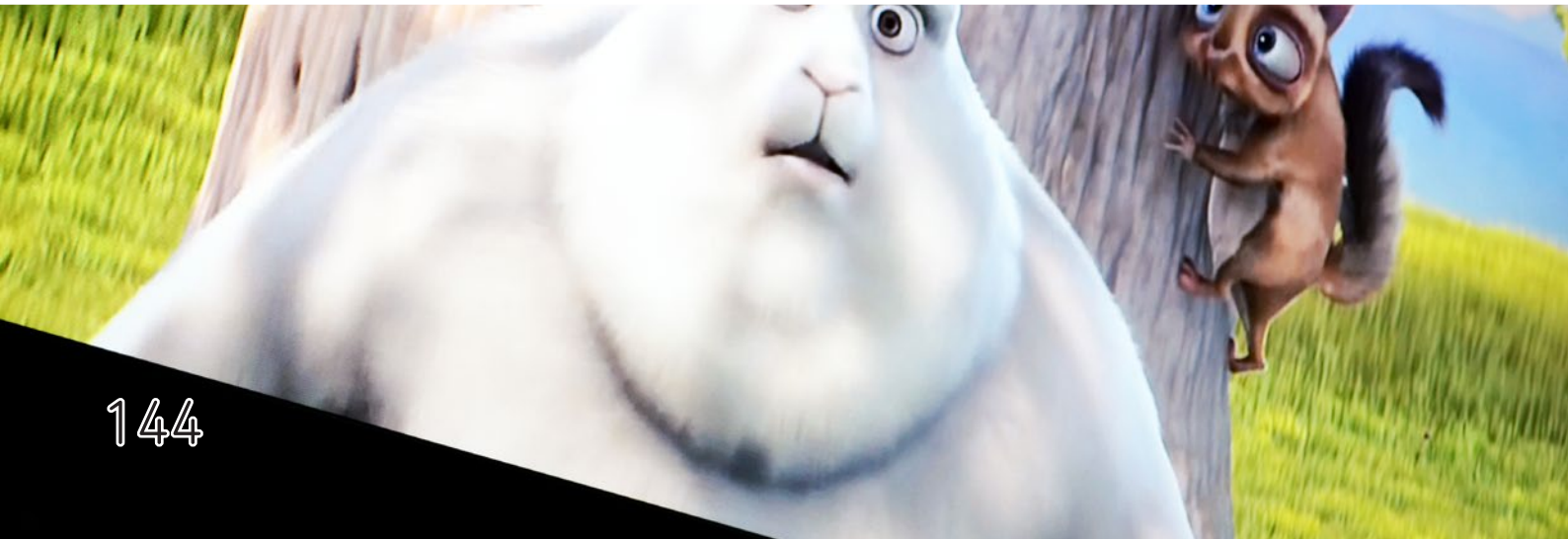
## Equipment

**Required**

- Raspberry Pi
- SD Card (8 GB+ Recommended)
- HDMI Cable
- USB Keyboard
- USB Mouse

**Optional**

- Raspberry Pi Case
- Network Connection



# Installing Kodi from Packages

There may be various reasons why you would want to install Kodi from apt-get instead of using one of the various pre-built operating systems.

One of those reasons may be that you want to retain the usage of the base Raspbian operating system, packages like OpenElec use an extremely bare-bones operating system, meaning it's hard to do anything that's outside of Kodi.

The downside to this, of course, is that Kodi won't run to its fullest extent, but it is the only option if you want to retain the functionality of the Raspbian operating system.

Skip this page if you only intend on using your Raspberry Pi as a media center.

To install Kodi on the Raspbian operating system, you simply need to open Terminal and type the following two commands.

```
sudo apt-get update
sudo apt-get install kodi
```

After this has finished completing you should now have Kodi installed on your Raspberry Pi. You can launch this by just typing the following command into the terminal.

```
sudo kodi
```

# The various flavours of Kodi Media Center

## LibreELEC - https://libreelec.tv/

LibreELEC is a fork of the once popualar OpenElec and is one of the most popular choices when it comes to setting your Raspberry Pi up as a media center. It makes use of an extremely bare-bones operating system. By bare-bones, we mean all you are getting is Kodi.

This allows it to be one of the fastest, most reliable, and slimmest distributions of Kodi.

Its only limitation comes if you want to do anything that requires a more fully fledged operating system.

## OSMC - http://osmc.tv/

OSMC is the continuation of the once popular Raspbmc and is another fork of the Kodi media center.

OSMC differs from Openelec in that it features a fully-fledged operating system underneath. It is one of the most fully featured media centers available for the Raspberry Pi.

It makes use of its theme, rather than the default Kodi one and allows the user to make numerous changes to the Raspberry Pi settings, such as overclocking the device.

It is probably the most fully featured choice out of the three.

## xbian - http://xbian.org

Xbian is designed as a small, fast, and lightweight media center distribution for the Raspberry Pi, like the others here, is based on Kodi.

It is built off a minimal Debian installation, which gives it some advantages over openelec, which mainly being you still have the freedom to install more packages to the operating system.

Xbian is also built to always be on the bleeding edge. They try to make sure that it is always utilizing the latest version of Kodi, which means while it will always have the most features, it may not be the most bug-free choice.

# 1. Download & Format the SD Card

**1.** Decide what version of Kodi you want to install from the list below.

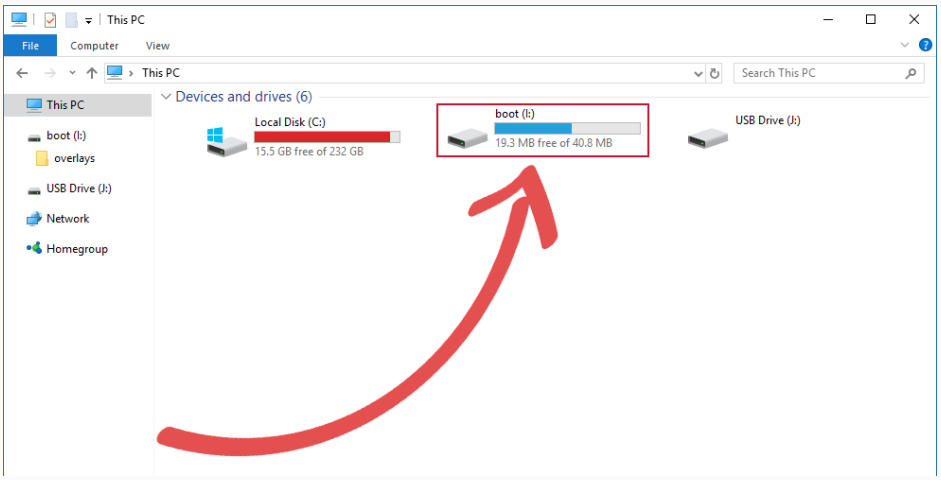
Download the **librelec** from https://libreelec.tv/downloads/ under "**DIRECT DOWNLOADS**".
Download **Xbian** from http://www.xbian.org/getxbian/ under "**Xbian Images**".
Download the **OSMC** from https://osmc.tv/download/ under "**Disc Images**".

**2.** To get started, you need to download a SD Formatter. On Windows and Mac you can grab the one from the SD Association here: **https://www.sdcard.org/downloads/formatter_4/**.

Once on the site download and install the latest version of the SD Formatter for your OS.

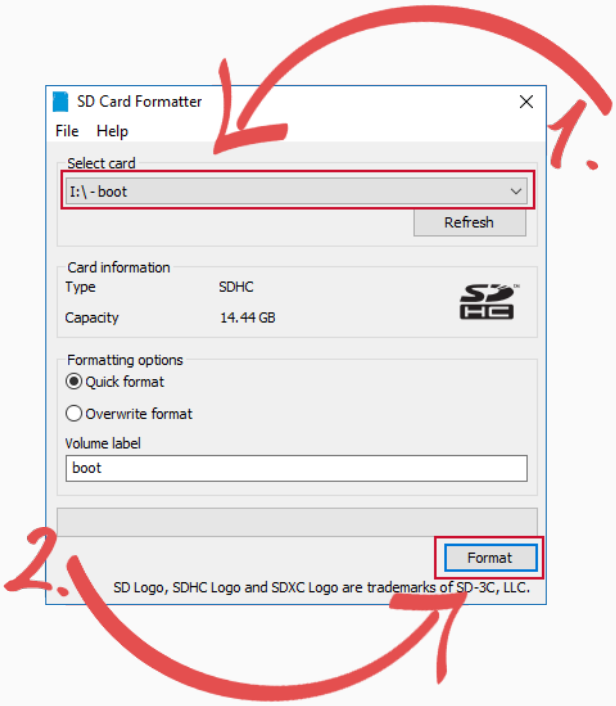
**3.** Insert your SD card into the computer or laptop's SD card reader and check the drive letter allocated to it, e.g., **G:/**.

Ensure you have found the correct drive letter otherwise you will end up formatting the wrong device.




**4.** Launch the SD Formatting tool that you installed earlier, make sure you select the correct drive letter for your SD card (e.g., **G:/**) (**1.**)

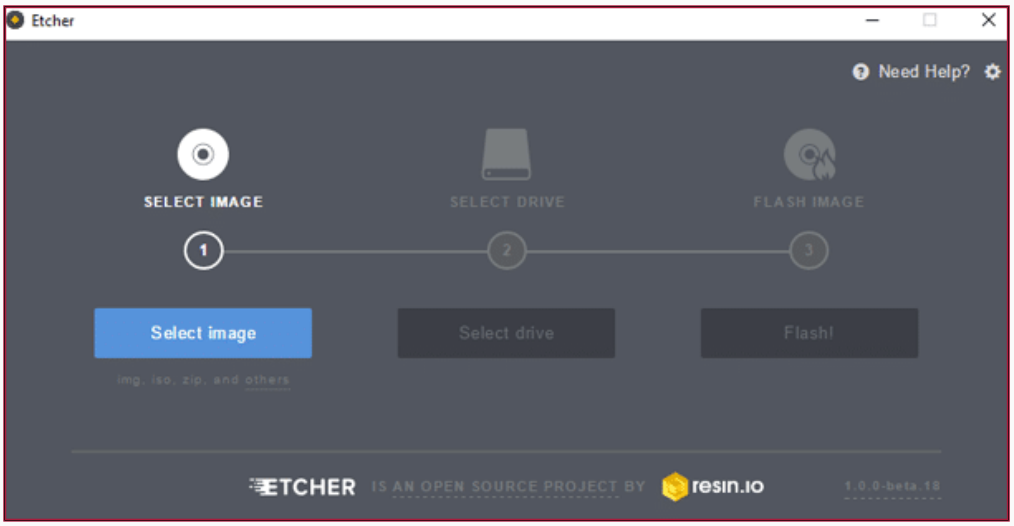
Once you are certain you have everything set correctly, feel free to hit the **"FORMAT"** button (**2.**)



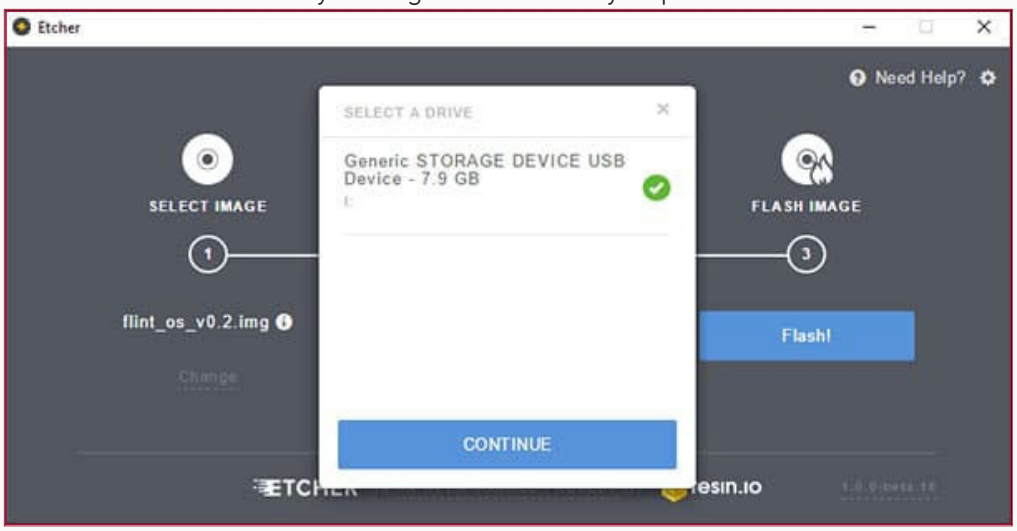
## 2. Installing your version of Kodi

**1.** Next, you will need some tools to be able to write the image to the SD card. For this we will use a tool called Etcher, you can download it from there website at https://etcher.io/.

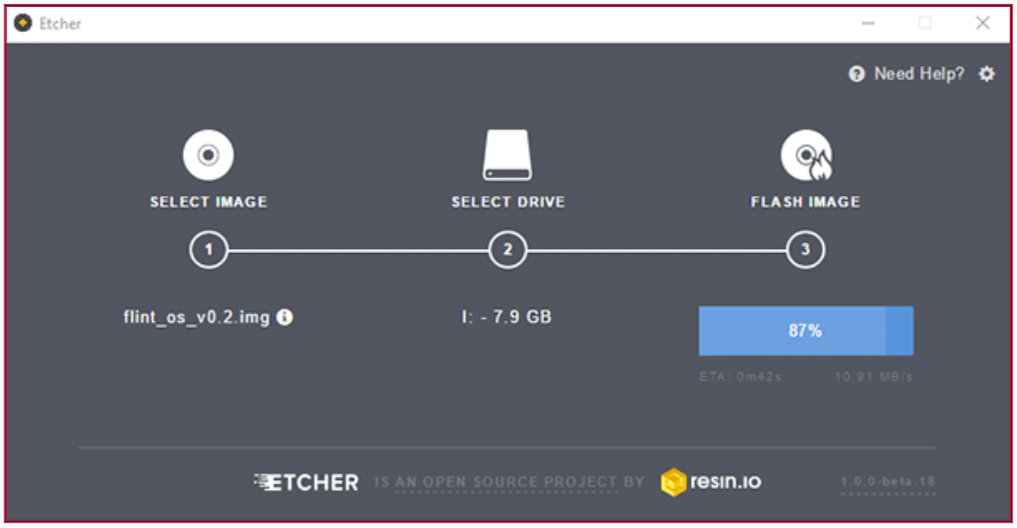
**2.** Open Etcher and press the "**Select Image**" button. Browse to the image you downloaded in **Step 1** and select it.




**3.** Next click the "**Select Drive**" button. With the correct drive selected, press "**Continue**". Double check to ensure it is the SD Card otherwise you might accidentally wipe a hard drive.




**4.** Finally, click the "**Flash!**" button to begin the flashing process. This can take some time. Once completed, take out your SD Card and put it into the Raspberry Pi.



## 3. Starting up the Raspberry Pi

Turn the Pi on by plugging in the micro USB cord from the power supply. It will now boot and will take about 5-10 minutes getting everything set up.

This process may be faster depending on your Raspberry Pi model.

Now proceed to set up any specific options you might want within the media center, most like Libre-ELEC come with a handy suite of tools to get things setup how you want it.

## Loading your videos into Kodi

Loading your video into Kodi is an easy thing to do, but it can be confusing if you have never touched a media center like Kodi before.

**1.** First go to **Videos**->**Files**->**Add video**. This screen allows us to add videos for Kodi to scan.

**2.** In here browse to the top-level directory where one lot of your videos are.
Make sure TV Shows and movies are seperate as they require different scrapers.

**3.** Once you have found the directory you want, select it by pressing **OK**. You can also add more directories by pressing add then browsing to the other directories.

**4.** Once you have finished doing this **press OK**.

**5.** You should now add a scraper. The scraper is what scans your files then grabs information about them from the internet to display to you in Kodi's interface such as banners and actor info.

For **TV Shows** I highly recommend you use **TVDB**, it is one of the best community resources for TV Shows and is one of the most accurate and well preserved.

For **Movies**, I highly recommend using the **Universal Movie Scraper**. This option scans a variety of different websites ensuring that it will get the best information possible for your movies.

**6.** Once you are happy with your choice, you can **press OK**.

**7.** You will then be asked if you would like to refresh info for all items in the path, **press Yes**.

**8.** It will now scan and add all the files it can recognize into a convenient easy to read list. If you find most of your stuff is not being picked up, then you should take a look at the names of the files and compare them to the information on the TVDB website. (Or the scraper you're using)

This process can take some time, as it needs to scan through every file, and then check against online databases. So, sit back and relax.

Repeat these steps for any additional folders of TV Shows and

## Raspberry Pi Kodi Remote

There are various ways to control your Kodi based media center, from setting up an IR receiver, using a tablet or phone, or having a TV that supports HDMI CEC

Most of the distributions of Kodi, such as Openelec and OSMC, offer the support for a large variety of different IR remotes controls.

You can also obtain some Android-based tablets and phones relatively cheaply these days, so a simple cheap phone or tablet can make a cost-effective remote control, along with it containing more features then a simple IR remote control can.

## HDMI CEC

HDMI CEC is the easiest way to control your Raspberry Pi media center. Kodi supports HDMI CEC right out of the box, what this means is that the remote control you use to control your TV will work automatically with Kodi.

These features work thanks to HDMI CEC's capabilities, as it sends the keypress commands that the TV receives to the Raspberry Pi over the HDMI cable.

The only real downside is that a lot of TV's sadly don't implement HDMI CEC.

## IR Remote

Setting up an IR remote is probably the most complicated out of the three options. Probably the easiest one to setup and use is the FLIRC. The FLIRC allows you to set it up and configure it on the computer and then plug it into your Raspberry Pi with it all configured.

There is also an add-on available that allows you to setup FLIRC within Kodi itself as well.

If you are interested in the FLIRC, you can purchase it directly from the company who develops it at:
https://flirc.tv/flirc-usb

## Android/iOS Remote Control

Another alternative to using an IR remote, or HDMI CEC's capabilities is to make use of your Android or iOS device. They can both be easily set up for communication with your Raspberry Pi, allowing you to control Kodi from your phone or tablet.

For Android, we highly recommend you use the app called "**Yatse**". It is one of the most functional and continually developed remote control applications for Kodi.

You can download it from the Play Store at:
https://play.google.com/store/apps/details?id=org.leetzone.android.yatsewidgetfree

For iOS devices, your best choice is the official Kodi remote control app.

You can obtain this app from the Apple app store at:
https://itunes.apple.com/au/app/official-kodi-remote/id520480364

SANYO

# Headless Music Player

## Project Description

In this project, I am going to be using a pre-built software package called the Pi Musicbox. Pi Musicbox contains plenty of features & functionality that make it great as a Music player.

This package is a headless music player so you will need to use a different device to be able to control it. The good thing is you can pretty much use any device that has a browser to be able to interact with it.

Now since this project is completely based on audio, it might be worth getting a USB Audio Card or a Raspberry Pi soundcard.

There two supported add-on audio card brands and they are HiFiBerry and IQAudio. (These both typically require the GPIO pins) however, most audio cards that are built for the Raspberry Pi should work as this is based on Raspbian.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

**Optional**
- Raspberry Pi Case
- USB Keyboard
- USB Mouse
- External Hard Drive or USB Drive

# Step 1 - Formatting your SD Card

The following steps will show you the best way to format your SD Card for the Raspberry Pi:

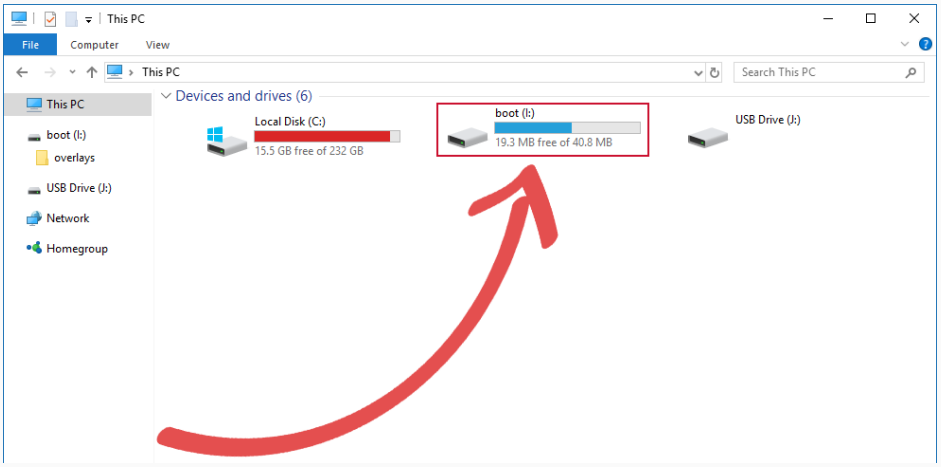
**1.** To get started, you need to download a SD Formatter. On Windows and Mac you can grab the one from the SD Association here: **https://www.sdcard.org/downloads/formatter_4/**.

Once on the site download and install the latest version of the SD Formatter for your OS.

**2.** Once downloaded, run the file and follow the prompts to install the SD Card Formatting tool onto your computer.

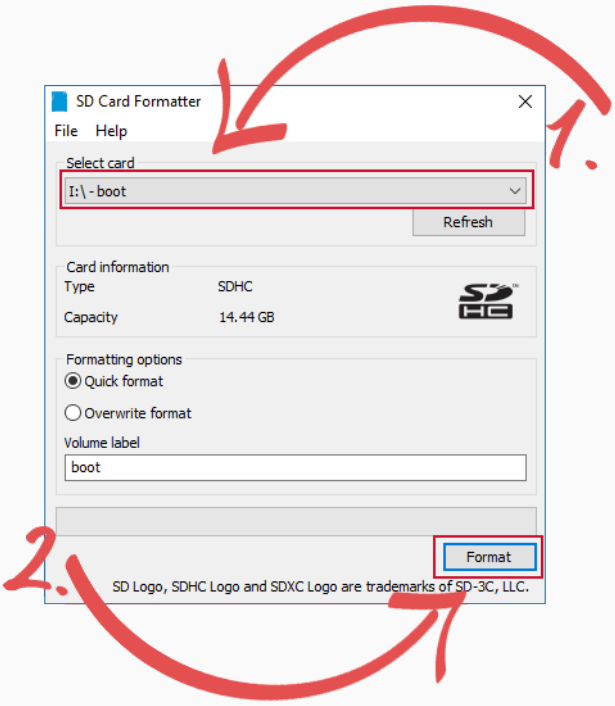
**3.** Insert your SD card into the computer or laptop's SD card reader and check the drive letter allocated to it, e.g., **G:/**.

Ensure you have found the correct drive letter otherwise you will end up formatting the wrong device.




**4.** Launch the SD Formatting tool that you installed earlier, make sure you select the correct drive letter for your SD card (e.g., **G:/**) (**1.**)

Once you are certain you have everything set correctly, feel free to hit the **"FORMAT"** button (**2.**)



# Step 2 - Installing Raspberry Pi Music Box

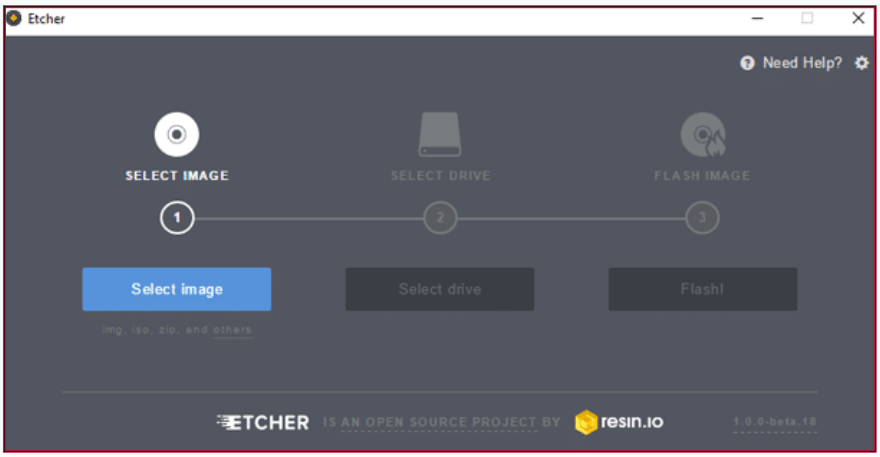
**1.** First head over to the Pi Music Box's Github at **https://github.com/pimusicbox/pimusicbox/releases/** and download the latest zip file.

Once downloaded, unzip the file so, you are left with a **.img** file.

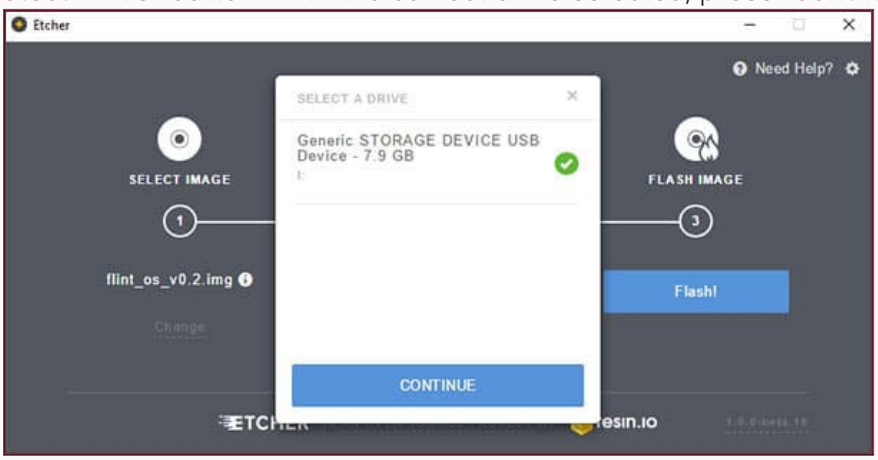
**2.** Next, you will need some tools to be able to write the image to the SD card.

For this we will use a tool called Etcher, you can download at https://etcher.io/.

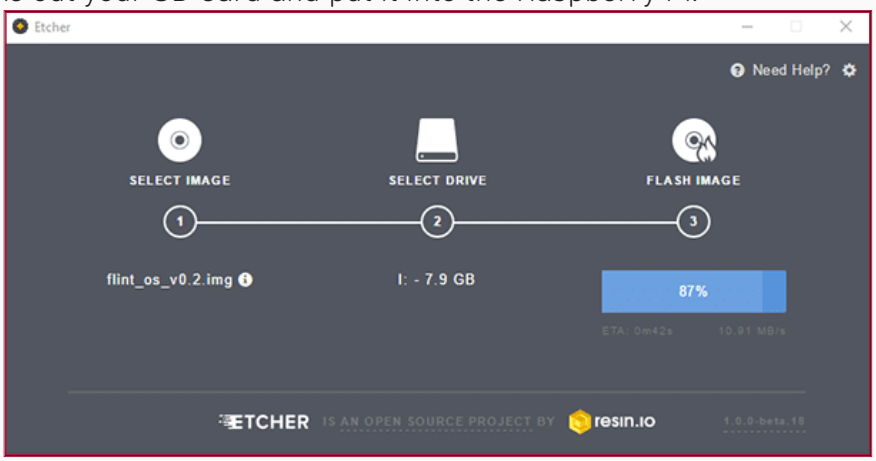
**3.** Open Etcher and press the "**Select Image**" button. Browse to the image you downloaded in **Step 1** and select it.




**4.** Next click the "**Select Drive**" button. With the correct drive selected, press "**Continue**".




**5.** Finally, click the "**Flash!**" button to begin the flashing process. This can take some time. Once completed, take out your SD Card and put it into the Raspberry Pi.



## Booting/Setting up the Raspberry Pi Music Player

Now we are all good to connect everything up and turn the Raspberry Pi on.

We will be opening the Music box in a browser so, it's essential that you have an Ethernet cord or Wi-Fi dongle connected. (The Wi-Fi dongle required a bit of extra setup mentioned above)
To connect to the Raspberry Pi music server, open up the browser and enter the following URL.

http://musicbox.local

If this doesn't work or you're using an Android device, then you will need to use the IP of the Raspberry Pi.

If you have your Pi connected to a screen you should be able to see the IP in the command line.

Once connected in your browser you should be presented with a screen like the one below.

I am just going to mention some of the things you can setup so now go to settings. You will find the settings in the following order.

## Network

In here you will find all the network settings you will need to correctly setup the Raspberry Pi music player.

- **Wi-Fi Network Name:** This is the SSID or the name of your Wi-Fi network you wish to connect to.
- **Wi-Fi Password:** The password required to access the local network.
- **Workgroup:** This is the workgroup name of the local Windows Network.
- **SSH:** Only turn this on if you need to edit the Raspberry Pi remotely via the command line.

## Audio

- **Initial Volume:** This is the volume that the device will start at by default.
- **Audio Output:** Here you can manually select the output for the audio.
- **Downsample USB:** This is the workgroup name of the local Windows Network.

## MusicBox

In here you will find all the network settings you will need to correctly setup the Raspberry Pi music player.

• **Device Name:** This is the name of your Pi and how it would be called on your local network.

Change this if you wish it to be something different to "**musicbox**".

Keep in mind changing it will change the URL in the bowser to the new name.

• **Autoplay Url:** You can enter a URL here to start a radio station or stream for when the device boots up.

If it doesn't automatically start you can increase the wait time to a little higher

• **Wait Time:** The time it waits before playing the autoplay URL after boot.

The time set here is the wait time in seconds.

• **Root Password:** By default, the root password to the device is "musicbox" this should be changed to something more secure especially if you are going to enable SSH.

• **AirPlay Streaming:** This allows you to stream directly from your iOS device to the Raspberry Pi Music Server.

When enabled this will show up in your AirPlay device list.

• **DLNA/UPnP/OpenHome Streaming:** If you need to stream over any of these protocols then just enable this.

## Music Files

• **Scan Music Files:** Enable this if you would like to scan for music files on boot.

However, you could find that this slows down your boot time considerably.

• **Network Drive:** If you have a network drive setup that has all your music in it then you can set it in here.

Make sure you enter the correct username and password.

• **Resize Filesystem:** This setting is critical if you are looking to make the most of your SD card.

Once you enable this and reboot, you will gain access to the entire SD Card.

## Services

If you are looking to use services such as Spotify, YouTube, Google Music, SoundCloud and much more, then you will need to set them up in here.

The setup for each of the services are straightforward and are often explained if any extra information is required. More services may be added in the future.

That's all the information you should need for setting up the music player correctly. There are a few things you should probably be aware of that I will quickly mention below.

## Other things to note

Now with the Pi music box, there are a few things that are mentioned you should probably be aware of.

The software package is not secure enough to be run outside a security system. Keep this in mind if you decide to connect to a public Wi-Fi spot or, you are on an unsecured internet connection.

The passwords for the Spotify service and a Wi-Fi connection are stored in plain text on the SD-card.

You can avoid the Wi-Fi issue by setting it up manually and not use their inbuilt system but, there is no workaround for the Spotify service.

You should change the root login password if you decide to turn SSH on. As I mentioned above the password is default so anyone who knows of the music box software can quickly gain access to your system if SSH is enabled. (However, they would need to be on your local network)

If you want to access the mopify GUI instead of the Pi MusicBox GUI you can just enter the following into your browser.

```
http://musicbox.local/mopify
```

Again replace the **musicbox.local** with the IP of the Raspberry Pi if the hostname is not working.

Mopify is a GUI designed for Spotify only so if you want to use the other services then you will need to use the Pi music box GUI.

# Plex Server

## Project Description

Plex is a client-server setup where the client just streams data from the Plex media server. The Plex media server allows you to have all your movies, music and photos located on the one device, the server.

In this case, we will be using the Raspberry Pi. You can then have multiple clients connect to the same server, which is excellent as you don't need to have multiple copies of the same media across several devices.

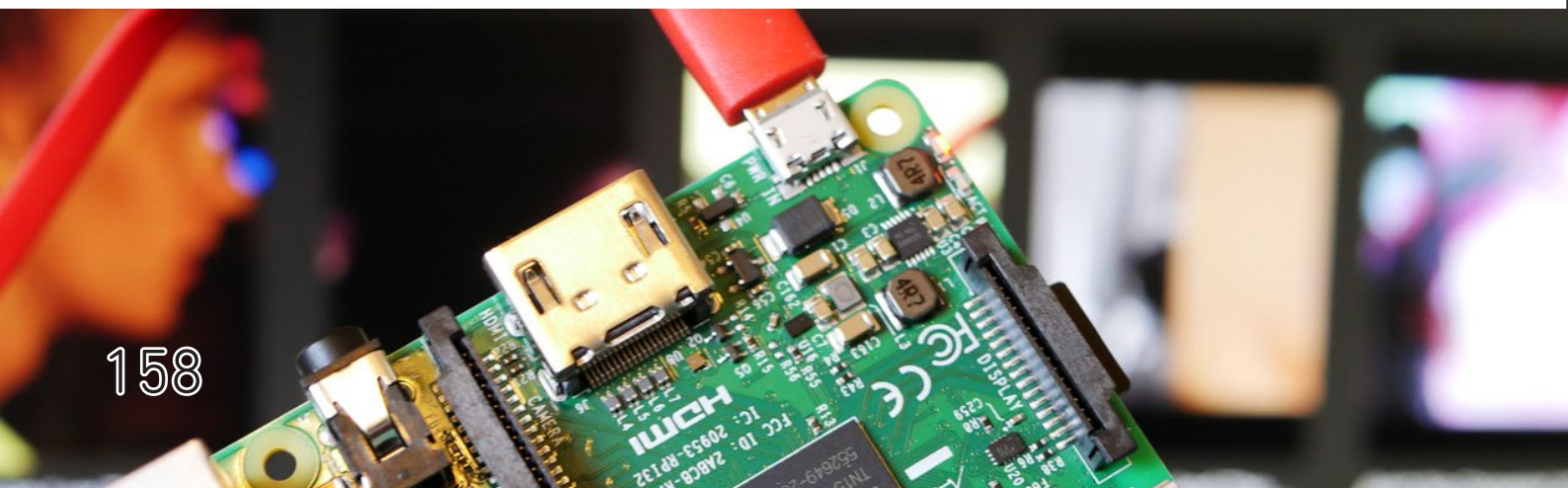
The Plex Client is supported on various operating systems, such as Windows, Apple, Android, Amazon FireTV, Chromecast, Xbox, PlayStation, Linux and so many more.

**Note:** This will not work on older versions of the Pi. Only the Raspberry Pi 2 or later for this to work.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection (Ethernet Reccommended)

**Optional**
- Raspberry Pi Case
- External Hard drive or USB Drive
- USB Keybaord
- USB Mouse



## Setting up the Raspberry Pi Plex Server

Since there is no official ARM Plex server, we will need to download and use a repackaged ARM version from day2dev.

**1.** As with any software, tutorial let's first make sure our Pi is up to date

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

**2.** We next need to enable the HTTPS transport package so we can access HTTPS packages using apt-get.

Enter the following line to download, install & activate it.

```
sudo apt-get install apt-transport-https
```

**3.** Next, we need to add a crypt o key for the dev2day website to our keyring.

The **|** in the following command copies the output from the first command (**wget**) into the second command (**sudo apt-get add** – "first command output").

```
wget -O - https://dev2day.de/pms/dev2day-pms.gpg.key  | sudo apt-key add -
```

**4.** Next, we need to add the dev2day repository to our package source list.

To do this just enter the following command into the terminal.

```
echo "deb https://dev2day.de/pms/ stretch main" | sudo tee /etc/apt/sources.list.d/pms.list
```

**5.** Now we need to update the package list.

You can do this by running the following command on your Raspberry Pi.

```
sudo apt-get update
```

If you get the error **"/usr/lib/apt/methods/https could not be found."** Then the https transport package hasn't been installed. Double check that it has been installed correctly.

**6.** Now run the following to install the Plex media server onto the Raspberry Pi.

```
sudo apt-get install -t jessie plexmediaserver
```

**7.** To avoid any annoying permission problems, change Plex to run under the Pi user.

To do this open the following file.

```
sudo nano /etc/default/plexmediaserver
```

**8.** Change the PLEX_MEDIA_SERVER_USER from plex to pi.

```
PLEX_MEDIA_SERVER_USER=pi
```

With the change now made make sure you save and exit the file by pressing **Ctrl+X** then pressing **Y** and pressing **Enter**.

**9.** Now restart the plex media server by using the following command:

```
sudo service plexmediaserver restart
```

## Setting up the Plex Server - Continued

**10.** Now it should all be installed, but before we get started, we should make sure the Pi has a static IP, so it's easy to remember the IP address.

To get your current IP address enter the following:

```
hostname -I
```

**11.** Now open the boot **cmdline.txt** file with the following command:

```
sudo nano /boot/cmdline.txt
```

**12.** At the bottom of this file, add the following line: (Replacing "**YOUR IP**" with the IP you got from using **hostname -I** command)

```
ip=YOUR IP
```

With that change, make sure you save and exit the file by pressing **Ctrl+X** then pressing **Y** and pressing **Enter**.

**13.** Now just restart the Raspberry Pi by running the following command.

```
sudo reboot
```

**14.** The Raspberry Pi should now always try to start with the same local IP address.

You can help ensure this by setting it on your routers by tying the mac address of your network device (WiFi or Ethernet) to a particular local IP address.

The majority of routers should support this.

Now the Raspberry Pi Plex media server should be all set up and ready to scan your media and stream it any client that wishes to connect.

## Storing Media on the Raspberry Pi

Now there are several ways to store your media on the Raspberry Pi. I will mention each of the methods below.

## External Harddrive

You can just hook up an external hard drive with all your music, movies and whatever else you may have.

Setting the Plex program to run as the **Pi** user means you can just plug a USB hard drive in and access the media in Plex without any issues.

You can also permanently mount drives to a certain location.

I have already covered this in a previous tutorial so be sure to check out the guide available in our "**Getting Started with the Raspberry Pi**" book titled "**Mounting a USB Drive**".

Make sure you set the user & group owner of the drive to **Pi**.

## Network Attached Storage (NAS)

You could also set your Pi up to act as a NAS so you can transfer your media across to it without needing to disconnect/reconnect a hard drive.

You can set this all up by following my tutorial on setting up a Raspberry Pi network attached storage.

Again, make sure you set the group & user owner to **Pi** or whatever the user Plex is running as.

## SD Card

Lastly, you can just use the SD card for storage but as you could imagine this will quickly run out of space. You can setup a folder on the SD card to be accessed via the network.

# Connecting Clients to the Plex Media Server

If you are using an app on your phone, computer, Xbox, PlayStation, or any other device then the Plex client should be able to notice the server automatically.

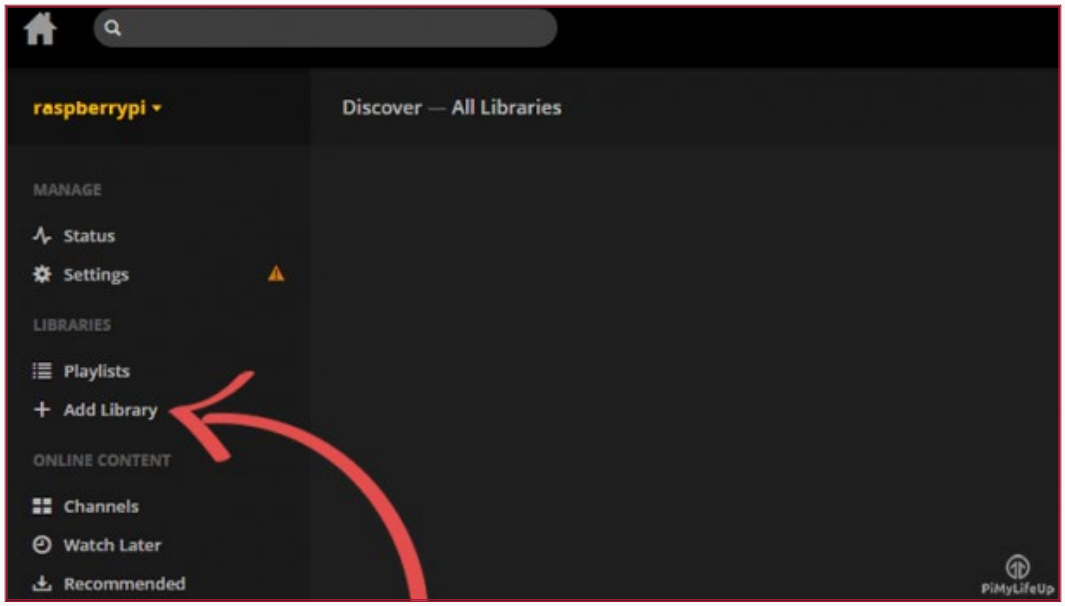
You will be prompted to log in, simply sign up or sign in to an existing plex account. You can skip this by just entering by entering the address above in again.

To connect in the browser simply enter the IP followed by the **port 32400** and **/web/**. For example, mine is, **192.168.1.100:32400/web/**.

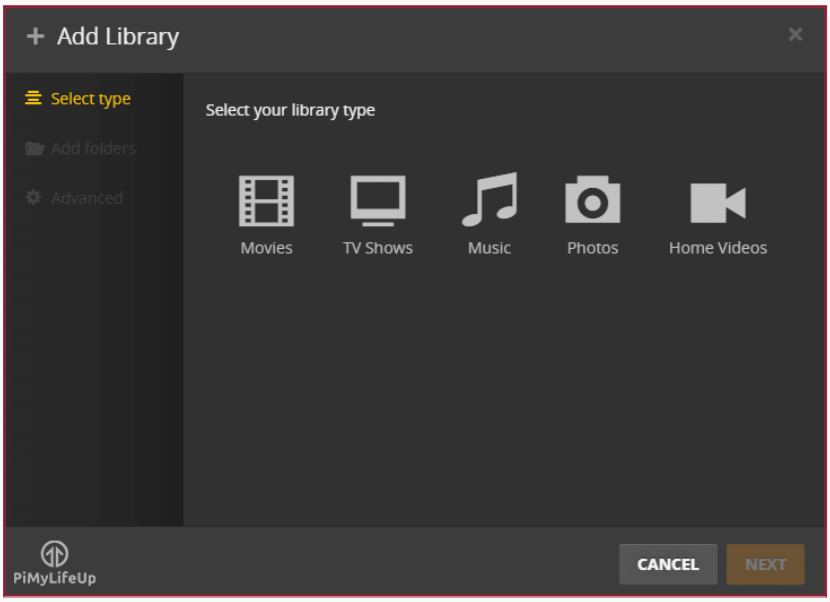
```
192.168.1.100:32400/web/
```

You will be prompted to log in, just sign up or sign in to an existing plex account. You can skip this by just entering by entering the address above in again.
Next, you will need to setup your music, movies, and TV show libraries. This is incredibly easy and shouldn't be too hard in getting it set up correctly.

**1.** First select add library in the left-hand side column.



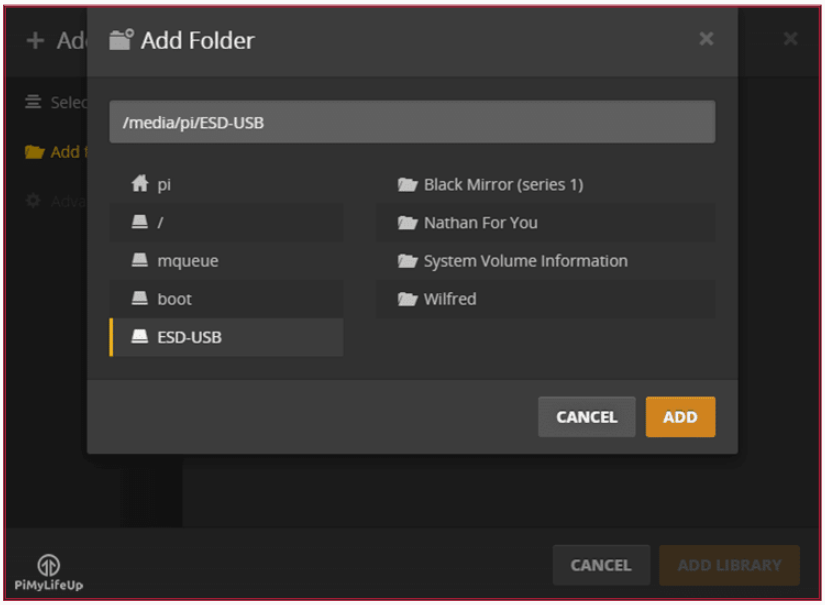

**2.** Next, select the type of media that is in the folder. If you have more than one type, then you will need to add a new library for each type of media.



## Connecting Clients to the Plex Media Server

**3.** Next, you will need to select the folder that has all your media in it.

For example, mine is on a USB drive that is displayed in the left-hand side column or can be found at **/media/pi/ESD-USB**




**4.** Once you add the library, it will now organize your clips in a nice easy to browse interface.

If you need more information on how to setup, name and organize your media library then Plex has an amazing amount of documentation that you can find it here:
https://support.plex.tv/hc/en-us/categories/200028098-Media-Preparation

## Troubleshooting

One issue I came across when it came to downloading and installing the package was that it couldn't connect to dev2day.de servers.

I got the following error. (This is just a snippet of the error)

```
"E: Failed to fetch https://dev2day.de/pms/pool/main/p/plexmediaserver/plexmediaserver
_0.9.16.4.1911-ee6e505-2~jessie_all.deb
Failed to connect to dev2day.de port 443: Network is unreachable

E: Unable to fetch some archives, maybe run apt-get update or try with --fix-missing?"
```

Fixing this problem was surprisingly easy after spending hours of troubleshooting. Just edit the file with our package source list where we added the dev2day.

```
sudo nano /etc/apt/sources.list.d/pms.list
```

In here just change dev2day to just normal HTTP. For example, it should look like this.

```
deb http://dev2day.de/pms/ stretch main
```

Once that is done simply run sudo apt-get update and then try updating again.

I hope that you have been able to get the Raspberry Pi Plex server up and running without any trouble. If you come across any issues or have some feedback, then please don't hesitate to leave a comment below.

# AirPlay Receiver

## Project Description

In this tutorial, we will be exploring how to setup an AirPlay Receiver by utilizing your Raspberry Pi. To do this, we will be making use of the open source software called Shairport Sync.

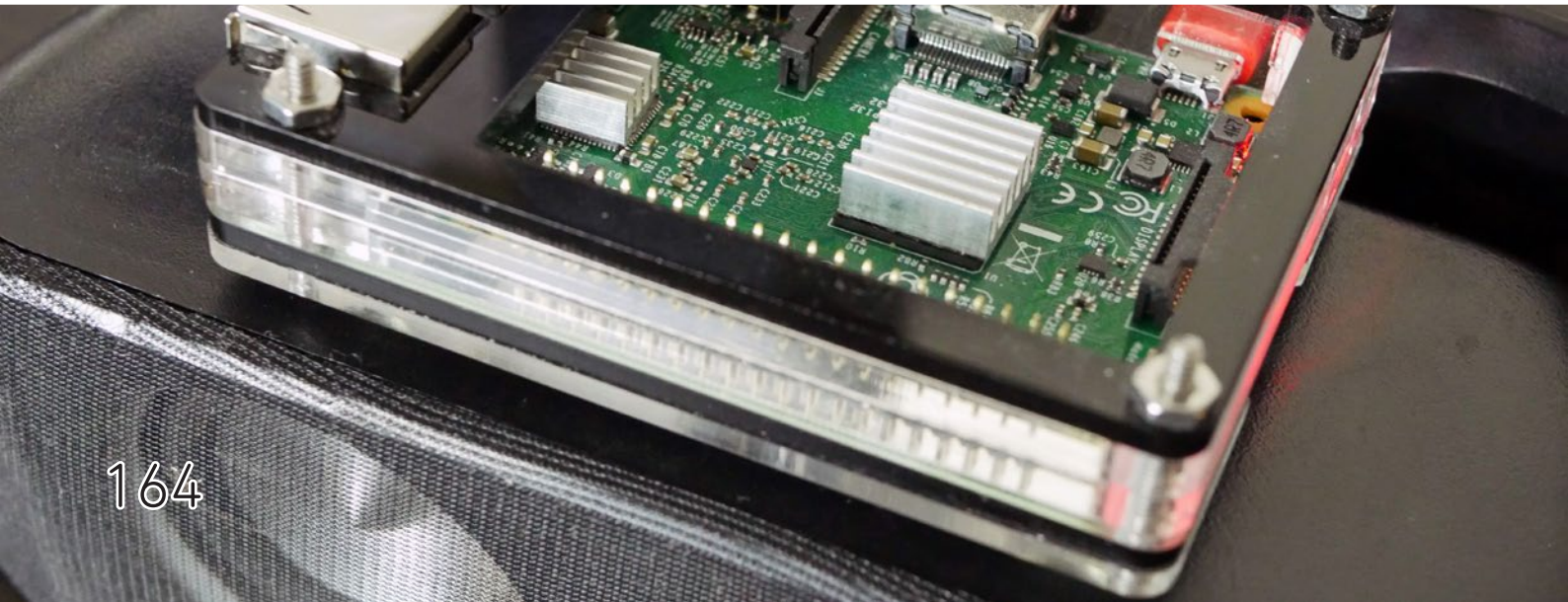
This software allows the Raspberry Pi to act as an airplay receiver by implementing Apple's proprietary protocols.

Shairport Sync will allow you to play music from any AirPlay-enabled device to your Raspberry Pi, allowing a straightforward and cost-efficient wireless speaker setup.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection (Wifi Recommended)
- Speakers

**Optional**
- Raspberry Pi Case
- External Hard drive or USB Drive
- USB Keybaord
- USB Mouse



## Setting up an Apple AirPlay Receiver

Setting your Raspberry Pi up as an AirPlay receiver is an incredibly simple task if you have a good Wi-Fi connection and a set of speakers it is a straightforward process.

**1.** First, run an update and upgrade to ensure we are running the latest software.

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

**2.** Once that has completed we need to install several different packages, run the following command to install them all.

```
sudo apt-get install autoconf libtool libdaemon-dev libasound2-dev libpopt-dev libconfig-dev
sudo apt-get install avahi-daemon libavahi-client-dev
sudo apt-get install libssl-dev
```

**3.** We will now clone the shairport-sync source to our Raspberry Pi. Shairport-Sync is the best fork of the original Shairport code and allows synching across multiple rooms.

Run the following commands to download the source code to your Raspberry Pi.

```
cd ~
git clone https://github.com/mikebrady/shairport-sync.git
```

**4.** Now let's build and install the Shairport software, to do this we must run a few commands. To get started lets first move into the shairport-sync folder and configure the system.

```
cd shairport-sync
autoreconf -i -f
./configure --with-alsa --with-avahi --with-ssl=openssl --with-systemd --with-metadata
```

The autoreconf command setups the necessary config file. The configure command further sets up the build system, telling it to utilize the ALSA audio backend, the Avahi network and set it to use OpenSSL for encryption.

**5.** With the configuration now done, let's run the make command and install the software. This will setup numerous things including the auto start script.

```
make
sudo make install
```

**7.** To enable the Shairport Sync software to start automatically at system startup you need to enter the following command into terminal:

```
sudo systemctl enable shairport-sync
```

**8.** Finally, we can startup Shairport immediately by running the command below:

```
sudo service shairport-sync start
```

You should now be able to play audio files through your Raspberry Pi using any AirPlay-capable devices, the most prominent of these being iOS devices.

It will pop up in your list of devices as RaspberryPi if you are using the default host name.

## Improving the Analogue Audio output

With our Raspberry Pi AirPlay receiver now setup. There are several different things we can do to improve it. The first of these is to change the Raspberry Pi so it will utilize a newer version of the audio driver.

To run this improved audio driver and get the benefits of it fully there are a few changes we will have to make.

**1.** Firstly we need to update the Raspberry Pi's firmware. We can do this by running the following command. It can take some time. Make sure your Raspberry Pi doesn't lose power during this.

```
sudo rpi-update
```

**2.** When the firmware update has completed, turn off your Raspberry Pi and take out your SD Card and insert into a reader connected to a computer. The reason for this is that we need to modify the Raspberry Pi's boot config file

The file we are after is located at **/boot/config.txt** on the SD Card, open it up with your favorite text editor. Add the following new line to this file:

```
audio_pwm_mode=2
```

Once you have edited this file, you can save it and place the SD Card back into your Raspberry Pi and turn it back on.

**3.** With your Raspberry Pi back on, there are two more things we need to do before the improved analog audio driver will work properly with Shairport.

The next step is to set it so, the analog jack is the main audio out.

Run the following command in terminal to do this:

```
amixer cset numid=3 1
```

**4.** Now there is one final thing we must do, and that is modify the volume db Range that Shairport uses, we can modify this by changing it in the configuration file.

Run the following command to begin editing the configuration file.

```
sudo nano /usr/local/etc/shairport-sync.conf
```

**5.** Within this file make the following changes:

### Find

```
//      volume_range_db = 60 ;
```

### Replace with

```
        volume_range_db = 40;
```

We can now save the file by pressing **Ctrl + X** then pressing **Y** and then **Enter**.

**6.** Now to make sure all these changes are properly loaded in, we will restart the Raspberry Pi by running the following code.

```
sudo reboot
```

## Improving the Wi-Fi performance

**1.** To improve the WiFi performance of your AirPlay device, you will want to disable the Raspberry Pi's WLAN adapter power management.

The reason for this is that it can prevent Shairport from being visible on your list of Airplay devices.

Luckily it is easy to do this by modifying the network interfaces, open the file by using the following command in terminal:

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

**2.** Within this file locate and add the following text to the interfaces file.

This will tell the Raspberry Pi not to manage its power and will not turn it off to save power.

### Find

```
iface wlan0 inet manual
```

### Add Below

```
wireless-power off
```

We can now save the file by pressing **Ctrl + X** then pressing **Y** and then **Enter**.

**3.** Now to make sure all these changes are properly loaded in, we will restart the Raspberry Pi by running the following code.

```
sudo reboot
```

# Pirate Radio

## Project Description

In this tutorial, we will be showing you the basics of how to set up a Raspberry Pi Pirate Radio utilizing the FM Transmitter software and a piece of wire.

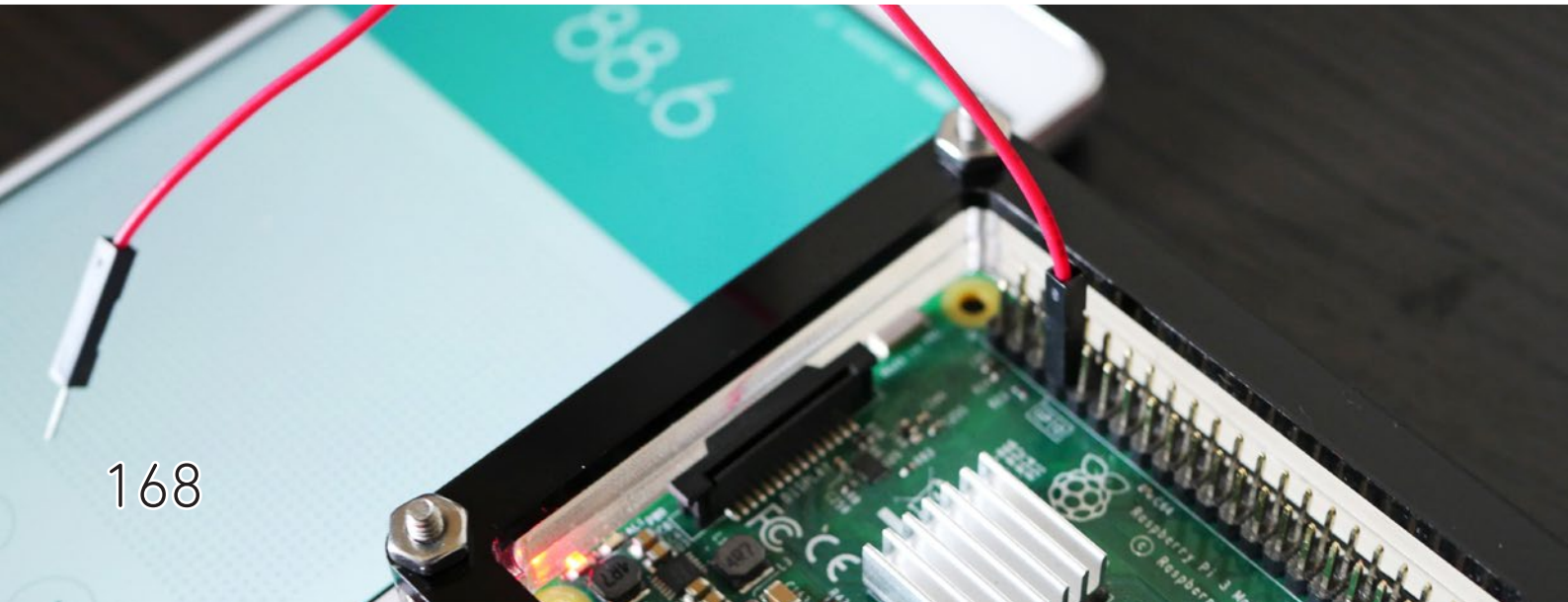
This piece of software allows you to broadcast an FM signal through your GPIO 4 pin. This means you can set up your own little radio station, however, please be noted that the sound quality and range will be quite poor.

Also, please note that there are laws and regulations around broadcasting on FM Frequencies, make sure that you check with the laws of your country before proceeding with this tutorial to ensure you aren't breaking any laws.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection (Wifi Recommended)
- 30cm 12 AWG wire

**Optional**
- Raspberry Pi Case
- External Hard drive or USB Drive
- USB Keybaord
- USB Mouse



## Setting up your Pirate Radio

Before we get started with this tutorial, we first need to hook up a wire to the GPIO 4 pin, which is also known as the seventh physical pin.

If you are unsure of what GPIO pin we are talking about you can check out our "**Raspberry Pi Circuitry**" book as this will touch on GPIO pins more and features a helpful GPIO reference image.

This wire will act as our antenna, for the tutorial we just used a simple breadboard wire which isn't the best antenna but is enough to work as a proof of concept.

A 30cm piece of 12 AWG wire should work good enough to broadcast over a small area.

**1.** Now before we set up our basic pirate radio and install the FM transmitter we need to run an update on the Raspberry Pi by using the following two commands.

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

**2.** With the update now done, let's install all the packages we will need to be able to fully set up our pirate radio.

Run the following install commands to grab all the packages we need.

```
sudo apt-get install -y sox make gcc g++ git arecord libmp3lame-dev
```

**3.** Now that we have all the packages we need installed, we can move onto cloning the FM Transmitter software to our Raspberry Pi.

This code will allow us to broadcast music through our antenna that's placed on the GPIO 4 pin.

We can clone the software to our Raspberry by running the following commands on it.

```
cd ~
git clone https://github.com/somu1795/fm_transmitter.git
```

**4.** Let's now compile the software. We can just do this by moving into the directory and running make.

Run the following two commands on your Raspberry Pi.

```
cd ~/fm_transmitter
make
```

**5.** With the FM Transmitter software now compiled onto our Raspberry Pi we can now try playing our first music file.

To be able to do this we first need a supported file, let's run the following commands to grab a test file from our file server.

```
wget https://pimylifeup.com/out/star-wars.wav
```

## Setting up your Pirate Radio

**6.** Now that we have a supported file downloaded to our Raspberry Pi let's try running the software.

To do this, we can simply type in the following command to run it.

This command converts the file into a supported format and pipes it to our FM Transmitter software and then plays it on the 100.6 band.

```
sox /home/pi/fm_transmitter/star-wars.wav -r 22050 -c 1 -b 16 -t wav - | sudo
```

The arguments for this command do several things:
**-r** defines the sample rate that SOX will convert the file.
**-c** defines the number of channels. Limitations of the software force us to set it to a single channel.
**-b** defines the bit rate that the output should be sampled at.

**7.** You can now test to see this is working, by tuning to the 100.6 FM band on your radio device. You should hear the music come through if it is successfully working.

Once you are happy that it is working we can move onto the next segment of this tutorial.

This segment will involve setting it up so our Pirate Radio can also play MP3 files since this is not supported by default.

To achieve this, we will need to compile several pieces of software, go over to the next page to learn how to set up everything you need.

## Adding MP3 Support to your Pirate Radio

To add MP3 support, we need to compile and install FFMPEG, since FFMPEG is not available through packages we will have to do all of this manually.

**1.** To start off lets first clone and compile x264 support, this is needed to compile the latest versions of FFmpeg.

Run the following six commands on your Raspberry Pi to do this.

```
cd /usr/src
sudo git clone git://git.videolan.org/x264
cd x264
sudo ./configure --host=arm-unknown-linux-gnueabi --enable-static --disable-opencl
sudo make
sudo make install
```

**2.** With the x264 library now installed we can go into compiling FFmpeg.

To do this, we need to run another six commands on our Raspberry Pi.

```
cd /usr/src
sudo git clone git://source.ffmpeg.org/ffmpeg.git ffmpeg
cd ffmpeg
```

**Raspberry Pi 3 and later:**

```
sudo ./configure --arch=armv7-a --target-os=linux --enable-gpl --enable-libx264
--enable-nonfree --extra-cflags='-march=armv7-a -mfpu=neon-vfpv4 -mfloat-abi=hard'
```

**Raspberry Pi 2 and earlier:**

```
sudo ./configure --arch=armel --target-os=linux --enable-gpl --enable-libx264 --enable-nonfree
```

**Finally for both run:**

```
sudo make -j4
sudo make install
```

**3.** With FFmpeg now compiled and setup we can now actually run MP3 files. To do this, we will make use of the "**PiStation.py**" script that is included with the FM Transmitter software.

While we do not have a sample mp3 file handy to get you to download, you can simply run any mp3 file you want by utilizing the following command.

Make sure you replace the last segment with the location of the file.

```
cd ~/fm_transmitter
sudo python ./PiStation.py -f 100.3 example.mp3
```

Hopefully, by now you have learned how to set up a basic pirate radio using the FM Transmitter software. This pirate radio will allow you to broadcast any music you want by just using your Raspberry Pi.

Remember though that you need to adhere to your countries radio laws, most countries have a lot of restrictions on what frequencies you can use and the range that you can broadcast to.

# Volumio

## Project Description

In this tutorial, we are showing you the basics of installing the headless music player, Volumio. We will also be walking you through how to get Wi-Fi up and running on your Raspberry Pi through Volumio's interface. Additionally, the tutorial also shows you the steps to go through to install and setup Volumio's Spotify plugin.

Volumio is an open source and headless music player that is designed to be able to play almost every single music file, providing support for MP3, FLAC, WAV, AAC, ALAC, DSD and many more. Also, to support of a large variety of audio codecs, it also provides support for UPNP\DLNA, Airplay, Spotify and Web Radios making it an all in one audio player.

Additionally, Volumio also provides support for USB 2.0 DAC's and a large amount of Raspberry Pi DAC's that work through I2S.

To load music files on to Volumio make sure you put them on a USB device and plug that into your Raspberry Pi.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection (Wifi Recommended)
- External Hard drive or USB Drive with music

**Optional**
- Raspberry Pi Case
- USB Keybaord
- USB Mouse

# Step 1 - Formatting your SD Card

First off you should do when installing a new image to an SD card is format it correctly.

The following steps will show you the best way to format your SD Card for the Raspberry Pi:

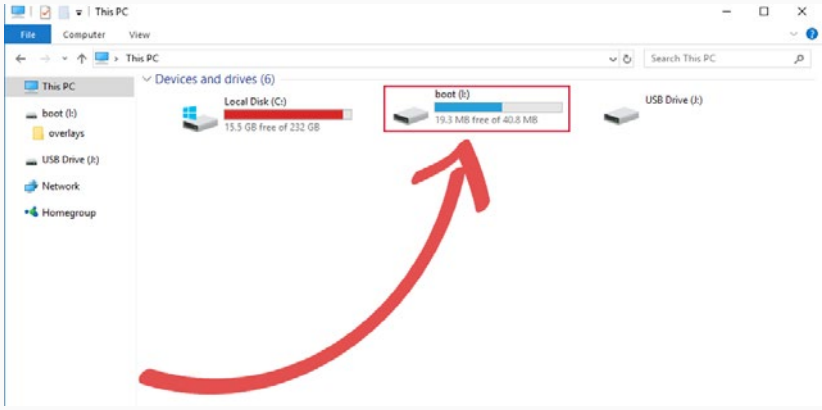
**1.** To begin you need a tool to format your SD Card. You can obtain one of the most robust ones from here: **https://www.sdcard.org/downloads/formatter_4/**

Once you are on the website will find a download link for the SD Formatter.

**2.** Download and install the SD Card Formatter from the SD Card association.

**3.** With the SD Formatter installed to your computer, insert your SD card and check the drive letter that is allocated to it, e.g., **G:/**.

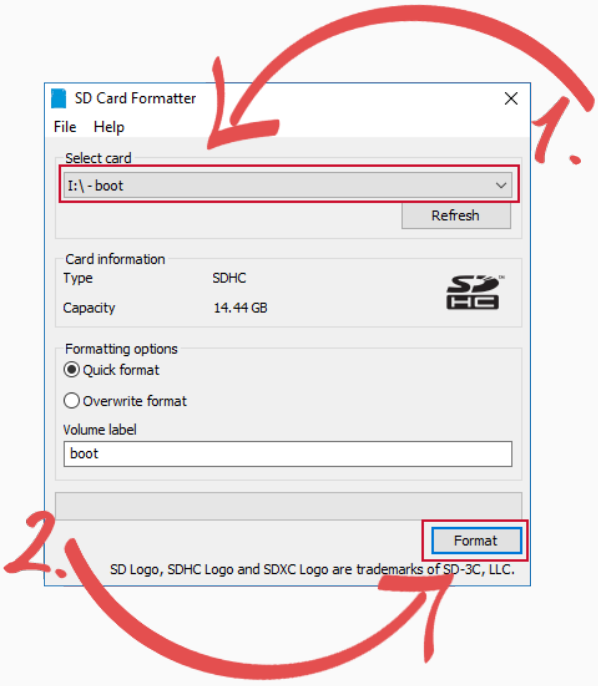
Make sure that you have the correct drive letter otherwise you could accidentally format the wrong device.




**4.** With your SD Card now plugged into your computer, launch the SD Formatter software that we installed earlier.

First set the drive letter to the correct one **(e.g., G:/)**.

Once you have everything set correctly, press the "**FORMAT**" button.



# Step 2 - Installing Raspberry Pi Music Box

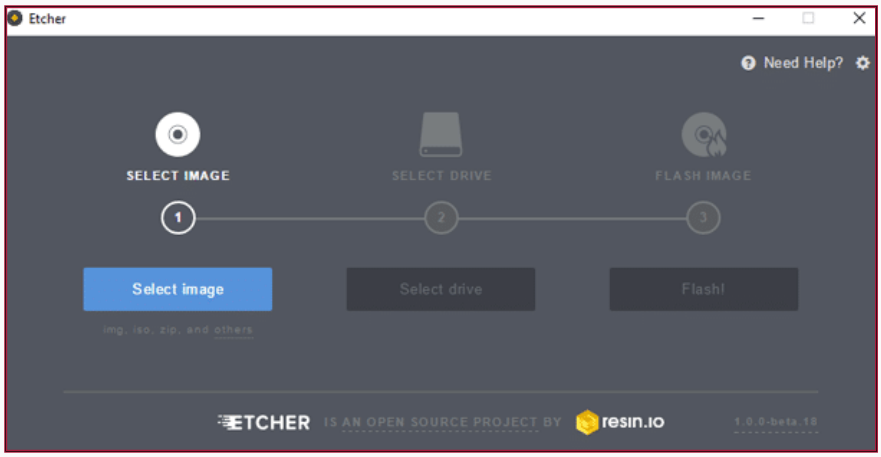
**1.** First head over to the Pi Music Box's Github at **https://volumio.org/get-started/** and download the latest zip file.

Once downloaded, unzip the file so, you are left with a **.img** file.

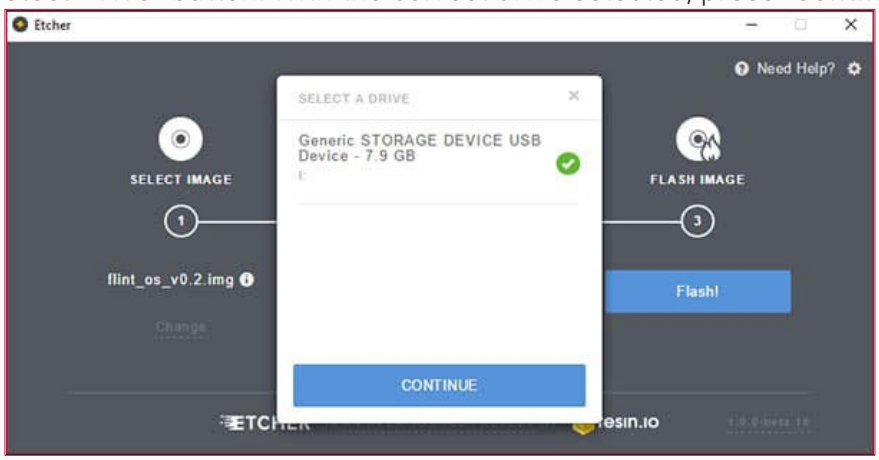
**2.** Next, you will need some tools to be able to write the image to the SD card.

For this we will use a tool called Etcher, you can download at https://etcher.io/.

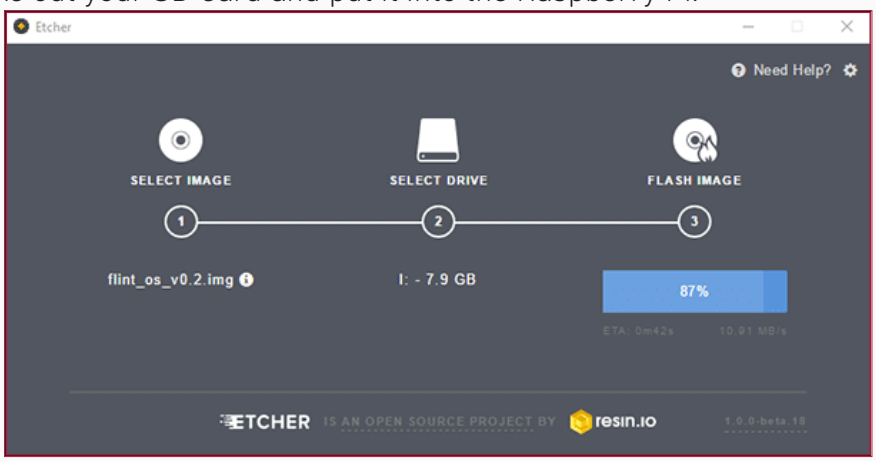
**3.** Open Etcher and press the "**Select Image**" button. Browse to the image you downloaded in **Step 1** and select it.




**4.** Next click the "**Select Drive**" button. With the correct drive selected, press "**Continue**".




**5.** Finally, click the "**Flash!**" button to begin the flashing process. This can take some time. Once completed, take out your SD Card and put it into the Raspberry Pi.



# Setting up Wi-Fi on Volumio

**1.** Now that you have your Raspberry Pi turned on, Volumio should have automatically started.

Please note that the first boot of Volumio can take up to 6 minutes to finish as it does the initial setup process.

Now here is the point that the tutorial splits up, if you are using an ethernet connection you can skip this section.

Otherwise, if you want to use Wi-Fi, you will need to complete the following steps to setup Wi-Fi.

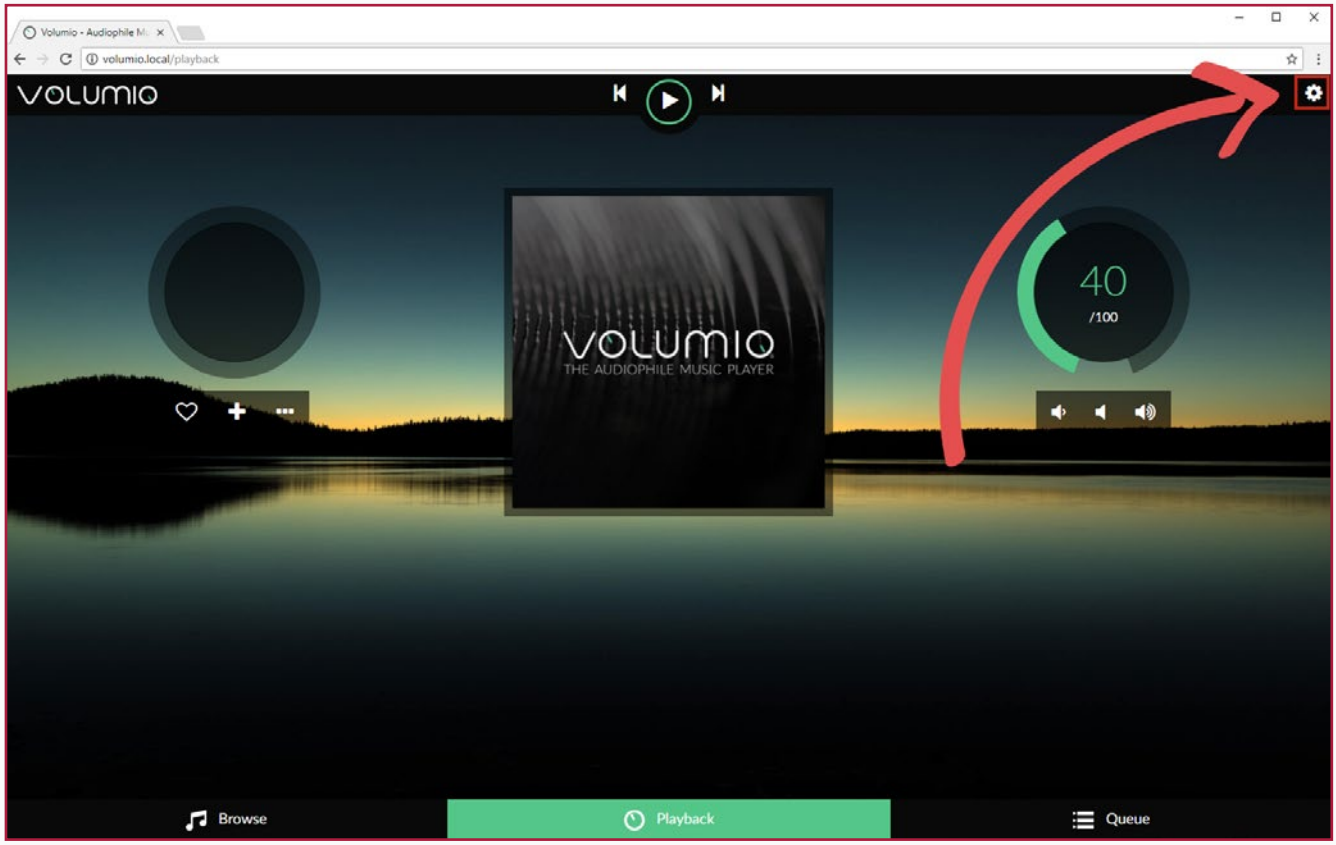
To set up Wi-Fi Volumio generates its temporary Wi-Fi hotspot under the name "**Volumio**".

You will need to connect to this on your computer to proceed with this tutorial when prompted for the password type in "**volumio2**".

**2.** Once you have made a connection to Volumio's temporary Wi-Fi hotspot, you then need to go to http://volumio.local in your favorite web browser.

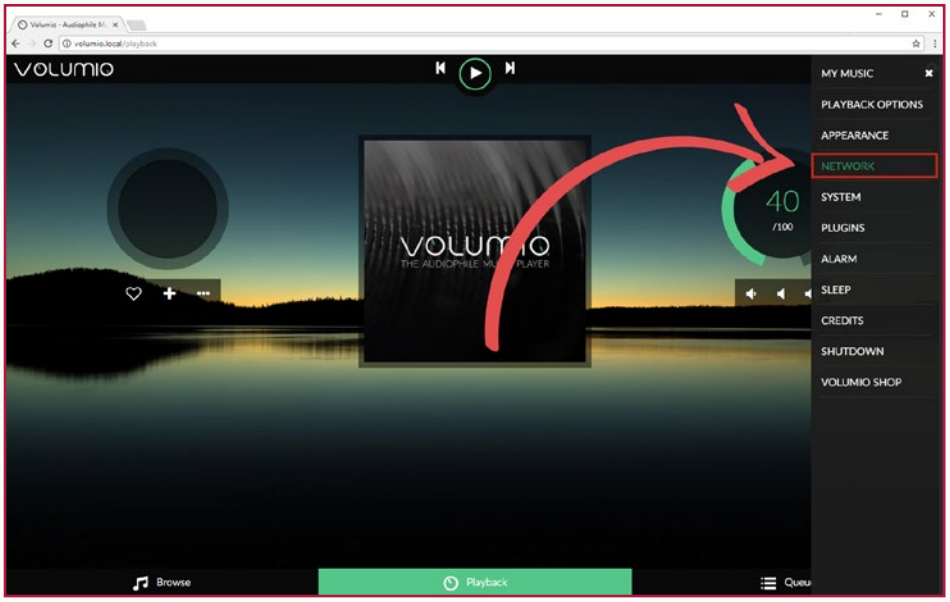
Once you have browsed to the address, you should be greeted with the following screen.

On this screen, **click** the **cog** in the top right-hand corner to bring out the navigation menu.



# Setting up Wi-Fi on Volumio

**3.** Now on the navigation menu, click "**Network**" as highlighted below.
This will take us to the screen where we can finally setup our Wi-Fi.



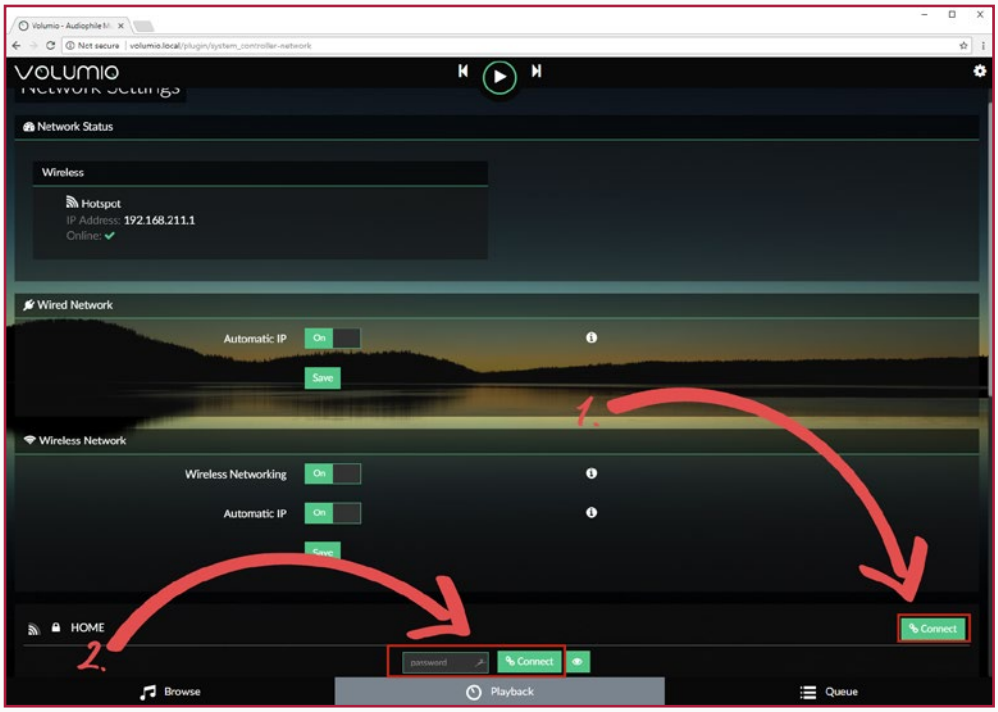

**4.** On the network screen, scroll down till you see the "**Wireless Network**" section.

Once you have found your Wi-Fi connection, click the "**Connect**" button (**1.**)

After clicking the first "**Connect**" button, you will then be asked to enter the password for your Wi-Fi connection.

Make sure you enter your password correctly here as this will disable the Wi-Fi hotspot, you can click the button with the eye icon to be able to see what you are typing.

Once you are certain, you have typed in the password correctly press the second "**Connect**" button (**2.**).



## Setting up Wi-Fi on Volumio

**5.** You will now be automatically disconnected from the Wi-Fi hotspot as Volumio reconfigures itself to connect to your Wi-Fi Network.

You should now be able to successfully connect back to your internet connection and again browse to https://volumio.local.

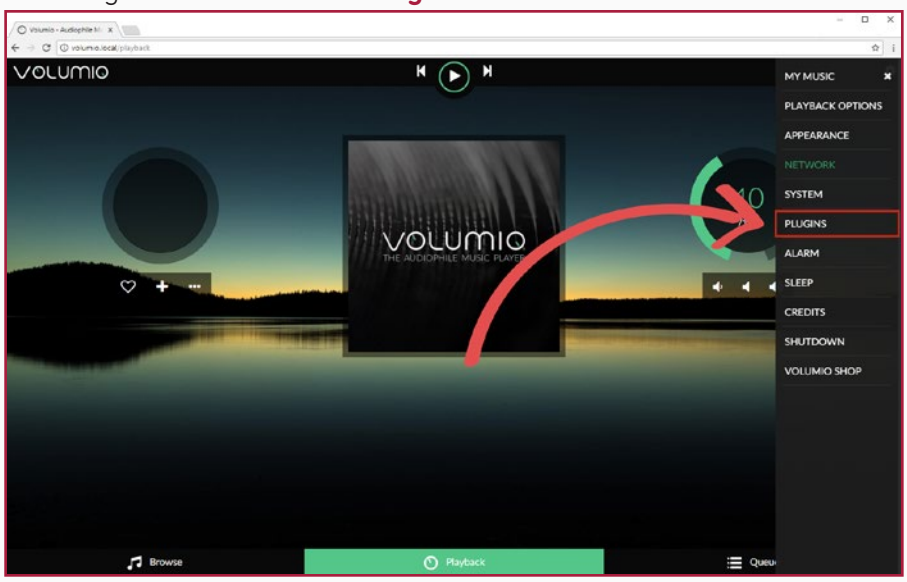
If it doesn't load straight away, then give it a few minutes so your Raspberry Pi can finish reconfiguring itself.

## Setting up Spotify on Volumio

**1.** Now that we have our internet connection up and running on Volumio one of the first things you will likely want to do set it up with Spotify.

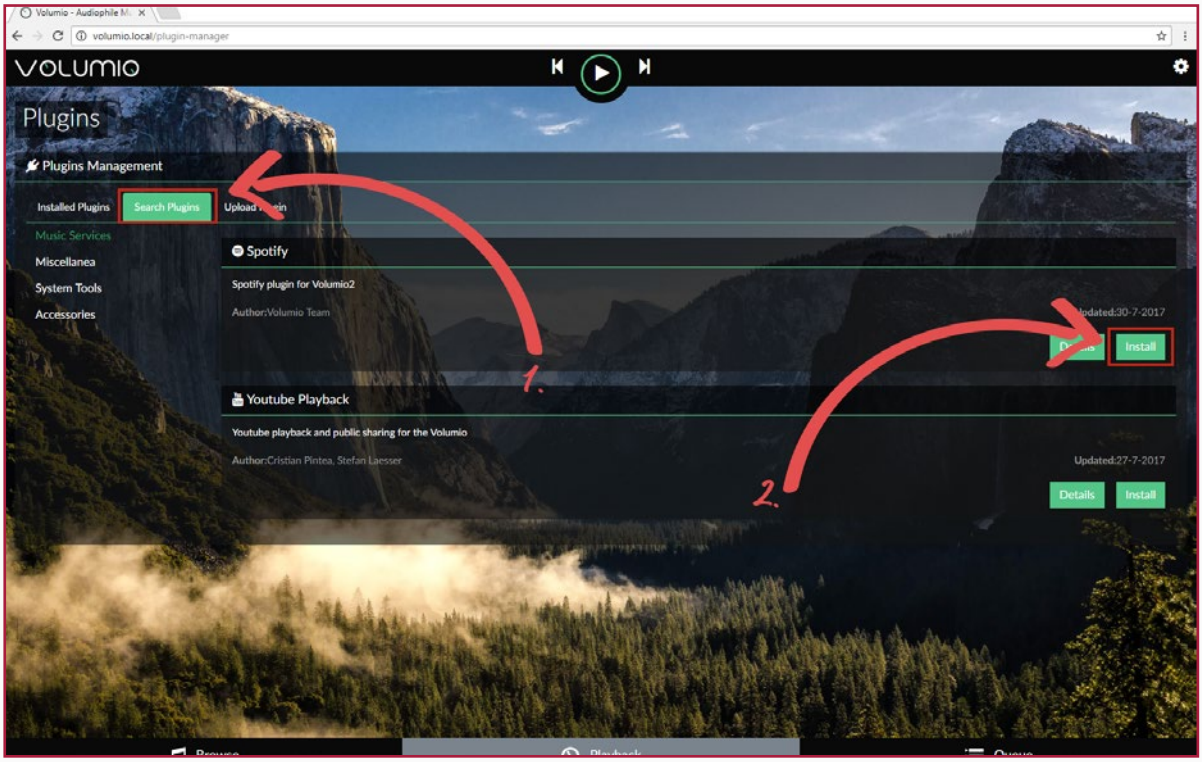
To be able to do this again press the cog in the top right-hand corner of the screen as we have shown below.

**2.** Now again on the navigation menu click "**Plugins**" as we have shown below.



# Setting up Spotify on Volumio

**3.** On the **Plugins Management** screen click the "**Search Plugins**" tab (**1.**), then the "**Install**" button (**2.**) next to Spotify.

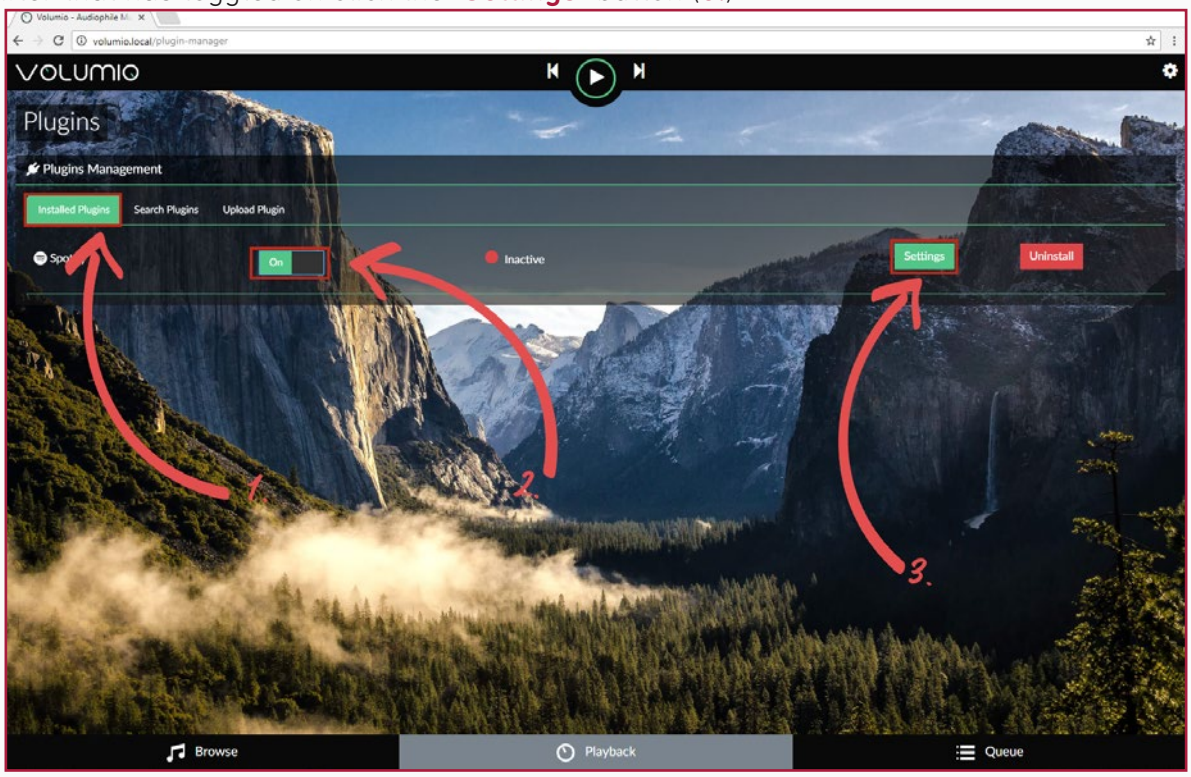



**4.** After the installation process has completed.

Click the "**Installed Plugins**" tab (**1.**).

On this new tab, click the "**On/Off**" Button (**2.**) to toggle the plugin on.

Finally, after that has toggled on click the "**Settings**" button (**3.**)

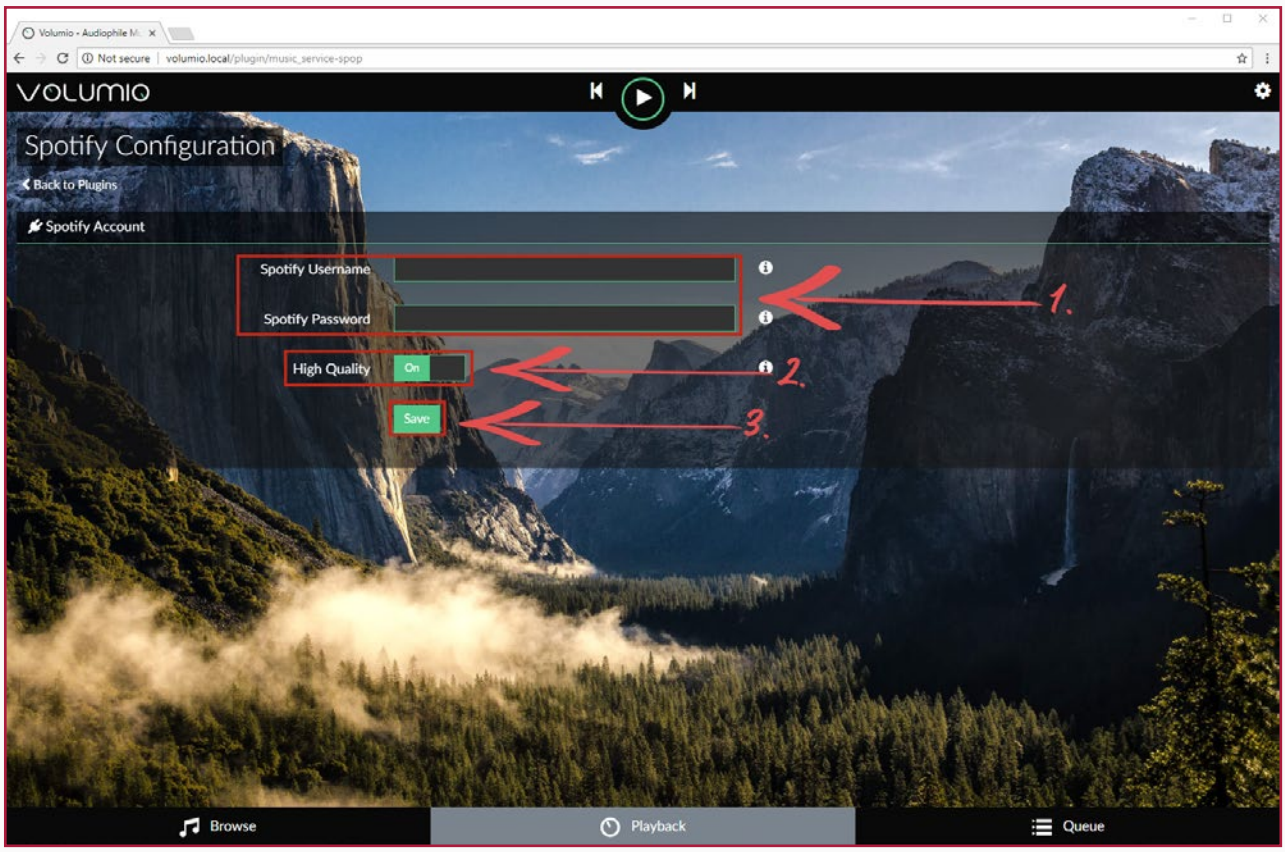


# Setting up Spotify on Volumio

**5.** Begin by filling out your Spotify Username and your Spotify Password in the first two text boxes (**1.**)

Afterward, you must disable "**High Quality**" (**2.**) if you are running a free Spotify account, it should be safe to leave on for a paid account.

Once you have set your Spotify Username and Spotify Password and set High Quality to on or off you can finally press the "**Save**" (**3.**) Button.




After pressing the save Volumio button, Volumio should automatically make a connection with Spotify and configure itself to be able to load songs from it.

Please note if you are using 2-Factor authentication this may fail to work, you will first have to generate an App Password and use that instead as Volumio does not support logins through 2-Factor.

You should now be able to go back to the main menu and click browse in the bottom right-hand corner.

On here you will be able to see all your available music that is located on any plugged in USB devices, you will also now be able to see a category called Spotify, on here you can find all your playlists and have access to all songs on Spotify.

# Viewing RTMP Streams

## Project Description

In this tutorial, we will be showing you how to view RTMP streams on your Raspberry Pi using three simple pieces of software.

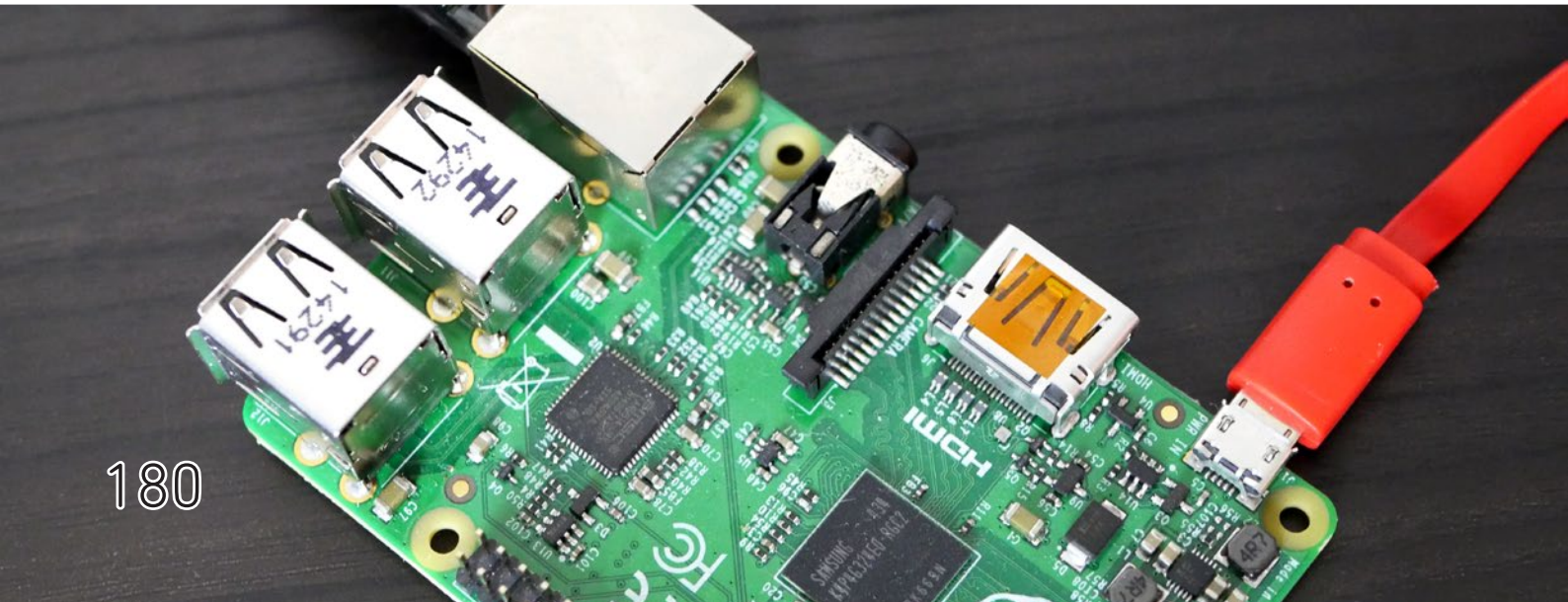
For those who do not know, RTMP stands for the Real-Time Messaging Protocol. It was initially a proprietary protocol developed by Macromedia for streaming audio, video and data over the Interface, primarily between there Flash player technology and a server.

However, when Adobe purchased Macromedia, they released an incomplete version of the specification of the protocol allowing the development of software such as rtmpdump and allowing other companies to be able to implement the protocol in their software.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection (Wifi Recommended)

**Optional**
- Raspberry Pi Case
- USB Keybaord
- USB Mouse



# Viewing RTMP Streams on the Raspberry Pi

**1.** Before we get started with setting up the Raspberry Pi to view RTMP Streams, we will first need to ensure our Raspberry Pi is completely up to date.

We can do this by running the following two commands on the Raspberry Pi.

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

**2.** With the Raspberry Pi up to date, we can now proceed with installing the software that we will be utilizing.

There is three pieces of software that we require, the first of these being **rtmpdump**.

**rtmpdump** is a toolkit designed to handle RTMP streams. We will use this to connect to the RTMP server and capture the stream from the network.

While you would typically pass this stream to a file, we will be passing it to a video player.

The second crucial piece of software that we will be utilizing for viewing RTMP streams on our Raspberry Pi is **mkfifo**.

This piece of software allows us to create a specially named pipe to which we will be dumping out RTMP stream to.

The third and final piece of the puzzle is **omxplayer**. This software is the video player that we will be utilizing to playback our RTMP stream on our Raspberry Pi.

Omxplayer is a video player that is specifically made for the Raspberry Pi's GPU, so it is perfect for our task.

Two of these may require installing, while omxplayer comes on most installations of Raspbian some lite versions of it may exclude it so we will include it in our installation command.

Run the following command on your Raspberry Pi to install rtmpdump and omxplayer.

```
sudo apt-get install omxplayer rtmpdump -y
```

**3.** Now with the software that we need installed, we can begin setting up everything we need. The first thing we need to do is create a named pipe.

This named pipe is essentially a FIFO (First in First Out) memory buffer that we can reference by a name. We will be dumping our RTMP stream into this pipe.

To create this named pipe we need to run the following command on the Raspberry Pi. In our example we will be naming our pipe "**rtmp_pipe**", but you can name this anything you want.

If you change this, then remember to change everything instance of rtmp_pipe in this tutorial with it.

```
mkfifo rtmp_pipe
```

## Viewing RTMP Streams on the Raspberry Pi

**4.** Finally, we get to the interesting part of the stream, and that is dumping and playing back our RTMP Stream.

For this example, we will be using "**rtmp://184.72.239.149/vod/mp4:BigBuckBunny_115k.mov**" as our example stream. This link is a stream of Big Buck Bunny by the Blender Institute.

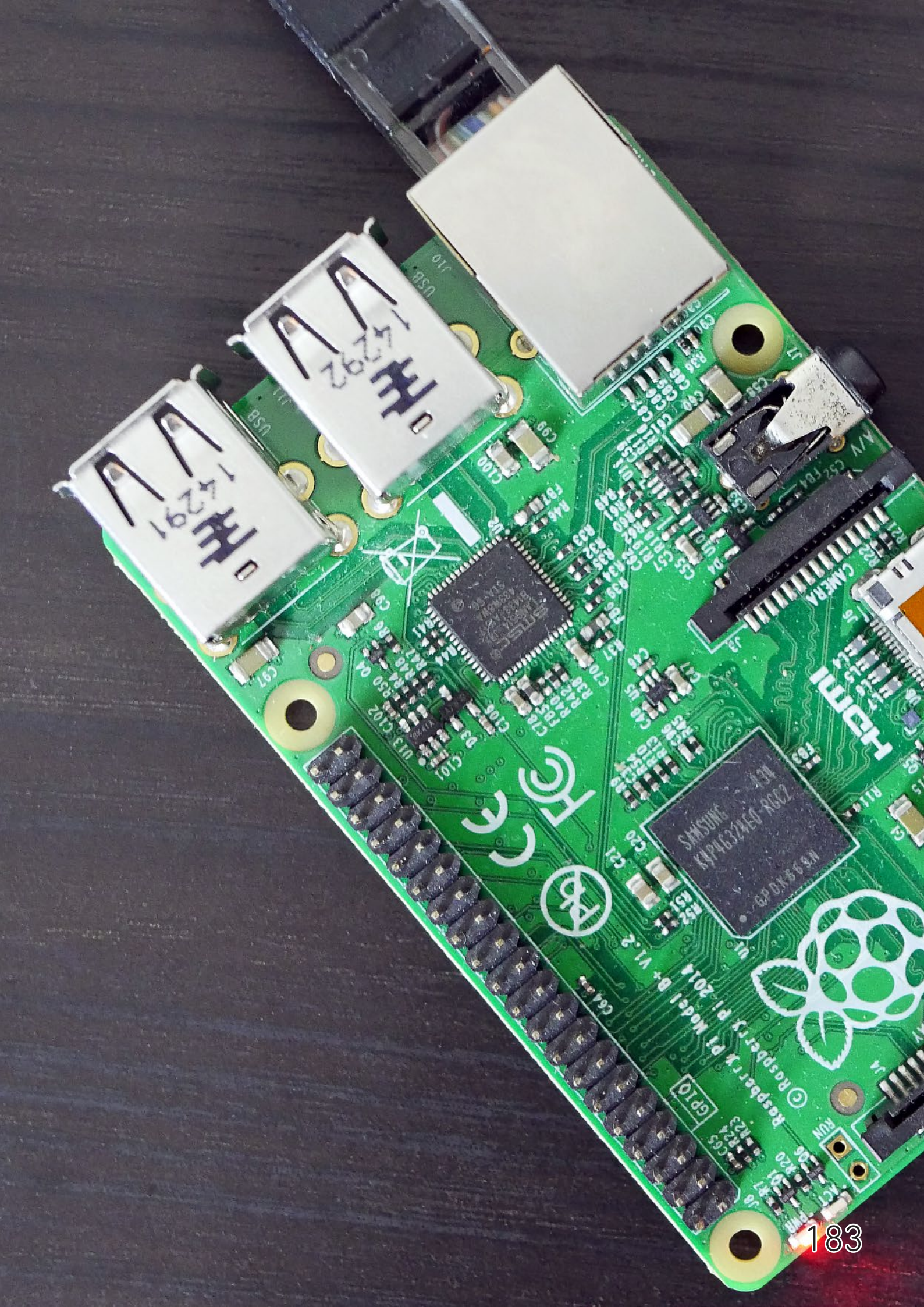
To view this stream, we will be passing it into rtmpdump. This program will then dump the stream into our pipe, which will then be loaded in by omxplayer.

Run the following command on your Raspberry Pi's terminal.

```
rtmpdump -r rtmp://184.72.239.149/vod/mp4:BigBuckBunny_115k --live -o rtmp_pipe  | omxplayer rtmp_pipe
```

**5.** The stream should now be loaded in omxplayer and be playing on your screen.

If the video successfully loads and plays then you have completed this tutorial correctly.


SAMSUNG
K4P4G324EQ-RGC2
Raspberry Pi Model B+ V1.2
HDMI
CAMERA

# Kiosk Mode
## With Chromium

## Project Description

For this tutorial, we will be showing you how you can set up your Raspberry Pi as a Kiosk using the popular web browser, Chromium.

We utilize Chromium as it is one of the best-supported web browsers and openly supports the functionality to act in a kiosk mode. It is also easy to control through keypresses which we can simulate using the xdotool that we install during this guide.

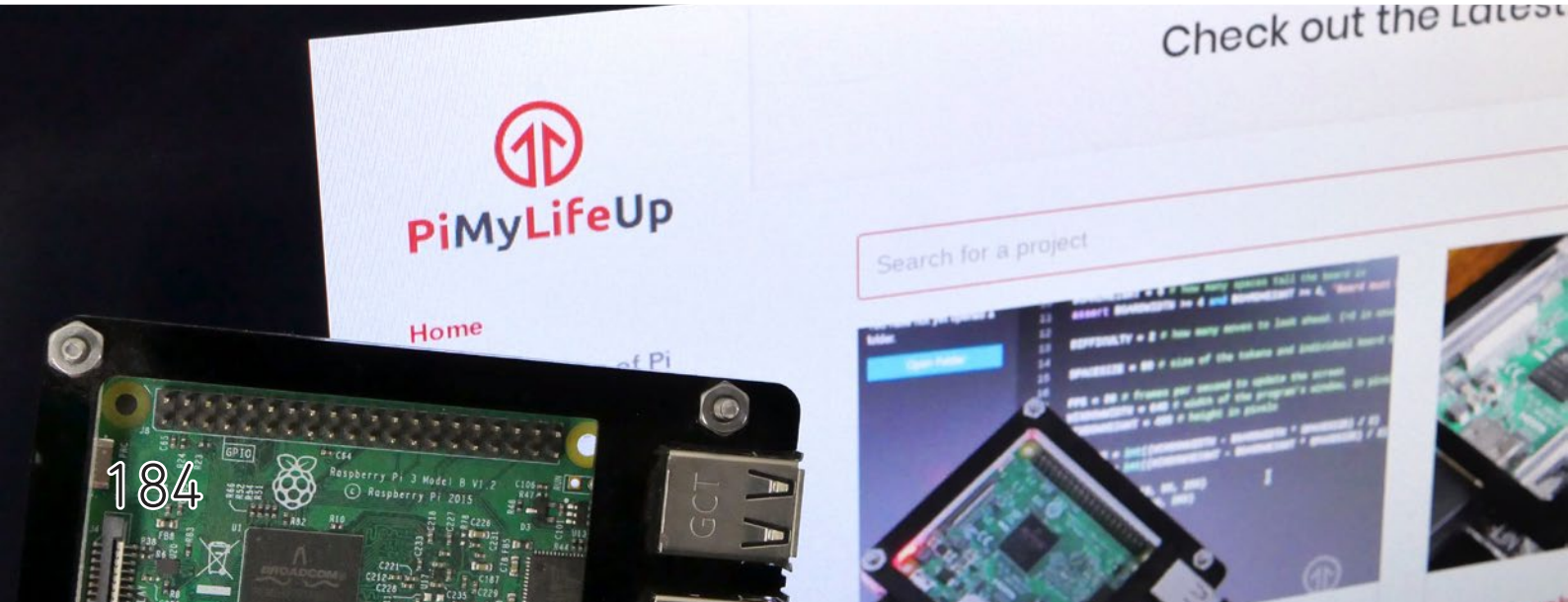
This guide will give you a good idea on how to use autologin tasks and simple bash scripts to perform numerous tasks.

Please note to do this tutorial you will need to be running the full version of Raspbian as this relies on the GUI that comes with it to properly operate.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection (Wifi Recommended)
- Screen

**Optional**
- Raspberry Pi Case
- USB Keybaord
- USB Mouse



## Preparing Raspbian for the Chromium Kiosk

**1.** Before we get started with this tutorial, we will be first removing some packages that we don't need for our Raspberry Pi kiosk.

Removing these packages will free up some much-needed memory and reduce the number of packages that will be updated every time you update your Raspberry Pi.

To do this just run the following three commands on your Raspberry Pi. We have split these into three different commands to make them easier to copy and write out.

```
sudo apt-get purge wolfram-engine scratch scratch2 nuscratch sonic-pi idle3 -y
sudo apt-get purge smartsim java-common minecraft-pi libreoffice* -y
```

**2.** Now that we have removed these numerous packages from our Raspberry Pi Kiosk we will need to run some cleanup commands.

To remove any unnecessary lingering packages and to clean out the local repository of retrieved packages run the following two commands on your Raspberry Pi.

```
sudo apt-get clean
sudo apt-get autoremove -y
```

**3.** Now that we have removed the bulk of the unnecessary applications, we must now make sure that our installation of Raspbian is up to date.

We can use the following two commands within the terminal on our Raspberry Pi to update all the packages to their latest versions.

```
sudo apt-get update
sudo apt-get ugprade
```

**4.** We now need also to install xdotool. This tool will allow our bash script to execute key presses without anyone being on the device. We also install the unclutter package, this will enable us to hide the mouse from the display.

Just run the following command on your Raspberry Pi to install the package.

```
sudo apt-get install xdotool unclutter sed
```

**5.** Now that those packages have been installed we can now proceed to the next stage of this tutorial. That is setting up Raspbian to auto login to our user. Having to log in every time for a kiosk would be an annoyance.

Desktop autologin is the default behavior but if for some reason you have changed it follow the next few steps to switch it back. Otherwise, skip to **Step 6** of this tutorial.

**5a.** Run the following command on your Raspberry Pi to load up the Raspi-config tool. We will be using this tool to enable auto login.

```
sudo raspi-config
```

**5b.** Now within the tool go to **3 Boot Options** -> **B1 Desktop / CLI** -> **B4 Desktop Autologin**

**5c.** Desktop autologin should now be enabled and you can safely quit out of the **raspi-config** tool.

## Writing the Chromium Kiosk Script

**1.** Now that we have prepared the Raspbian operating system to make it more light weight and have also enabled desktop autologin we can go ahead and write our kiosk script.

The kiosk script will handle the bulk of the work for our Raspberry Pi Kiosk, including launching Chromium itself and also simulating key presses.

Begin writing our kiosk bash script by running the following command on the Raspberry Pi.

```
nano /home/pi/kiosk.sh
```

**2.** Within this file enter the following lines of code, we will explain the important parts of the script so you can better bend it to your needs.

```
#!/bin/bash
```

The very first line defines what the command line interpreter (CLI) should use to try and execute this file. This is useful for cases where you don't want to have to specify the specific application required every time you run the script.

```
xset s noblank
xset s off
xset -dpms
```

These three lines are pretty important as they help us stop the Raspberry Pi's display power management system from kicking in and blanking out the screen.

Basically, these three commands set the current xsession not to blank out the screensaver and then disables the screensaver altogether.

The third line disables the entire "display power management system" meaning that the desktop interface should never blank out the screen.

```
unclutter -idle 0.5 -root &
```

This line runs the program we installed earlier called **unclutter**. This application will hide the mouse from the display whenever it has been idle for longer then **0.5 seconds** and will remove it even if it is over the root background.

You can adjust the idle timer to the number of seconds you want with each decimal place being a fraction of a second.

If you would prefer to remove the mouse instantly, then remove the **-idle 0.5** from the command.

```
sed -i 's/"exited_cleanly":false/"exited_cleanly":true/' /home/pi/.config/chromium/Default/Preferences
sed -i 's/"exit_type":"Crashed"/"exit_type":"Normal"/' /home/pi/.config/chromium/Default/Preferences
```

These two lines of the script use sed to search through the Chromium preferences file and clear out any flags that would make the warning bar appear, a behavior you don't really want happening on your Raspberry Pi Kiosk.

If Chromium ever crashes or is closed suddenly, the lines above will ensure you don't have to get hold of a mouse and keyboard to clear the warning bar that would typically appear at the top of the browser.

## Writing the Chromium Kiosk Script

```
/usr/bin/chromium-browser --noerrdialogs --disable-infobars --kiosk \
https://pimylifeup.comhttps://arduinomylifeup.com &
```

This line launches Chromium with our parameters. We will go through each of these parameters so you know what you can modify, and how you can modify it.

### --kiosk

This flag sets Chromium to operate in Kiosk mode, this locks it into a particular set of features and only allows limited access to both the web browser and any other OS functionality.

Chromium's kiosk functionality takes full control of the screen, maximizing Chromium to the full size of your screen and stops user input from being accepted by the OS, effectively trapping the end user within a sandbox.

### --noerrdialogs

This option tells Chromium that it should not display any of its error dialogs to the end user.

It is crucial if you don't want the end user to know if anything has or is going wrong with Chromium, this goes alongside our code to clear the "**exited_cleanly**" and "**exit_type**" state earlier in the code.

### --disable-infobars

We use this to disable Chromium from displaying its info bar to the end user.

The info bar can be used by Chromium to notify them of certain things such as that Chromium is not their default web browser.

Of course, as we are using this as a kiosk, we don't need the user to know any information that Chromium might want to display.

### https://pimylifeup.com https://arduinomylifeup.com

These are the two web pages that the script will open, each will be opened in a new tab. You can add additional web pages/tabs by adding to this list by separating each one with a space.

```
while true; do
    xdotool keydown ctrl+Tab; xdotool keyup ctrl+Tab;
    sleep 15
done
```

These lines run a very simple infinite while loop that uses **xdotool** to mimic pressing **CTRL + Tab**. Making Chromium switch to the next tab.

After **xdotool** has executed its key presses, it then puts the loop to **sleep for 15 seconds**. To change how long the loop sleeps for before it executes the xdotool command again just change the **sleep 15** command.

You can also use this method to add a screen refresh, this may be important when you want to display live scores.

The command to refresh the webpage should look something like what we have shown below, you can swap out the tab switching code with this, or utilize both of them.

```
xdotool keydown ctrl+r; xdotool keyup ctrl+r;
```

## Writing the Chromium Kiosk Script

**3.** Once you have entered all the code for our Raspberry Pi kiosk script, it should look somewhat similar to what we have displayed below.

```
#!/bin/bash
xset s noblank
xset s off
xset -dpms

unclutter -idle 0.5 -root &

sed -i 's/"exited_cleanly":false/"exited_cleanly":true/' /home/pi/.config/chromium/Default/Preferences
sed -i 's/"exit_type":"Crashed"/"exit_type":"Normal"/' /home/pi/.config/chromium/Default/Preferences

/usr/bin/chromium-browser --noerrdialogs --disable-infobars --kiosk \
https://pimylifeup.com https://arduinomylifeup.com &

while true; do
  xdotool keydown ctrl+Tab; xdotool keyup ctrl+Tab;
  sleep 10
done
```

**4.** Once you are sure that everything is correct, save the file by pressing **CTRL + X** then **Y** and finally **ENTER**.

**5.** You can now verify whether the kiosk script is working correctly by running the following command on your Raspberry Pi. Make sure you do this in desktop mode and not over SSH.

```
sudo bash /home/pi/kiosk.sh
```

**6.** If everything is working as intended the Chromium web browser should open up and take over your desktop, the mouse should dissapear and the tab should switch every 10 secconds.

In the next section we will show you how to bring this alltogether and have the **kiosk.sh** script launched at startup.

## Bringing it altogether

**1.** Now that we have enforced desktop auto login and written our kiosk script, we need to go ahead and modify the lxsession autostart script by running the following command on the Raspberry Pi.

```
nano /home/pi/.config/lxsession/LXDE-pi/autostart
```

**2.** As we don't want any screensavers to run, we definitely don't want the xscreensaver application to be getting launched on our Raspberry Pi Kiosk.

So, the first thing we need to do in this file is to comment out a particular line that launches up the xscreensaver application.

### Find

```
@xscreensaver -no-splash
```

### Replace With

```
@xscreensaver -no-splash
```

**3.** Now that you have commented out the xscreensaver line you need to enter the following lines to the bottom of the file.

```
@bash /home/pi/kiosk.sh
```

This line that we are going to add will trigger a batch script that we wrote earlier in this tutorial.

**4.** With all the code entered into the **autostart** file, it should end up looking like what we have shown below.

If you are happy with what is there, then you should save the script by pressing **CTRL + X** then **Y** and then finally **ENTER**.

```
@lxpanel --profile LXDE-pi
@pcmanfm --desktop --profile LXDE-pi
#@xscreensaver -no-splash
@point-rpi
@bash /home/pi/kiosk.sh
```

**5.** Finally, let's restart our Raspberry Pi. Upon reboot, Chromium will be opened up in Kiosk mode, and your chosen websites will be displayed.

Hopefully, by this point, everything is working exactly as intended and you have tested the script in the last section.

Restart the Raspberry Pi by running the following command.

```
sudo reboot
```

**6.** Now with everything up and running correctly, if there is for any reason you want to stop the script from running, you can utilize the following command.

```
sudo killall kiosk.sh
```

This command will kill any instances of the **kiosk.sh** script that is currently running.

In the next section we will show you how to enforce the resolution on your Raspberry Pi.

## Enforcing a resolution

**1.** One thing you might want to do is to enforce the resolution that the Raspberry Pi is going to use. Setting the resolution can be quite handy as the Raspberry Pi's built-in detection can sometimes be a bit flakey.

To begin setting the resolution, we must first load up the Raspberry Pi configuration tool by running the following command.

```
sudo raspi-config
```

**2.** Within the configuraiton tool, you will want to start by going to "**7 Advanced Options**".

**3.** Now that we are in the advanced options section you should see an option labeled "**A5 Resolution**" select that option.

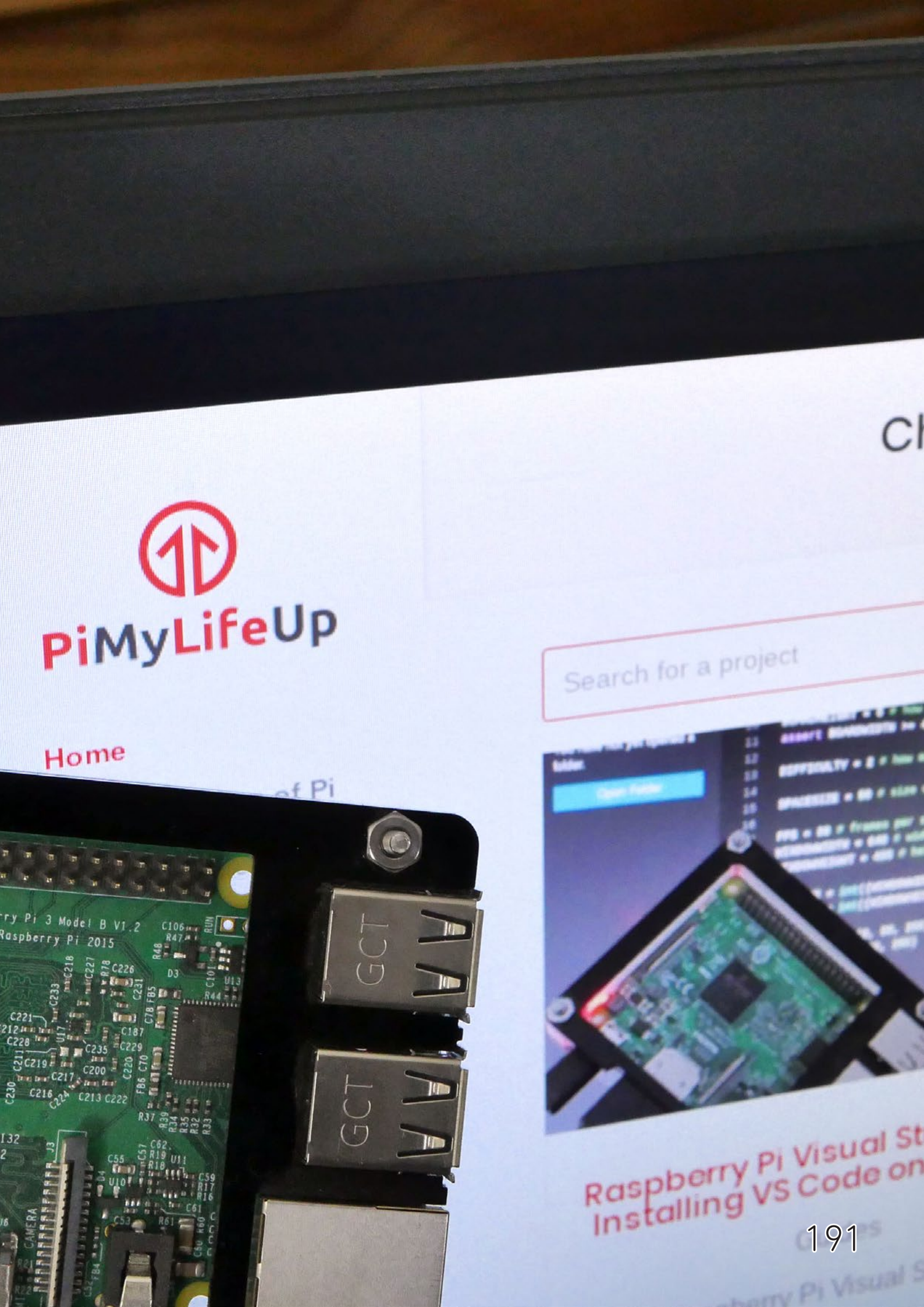
**4.** Within here find and select the resolution that best fits your screen and press **ENTER**.

**5.** With your resolution now set you will need to restart the Raspberry Pi.

Do this by first exiting out of the configuration tool by pressing **ESC**, then entering the following command into the Raspberry Pi's terminal.

```
sudo reboot
```

**6.** Your Raspberry Pi should now restart and be running at the specified resolution.


PiMyLifeUp
Search for a project
Home
Raspberry Pi Visual St
Installing VS Code on
Raspberry Pi 3 Model B V1.2
Raspberry Pi 2015
GCT
GCT

# Raspberry Pi Chromecast

## Project Description

In this tutorial, we will be showing you how you can set up your Raspberry Pi to act as a Chromecast device. While we can't implement the same protocol that Chromecast relies on we can replicate this on the Raspberry Pi.

We replicate its behavior by using two key pieces of software on the Raspberry Pi. The first of these pieces of software is omxplayer. This software handles the video and audio files that are cast to our Raspberry Pi.

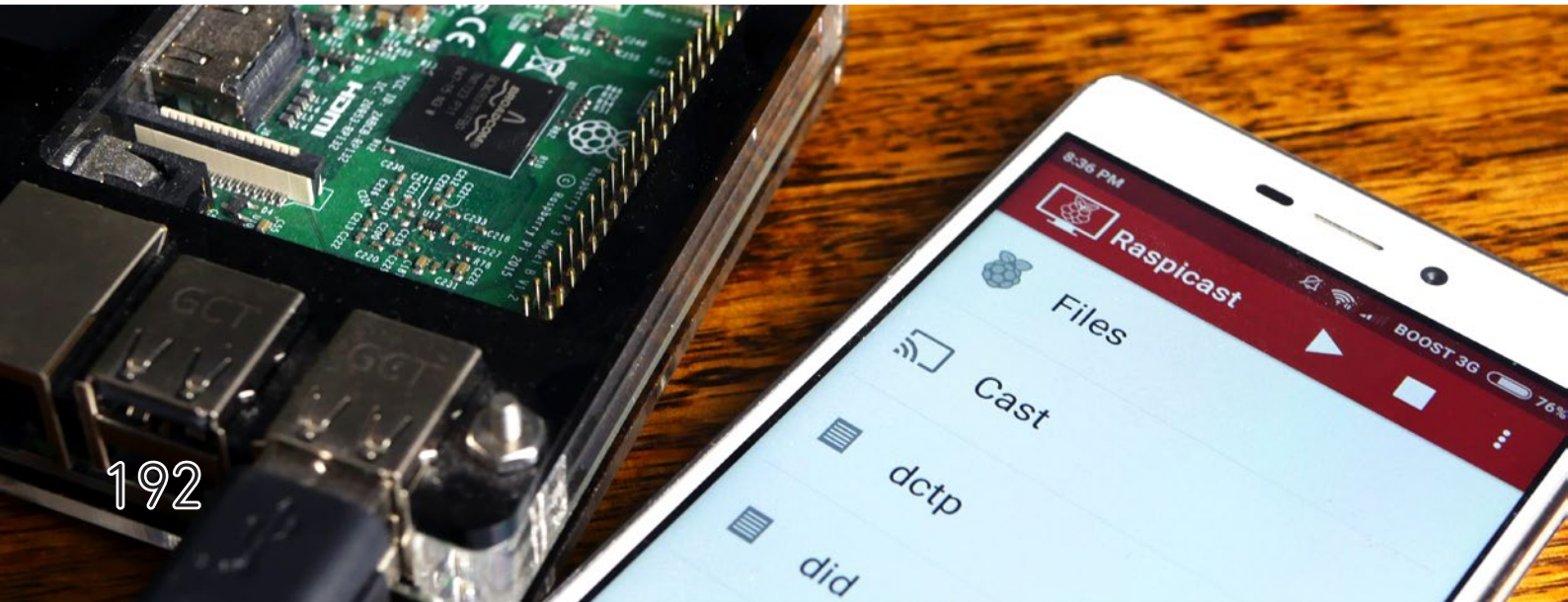
Our second piece of software is the OpenMax, Image viewer. We utilize this piece of software to handle images sent to the device.

Lastly, we rely on an Android application called Raspicast. This application is what we will utilize to cast videos, songs, and images to the Raspberry Pi.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection (Wifi Recommended)

**Optional**
- Raspberry Pi Case
- USB Keybaord
- USB Mouse



## Setting up Raspbian to operate as a Chromecast

**1.** Now before we get started with setting up our Raspberry Pi as a Chromecast, we must first ensure that our Raspberry is actually up to date and that we have the latest package list.

We can update our Raspberry Pi by running the following two commands on it.

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

**2.** For this tutorial, we will require to use GIT and Make. While these packages should be pre-installed on Raspbian, we will run the command below just to make sure they are.

To ensure this is installed just run the following command on the Raspberry Pi.

```
sudo apt-get install git make -y
```

**3.** With our Raspberry Pi now up to date, we need first to make sure we have OMXPlayer installed, while Raspbian usually comes with this installed we will make sure it exists anyway.

The reason we need OMXPlayer is so that it can handle video/audio processing for any video or audio files that we cast to our Raspberry Pi Chromecast.

One advantage of OMXPlayer to other video and audio players is that it was designed specifically for the Raspberry Pi's GPU meaning it offers some of the best performance, something that is critical for any files that are cast.

Run the following command to install OMXPlayer on to your Pi.

```
sudo apt-get install omxplayer -y
```

**4.** With OMXPlayer now installed to our Raspberry Pi, we need to move onto setting up the next piece of the pie. While we now have a piece of software ready that can handle our video and audio files we still need something that can handle images.

For the task of handling images on our Raspberry Pi Chromecast, we will be installing a piece of software that is called OpenMax Image Viewer.

OpenMax Image Viewer is a must for this project as it is a GPU accelerated image viewer explicitly designed for the Raspberry Pi's GPU, meaning we can display any images as fast as possible and reduce the load on the Raspberry Pi's CPU.

To begin the process of setting up OpenMax Image Viewer, we must first clone its code repository from Github.

Just run the following two commands on your Raspberry Pi to clone this repository to your Raspberry Pi.

```
cd ~
git clone https://github.com/HaarigerHarald/omxiv.git
```

## Setting up Raspbian to operate as a Chromecast

**5.** With the OpenMax Image viewers code now cloned to our Raspberry Pi, we must now install a couple more packages.

These two packages that we need are libjpeg8-dev and libpng12-dev. These packages contain development libraries for the PNG format and the JPEG format, both which are required to compile the code.

Let's download and install these packages to our Raspberry Pi by running the following command.

```
sudo apt-get install libjpeg8-dev libpng12-dev
```

**Note:** If for some reason Raspbian fails to find the packages, try reunning sudo apt-get update.

**6.** We can now finally proceed on to the steps to compiling and installing OpenMax Image viewer to our Raspberry Pi Chromecast device.

To start off, we must first change into the directory where we cloned our code, do this by running the following command on our Raspberry Pi.

```
cd ~/omxiv
```

**8.** Now that we are in the right directory we need to initialize the compiling process now. Compiling will give us the files we need to run the OpenMax Image Viewer software for our Raspberry Pi Chromecast device.

Thanks to the **make** software, compiling the code is a very simple and straightforward process.

To do this run the following two commands on the Raspberry Pi.

```
make ilclient
make
```

**9.** With the OpenMax Image Viewer now compiled, there is one last command we must utilize.

This last command is what will setup OpenMax on our Raspberry Pi so that it is available to the general operating system.

Type the following command into the terminal on your Raspberry Pi, and you will be all set to proceed to the next section of our guide.

```
sudo make install
```

**10.** Now the last thing we need to do is grab our Raspberry Pi's local IP address.

Run the following command to grab to do it, make sure the '**I**' is capitalized otherwise this command won't work correctly.

```
hostname -I
```

# Using Raspicast to cast to the Raspberry Pi

**1.** With our Raspberry Pi now setup to act like a Chromecast, we now need to download an app to our Android device.

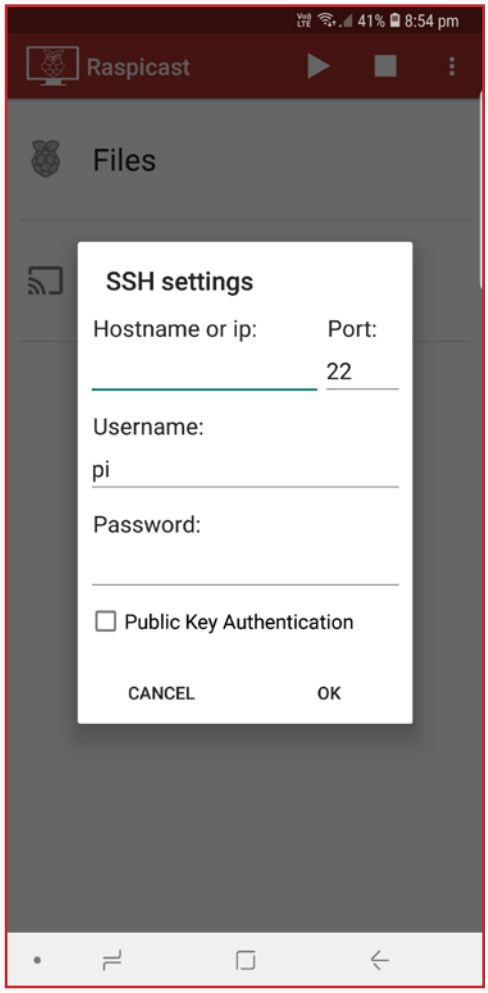
This application is Raspicast. Raspicast is the piece of software that will allow us to cast images and videos from our Android devices to the Raspberry Pi.

Either search up "**Raspicast**" on the Google Play Store or go to the following URL to go directly to the app on your device: https://play.google.com/store/apps/details?id=at.huber.raspicast&hl=en

Download, install and run the Raspicast application.

**2.** You should now be greeted with the following screen. This screen will ask you to enter your SSH details as it will use these to talk to your Raspberry Pi.

If you haven't already set up SSH on your Raspberry Pi, then you should do this now as it is required for the Raspicast software to talk with it.




**3.** Once you have entered the correct information, you should now be able to cast videos, song, and images from your Android device to the Raspberry Pi.

# Server
## Projects

# Wireless Print Server

## Project Description

In this tutorial, I will be looking at how to set up a Raspberry Pi Print Server. Getting the software installed is a simple process, but configuring it and getting a Windows network to find your print server is a bit more of an involved process.

Setting up a Raspberry Pi Print server is a simple way of bringing your regular USB printer to more modern standards without you having to shell out hundreds of dollars. It allows you to move the printer to whether you want and not have to connect it straight to any singular computer. The Print Server is also a good way to share a printer without needing a costly tower computer running all the time.

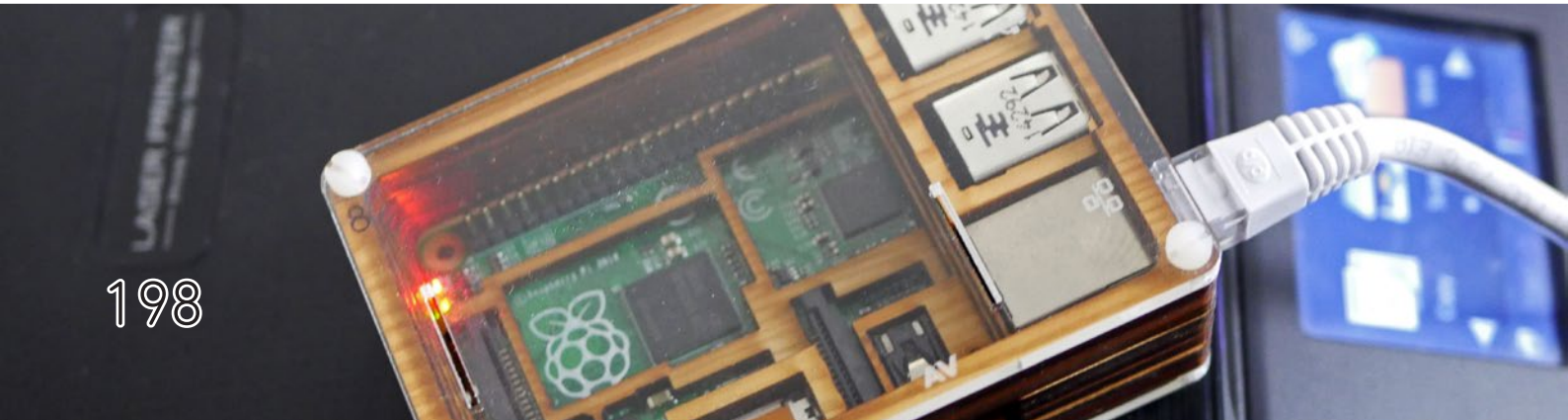
To make this all work, we will be making use of a piece of software called CUPS. CUPS stands for Common Unix Printing System and is the backbone of most Linux printing software. It is the software that communicates with your Printer and gets it to print files correctly.

If you want to make sure your Printer is supported by the CUPS printing software, then go to their website at http://www.openprinting.org/printers and look up your specific model number.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection
- USB Printer

**Optional**
- Raspberry Pi Case
- USB Keyboard
- USB Mouse



## Installing the Print Server Software

Installing the Print Server for the Raspberry Pi is a simple process since it is available through the Debian Jessie packages.

**1.** To get started we should first update the Raspberry Pi to ensure we are running the latest software by running the following commands into the terminal:

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

**2.** Once the Raspberry Pi has been updated, we can now go about installing the Print Server software. In this case, we will be installing CUPS.

This software manages printers connect via USB or over the network and has the bonus of providing a management interface you can view over the internet.

Install this software by typing the following command into the terminal:

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

**3.** When CUPS has finished installing there is are few extra things we will need to do.

The first of these is to add the **pi** user to the **lpadmin** group. This command will allow the **pi** user to access the administrative functions of **CUPS** without needing to use the superuser.

```
sudo usermod -a -G lpadmin pi
```

**4.** There is one other thing we need to do to CUPS to ensure it runs well on your home network. That is to make CUPS accessible across your whole network, at the moment it will block any non-localhost traffic.

We can get it to accept all traffic by running the following two commands:

```
sudo cupsctl --remote-any
sudo /etc/init.d/cups restart
```

**5.** Now we should be able to access the Raspberry Pi Print Server from any computer within your network.

If you are unsure of what your Raspberry Pi's local IP Address is then you can make use of the following command:

```
hostname -I
```

**6.** Once you have your Raspberry Pi's IP Address, go to the following web address in your favorite web browser, make sure to swap out my IP address (**192.168.1.105**) with your own.

```
http://192.168.1.105:631
```

Over the next few pages, we will look at setting up SAMBA correctly to ensure Windows can properly identify the Raspberry Pi Print Server.

We will also show you how to add a printer using the CUPS interface.

## Setting up SAMBA for the Pi Print Server

If you intend on using your Raspberry Pi Print Server with Windows, then setting up SAMBA correctly is necessary.

We will need to install SAMBA and make a few changes to its configuration to ensure that it runs correctly and utilizes CUPS print drivers.

**1.** Now firstly, we should make sure we have SAMBA installed, the easiest way to do this is to just issue an install command for it.

We can do that by running following command within the terminal on the Raspberry Pi:

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

**2.** With SAMBA now installed to our Raspberry Pi, we will need to open its configuration file and make several edits, we can open the file with the following command:

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

**3.** Now with the file open, we will need to scroll to the bottom of the file. The quickest way to do this is to use **Ctrl+V.**

Once at the bottom of the file you should add or change the following lines.

In my case, the **[printers]** and the **[print$]** sections were already in the file, so I just needed to change the values to match.

```
# CUPS printing.
[printers]
comment = All Printers
browseable = no
path = /var/spool/samba
printable = yes
guest ok = yes
read only = yes
create mask = 0700

# Windows clients look for this share name as a source of downloadable
# printer drivers
[print$]
comment = Printer Drivers
path = /var/lib/samba/printers
browseable = yes
read only = no
guest ok = no
```

Save the file by pressing **Ctrl+X** and then pressing **Y** and then **Enter**.

**4.** We can now restart SAMBA to get it to load in our new configuration, to do that, all we need to do is type the following command into the terminal:

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

# Adding a printer to CUPS

**1.** Adding a printer to CUPS is a rather simple process, but first, we need to load up the CUPS web interface on our Raspberry Pi.

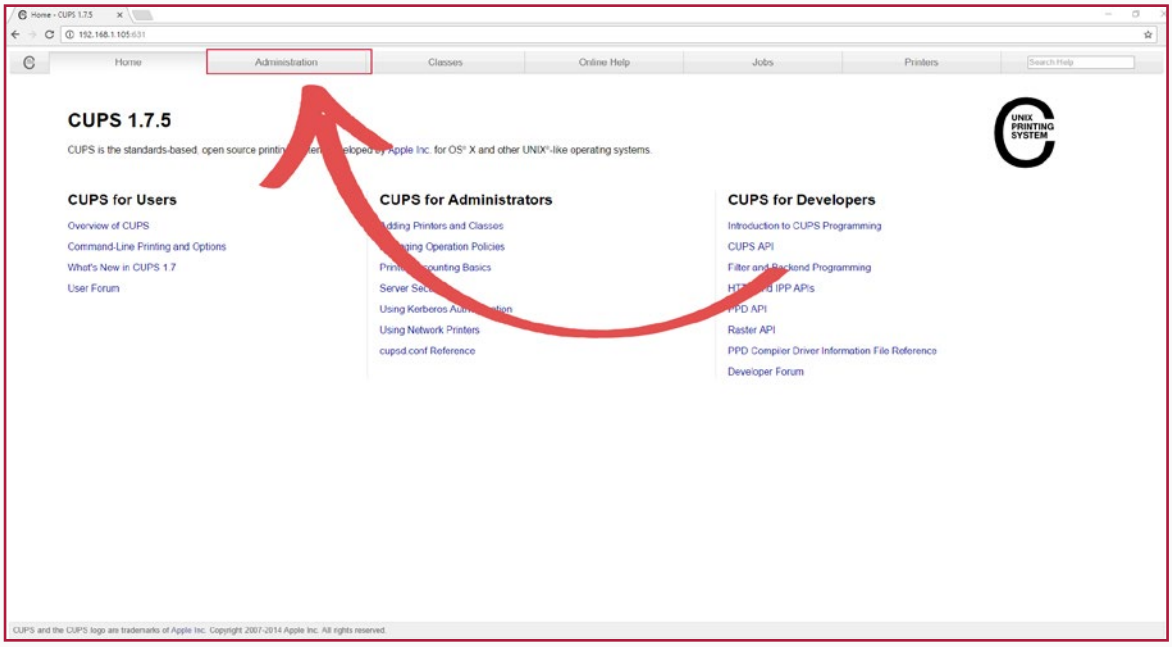
If you are unsure on your Raspberry Pi's IP address, then run the following command:

```
hostname -I
```

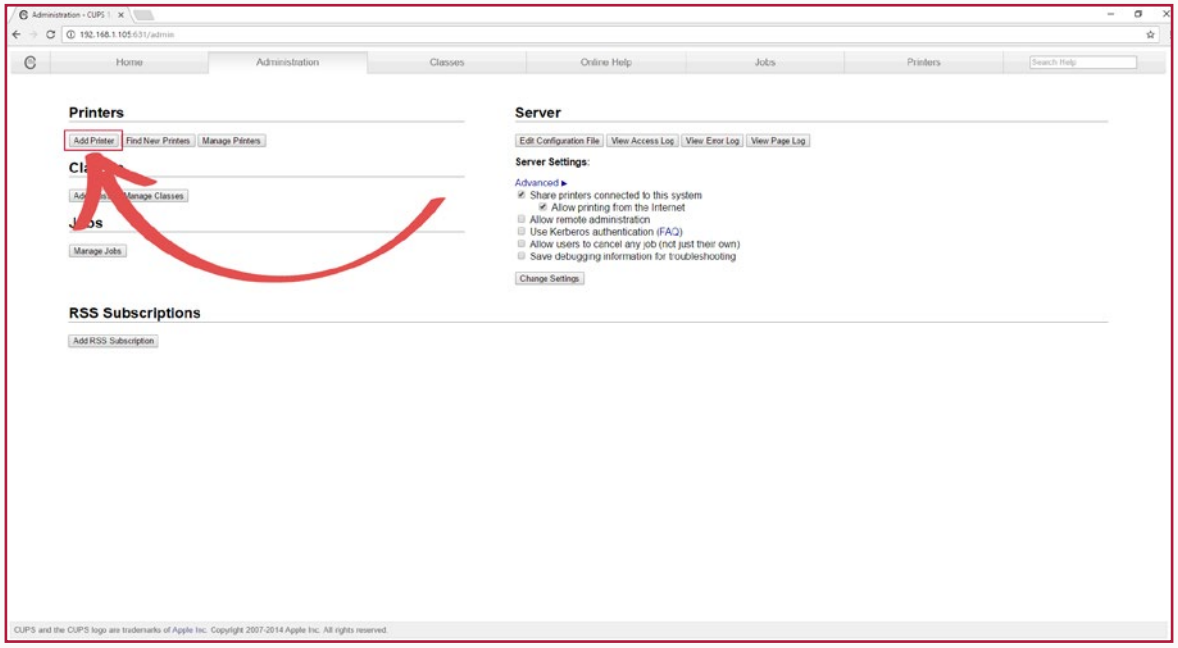
**2.** Once you have your Raspberry Pi's IP Address, go to the following web address in your favorite web browser, make sure to swap out my IP address (**192.168.1.105**) with your own.

```
https://192.168.1.105:631
```

**3.** You should be greeted with the following screen, on here we need to click "**Administration**".




**4.** Now that we are on the administration screen, we need to click on the "**Add Printer**" button.

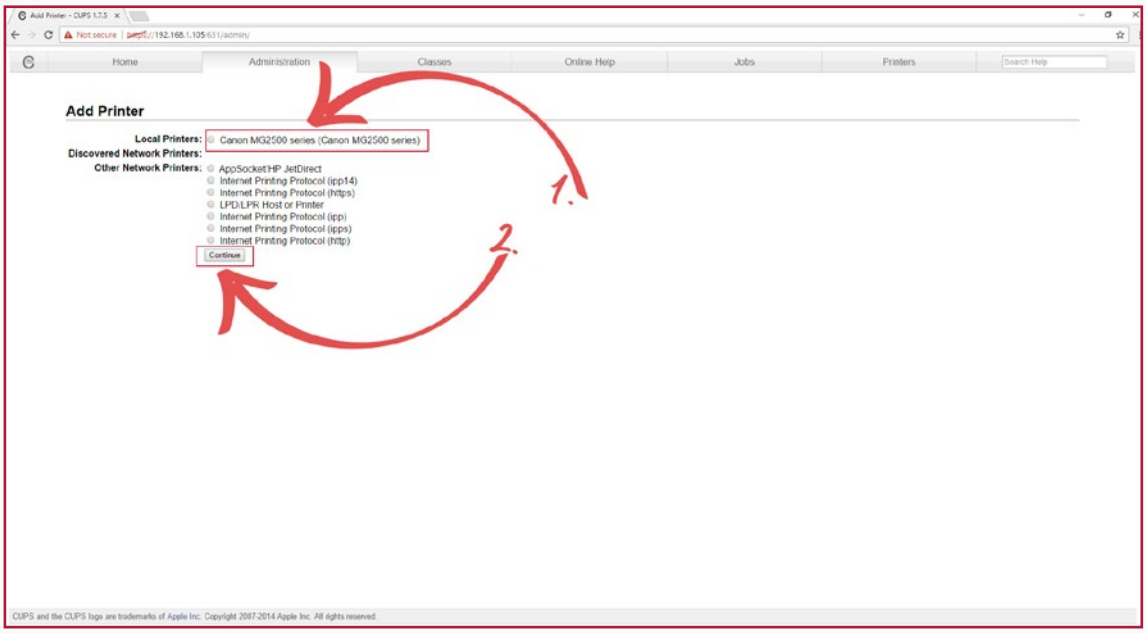


# Adding a printer to CUPS - Continued

**5.** With the **Add Printer** screen now loaded, we can select the printer we want to set up. In our case, that is the **Canon MG25000 series (Canon MG2500 series)** printer. Once selected, press the "**continue**" button.

If your printer is not showing up on this screen, then make sure you have plugged it into one of the USB ports on the Raspberry Pi and is turned on.

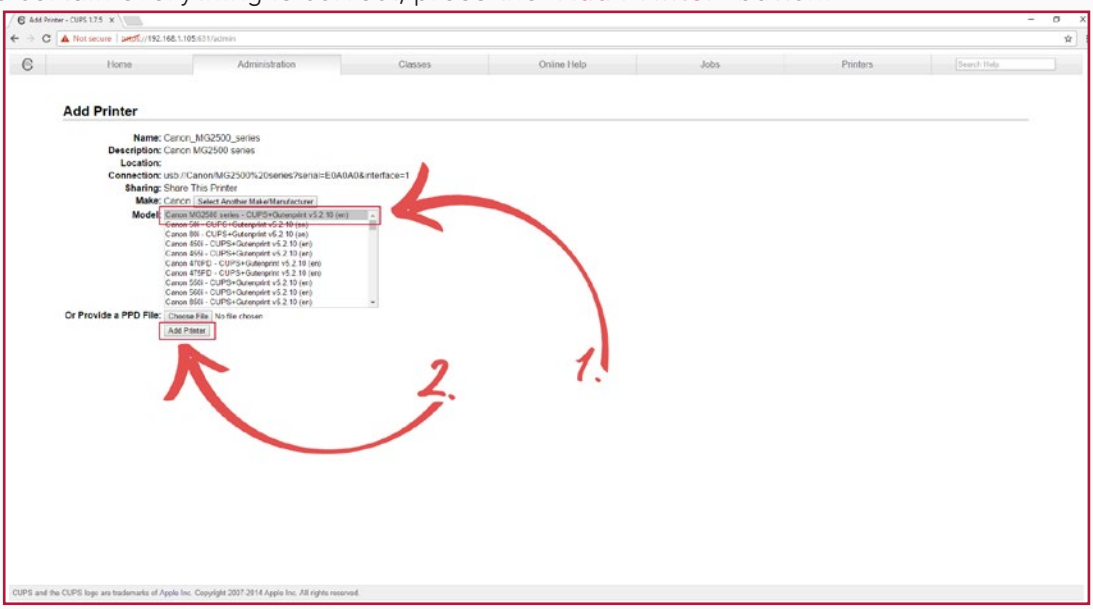
You may have to restart your Raspberry Pi if it is still refusing to show up, ensure the printer is turned on and plugged in when you restart.




**6.** On this screen, you need to select the model of your printer. CUPS will try and automatically detect the model and pick the correct driver.

However, in some cases this will not function correctly, so you will have to go through the list yourself and find the most relevant driver.

Once you are certain everything is correct, press the "**Add Printer**" button.

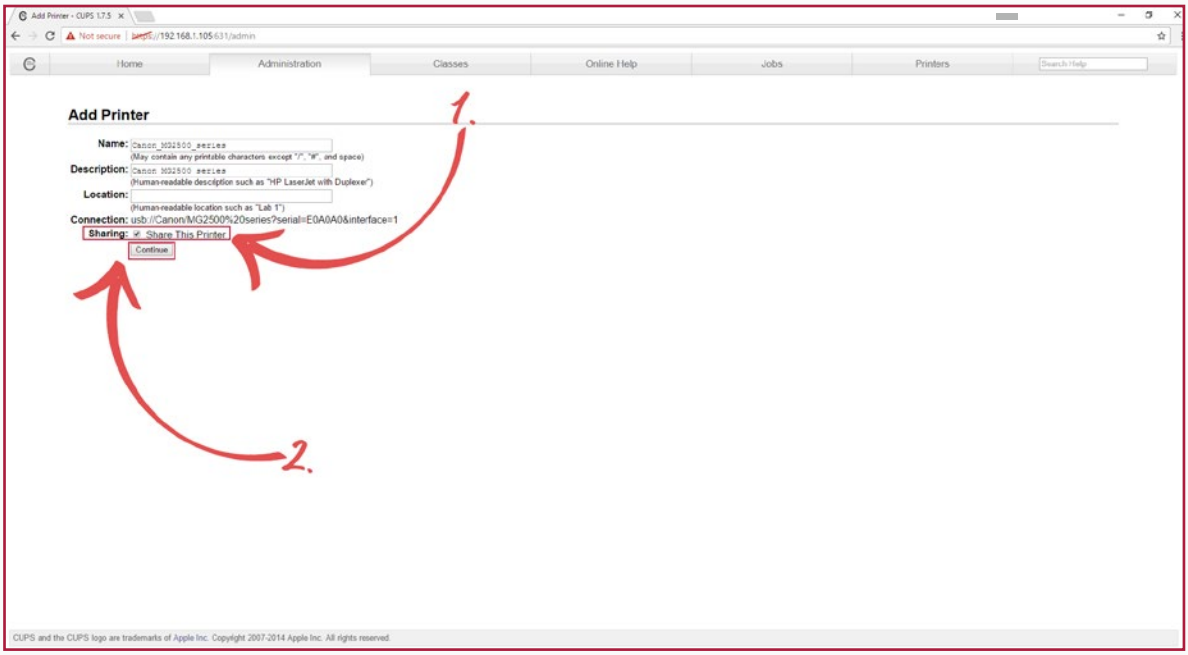


# Adding a printer to CUPS - Continued

**7.** Now, this is the last screen you need to deal with before the printers successfully added, here you can set the names to whatever you want. It is handy setting location if you have multiple printers in your house that you need to deal with.

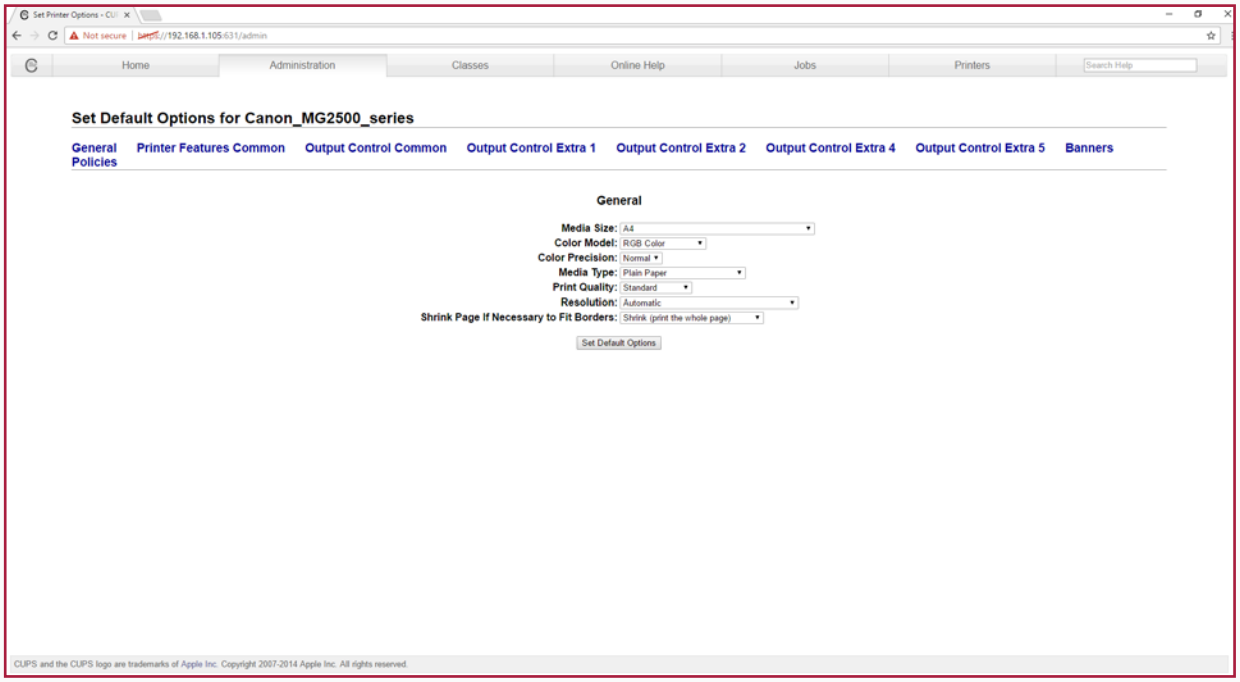
Also, make sure you enable "Share this printer", otherwise other computers will not be able to access it.

Once you are happy with what, feel free to press "Continue".




**8.** The final screen you will be presented with after setting up your printer is the following.

This screen allows you to change a few of the printer's specific settings. Such as the page print size, the print quality, and various other options.



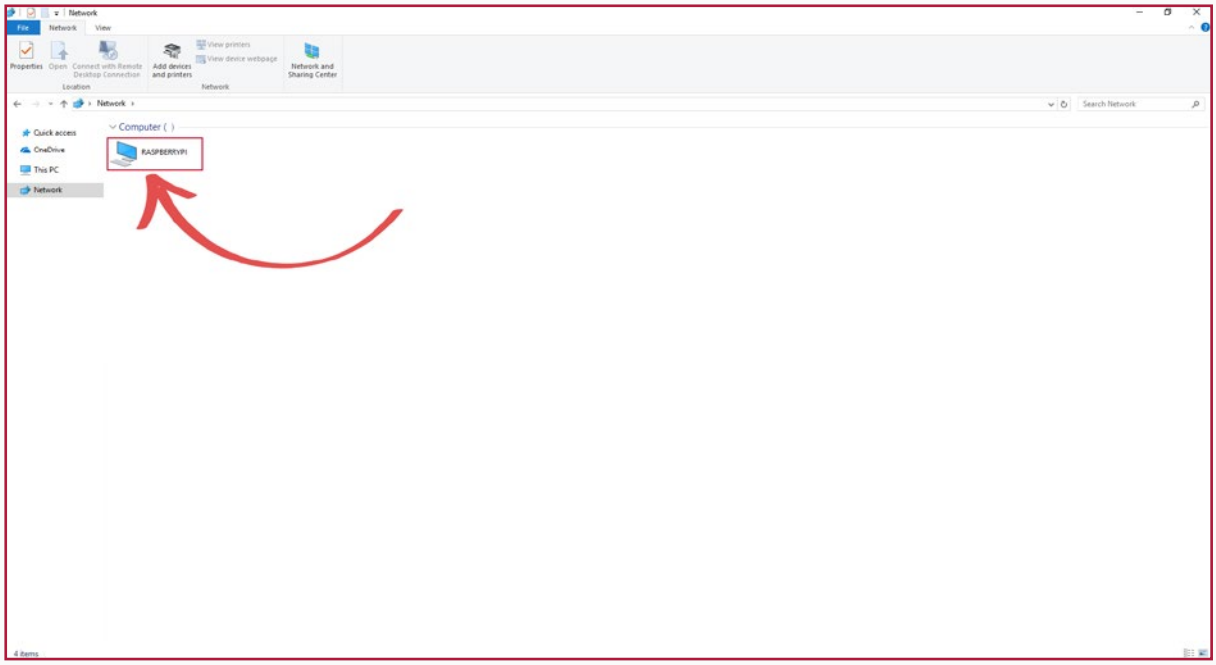

On the next page, we will go over how to add our newly setup Raspberry Pi print server to Windows. This should be a relatively easy process thanks to setting up SAMBA earlier in the tutorial.

# Adding a CUPS printer to Windows

**1.** Adding a CUPS printer to Windows can be a bit of work, mainly because you need to select the driver for Windows to be able to connect to and understand the printer.

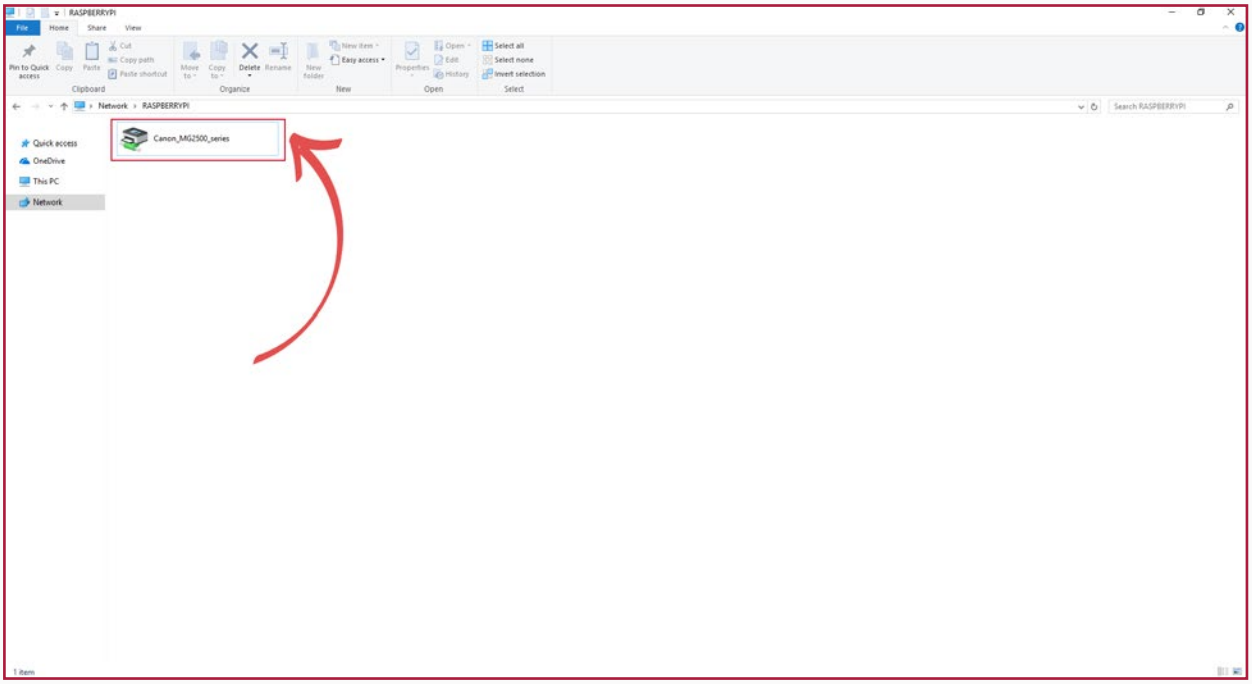
To get started, first go to the Network page in Windows, one of the fastest ways to get to this is to load up "**My Computer**" or "**This PC**" and click on "**network**" in the sidebar. Once there you should have a screen somewhat like below with your Raspberry Pi's hostname there.

Double click on your Raspberry Pi's share, please note it may ask for a username and password. If just pressing enter doesn't work, try entering the username **pi**.



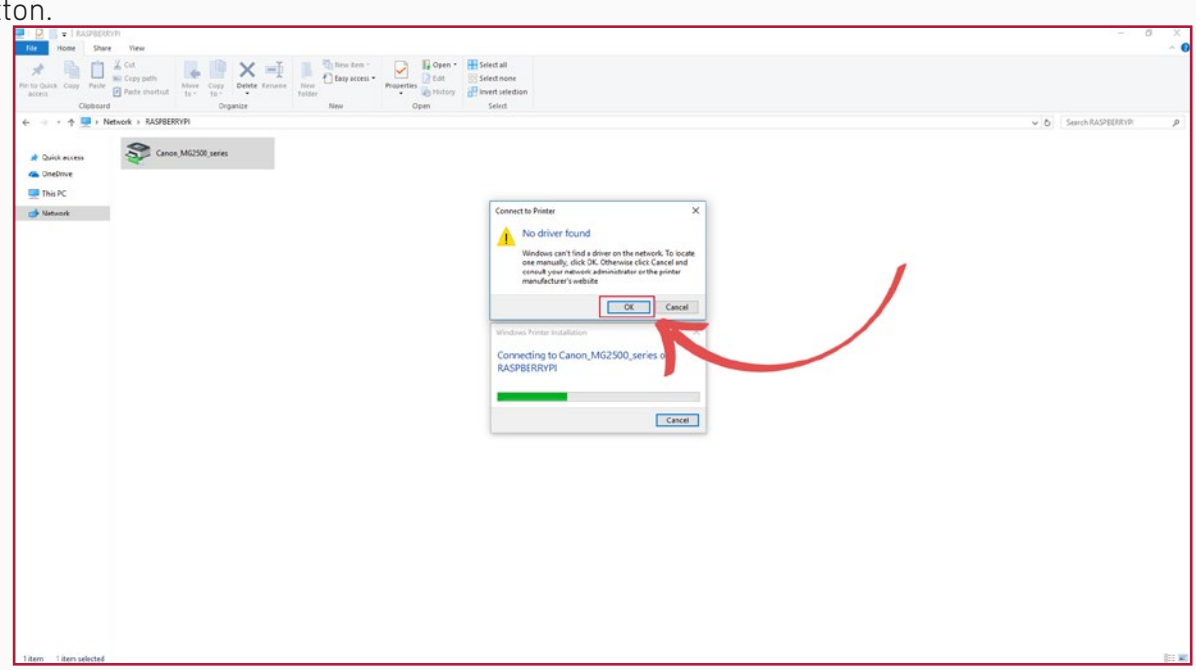

**2.** You should now be greeted with a screen displaying the printers available on your Raspberry Pi Print Server.

Double click on the printer you want to have connected to your computer.



# Adding a CUPS printer to Windows - Continued

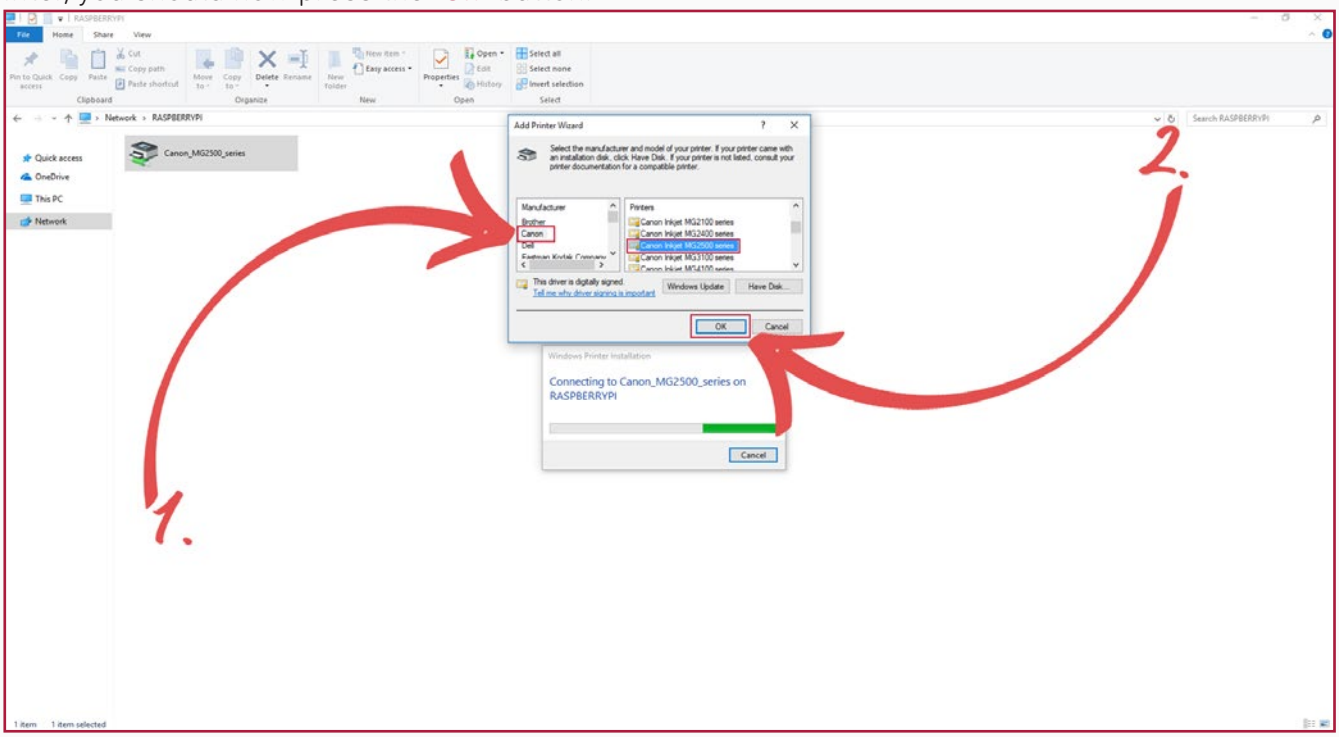
**3.** Upon double clicking this, you will likely be greeted with the warning message below, just click the "**OK**" button.




**4.** Now you will need to find your printer within this list, on the left is a list of all the brands, and on the right, is a list of all the printers for that brand that Windows has drivers for.

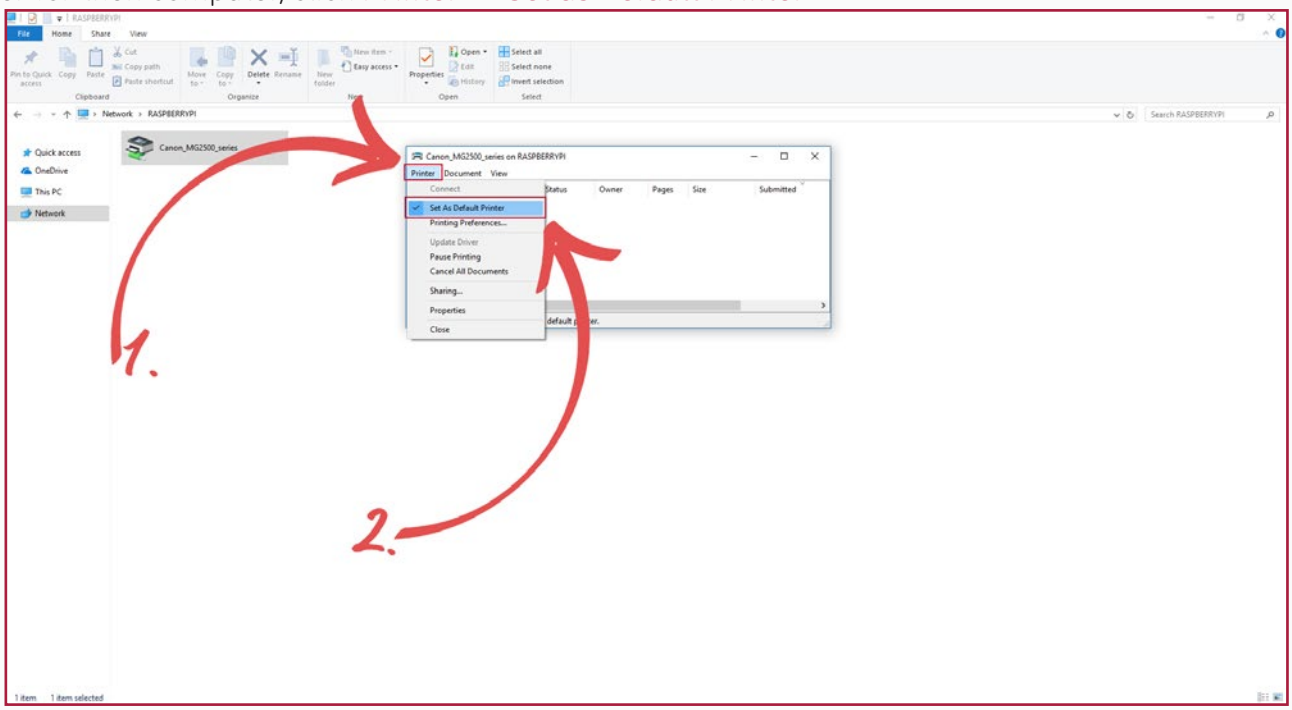
If you don't find your printer on here, then try looking up your printer's model online and download the appropriate drivers for it.

In my case, I had to look for the Cannon MG 2500 series as shown below. Once you have selected your printer, you should now press the "**Ok**" button.



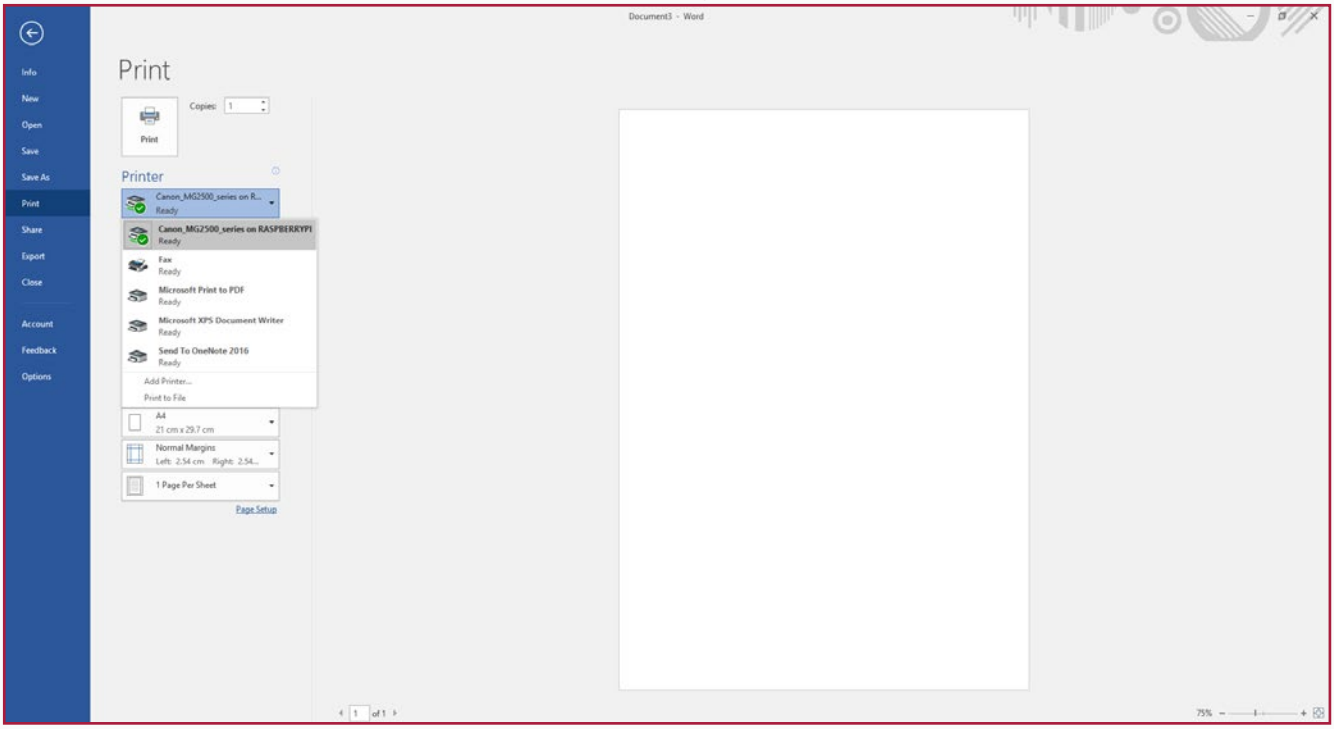
# Adding a CUPS printer to Windows - Continued

**5.** Doing this will now load up a connection with your printer, if you want to make this the default printer for then computer, click **Printer** -> **Set as Default Printer**



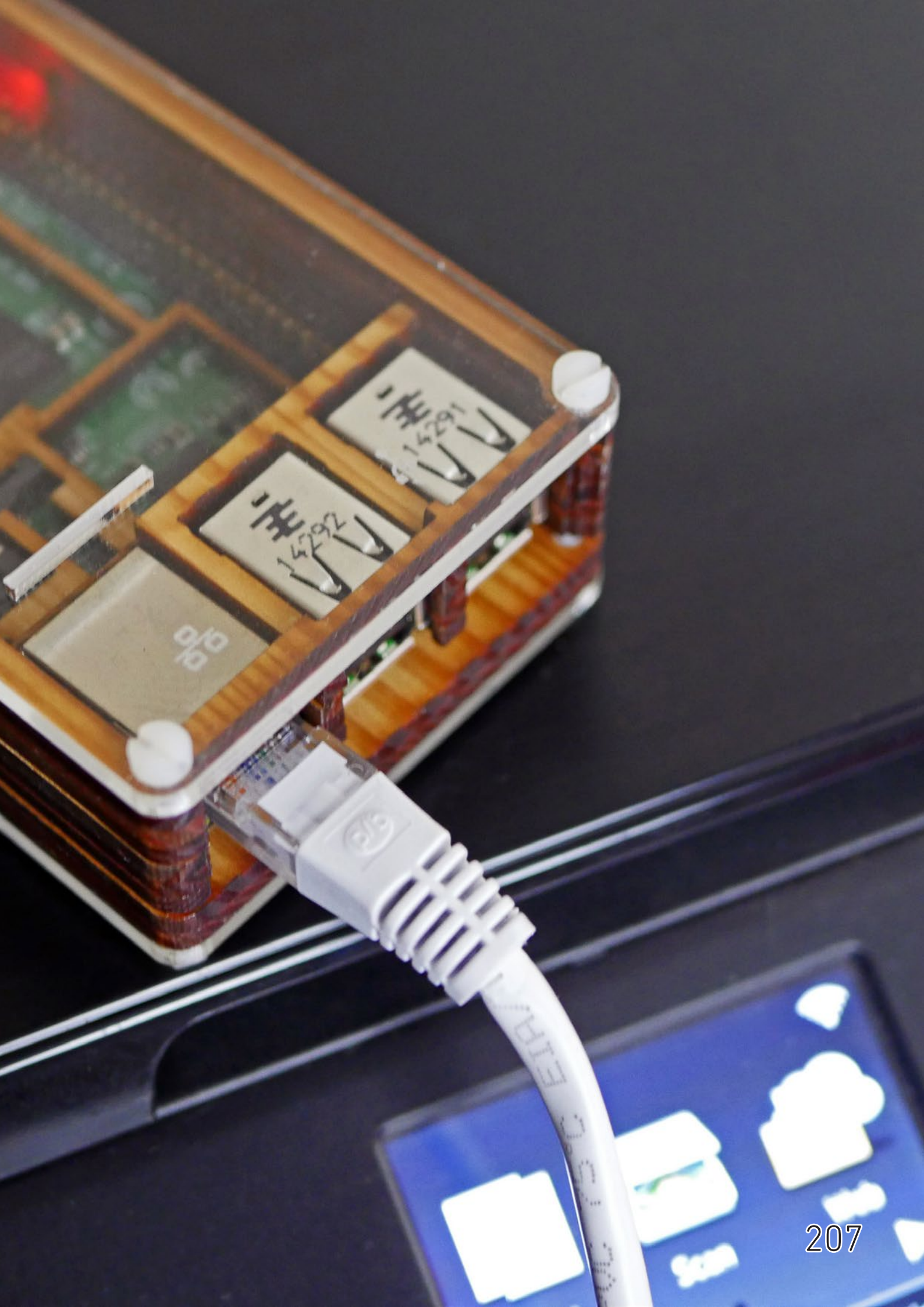

**6.** The printer should now be successfully added to your computer and be available for any program to use.

You can ensure the printer is correctly setup by printing any file.




If you have any issues with the file printing, make sure that you have selected the correct Printer Driver on both CUPS and Windows.

Make sure your printer is also switched on, some printers like the Canon MG2500 series don't automatically turn back on.


4292
4291

# Setting up a Mail Server

## Project Description

This Raspberry Pi Mail Server tutorial will take you through on how to have your very own mail server that is both low cost and relatively versatile.

One of the various reasons to set up a Raspberry Pi mail server is because it is incredibly low cost, the initial cost is cheap and then ongoing expenses is relatively low due to the Raspberry Pi's low power usage.

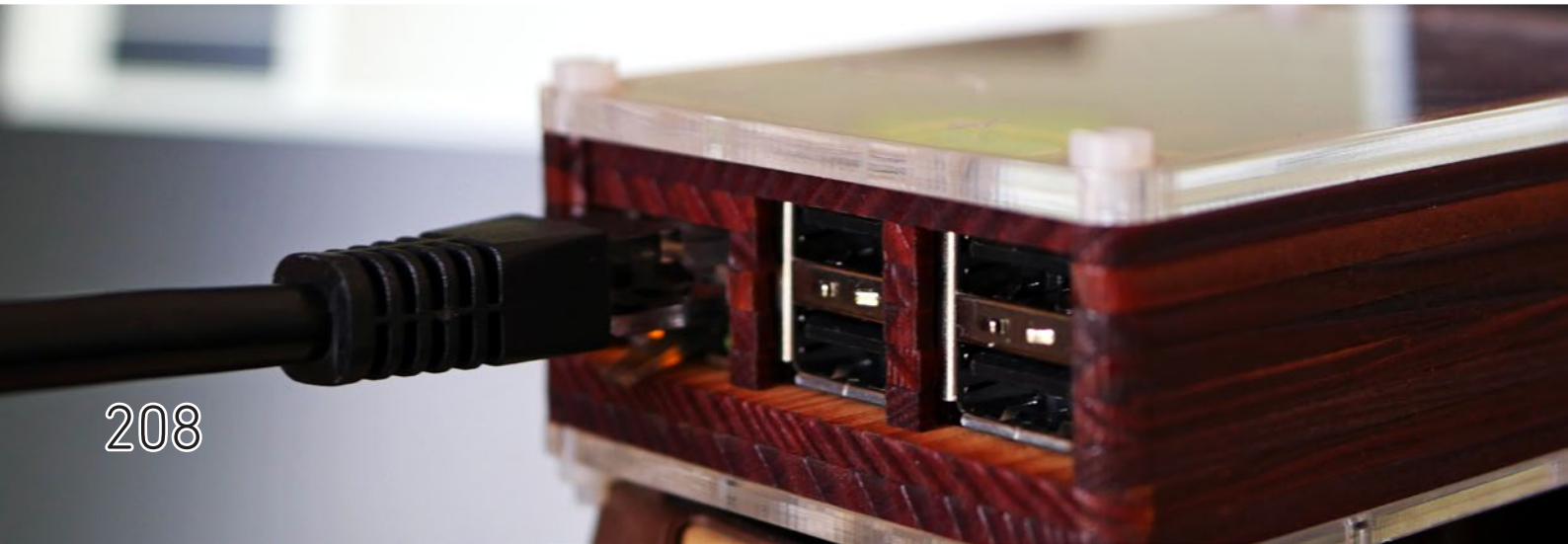
There are several other advantages to setting up a Raspberry Pi mail server such as being able to utilize a custom domain name and having full control over your communications.

The emails are stored on your server and will have no one scanning them to try and present you with better advertisements.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

**Optional**
- Raspberry Pi Case
- External Hard Drive or USB Drive



# Installing the Citadel Mail Server

There are a few ways to install a mail server on the Raspberry Pi, one of the most complicated ways is to install Postfix and Dovecot separately then installing a web frontend.

In our tutorial, we will be utilizing a software package called Citadel. Citadel is easy to set up while being a powerful yet versatile mailing server.

It heavily simplifies the process of setting up a mail server and does not require extensive configuration it to run.

**1.** First, let's update our package repository by running the following command:

```
sudo apt-get update
```

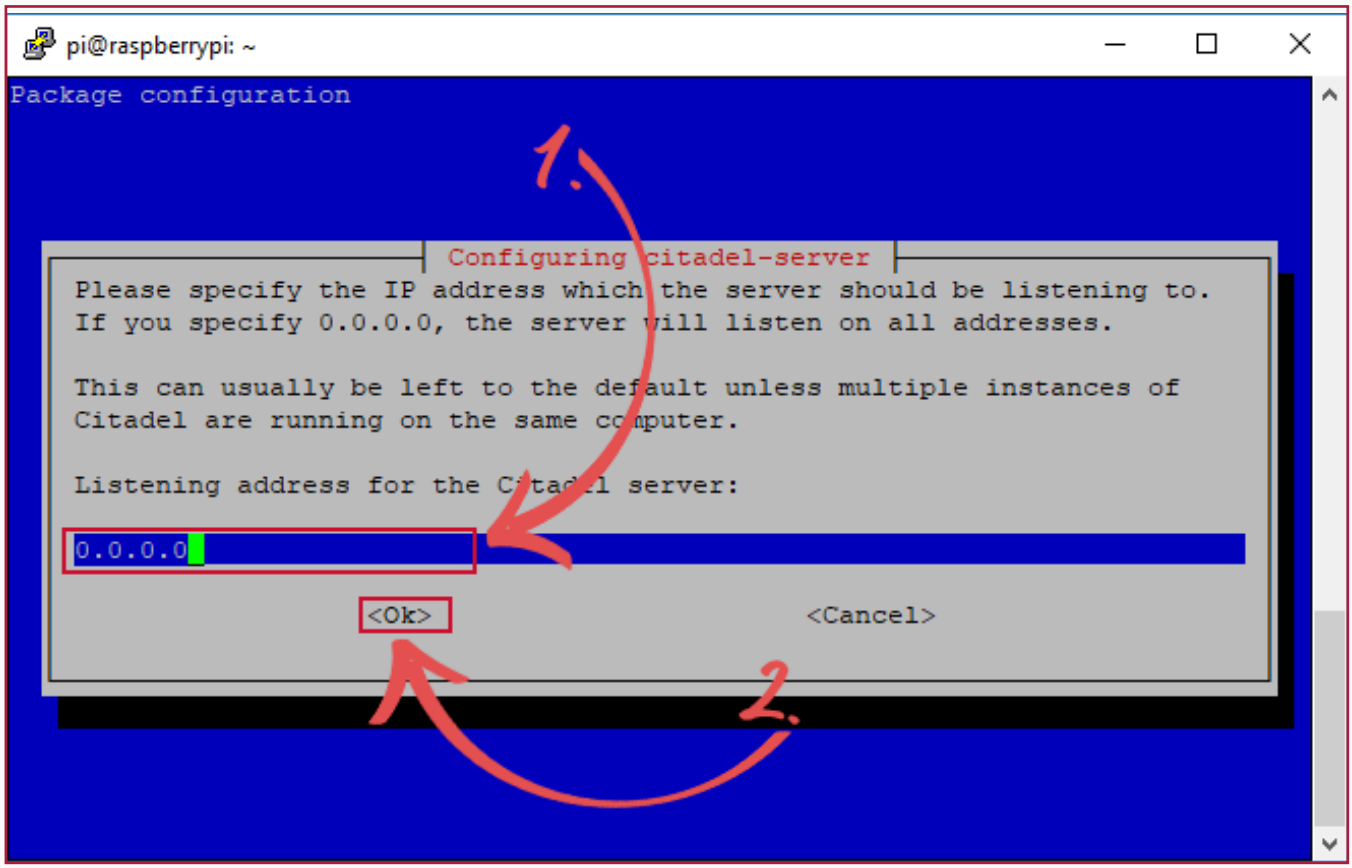
**2.** Now let's start the installation process of Citadel, we can do that easily with the following command:

```
sudo apt-get install citadel-suite
```

**3.** For this, leave the field set to **0.0.0.0**, this allows Citadel to listen on all incoming address.

The only reason to change this will be if you are running multiple instances of Citadel.

Press **Enter** to continue.

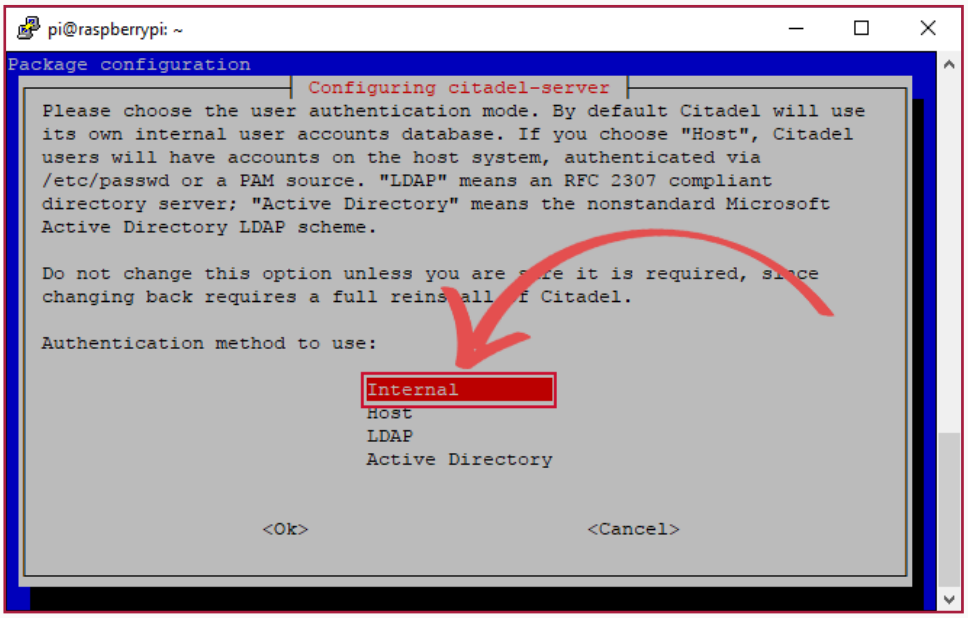


# Installing the Citadel Mail Server

**4.** Now for the user authentication mode, we are going to keep this set to **Internal**.

This mode allows Citadel itself to utilize its internal user accounts database. Only change this if you need the other three options.

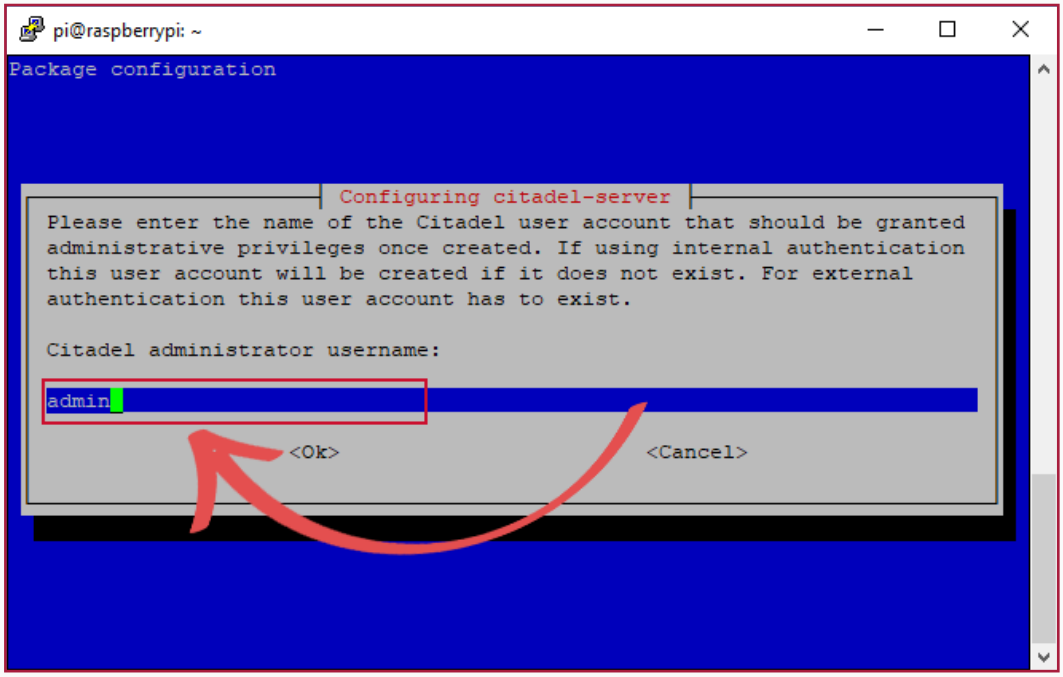
Press **Enter** to continue.




**5.** Here we get to set the name we want for the administrator user, to keep this tutorial as simple as possible we will be just utilizing the default username admin.

However, we recommend changing the username to something else. Remember what you set here for later in the tutorial as this is what you will need to log in to Citadels web interface.

Press **Enter** when you are happy with your choice.

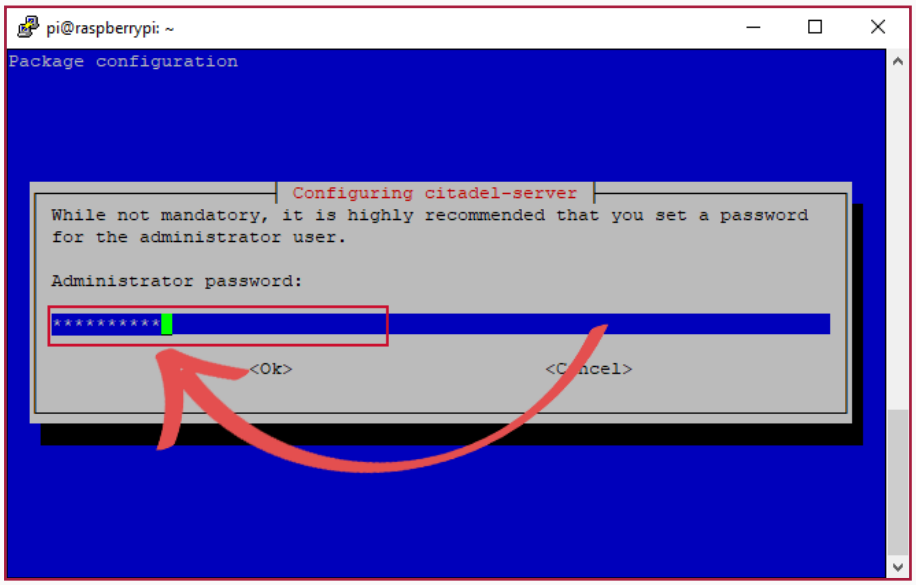


**6.** Now is the time to enter a password for this new administrator user, make sure you make this password something secure.

A secure password is crucial for anyone who gains access to your admin account could view all your emails and potentially damage your server's settings. It is even more crucial if you also plan on port forwarding port 80 to allow access to the web interface.

Press **Enter** when you are happy with the password you have entered.

You will be asked to verify this password on this screen, enter the same password again and press **Enter**.




**7.** For this tutorial, we will be selecting **Internal** to make Webcit utilize its HTTP server facilities.

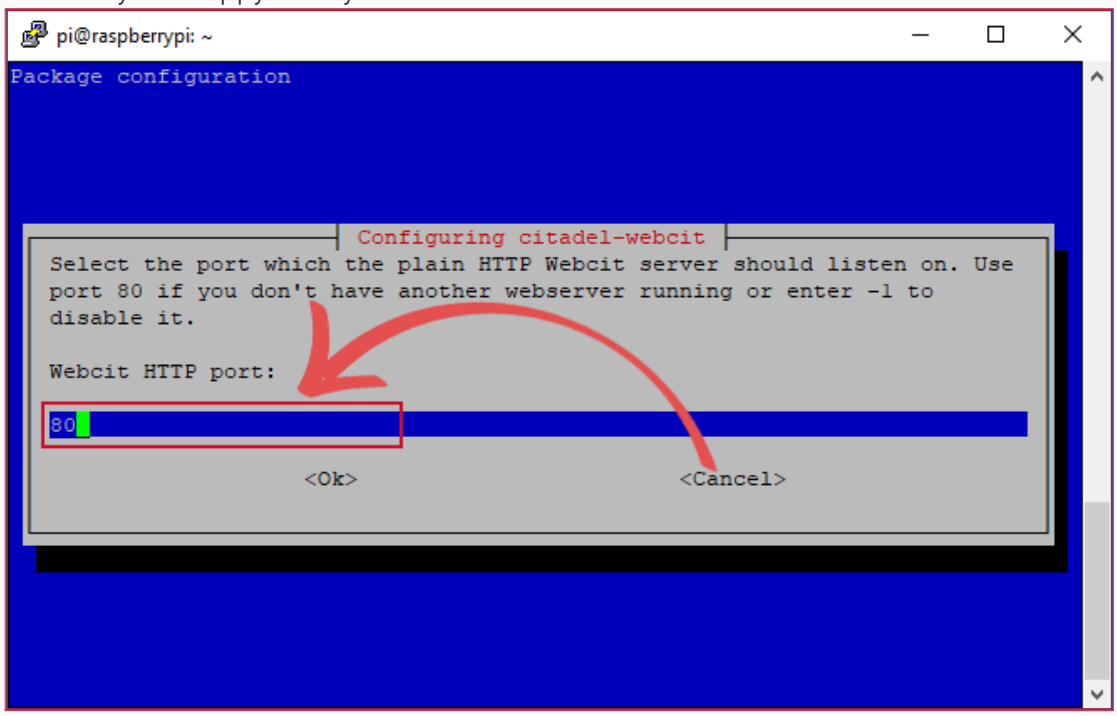
However, if you are utilizing an Apache2 server already, then select **Apache2**.

Use the **arrow keys** to select **Internal** then press **Enter** to continue.

**8.** Now it will prompt you to select the port you want to listen for an HTTP connection on if you have no other web servers running it's safe to keep it set to **80**.

Otherwise, try changing it to another port such as **8080**.

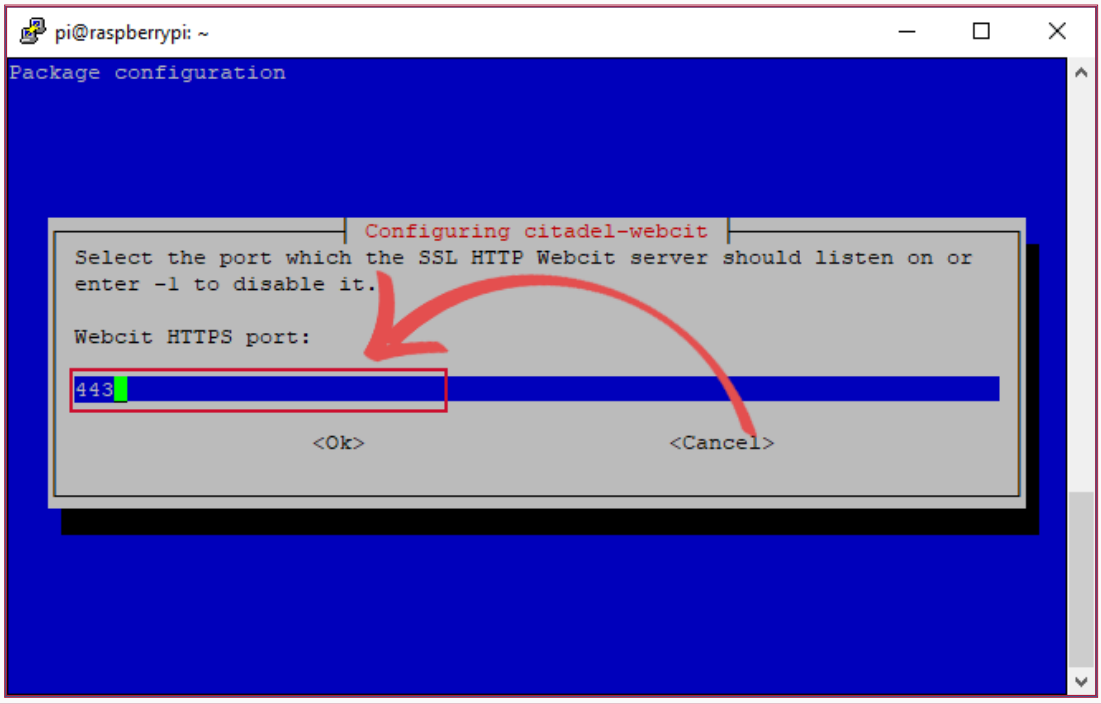
Press **Enter** when your happy with your choice




**9.** Now it will prompt you to select the port you want to listen for an HTTPS connection on if you have no other web servers running it's safe to keep it set to **443**.

Otherwise, try changing it to another port such as **4434**.

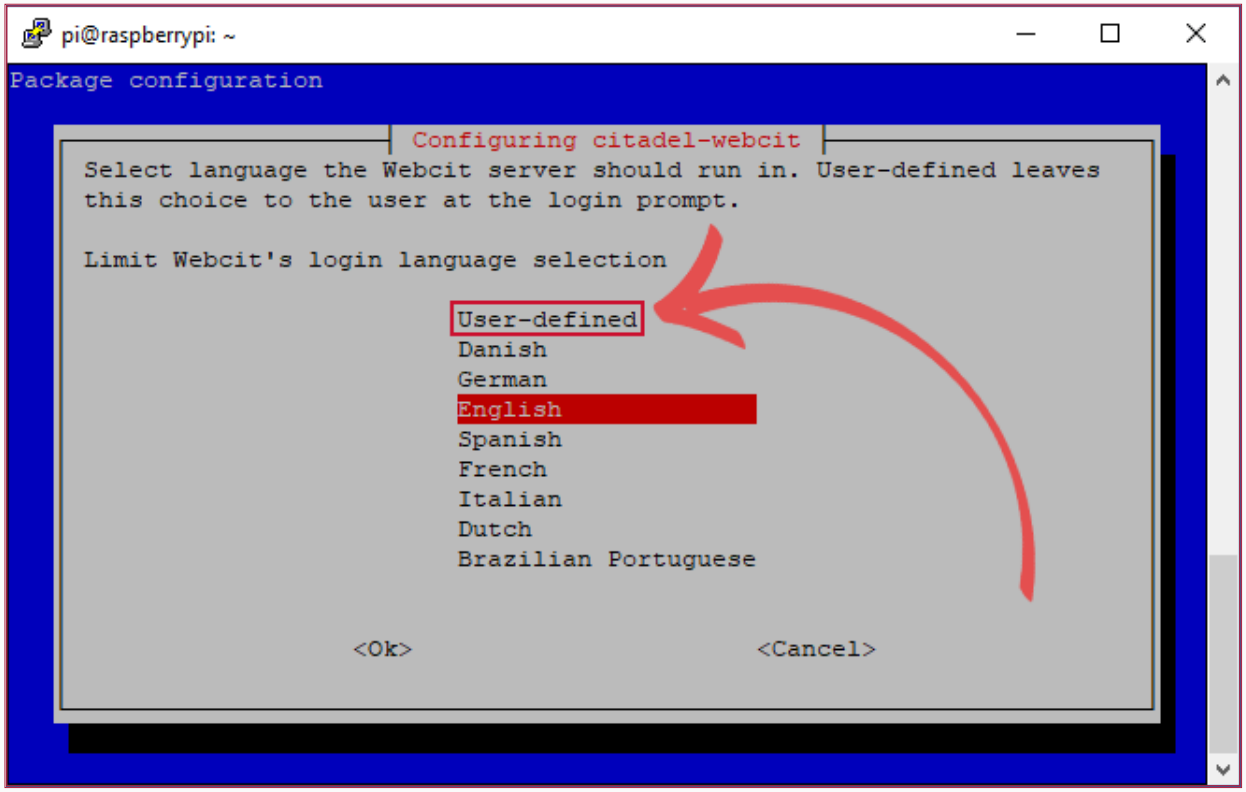
Press **Enter** when your happy with your choice



# Installing the Citadel Mail Server

**10.** Now we can select whether we want to enforce a default language or let the user choose it on login. The straightforward option is just to select "**User Defined**".

When you are happy with your choice, press **Enter**.

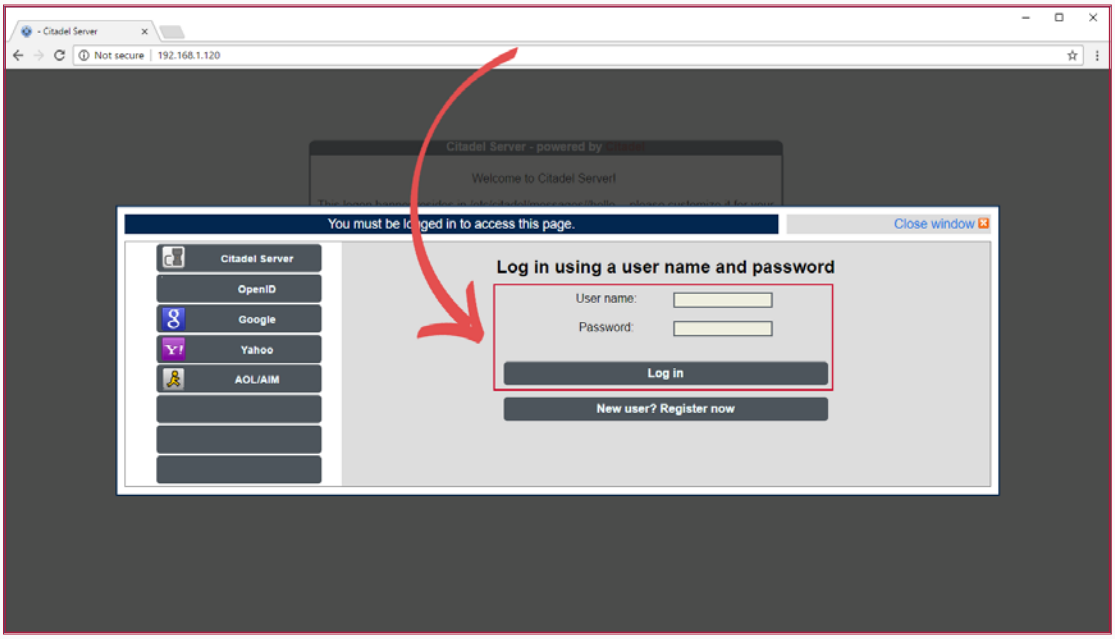


# Setting up the Citadel Mail Server

**1.** Now we need to load up Citadels web interface, to do this go to your Raspberry Pi's IP Address in your favorite web browser.

In my case, I would go to http://192.168.1.105/. If you don't have Raspberry Pi's IP on hand you can type **hostname -I** into the Raspberry Pi's terminal to get your local IP address.

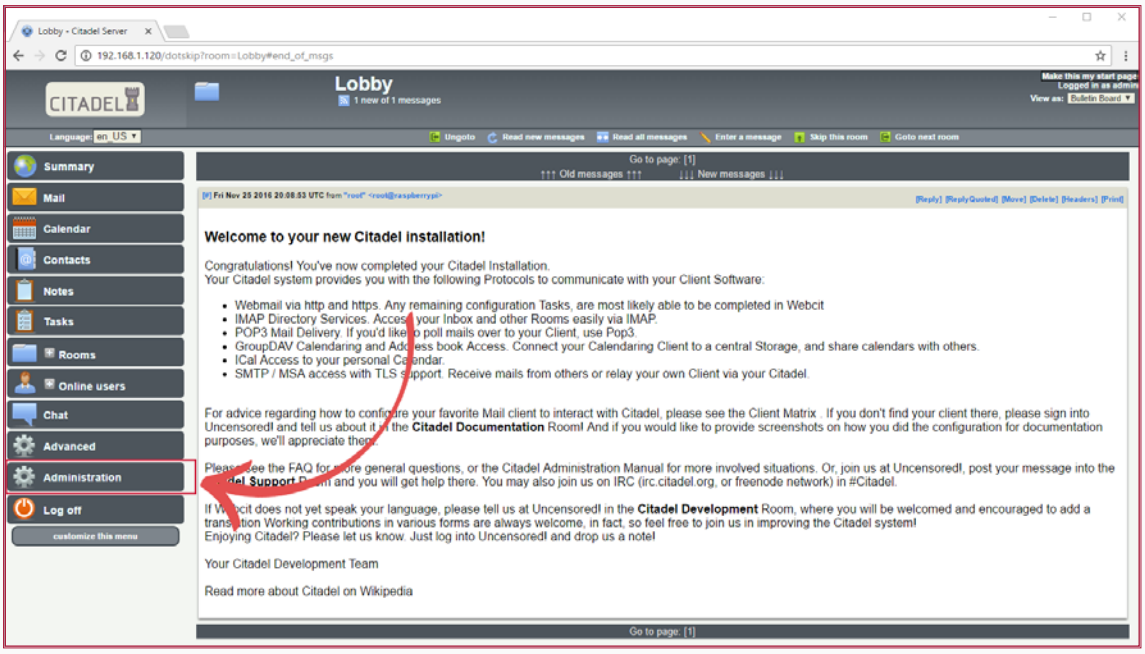
Log in by using the admin username and the admin password you set during the installation process.




**2.** Now, you will see the default dashboard.

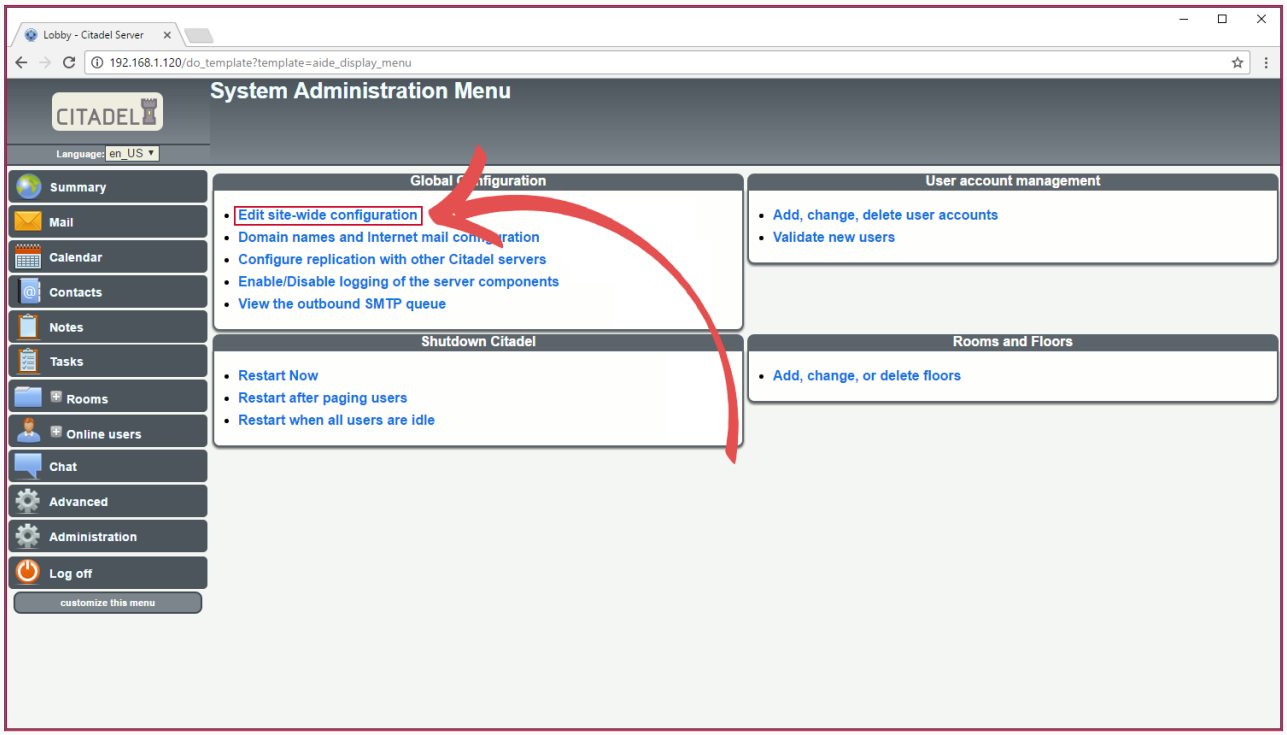
You can change the homepage by opening another page and pressing "**Make this my start page**" in the top right-hand corner.

For now, just click on **Administration** in the sidebar.



# Setting up the Citadel Mail Server

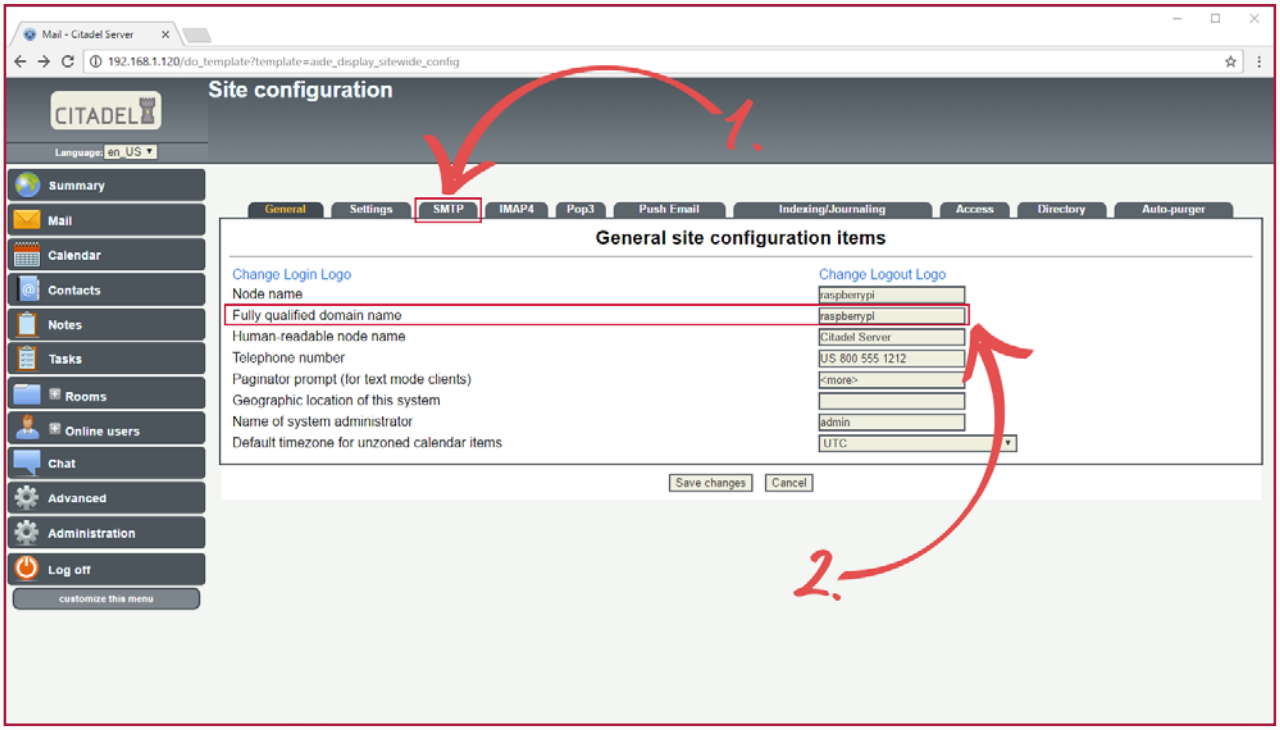
**3.** Now where in the System Administration Menu, there's a wide variety of different options in here but what we are after is "**Edit site-wide configuration**", so click on that.




**4.** Now on this screen, we will have to make a few changes to get everything working.

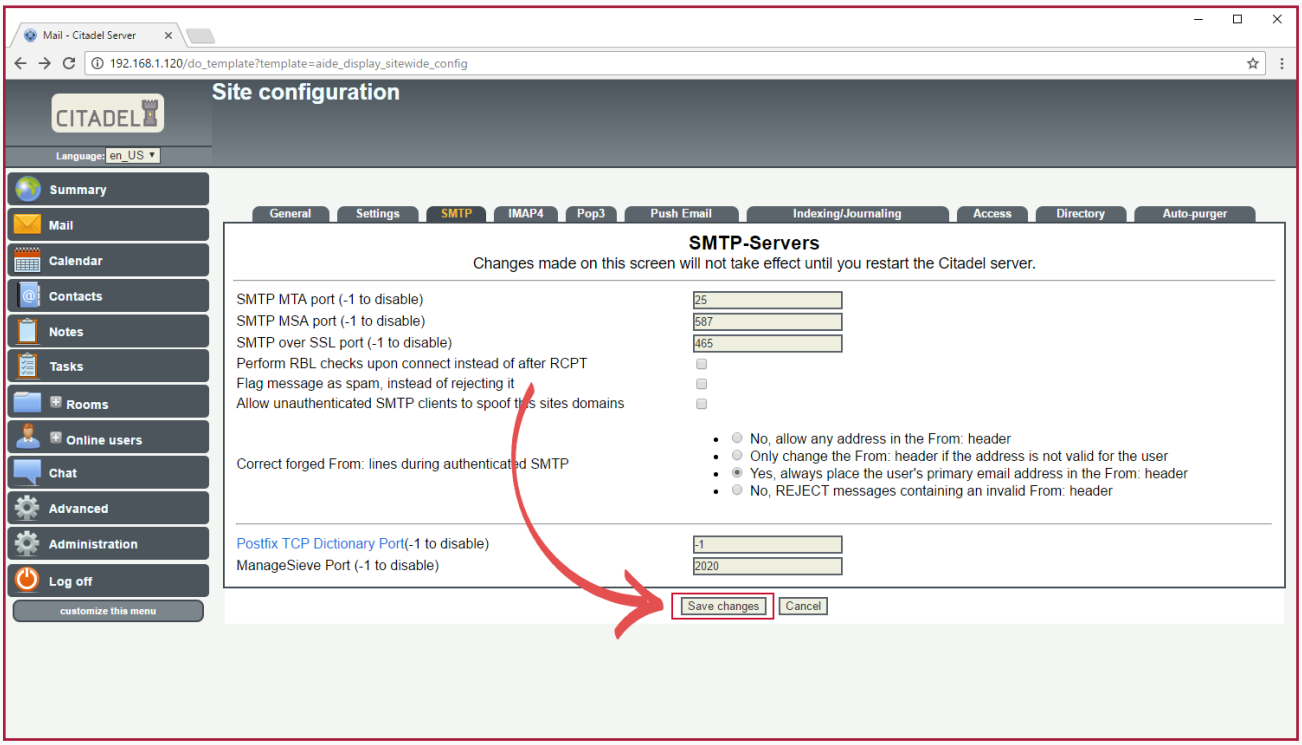
Here we need to change the "**Fully Qualified domain name**" to the custom domain we plan to use for our email. In our case, we will set this to **mail.pimylifeup.com**.

Once done, click on the "**SMTP**" tab.

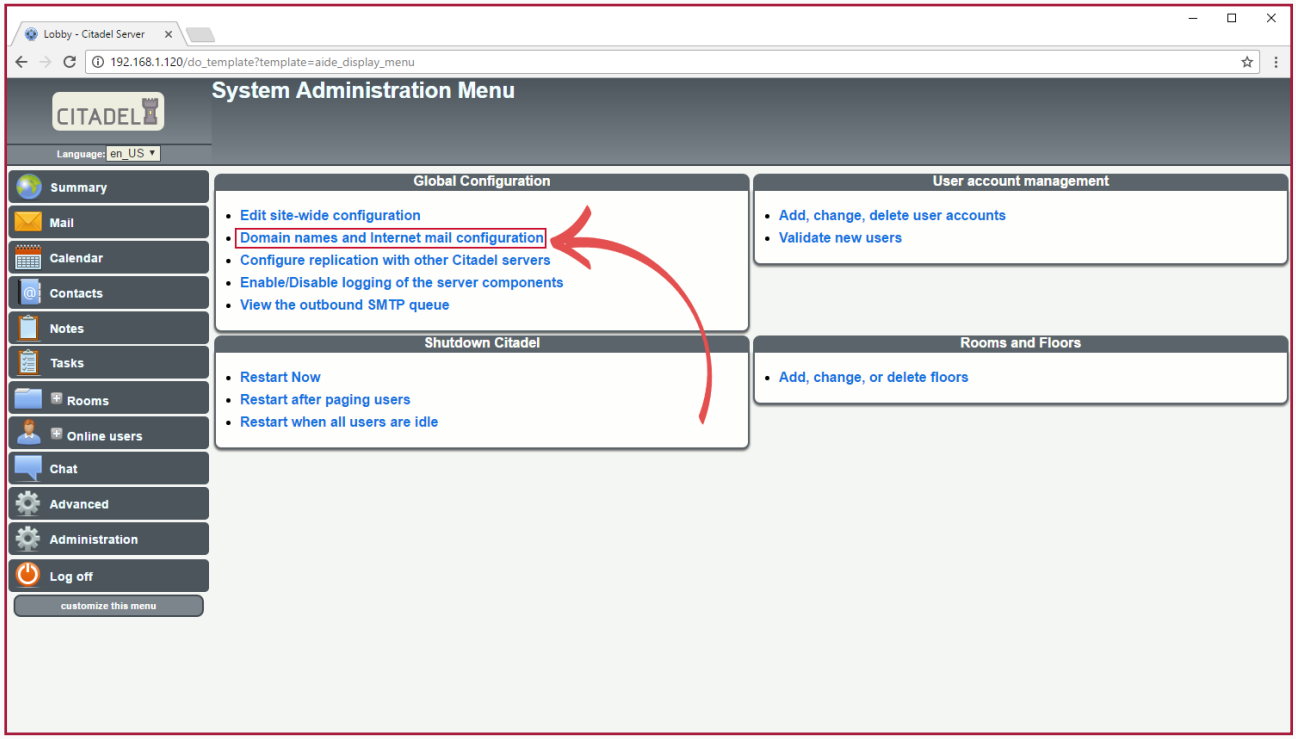


**5.** Now on this screen make a note of the ports mentioned here, we will need to port forward these to allow the server to receive emails over SMTP.

Once done, press the "**Save Changes**" button, then click on "**Administration**" in the sidebar.

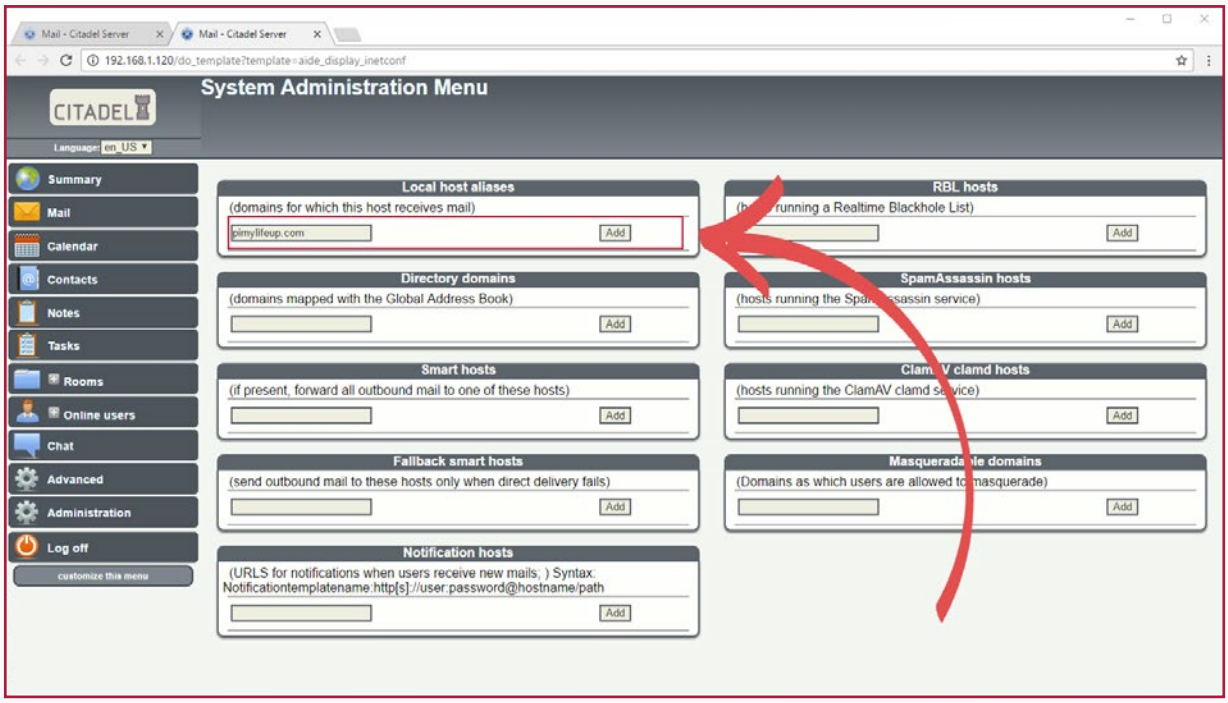



**6.** Now back in the "**System Administration Menu**" click on "**Domain names and internet mail configuration**" This link will take us to the menu that we will be using to add additional domain names to Citadel.



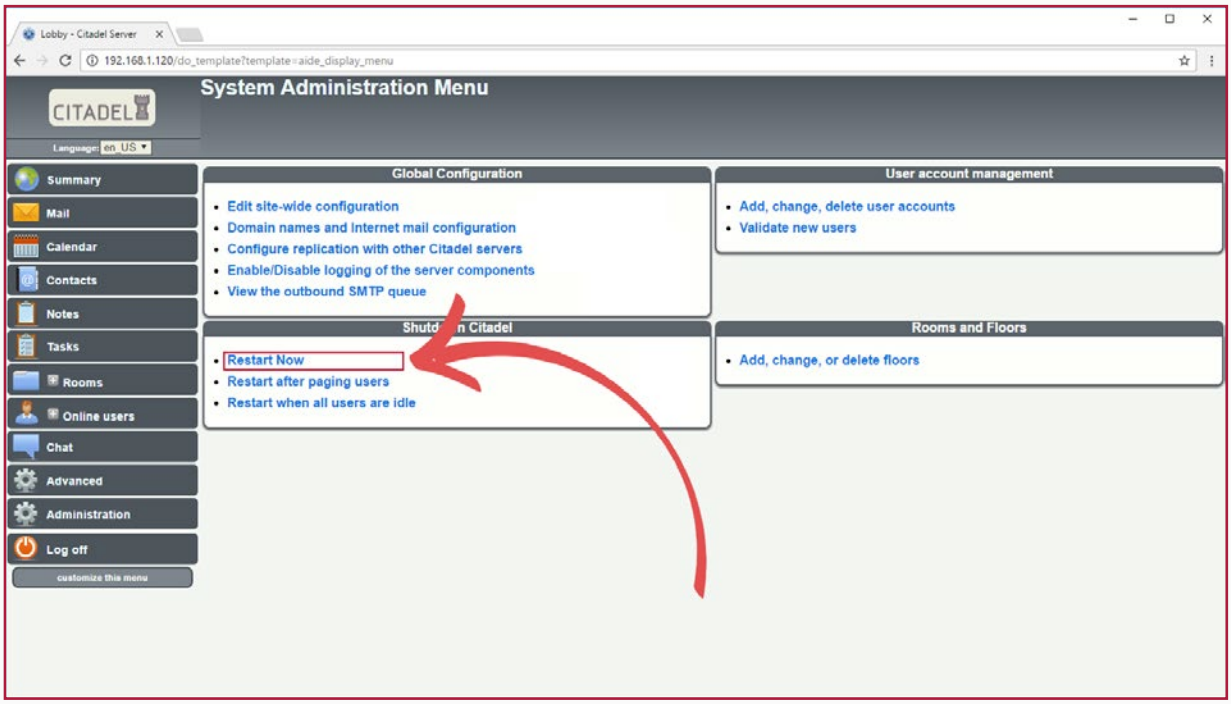
**7.** Now on this screen, under "**Local host aliases**" type in the domain name you want to use then press the "**Add**" button.

Once done, click back on "**Administration**" in the sidebar.




**8.** Now we are back on this screen.

We need to click "**Restart Now**" under "**Shutdown Citadel**" this ensures that all our settings changes will now be loaded in.



# Setting up your DNS

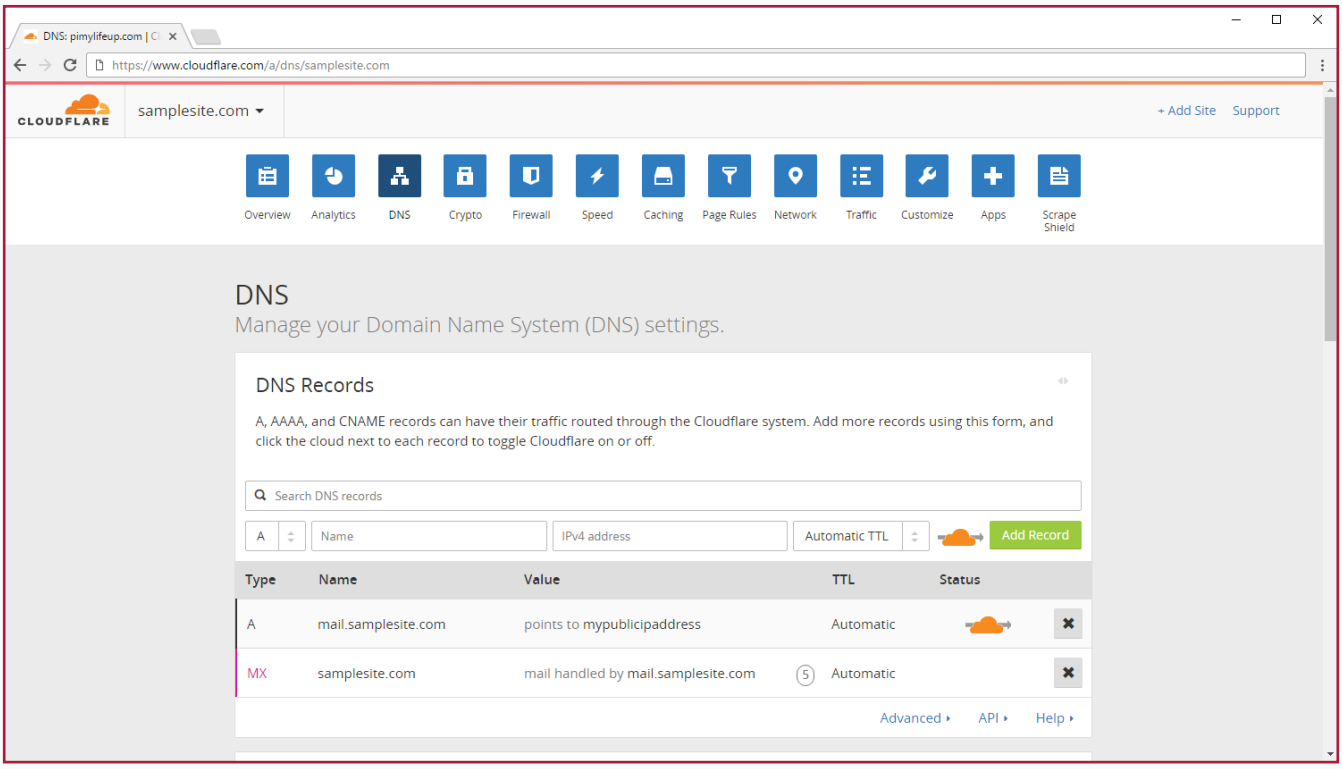
The easiest way to setup your DNS is to utilize a service like Cloudflare. Cloudflare offers a stable and redundant DNS service that can be easily setup to also act as a dynamic DNS service as well which is incredibly useful for anyone that is behind a dynamic IP address.

An added advantage is that it also makes it incredibly easy to modify the DNS records, and updates are propagated much faster than most DNS services.

**1.** Go to your domain name on Cloudflare, then got to the DNS tab.

**2.** In here you need to add an **A name record** that points towards your networks public IP address. Adding this record will allow you to access your webmail client after you port-forward. It is also needed so we can point the **MX Record** to it.

**3.** Now add an **MX record** that points towards the domain name that you set up as an **A record** in the previous step.




**4.** With that all done, you will now need to port forward various ports from earlier, make sure you port-forward **port 80**, **port 25**, **port 587** and **port 465**. The last 3 of these ports should be your SMTP ports mentioned in the SMTP settings page from earlier.

If you are unsure on how to port-forward your router, you can follow our generic guide in our "**Gettting Started**" book. Otherwise look up your router at https://portforward.com/.

Also, if you have a dynamic IP address you will need to set up a dynamic DNS client, you can follow our guide on setting up one earlier on in this book in the getting started section. Make sure you follow the steps on setting it up to work through Cloudflare.

## Setting up Spamassassin for Citadel

Now we will go through setting up Spamassassin. Spamassassin is a tool that handles sifting through SPAM. It is a crucial tool to have for any mail server. It will add a bit more workload to the Raspberry Pi, but it should be able to handle it.

**1.** First, let's install Spamassassin from packages by running the following command. The installation process of Spamassassin can take some time as it has to compile several things.

```
sudo apt-get install spamassassin
```

**2.** After Spamassassin has installed, we now need to make some changes to its configuration, run the following command to edit the first configuration file.

```
sudo nano /etc/spamassassin/local.cf
```

**3.** Change the below lines so that they match what we have written if any of these lines have a # in front of it, then remove it. We will explain what each line does.

```
rewrite_header Subject [***** SPAM _SCORE_ *****]
```

This line makes Spamassassin modify the Subject header of spam e-mails to include SPAM and the spam score that Spamassassin has assigned to it.

```
report_safe 0
```

This line tells Spamassassin only to modify the headers of an email and not make any changes to the actual body.

```
required_score 2.0
```

Setting the required score low means you will initially get lots of false positives, but it will help you teach Spamassassin to know what emails are good and what emails are bad.

```
use_bayes 1
```

This line sets Spamassassin to use the Bayesian filter. Bayesian is a way of estimating the probability of whether an email is Spam or not. It is a commonly used method that improves as the sample size increases.

```
bayes_auto_learn 1
```

This line turns on the automatic learning for the Bayesian filtering.
Saves and quit out of the file by pressing **Ctrl+X** and then **Y** and then pressing **Enter**.

**4.** Now there is one final file we need to edit to finish setting up SpamAssassin, run the following command.

```
sudo nano /etc/default/spamassassin
```

```
ENABLED=1
```

This line allows SpamAssassin to work through systemctl and means we can get it to boot at startup easily.

```
CRON=1
```

Allows a cron job to automatically update SpamAssassin rules.

Saves and quit out of the file by pressing **Ctrl+X** and then **Y** and then pressing **Enter**.

# Setting up Spamassassin for Citadel

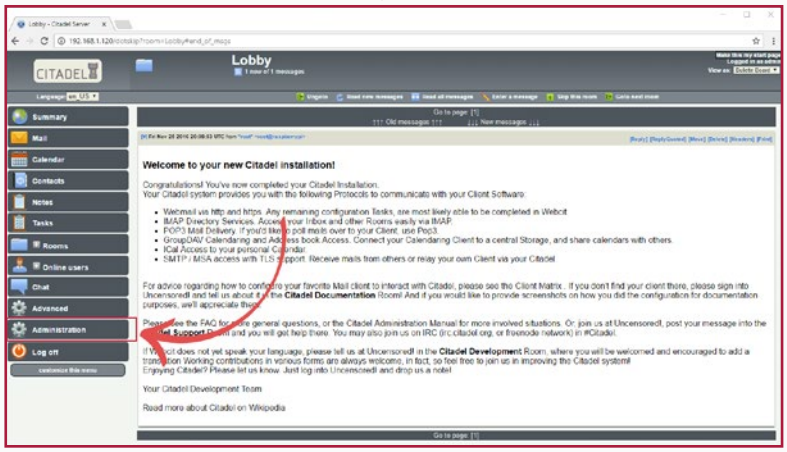
**5.** We can now start the SpamAssassin daemon with the following command

```
sudo service spamassassin start
```

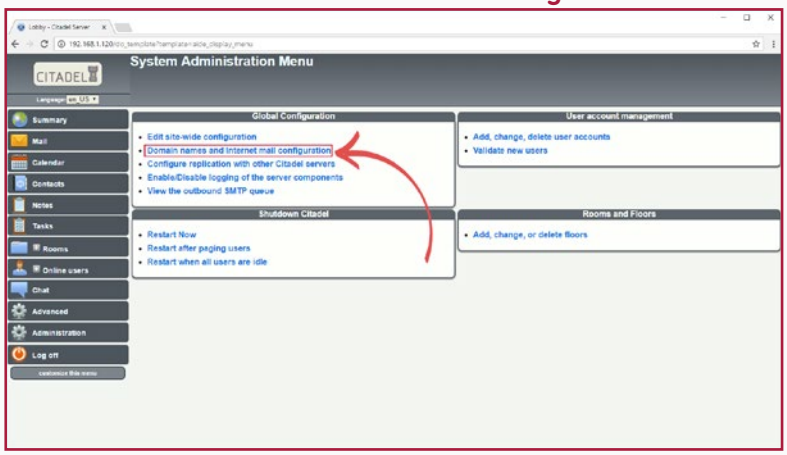
**6.** Now add SpamAssassin to the init system with the following command, so it automatically starts on boot.

```
sudo systemctl enable spamassassin
```

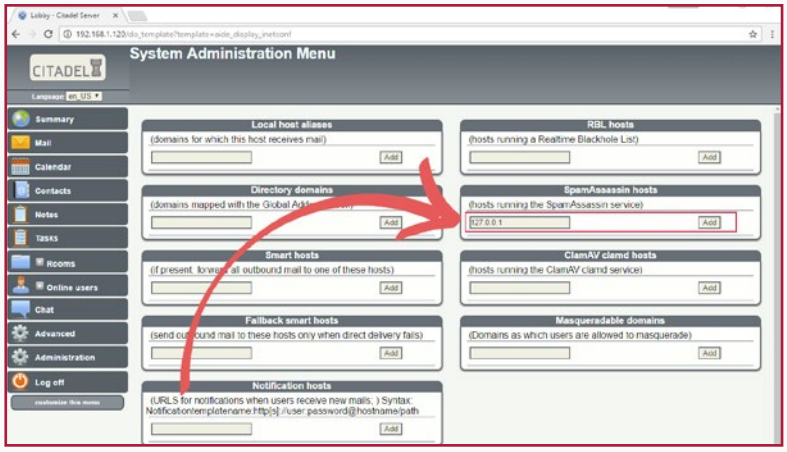
**7.** We now need to go to Citadels web interface and go back to the **Administration** screen.




**8.** Now we need to go to "**Domain names and Internet mail configuration**".




**9.** Now that we are in here, we need to type in the IP Address, **127.0.0.1** under "**SpamAssassin Hosts**". Once you have typed it in, we can just press the "**Add**" button.



# Setting up ClamAV for Citadel

Now onto the final part of our mail server installation guide, we are going to be setting up and installing the ClamAV software. This software will scan all your incoming emails for viruses and helps protect your Raspberry Pi from becoming infected.

**1.** Installing ClamAV is incredibly easy as its already included in the Debian packages, we can just run the following command to install it.

```
sudo apt-get install clamav
```

**2.** Now we want to get ClamAV to download the latest version of its virus databases. We can do this by running the following command.

```
sudo freshclam
```

You may run into the error below, but we can deal with that:

```
ERROR: /var/log/clamav/freshclam.log is locked by another process
```

To deal with this error, we need to stop clamav. We can do this by running the following command. Then rerun **sudo freshclam**.

```
sudo service clamav-freshclam stop
```

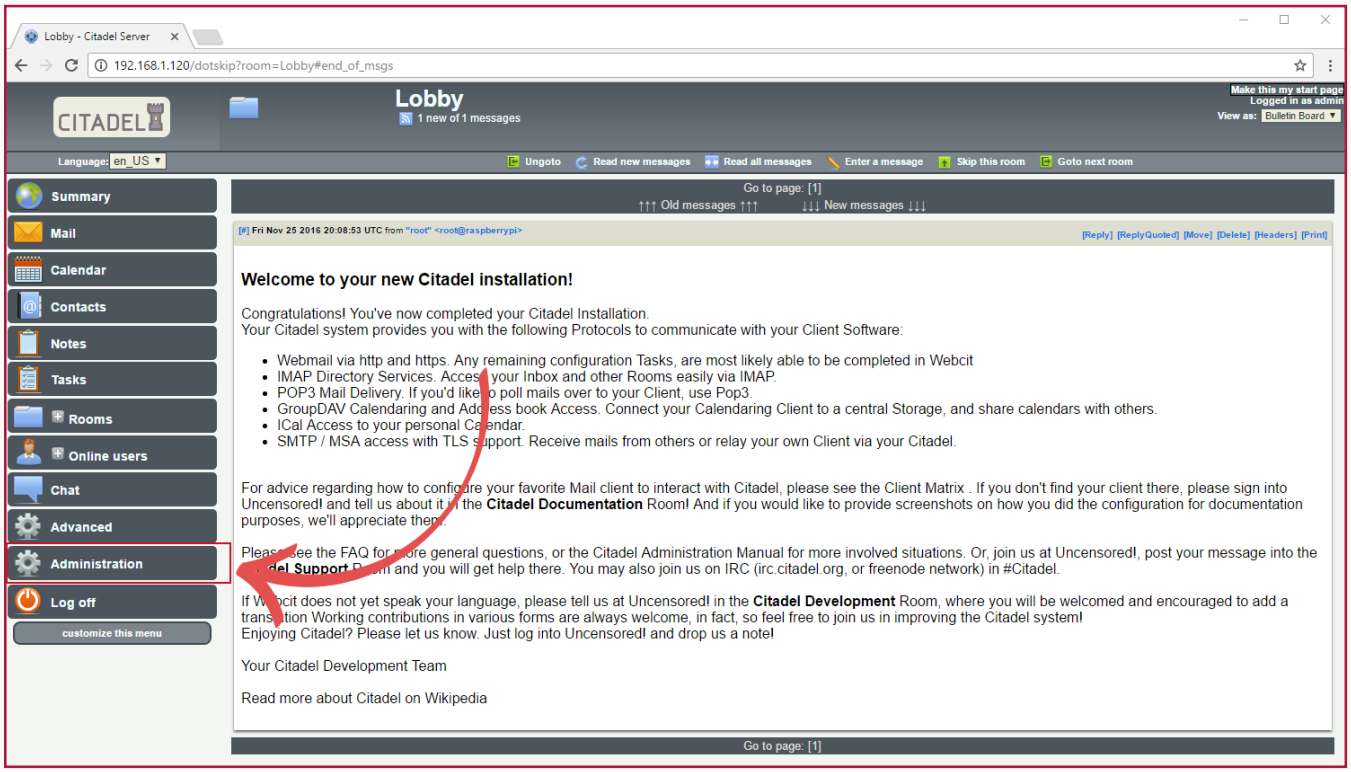
**3.** After the update has completed, run the following command.

```
sudo service clamav-freshclam start
```

**4.** Now enable ClamAV with systemctl so its ensured to start on bootup.

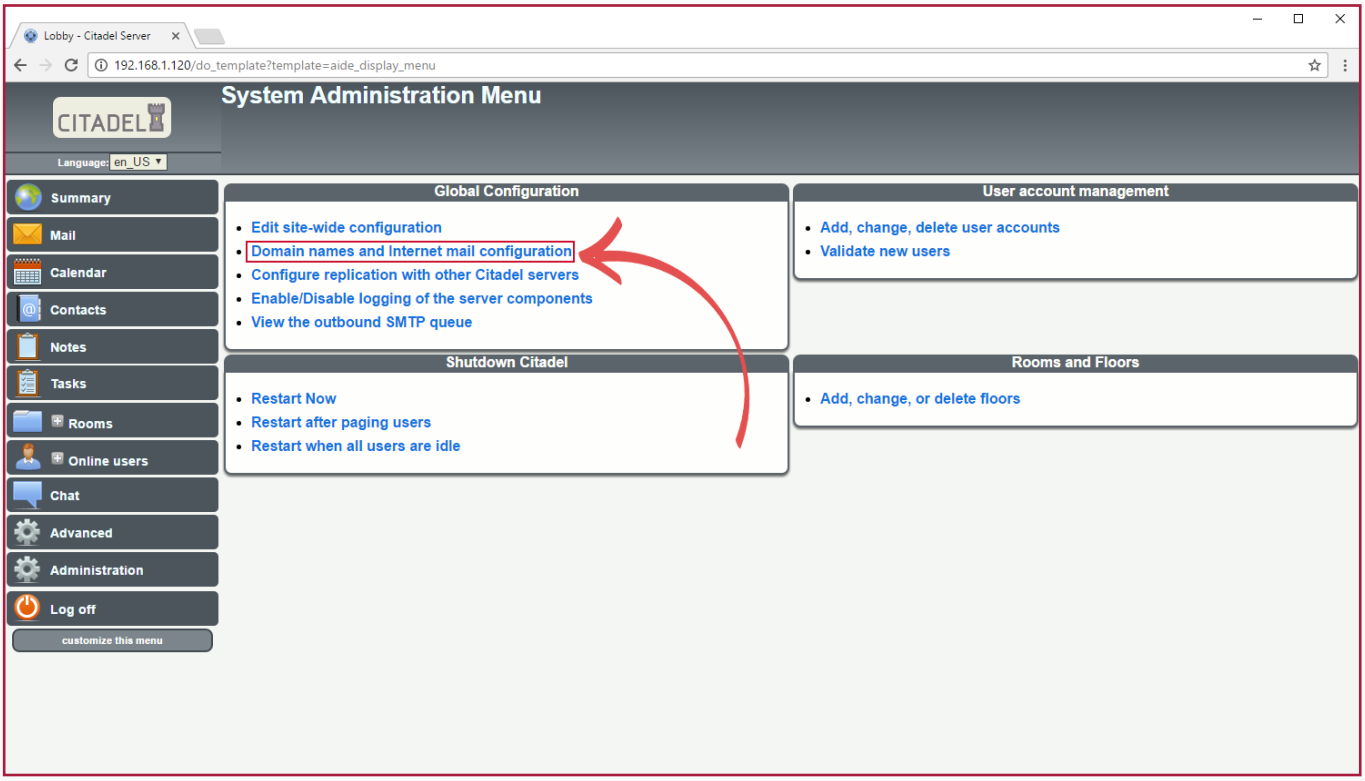
```
sudo systemctl enable clamav-freshclam
```

**5.** We now need to go to Citadels web interface and go back to the **Administration** screen.



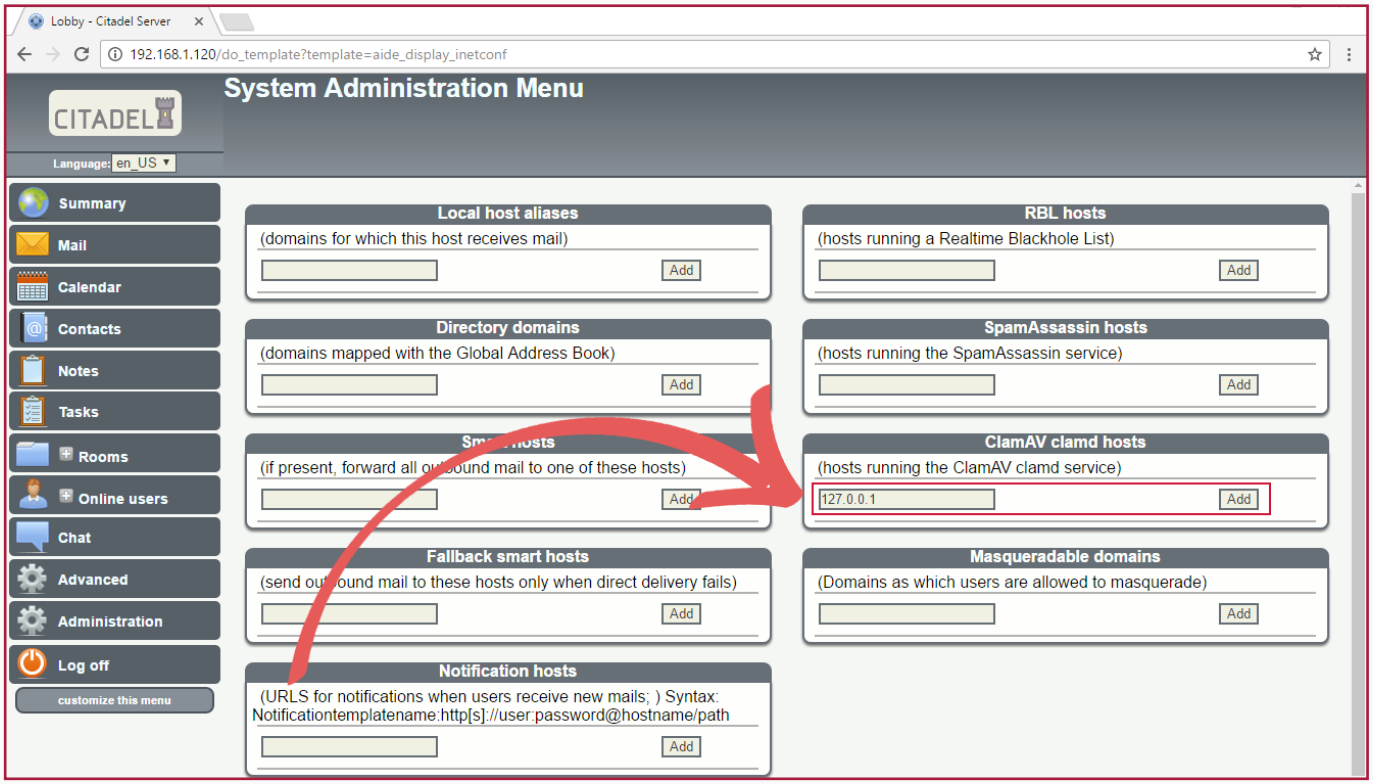
# Setting up ClamAV for Citadel

**6.** Back in here, we need to go to "**Domain names and Internet mail configuration**".




**9.** Now that we are in here, we need to type in the IP Address, **127.0.0.1** under "**ClamAV clamd hosts**".

Once you have typed it in, we can just press the "**Add**" button.



# Basic GIT Server

## Project Description

In this tutorial, we make a simple yet cool GIT server that is perfect for hosting your next code project.

If you're a programmer, then you probably have heard of GIT before. For those who haven't heard of it, GIT is a hugely popular version control software for the development of software.

There are plenty of hosted GIT solutions out there, but you may be wanting your own private GIT server for one reason or another.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection (Ethernet Reccommended)

**Optional**
- Raspberry Pi Case
- External Hard drive or USB Drive
- USB Keybaord
- USB Mouse

## Installing the Raspberry Pi Git server

The process of installing the Raspberry Pi GIT server is straightforward.

If you're not familiar with GIT and its commands, then I highly recommended that you take the time to learn them.

I will go into a few basics further down the page and point out a few good places to learn more about using GIT.

**1.** As always let's first make sure that the Raspberry Pi is up to date by running the following commands.

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

**2.** Now we need also to make sure that GIT is installed by running the following command.

```
sudo apt-get install git-core
```

**3.** We will also need to make sure you have SSH enabled.

If you haven't got this done, then you can find out how to do this in our getting started section of this book.

**4.** Now you might want to store this on a USB drive. If you want to do this, then you will need to mount the USB drive.

You will find the guide on mounting a USB drive on Raspberry Pi located in the getting started section of our book incredibly handy.

This is all you need to do to have the GIT server up and running. It is easy, the next part we will go into creating your first repository and pushing updates to it.

## Setting up your first repository

Now let's make our very own first repository so that we can start pushing code/data to it. There are more complex commands that you will probably want to learn but these basics will be perfect to get you going.

**1.** Firstly, we need to make directory for where our new repository will be stored. The –p tag will create any directories in our path that doesn't already exist.

```
mkdir -p /home/pi/git/myFirstRepository
```

**2.** Now let's move into the directory by using the cd command.

```
cd /home/pi/git/myFirstRepository
```

**3.** Now let's initialize the GIT repository using the bare command.

```
git init --bare
```

**4.** You will need to repeat these steps whenever you need to make a new repository. Now that's all done we're ready to do our first commit.

**5.** On a windows or mac computer you will need to download GIT to be able to start pushing code to our repository.

I recommend downloading and using git-scm, which you can find at https://git-scm.com/

**6.** Once you have installed it you should have some extra options when you right click in a folder.

**7.** In the folder you wish to commit, right click, and then go to GIT bash. This should open a window that looks like a command prompt.

**8.** If this is **not** already a GIT directory, then you will need to initialize it before we can push the code to our Pi.

To do this enter the following command:

```
git init
```

**9.** Now we need to add our remote GIT directory by adding the following line.

```
git remote add myFirstRepository pi@192.168.1.108:/home/pi/git/myFirstRepository
```

**10.** Now we can easily push to data to our repository.

To do this we just need to enter the following commands within the terminal on our Raspberry Pi.

```
git add .
git commit -m "Our first commit"
git push myFirstRepository master
```

**11.** It should come up with a success message. This means our code has been pushed to our git server.

## Setting up your first repository

**12.** To just test to see if everything is working correctly you can clone the repository we just setup to a new folder.

First open GIT bash in a different folder and then run the following commands:

```
git clone pi@192.168.1.108:/home/pi/git/myFirstRepository/
```

**13.** As you can see the Raspberry Pi GIT server is storing our code correctly. Now this is the very basics of git and there is so much more to learn.

Now there is a lot more to learn about GIT and the commands you're able to use.

I have come across a couple of websites that have been incredibly helpful with learning more about GIT and how it works.

If you're looking at getting into programming, then it is important that you learn about code version control systems.

## Accessing the Private GIT server remotely

Now you may want to access your GIT repositories outside your local network. To do this you will need to setup port forwarding on your router.

There are a few security risks involved with allowing communication through a port but is the easiest way for gaining access to your server.

It is essential that you update the pi user's password to something a lot stronger if you're going to allow external access.

If you want to setup external access, then check out my guide on port forwarding with the Raspberry Pi, you can find this earlier on in our book in our "Getting Started" section. You will need to port forward the SSH port (22) to gain access outside your local network.

If you use an external port other than 22 then you will need to specify this either in **~/.shh/ssh_config** or using a URL style address instead like as followed.

```
ssh://pi@XXX.XXX.XXX.XXX:23/home/pi/git/myFirstRepository.git/
```

Where **X** represents your **external IP** and **23** is the **external port.**

# Installing Gitea

## Project Description

In this tutorial we will be showing you all the steps to getting Gitea up and running on the Raspberry Pi.

Gitea, for those who don't know, is a self-hosted Git service, offering an interface much like Github for those who want to self-host there Git repositories but still want to be able to view them through a robust graphical interface.

Gitea is written in Go with the intention of having low minimal requirements meaning it is designed to even run fine on the Raspberry Pi's limited resources. This design helps make it a much better choice than the more demanding Gitlab.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection (Ethernet Reccommended)

**Optional**
- Raspberry Pi Case

## Getting prepared for Gitea on the Pi

**1.** Before we begin setting up Gitea on our Raspberry Pi's we need first to make sure that Raspbian is entirely up to date.

We can do this by running the following two commands on our Raspberry Pi.

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

**2.** Now that Raspbian is entirely up to date we can now install the packages that we require to get started with Gitea.

In this case, we need to install git and the MYSQL server. Use the following command to install these.

```
sudo apt-get install  git mysql-server -y
```

**3.** Let's now create a user that we will be running Gitea under. This command will also create a folder for the user.

We use **--disabled-login** as we don't want people logging into the Raspberry Pi as this user and use **--gecos** to allow us to set a name for the user, in this case, "**Gitea**".

```
sudo adduser --disabled-login --gecos 'Gitea' git
```

## Setting up MYSQL for Gitea

**1.** Before we go ahead and create an MYSQL user and database, we need first to improve the security of our MYSQL server.

To do this, we will by making use of the following command on our Raspberry Pi, this we will run the **mysql_secure_installation** tool.

```
sudo mysql_secure_installation
```

**2.** This tool will ask you several things. We will quickly run through these options over the next few steps.

The first of these is "**Enter current password for root (enter for none) :**", to this just press **Enter** as there is currently no password for the root MYSQL user.

**3.** Next, you will be asked if you want to "**Set root password?**", To this type in **Y** and press **Enter**. Setting a password is essential to ensure your data remains secure.

You will then be asked to enter a new password, make sure you set this to something secure and not easily guessable.

**4.** Now you will be asked if you want to "**Remove anonymous users?**", to this type **Y** and press **Enter**.

Removing anonymous users is essential for data security as it prevents the ability for people to log into the MYSQL server without having a user account created for them.

## Setting up MYSQL for Gitea

**5.** The next option is whether to "**Disallow root login remotely?**", to this type **Y** and press **Enter**.

Disallowing root login remotely makes it more difficult for someone to compromise your database.

To perform a root login, you will have to be on the device directly. A remote connection to the SQL database as a root user will be rejected.

**6.** Now one of the final options is to "**Remove test database and access to it?**", to this type **Y** and press **Enter**.

This final option removes some test data that is included with a default installation of MySQL. These are only there for testing purposes.

**7.** Now with our MYSQL installation far more secure, let's go ahead and log into it by running the following command on our Raspberry Pi.

Remember to have your MYSQL password handy, as you will need to enter it shortly.

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

**8.** Now that we are logged into the MYSQL client we can now start off by creating a new database which gitea will store its data.

To do this just run the following command within the MYSQL client, this will create a new database called gitea.

```
CREATE DATABASE gitea;
```

**9.** With the gitea database now created we can now go ahead and create a user for this database.

We will be naming this user **gitea**, make sure you replace "**ENTERPASSWORD**" with your password.

Make sure this password is different to your root password and is secure.

```
GRANT ALL PRIVILEGES ON gitea.* TO 'gitea'@'localhost' IDENTIFIED BY 'ENTERPASSWORD';
```

**10.** You by now should have created a new gitea database, and created a new gitea user that has all privileges on the gitea database.

To ensure this is all loaded into MYSQL, we need to flush the privileges table.

```
FLUSH PRIVILEGES;
```

**11.** Finally, quit out of the MYSQL client by typing in the following command.

```
exit
```

**12.** Now that we have everything set up and also have MYSQL all configured with a new database and user we can now actually install Gitea to the Raspberry Pi.

## Installing Gitea to the Raspberry Pi

**1.** To start off lets first switch to the "**git**" user that we created earlier, we will perform the next few steps within it.

To switch to this user use the following command on your Raspberry Pi.

```
sudo su git
```

**2.** Let's now change into the home directory for the git user, as this is where we will be creating folders and downloading the Gitea binary for our device.

Use the following command on your Pi to switch directories.

```
cd ~
```

**3.** Creating the folder we need is dead easy, just type the following command into the terminal on your Raspberry Pi.

```
mkdir gitea
```

**4.** Now we need to switch to our new folder and download Gitea itself to our Raspberry Pi.

We can do this by running the following command on our Raspberry Pi.

While we try to ensure we have the latest version here, you can always go to Gitea's official download website to check for newer versions of the program.

You can find the download site at https://dl.gitea.io/gitea.

```
cd gitea
wget https://dl.gitea.io/gitea/1.4.0/gitea-1.4.0-linux-arm-7 -O gitea
```

**5.** Let's now give execution rights to the file by entering the following command.

Giving the file execution permissions will allow us to use a service file to run it.

```
chmod +x gitea
```

**6.** Now that we have downloaded Gitea to our Raspberry Pi let's go ahead and now create a service for it.

This service will allow us to have Gitea launched automatically at startup and will enable us to stop and restart it easily.

Run the following command on your Raspberry Pi to begin writing the service file.

```
sudo nano /etc/systemd/system/gitea.service
```

## Installing Gitea to the Raspberry Pi

**7.** Within this file, we need to enter the following lines.

These lines tell the service manager how to handle the service, where to start the application and what user to run it under.

```
[Unit]
Description=Gitea (Git with a cup of tea)
After=syslog.target
After=network.target

[Service]
# Modify these two values and uncomment them if you have
# repos with lots of files and get to HTTP error 500 because of that
###
# LimitMEMLOCK=infinity
# LimitNOFILE=65535
RestartSec=2s
Type=simple
User=git
Group=git
WorkingDirectory=/home/git/gitea
ExecStart=/home/git/gitea/gitea web
Restart=always
Environment=USER=git
HOME=/home/git

[Install]
WantedBy=multi-user.target
```

**7.** Once you have entered everything into the service file and are sure everything is correct, we can save it by pressing **CTRL + X** then **Y** and finally pressing **ENTER**.

**8.** Finally, let's enable our Gitea service and start it up by running the following command on our Raspberry Pi.

This service will start up Gitea so we will be able to begin the process of configuring it.

```
sudo systemctl enable gitea.service
sudo systemctl start gitea.service
```

# Setting up Gitea on the Raspberry Pi

**1.** Now that we have installed and setup MYSQL and installed Gitea itself we can proceed to configure and set up Gitea itself.

To do this, you will need to have your Raspberry Pi's IP address handy.

In a web browser go to your Raspberry Pi's IP Address followed by :3000. We have an example of a valid URL below.

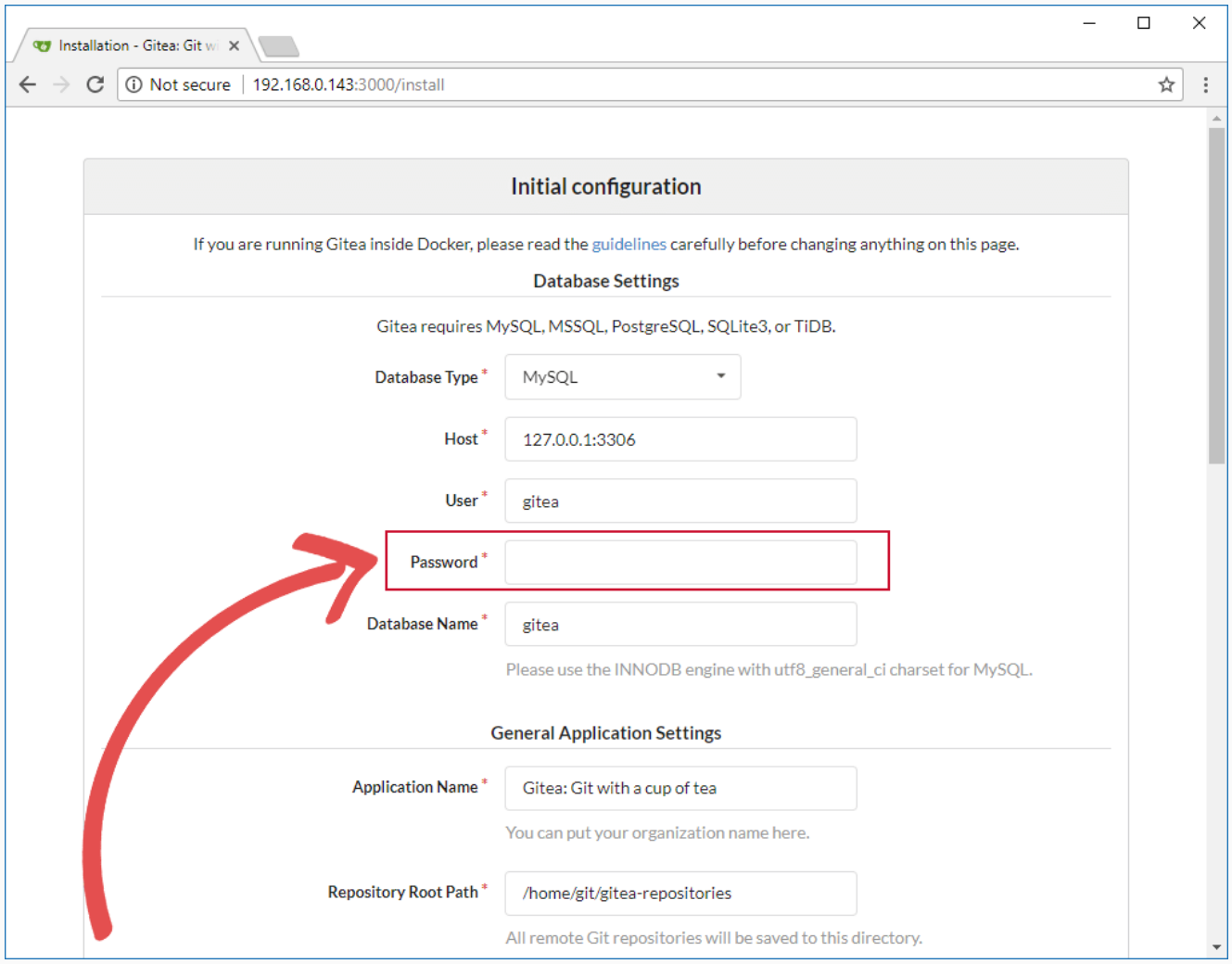
```
http://192.168.0.143:3000
```

**2.** When you go to your Raspberry Pi's IP Address followed by :3000 you should be initially greeted with the screen below.

This screen allows you to configure everything you might need for setting up Gitea such as changing the domain name, the HTTP Port the domain name and numerous other things.

The first thing we are going to deal with though is the database settings, for this you will need to have the password you set handy.

If you followed our guide correctly, you can keep user and database name the same.



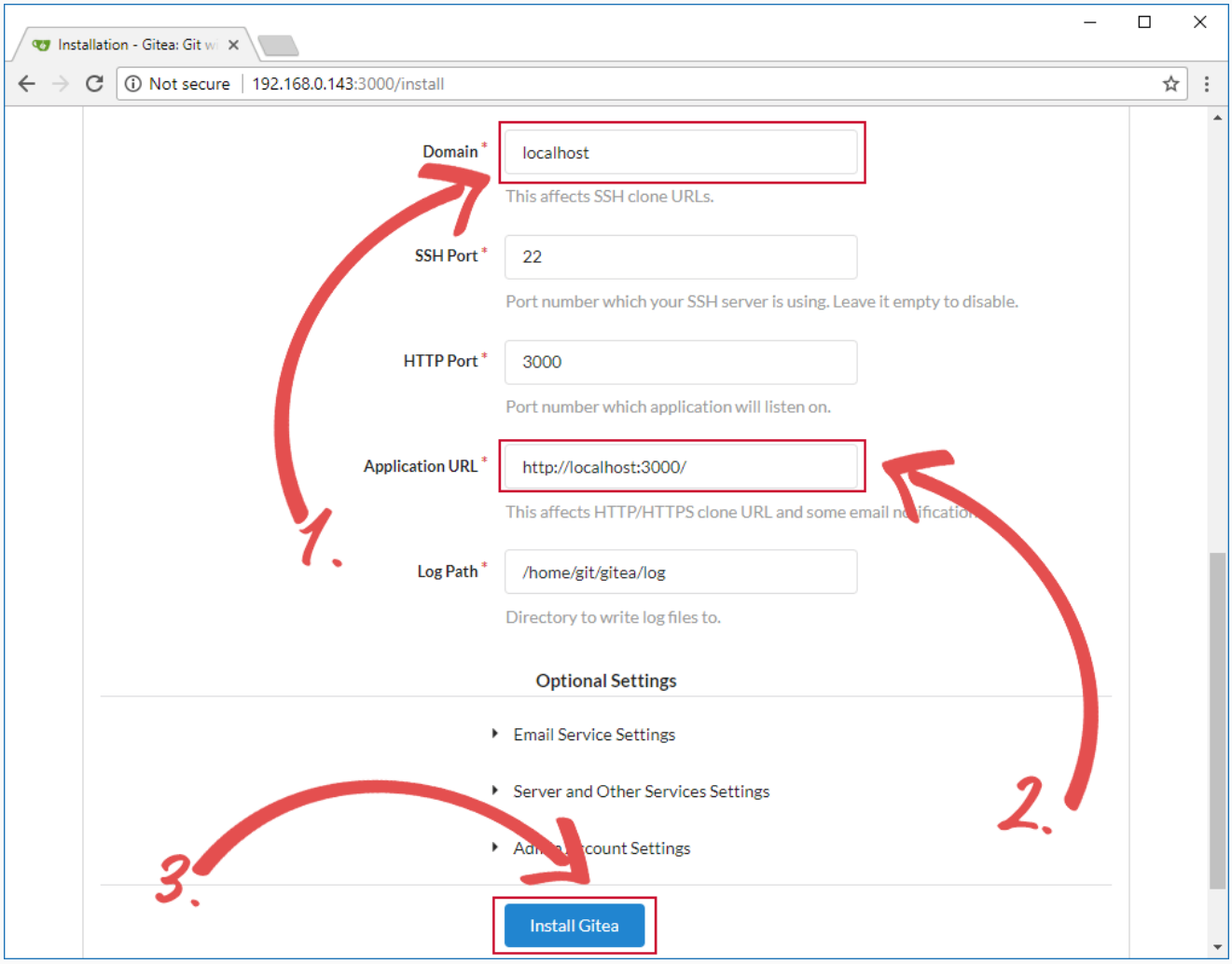
**3.** Scroll down till you see a setting for the Domain (**1.**), and Application URL (**2.**) These are quite important to set as they can create a very broken version of Gitea if they are set incorrectly.

You can use your Raspberry Pi's local IP address such as **192.168.0.143**. It's recommended to use your local IP address instead of localhost as it allows other devices on the network to interact with Gitea.

Alternatively, you could use your external IP address or a domain name with proper port forwarding. For example, you could use something like git.pimylifeup.com which is pointed towards your Raspberry Pi's IP address. Of course, to do this, you will require a static IP address.

Next, you need to use what you set for the Domain (**1.**) to replace "**localhost**" in the Application URL (**2.**)

Once you have set everything, you can go ahead and press the "**Install Gitea**" button located at the bottom of the web page.



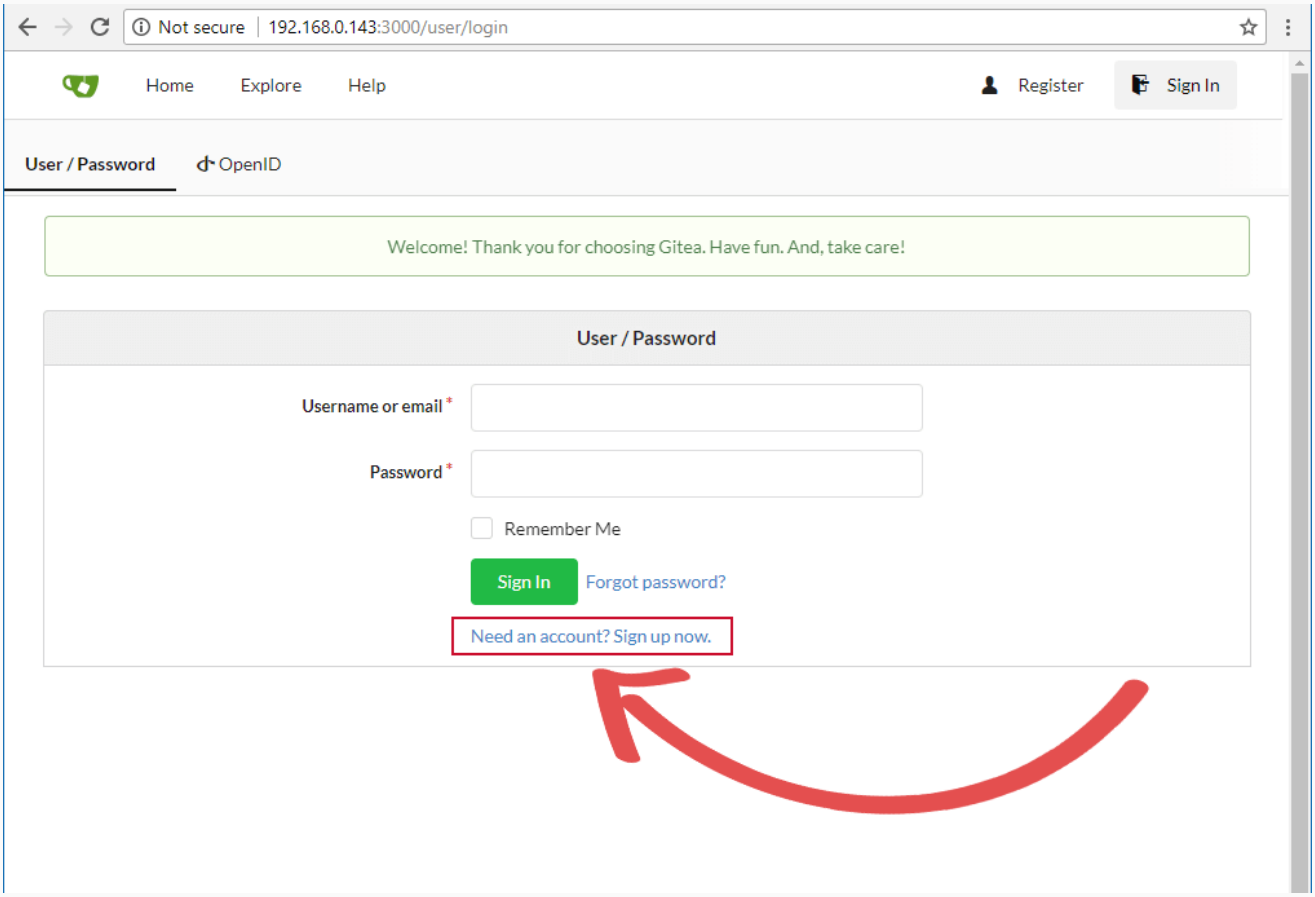
# Setting up Gitea on the Raspberry Pi

**4.** Now is a perfect time to grab a cup of coffee as the installation process of Gitea can take considerable time to complete.

Once the installation process has completed your web browser will be redirected to the login page as shown below.

Now all you have to do is register an account then you will have Gitea up and running perfectly.

Take some time to explore its interface and learn the ropes. If you are familiar with GitHub, you should be familiar with this.



# GITLab Server

## Project Description

In this tutorial, we will be setting up a piece of software called GITLab, this is a more fully featured version of our GIT tutorial, but is also a lot more demanding on the Raspberry Pi.

If you have ever used a GIT hosting service like GitHub, then you will find GITLab a lot like that, with it providing a web frontend to GIT. Having a web frontend can be quite useful as it provides a helpful visual interpretation of what is happening in your GIT repository.

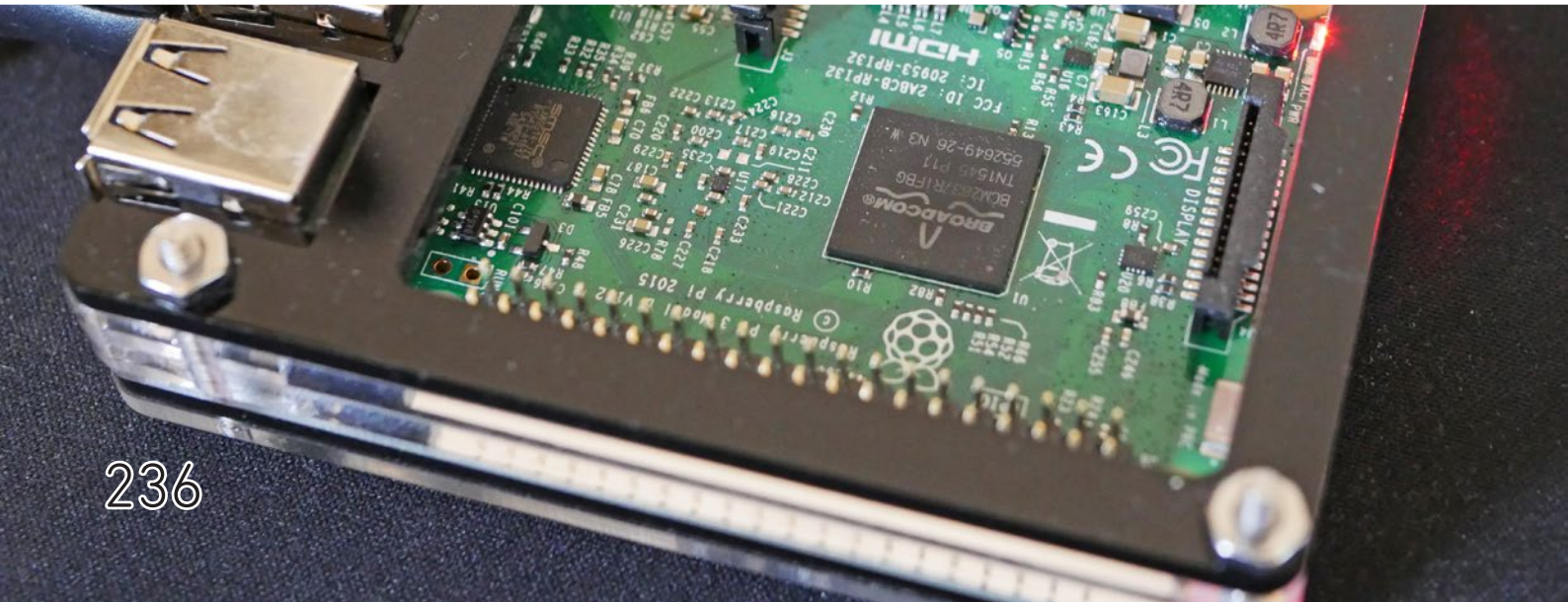
It also offers the ability to manage things like pull requests and forks easily. It can be very useful to setup if you plan to utilize GIT often, or with a team.

We recommend only doing this on a fresh install of Raspbian as GITLab can be quite intensive.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection (Ethernet Reccommended)

**Optional**
- Raspberry Pi Case
- USB Keybaord
- USB Mouse



# Installing GIT Lab

**1.** Before we install GIT Lab itself, we need to install a few other things. These packages we need to install are CURL, the open SSH server, postfix and a package that allows "**deb**" to support the HTTPS protocol.

We can install these using the following command:

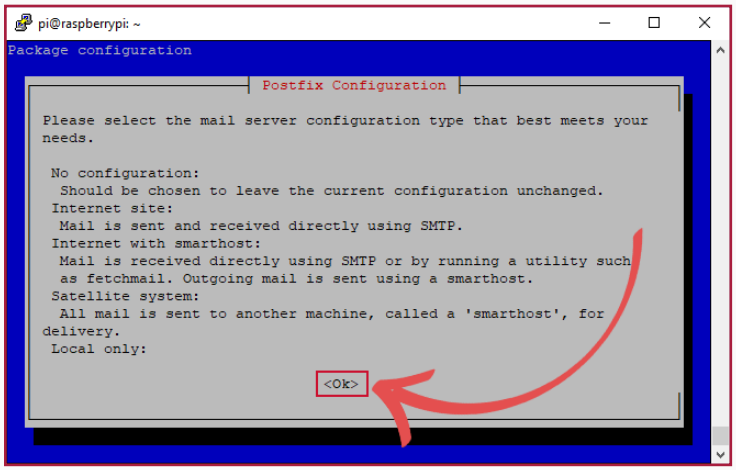
```
sudo apt-get install curl openssh-server ca-certificates postfix apt-transport-https
```

**2.** When Postfix begins to install you will be greeted with the following screen, this screen just explains the various options you can choose from.

The two easiest options to setup are "**Internet Site**" and "**Local Only**" delivery.

**Internet site** allows all your mail traffic to run through SMTP. **Local only** does what it says, and only routes mail traffic locally.

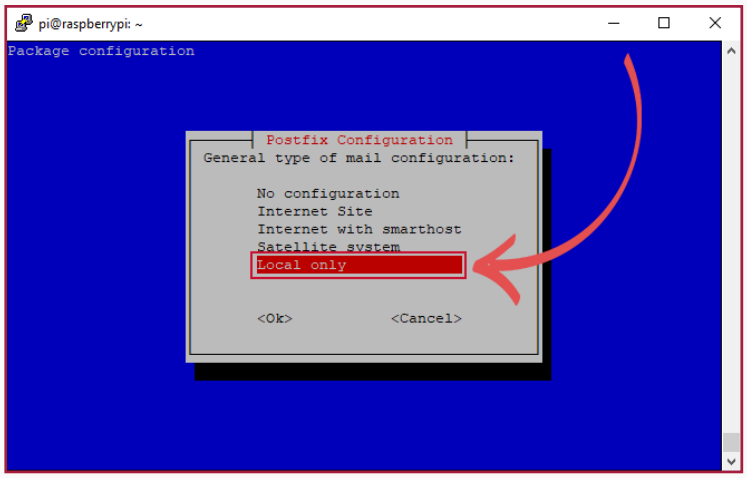
Select "**OK**" and press **enter** to continue.




**3.** On the next screen, we are going just to select "**Local only**", this is because it is the easiest to set up.

It is also useful as we only plan to use this GIT Lab installation within our local network, so we don't want emails getting sent outside it anyway.

Once you have selected the option, you want to utilize, select "**OK**" and press enter.

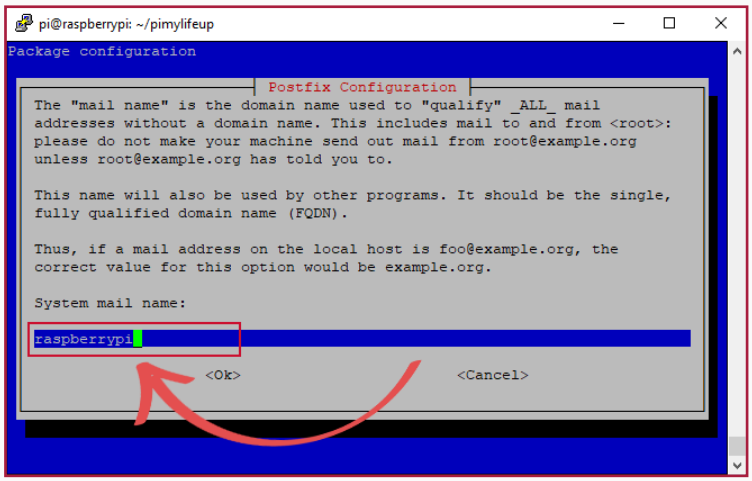


**4.** The screen will now ask us to set the "**mail name**" for Postfix to utilize.

If I wanted to utilize **gus@pimylifeup.com** I would set the "**System mail name:**" to **pimylifeup.com**

Since we don't plan to utilize mail, we are just going to leave it as **raspberrypi**.

Select "**OK**" and press **enter** to finalize the installation.




**5.** Once that is all done, we need to grab GIT Lab's GPG key. This key allows the Raspberry Pi to encrypt and sign your data when communicating with their servers. We can download and add the key with the following command:

```
curl https://packages.gitlab.com/gpg.key | sudo apt-key add -
```

**6.** With that finally done, we can start the next process. That is to begin the installation of all the packages that GIT Lab relies on upon, please note that this can take some time as there are a fair few things this script needs to do.

Run this command to download and run the installation script:

```
sudo curl -sS https://packages.gitlab.com/install/repositories/gitlab/raspberry-pi2/script.deb.sh | sudo bash
```

**7.** Once that has finished installing we can proceed with finally actually installing GITLab itself, this is another process that takes some time. It is about a 150mb download.

Run this command in terminal to begin the installation process:

```
sudo apt-get install gitlab-ce
```

**8.** The final command that needs running to finalize the installation of GITLab is the one shown below.

This command reconfigures and sets up all the settings for GITLab and installs all the required databases.

```
sudo gitlab-ctl reconfigure
```

Now GITLab should be successfully installed and ready to be used, we will go over creating your initial repository and making your first commit over the next couple of pages.

# Creating your first GIT Lab Project

**1.** To access our newly setup GitLab page we will need to go to the Raspberry Pi's local IP Address if you don't have the IP address you can quickly find it out with the following command:

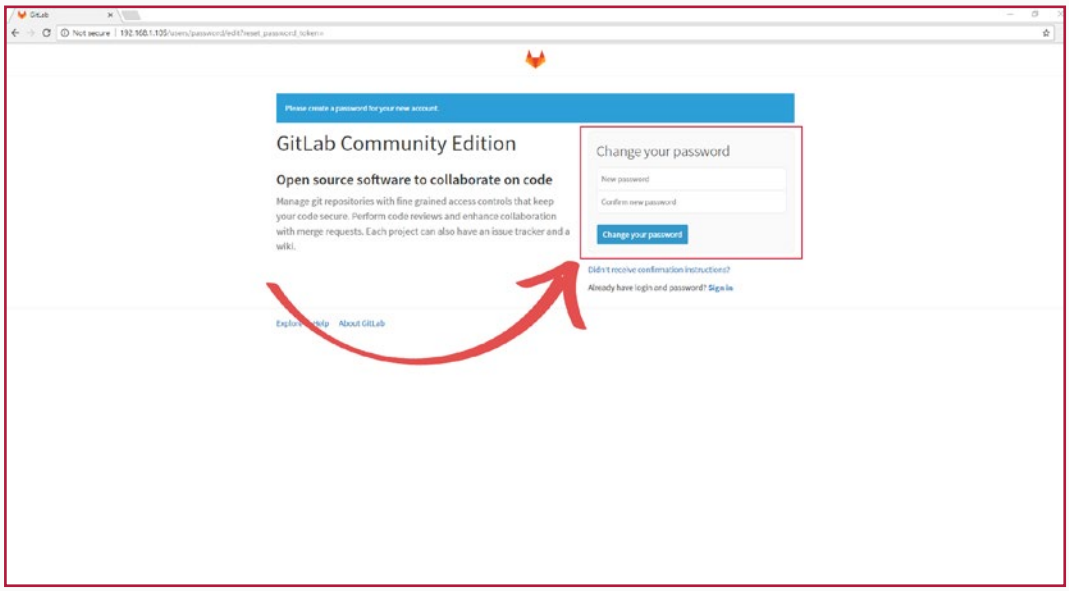
```
hostname -I
```

Once you have your Raspberry Pi's IP address, load it up in your favorite web browser, in my case, I would type the following into my web browser:

```
http://192.168.1.105/
```

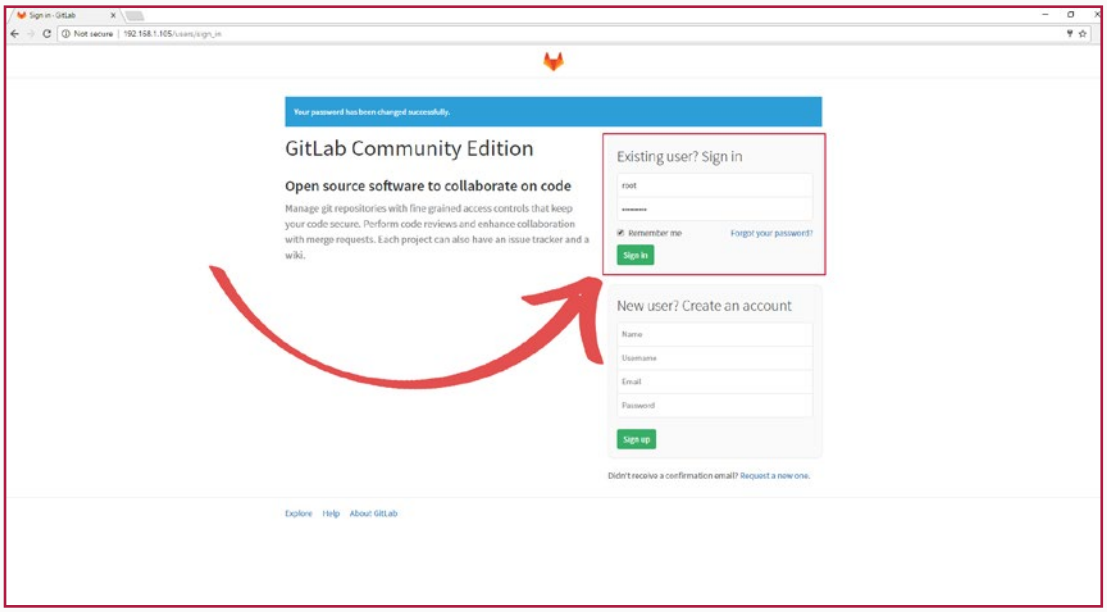
**2.** You will now be greeted by a screen asking you to set the password for the root account. We recommend setting something secure for this, as this account has full control over all of GitLab's.

Once you are happy with your password, press the "**Change your password**" button.




**3.** Now you can log in to GitLab with your new password, for the username make sure you use root.

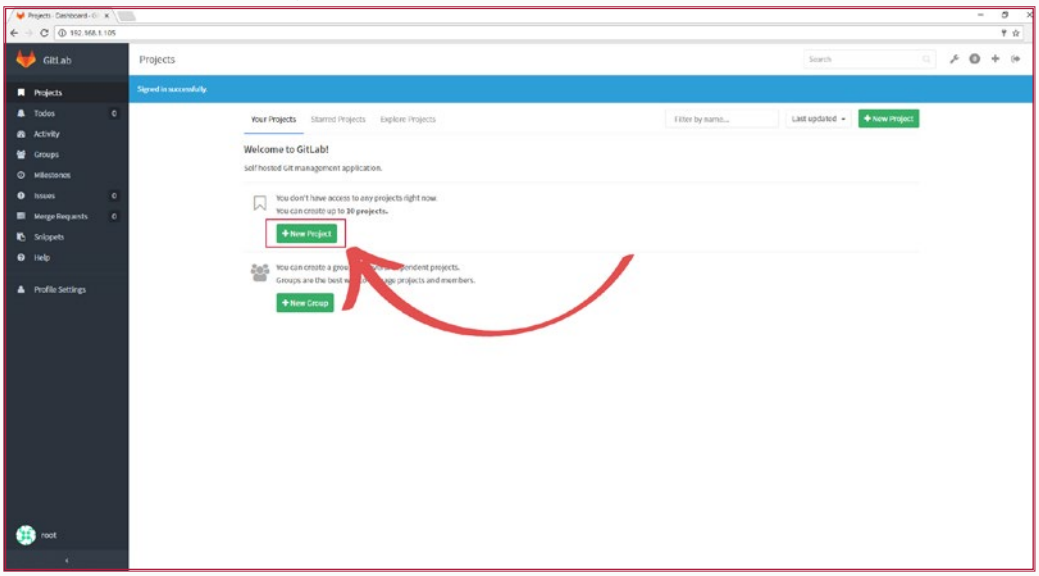
Once you are ready, you can press the sign in button.



# Creating your first GitLab Project

**4.** Once we are logged into GitLab we will be greeted with the dashboard page, at the moment this contains nothing but a few buttons to create a new group or project.

For our tutorial, we will be pressing the "**New Project**" button.



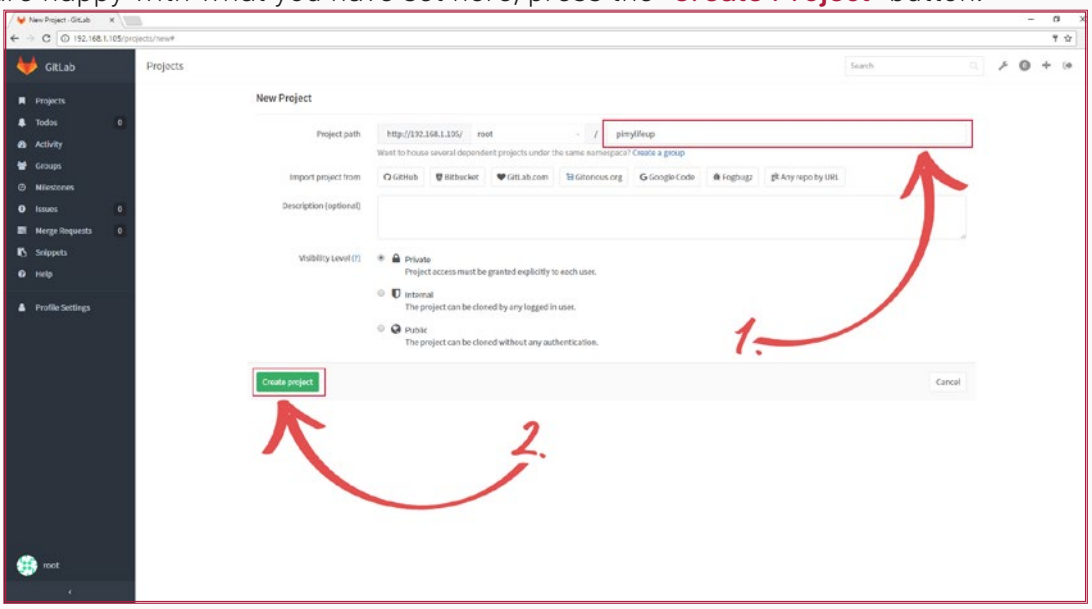

**5.** The New Project screen is filled with a variety of different options. GitLab allows you to import Git projects from a vast range of various services, such as GitHub and Bitbucket. However, for our tutorial we will be just creating a new repository.

The main thing we need to set in here is the project name. This is the far-right text box for the project path. We will be using the name "**pimylifeup**".

We will be leaving the visibility level set to "**Private**", this just means only our user can interact with the repository unless we specifically allow another one.

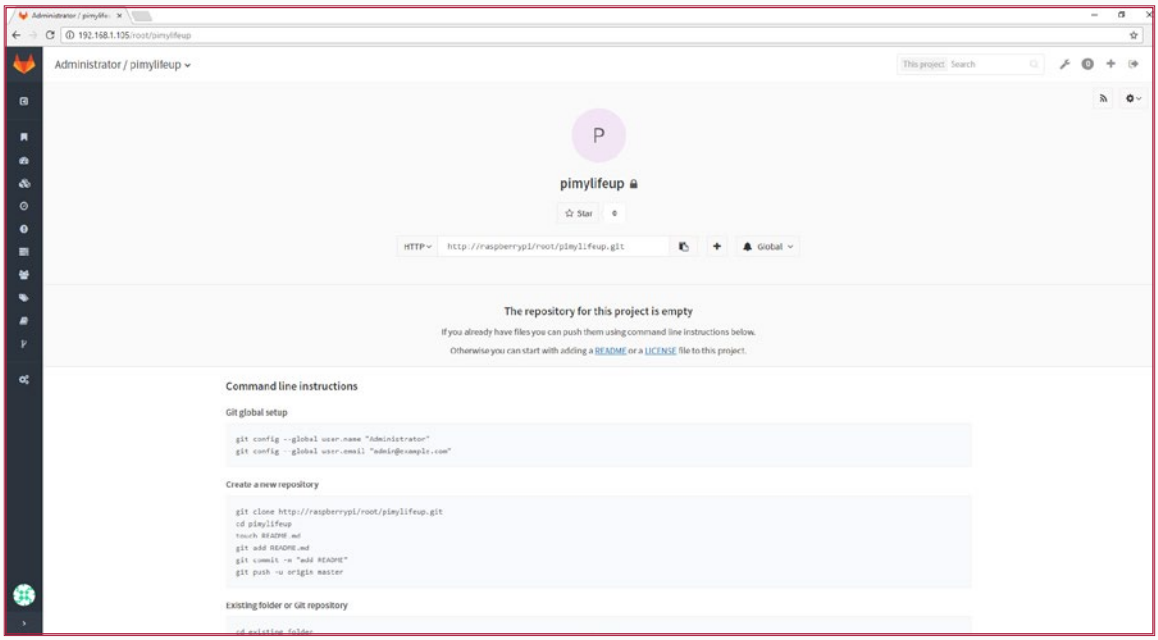
The two other options are "**Internal**" which allows any logged in user to be able to clone it, and "**Public**" which allows anyone to clone the repository.

Once you are happy with what you have set here, press the "**Create Project**" button.



# Creating your first GitLab Project

**6.** The screen you will be greeted with now is the empty project page, there is not a huge amount to do here at the moment, but we will soon be adding some files to this and populating it.




**7.** Now back in the terminal, we will clone our brand new repository and add our first file. To get started we need to first set some details.

You can change the username and the email below to whatever you want.

```
git config --global user.name "Administrator"
git config --global user.email "admin@example.com"
```

**8.** With that done we can go about cloning our new repository if you chose to use a different project name make sure you swap out "**pimylifeup**" with your name.

Now to clone the repository we will run the following couple of commands, this will grab the repository and move us into its directory.

```
cd ~
git clone http://raspberrypi/root/pimylifeup.git
cd pimylifeup
```

**9.** Now we have cloned the repository, let's create our first file and commit it. This file will just be an empty file named Readme, but this should give you an idea on how to commit files.

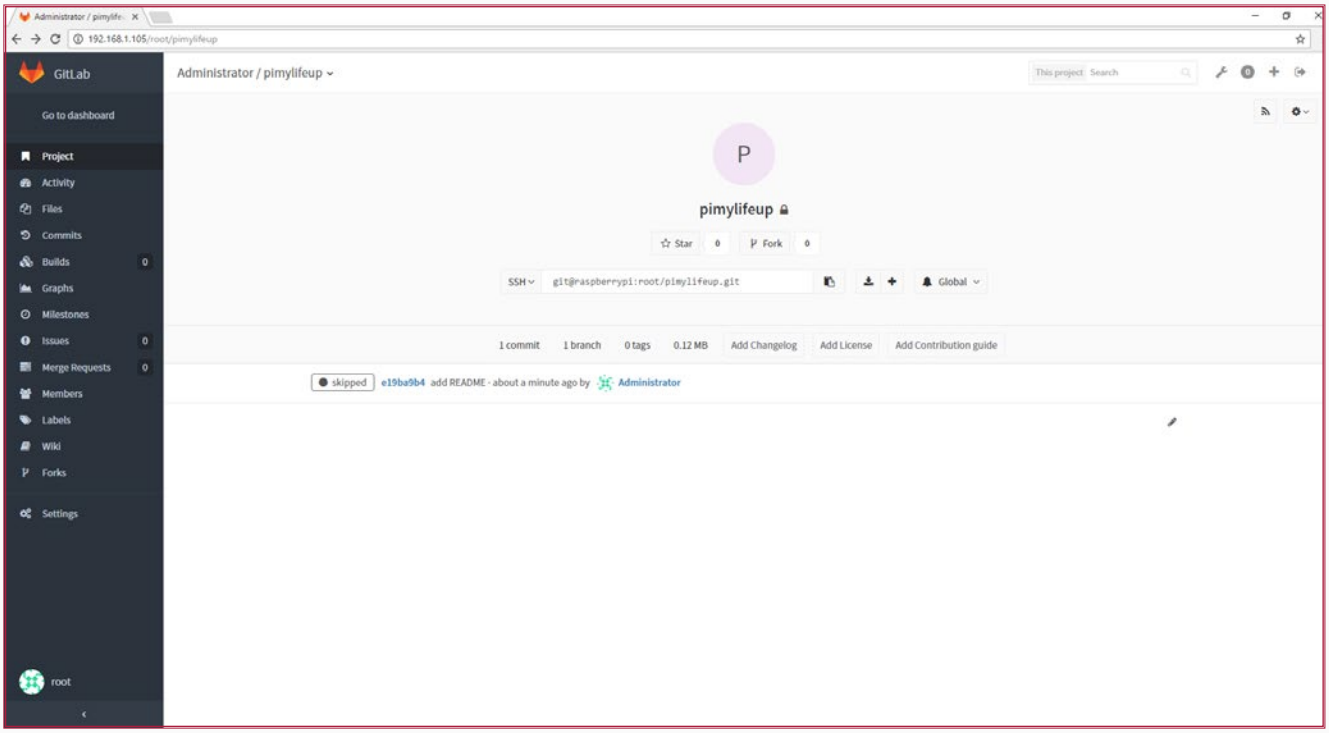
To add and commit this file, run the following few commands, they should be somewhat self-explanatory.

```
touch README.md
git add README.md
git commit -m "add readme"
git push -u origin master
```

## Creating your first GitLab Project

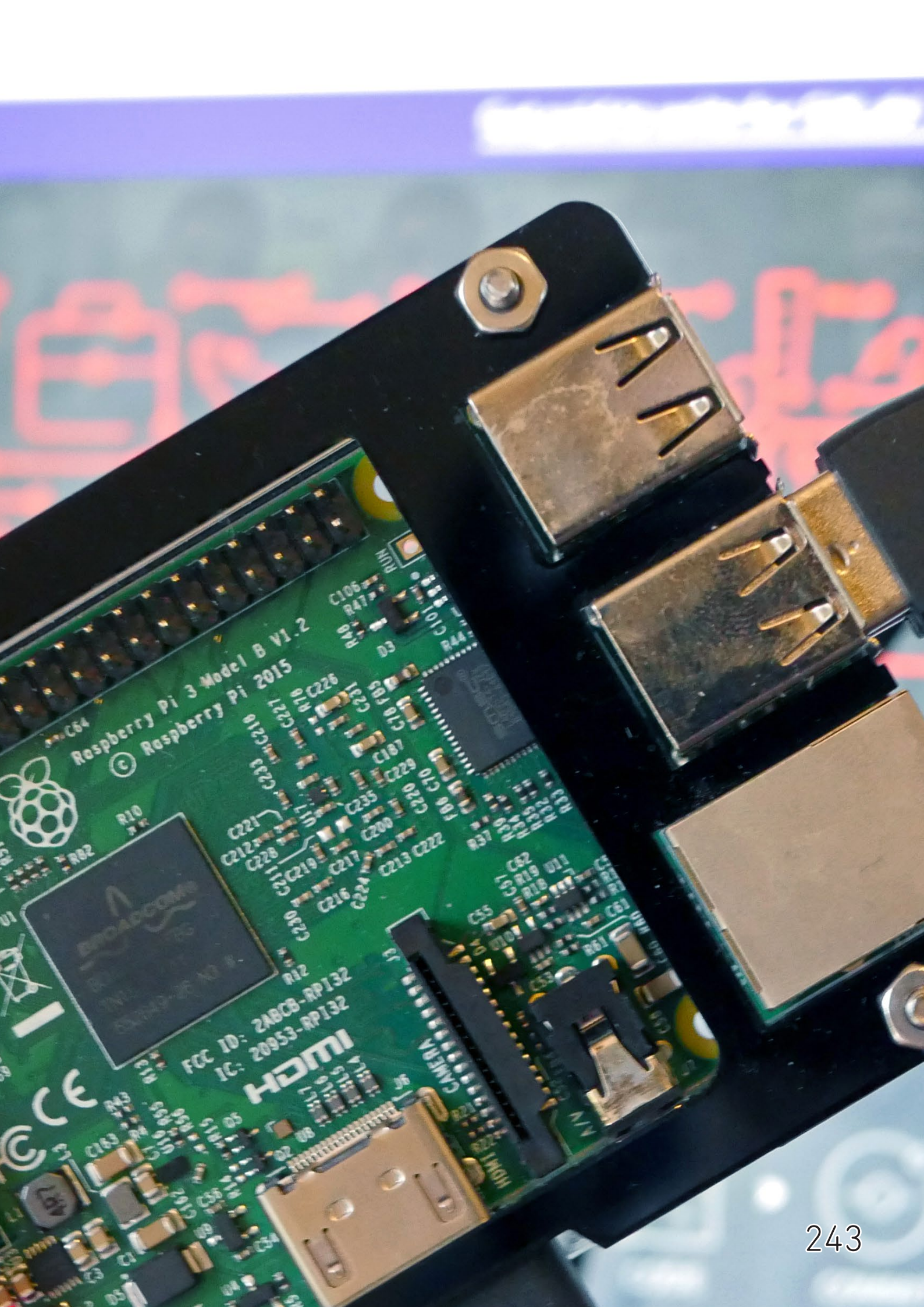
**10.** Now you can refresh your project page and now be able to see your new commit.

You can now click on the commit and check out the changes within the web interface rather than having to rely on Git's command line.




Congratulations, your Raspberry Pi GitLab server should now be up and running with its first project created.

We have only really scratched the surface of what GitLab is capable of doing, you can also use it to setup CI Builds, manage forks and merge requests, and also utilize it to help lodge issues and set project milestones.


Raspberry Pi 3 Model B V1.2
© Raspberry Pi 2015
FCC ID: 2ABCB-RPI32
IC: 20953-RPI32
HDMI
CAMERA
RUN

# Mumble Server

## Project Description

You will find that mumble is often used amongst gamers to co-ordinate over voice chat while playing video games (Same as Teamspeak). It can also be just used as an open voice chat room or pretty much anything you where you require voice chat rooms.

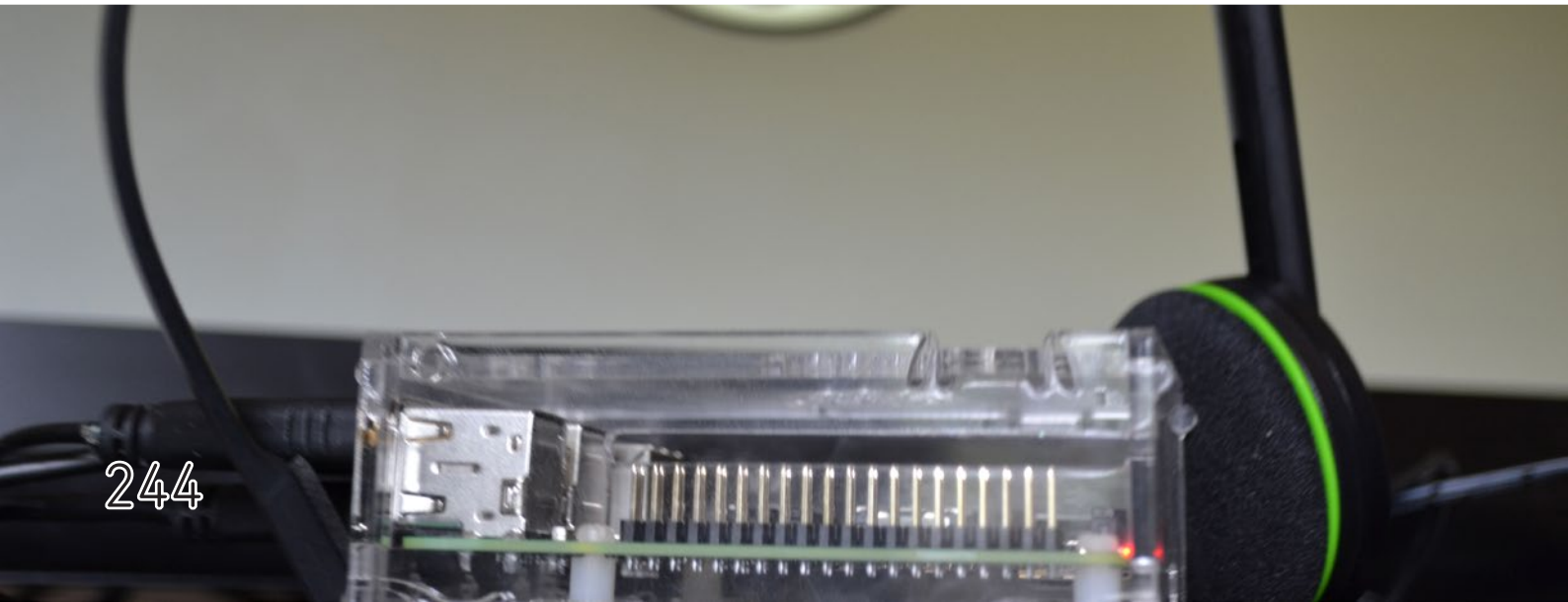
Mumble is not the only Raspberry Pi VOIP solution out their others such as Asterisk offers a much more complete package offering features such as inbound & outbound calls, audio/video conferencing and much more.

Mumble, however, is much easier to use software and provides enough functionality for most people.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection (Ethernet Reccommended)

**Optional**
- Raspberry Pi Case
- External Hard drive or USB Drive
- USB Keybaord
- USB Mouse



## Setting up Mumble on the Raspberry Pi

**1.** On the Raspberry Pi open the terminal or login to the Raspberry Pi over SSH

**2.** First, we can install the mumble server onto the Raspberry Pi by typing following simple command into the terminal.

```
sudo apt-get install mumble-server
```

**3.** Now we need to do some basic configurations such as set up a superuser so we can connect to the server and control it via the app, to do this run the following command.

```
sudo dpkg-reconfigure mumble-server
```

**4.** This command will present you with a few options, set these however you would like mumble to operate.We set the options to the following:

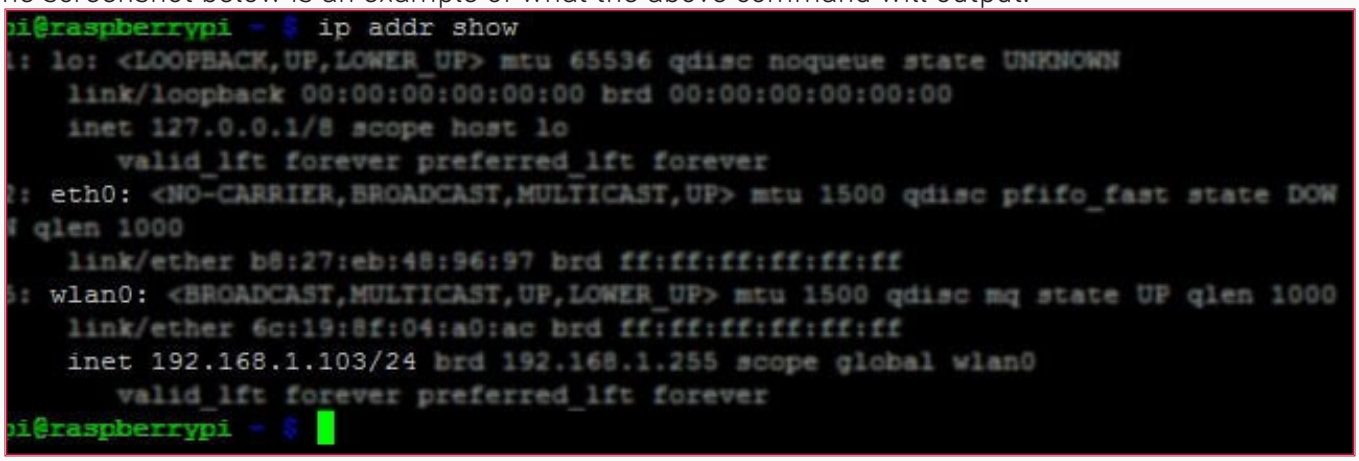
- **Autostart:** I selected **Yes**
- **High Priority:** I selected **Yes** (This ensures mumble will be given top priority even when the Pi is under a lot of stress by other applications)
- **SuperUser:** Set the password to whatever you want here. This account will have complete control over the server.

**5.** You need the IP address to connect to, to get this type the following into the terminal:

```
ip addr show
```

**6.** Write this number down next to inet (It will be under either eth0 or wlan0) as we will need it to connect to the server and setup port forwarding for outside access.

The screenshot below is an example of what the above command will output.



## Configuring the Raspberry Pi Mumble Server

Now we will probably want to set up a few things on the Mumble server such as welcome message, password (if you don't want anyone just joining).

**1.** In SSH or the terminal app enter the following command to begin editing the Mumble config.

```
sudo nano /etc/mumble-server.ini
```

**2.** Everything in the file is self-explanatory. However we will go through some options that can be worth changing.

- Find **welcomeText** and update to whatever you would like displayed when a user joins the channel.
- Find **serverpassword** and update if you would like a password for users looking to join the server.
- Uncomment **registersever** by removing the # and replace **mumble server** with the name you would like the base channel to be called.

Once you have finished changing the values you want, don't forget to save the file using **Ctrl +X** then pressing **Y** and then pressing Enter.

**4.** You will need to restart the server for changes to take place. Enter the following command

```
sudo /etc/init.d/mumble-server restart
```

Once the server has rebooted, you can now move onto accessing it via the mumble client.

## Accessing the Mumble Server

There are a few things we will need to do to access the server now, and this includes downloading the mumble software. The Mumble client is available for Windows, Mac OS X, and even Linux.

**1.** Download & install the Mumble client from their website https://mumble.org. (Don't install the server)

**2.** Open the program and go to **Server -> Connect**

**3.** In this screen we need to add all the details to log in to the server:

- **Label:** The name that will appear in your server list.
- **Address:** This is the IP Address we wrote down earlier
- **Port:** 64738 (Unless you changed it)
- **Username:** superuser
- **Password:** The one you set earlier using the dpkg command

You should now be able to connect to your server.

**4.** Now we have it working there is a few more things you may want to configure and setup.

## Setting up an Admin User

You will want to give yourself admin rights so that you don't need to log in to the superuser again.

1. In the Mumble client login as the superuser (After having logged in under a different account)
2. Right-click on the **server name** and go **edit**.
3. In here go to the group's tab and select admin from the drop-down box.
4. Type in or add your username in the drop down box next to add in the members' column.
5. Click add and then exit.
6. Now login under the username you just added and you should be able to add, edit channels and other complete other admin tasks on your Raspberry Pi mumble server.

## Setting up external access for the Mumble server

To set up external access we will need to change some settings in the Router. All routers are different so I will provide the details on what I have done to for our AC1750 TP-Link Router.

1. Head to your Router admin page (This will typically be **192.168.1.1** or **192.168.254**)
2. Enter the username and password. Default typically is admin & admin.
3. In here look for **Forwarding -> Virtual Server** and then click on add new.
4. In here enter the following details:
   - **Service port:** In this case, it is **64738**
   - **IP Address: 192.168.1.103** (Replace this with the local IP address of your Raspberry Pi)
   - **Internal Port:** We want this to be the same as the service so, **64738**
   - **Protocol:** All
   - **Status:** Enabled
5. These settings route traffic destined for port **64738** to our mumble server **(192.168.1.103)**
6. You should now be able to connect to the Raspberry Pi mumble server outside your network. You may need to restart the router for changes to take effect.

The best way to test if this is working is to get a friend who has Mumble installed or is willing to install to connect to your server.

Provide them with your external IP and or dynamic DNS, and they should be able to connect to the Raspberry Pi Mumble Server.

If you plan to use this server a lot, we do recommend following our Dynamic DNS tutorial. The advantage of that is that it means you won't have to give out your IP Address every time it changes.

If they are unable to connect you can try the following:

- Check your router settings and confirm they are correct.
- Check your IP hasn't changed (Some IPs will provide you with a dynamic IP rather than a static IP)
- Restart the router.
- Restart the Raspberry Pi

# Setting up a Minecraft Server

## Project Description

A Raspberry Pi Minecraft Server is a cost-effective way to have your very own private server with your friends or just for stuffing around on.

I have had mine running for quite some time now and haven't come across too many problems. It is important to know that this server will not be able to hold many people due to limitations with the Raspberry Pi's performance. From my experience, anything over five the server starts to get a bit wonky.

We are also going to be using the spigot version of Minecraft as the default Minecraft Server didn't work for me. I found it crashed a lot, lagged and was very unstable. Three things you don't want for a server.

In this project, we will install Java onto the Raspberry Pi and install the server. We will make a few changes to optimize the server for the Raspberry Pi 2 and Raspberry Pi 3 if you have one.

Also, we will set it up so you can access it on the web and reboot the software if the Raspberry Pi goes offline for any reason.

## Equipment

**Required**
- Raspberry Pi
- 4 GB (Micro) SD Card (8 GB+ Recommended)
- Network connection
- Raspberry Pi Power Supply

**Optional**
- USB Keyboard
- USB Mouse
- HDMI Cable
- Raspberry Pi Case

# Setting up the Raspberry Pi Minecraft Server

In this tutorial, we will work completely within the terminal. In fact, it is best to change the boot mode of the Raspberry Pi, so it boots directly into the terminal for the Minecraft server.

Minecraft servers are hugely demanding on the Raspberry Pi's hardware, so every extra bit of resources made available to it, the more it can handle.

**1.** First, let's make sure everything on the Raspberry Pi is up to date by running the following two commands.

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

**2.** Now we will need to make a couple of changes in the raspberry pi config tool. Let's bring the tool up by entering the following:

```
sudo raspi-config
```

**3.** First let's go to **Advanced Options**->**Memory Split**, update this to **16**. (This will allow for more memory to be free for the server)

**4.** Now is also a good time to change the bootup option, so it boots straight into the terminal.

Go to **Boot Options**->**Desktop/CLI**, Press **Enter** with **Console selected** to enable booting to console.

**5.** To truly make the most of the Minecraft server, you will need to overclock your Raspberry P.

The following steps only work with the Raspberry Pi 1, and the Raspberry Pi 2 as the raspi-config tool does not currently support the Raspberry Pi 3.

While still in the raspi-config tool, Go to **Overclock**, here pick the "**High**" overclocking option.

This option will give the Raspberry Pi a good overclock, though make sure you have a decent, stable power supply. Most USB chargers will not cut it.

**6.** If you haven't activated SSH before, make sure you go to **Advanced Options**->**SSH** and select "**Enable**". You will need this for easy access to your Raspberry Pi.

**7.** Now go to "**Finish**" and reboot your Raspberry Pi using the following command.

```
sudo reboot
```

**8.** Before we get too ahead of ourselves, now is a great chance to grab the IP address of the Raspberry Pi. You will need to write down the value from the next command. We will need to use this later.

In the terminal, enter the following command:

```
sudo hostname -I
```

**9.** Luckily for us, Java is already available on the latest versions of Raspbian, just in case your distribution doesn't have the latest version of java installed you can use the following command:

```
sudo apt-get install oracle-java8-jdk
```

## Setting up the Minecraft Server

**10.** Next, we will set up some folders that will act as our base for our Minecraft server, and download the Spigot build tools. These build tools are handy for setting up the latest version of Spigot.

Type the following commands into the terminal:

```
sudo mkdir /home/minecraft
cd /home/minecraft
sudo wget -O BuildTools.jar https://pimylifeup.com/out/spigotbuildtool
```

**11.** Now we have the **BuildTools.jar** file downloaded and ready to go, we can run this file to compile the latest available version of Spigot.

If you are using the Raspberry Pi 1 (A or B/B+) then go to step **11b**, newer versions of the Raspberry Pi you can go to step **11a**.

**11a.** To start compiling the latest version of spigot, just run the following command. Please note that this will take some time. Expect it to take at least **15-30 minutes** to finish.

```
sudo java -jar BuildTools.jar
```

**11b.** For those running an older Raspberry Pi, you can obtain the latest build we have compiled using the following command:

```
sudo wget -O spigot-1.10.2.jar https://pimylifeup.com/out/spigotlatestbuild
```

**12.** To make sure that spigot has successfully been compiled or downloaded we can use the following command to see all the files now present:

**Note**: Pay attention to the file starting with spigot, you will need to know the full name to be able to start it, for instance, the one the BuildTools.jar generated for us was **spigot-1.10.2.jar**.

```
ls
```

**13.** Now we are ready to launch the Minecraft server. We can do this by entering the following command. (You may need to change the version number depending on what version of Spigot you are using for example **spigot-1.10.2.jar**)

**Raspberry Pi 1**

```
sudo java -Xms256M -Xmx496M -jar /home/minecraft/spigot-1.10.2.jar nogui
```

**Raspberry Pi 2 & 3**

```
sudo java -Xms512M -Xmx1008M -jar /home/minecraft/spigot-1.10.2.jar nogui
```

This command launches the spigot server, utilizing a couple of parameters to make sure it runs as best as possible for the devices. **-Xms** defines the minimum amount of memory the server will allocate, and **-Xmx** defines the maximum amount of memory the server will allocate.

**14.** The server will start for a few seconds, then will request the user to agree to an EULA before being able to run the server. We can do this just by making an edit to a file by running the following command:

```
sudo nano eula.txt
```

## Setting up the Minecraft Server

**15.** In this file, change the following line:

```
eula=false
```

To

```
eula=true
```

**16.** Save the file and quit by pressing **Ctrl+X** and then pressing **Y**.

**17.** Now that we have done this, we can **run the commands in step 10 again** to launch up the server. It will take a few minutes to generate the map, give it about 3-5 minutes to generate it.

Don't worry it will only have to do this once. Upon any further restarts, it will only take about 30 seconds to load the map into memory.

**18.** The server should now successfully be running and accessible over your local network connection.

The Raspberry Pi Minecraft server will now run fine for the most part. but we may want to make some optimizations to the server to make it run even better.

## Connecting to the Raspberry Pi Minecraft Server

If you are on a local network, then it should be easy to connect to the Raspberry Pi Minecraft server. To test it out do the following:

Load up Minecraft on a PC within the same local network as the Raspberry Pi.

Go to multiplayer and then your server might pop up in the local list. If it doesn't just go to direct connect and enter the IP we got earlier on the Pi using the **command hostname -I**.

If you want to allow access to the Minecraft server via the internet, then you will need to setup port forwarding. You will need to port forward the **port 25565** (Unless you change it in the server properties) to the local IP address of your Raspberry Pi.

We have a basic guide on this in our Getting Started section of the book. It should give you an idea on how to port forward on your router.

## Spigot Plugins

Now to get the most out of our Raspberry Pi Minecraft server, we will want to install a plugin to help optimize the performance a bit. The more we can squeeze out of the Raspberry Pi, the better.

Firstly, let's install the NoSpawnChunks plugin, this will help save some of the Pi's very limited RAM. This plugin works by only loading in the world chunks that players are currently present in.

Type in the following commands to create the plugins folder and download the plugin:

```
sudo mkdir /home/minecraft/plugins
cd/home/minecraft/plugins
sudo wget http://dev.bukkit.org/media/files/586/974/NoSpawnChunks.jar
```

Another plugin that should help the performance of your Raspberry Pi and allow you to get more out of your Minecraft server is a plugin called **LaggRemover**.

It is a bit more advanced then **NoSpawnChunks** in that it actively tries to reduce the number of mobs and other entities to stop the server from being overloaded.

You can download the plugin into the plugins folder using the following couple of commands:

```
cd/home/minecraft/plugins
sudo wget https://addons-origin.cursecdn.com/files/910/762/LaggRemover-0.2.2.jar
```

There are other plugins out there that can help with performance, just simply use the wget command to download them to the plugins folder on the Raspberry Pi like we did above.

## Server Properties

Now you probably want to know how to edit the server properties. This file is vital for optimizing the server and customizing it to how you want the server to be.

If you want more information for each of the server settings you can find a good page on them at http://minecraft.gamepedia.com/Server.properties.

To begin editing  the server properties, enter the following command into the terminal:

```
sudo nano /home/minecraft/server.properties
```

Now in here, we will want to change a few settings to help optimize the performance of the server. The two settings we are going to configure are the maximum number of players, and the view distance. Reducing the view distance reduces the amount of RAM and processing power the server will use.

You change these and other settings to whatever you like however you want but keep in mind the Raspberry Pi can't handle too much processing.

Change the following two lines to match

```
view-distance=04
max-player=5
```

Now save and quit out of the file using **Ctrl+X** and then pressing **Y**.

## Configuring and Optimizing the Raspberry Pi Minecraft Server

If you intend on running your Raspberry Pi purely as a Minecraft server, it's well worth setting it up to starton boot.

Meaning if your Raspberry Pi goes down for whatever reason, your Minecraft server will start up when it comes back online.

**1.** Let's first start by making a simple startup script.

To do this lets first create the file by entering the following into the terminal:

```
sudo nano /home/minecraft-start.sh
```

**2.** In this file, we just need to add the following lines so that the Raspberry Pi Minecraft server will-startup correctly.

Make sure you use the right line based on the version of the Raspberry Pi your using.

**Note**: Make sure you swap out **spigot-1.10.2.jar** with the current version of spigot your using.

### Raspberry Pi 1

```
#!/bin/bash
cd /home/minecraft
sudo java -Xms256M -Xmx496M -jar /home/minecraft/spigot-1.10.2.jar nogui
```

### Raspberry Pi 2 & 3

```
#!/bin/bash
cd /home/minecraft
sudo java -Xms512M -Xmx1008M -jar /home/minecraft/spigot-1.10.2.jar nogui
```

**3.** Don't forget to save and quit out of the file by pressing typing **Ctrl+X** and then pressing **Y**.

**4.** Now to get the server to start up automatically on reboot then we need to add a line to the **rc.local** file.

Firstly let's open up the **rc.local** file using the following command:

```
sudo nano /etc/rc.local
```

### Find

```
exit 0
```

### Add Above

```
bash /home/minecraft-start.sh
```

**5.** Don't forget to save and quit out of the file by pressing typing **Ctrl+X** and then pressing **Y**.

**6.** To test this out properly, we should now reboot the Raspberry Pi by using the following command in the terminal.

Rebooting will make sure that it is correctly starting on boot.

```
sudo reboot
```

## Configuring and Optimizing the Raspberry Pi Minecraft Server

**7.** To check if the script is running either try to load the server in Minecraft or in the terminal of your Pi enter the following command:

```
ps aux | grep "minecraft-start.sh"
```

This command will return two results you only need to pay attention to the first.

If you need to kill the process get the process id from the above command, it is the first number after the username (In this case it should be **root**).

**8.** To kill the process just enter the following (Replacing **{pid}** with the process id):

```
sudo kill {pid}
```

# Minecraft Pocket Edition Server

## Project Description

In this tutorial, we will be showing you how to turn your Raspberry Pi into a Minecraft Pocket Edition server. Using your Raspberry Pi is a cheap way of providing you with an always-on, Pocket Edition capable server.

This software stops you from having to leave your phone or computer online if you want others to be able to have access to your world.

For this tutorial will be making use of the Nukkit software. We decided to utilize Nukkt since we found that it was one of the most stable Minecraft Pocket Edition servers currently available.

If you would like to know more about the Nukkit server software you can found out more about it at their website: https://nukkit.io

## Equipment

**Required**
- Raspberry Pi
- 4 GB (Micro) SD Card (8 GB+ Recommended)
- Network connection
- Raspberry Pi Power Supply

**Optional**
- USB Keyboard
- USB Mouse
- HDMI Cable
- Raspberry Pi Case

# Setting up the Raspberry Pi Minecraft PE Server

In this tutorial, we will work completely within the terminal. In fact, it is best to change the boot mode of the Raspberry Pi, so it boots directly into the terminal for the Minecraft server.

Minecraft Pocket Edition servers are hugely demanding on the Raspberry Pi's hardware, so every extra bit of resources made available to it, the more it can handle.

**1.** First, let's make sure everything on the Raspberry Pi is up to date by running the following two commands.

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

**2.** Now we will need to make a couple of changes in the raspberry pi config tool.

Let's bring the tool up by entering the following:

```
sudo raspi-config
```

**3.** First, let's go to **Advanced Options**->**Memory Split**, update this to **16**. (This will allow for more memory to be free for the server)

**4.** Secondly, now is a good time to change the bootup option, so it boots straight into the terminal.

Go to **Boot Options**->**Desktop/CLI**, Press **Enter** with **Console selected** to enable booting to console.

**5.** To make the most of the server, you will need to overclock your Raspberry Pi.

The following steps only work with the Raspberry Pi 1, and the Raspberry Pi 2 as the raspi-config tool does not currently support the Raspberry Pi 3.

While still in the raspi-config tool, Go to **Overclock**, here pick the "**High**" overclocking option.

This option will give the Raspberry Pi a good overclock, though make sure you have a decent, stable power supply. Most USB chargers will not cut it.

**6.** If you haven't activated SSH before, make sure you go to **Advanced Options**->**SSH** and select "**Enable**".

You will need this for easy access to your Raspberry Pi.

**7.** Now go to "**Finish**" and reboot your Raspberry Pi using the following command.

```
sudo reboot
```

**8.** Now is a great chance to grab the IP address of the Raspberry Pi.

You will need to write down the value from the next command. We will need to use this later.

In the terminal, enter the following command:

```
sudo hostname -I
```

## Setting up the Raspberry Pi Minecraft PE Server

**9.** With that all set up, we will now proceed with downloading and setting up the Nukkit server software. Before we get too far ahead of ourselves, lets first make a directory to keep this in, run the following commands to make the directory in the pi users root directory.

```
mkdir ~/nukkit
cd ~/nukkit
```

**10.** Finally, let's download Nukkit, we usually would have to compile the software. Thankfully some kind users have provided their CI (Continuous Integration) servers to the public.

Run the following command to download the latest build of Nukkit to your Raspberry Pi.

```
wget -O nukkit.jar https://pimylifeup.com/out/nukkit
```

**11.** With Nukkit now downloaded to our Raspberry Pi we can now run it, this will generate all the files we need to be able to configure the server further.

Running Nukkit is incredibly easy, we can run it with the following simple command.

```
sudo java -jar nukkit.jar
```

Upon running the server for the first time, you will be asked to choose a language, for this tutorial we will be using English. Therefore we typed **eng** into the console and pressed **enter**.

The server should immediately begin to boot up and the text below should be displayed to you.

```
[INFO] Preparing level "world"
[INFO] Starting GS4 status listener
[INFO] Setting query port to 19132
[INFO] Query is running on 0.0.0.0:19132
[INFO] Default game type: Survival Mode
[INFO] Done (3.109s)! For help, type "help" or "?"
```

While you can now begin to play on your server, you might want to first configure the server, so it is more to your liking. To do this first kill the process by pressing **Ctrl + C**.

**12.** Two configuration files come with Nukkit. One controls Nukkits options such as being able to change the number of chunks it loads in, the other controls actual Minecraft options.

You can read up on the **server.properties** file and what changes you can make to it by going to the following URL: http://minecraft.gamepedia.com/Server.properties

Run the following command to edit Nukkits configuration file.

```
sudo nano nukkit.yml
```

Or run the following command to edit the Minecraft Servers configuration.

```
sudo nano server.properties
```

Once you have finished with them, save by pressing **Ctrl + X** then pressing **Y** and then **Enter**.

Afterward, you can proceed to start up Nukkit again by typing in the following command:

```
sudo java -jar nukkit.jar
```

## Connecting to the Raspberry Pi Minecraft PE Server

If you are on a local network, then it will be easy to connect to the Raspberry Pi Minecraft PE server. To test out our recently setup Minecraft Pocket Edition Server we will need to do the following.

Load up Minecraft on a Mobile Phone or a Windows 10 PC (Minecraft Windows 10 Edition) that is located on the same local network as the Raspberry Pi.

Go to the Friends tab and then your server should pop up at the bottom of **LAN Games**. If it doesn't simply go to direct connect and enter the IP we got earlier on the Pi using the **command hostname -I**. We have highlighted the direct connect button in our screenshot below

If you want to allow access to the Minecraft Pocket Edition server via the internet, then you will need to setup port forwarding. You will need to port forward the **port 19132** (Unless you change it in the server properties) to the local IP address of your Raspberry Pi.

We have a basic guide on this in our Getting Started section of the book. It should give you an idea on how to port forward on your router.

# Terraria Server

## Project Description

In this project we will be walking you through all the steps to set up your very own Raspberry Pi Terraria Server, allowing you to set up a low cost and energy efficient Terraris Server.

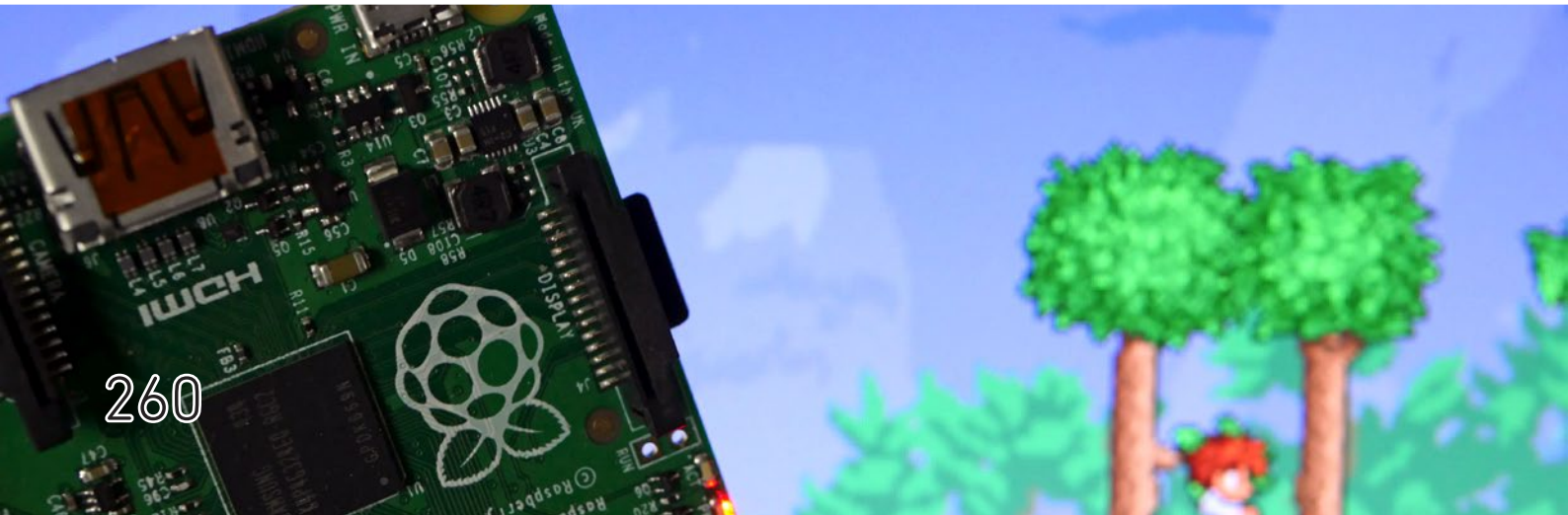
This project is made possible thanks to two open source initiatives, the first of those being the tShock Terraria server. This project implements everything you need to set up your own dedicated Terraris server and is the main piece of software we will be utilizing on the Raspberry Pi.

The second piece of software that we will be using is Mono, this is an open source implementation of the .net framework and allows us to run applications such as tShock on the Raspberry Pi.

## Equipment

**Required**
- Raspberry Pi
- 4 GB (Micro) SD Card (8 GB+ Recommended)
- Network connection

**Optional**
- USB Keyboard
- USB Mouse
- HDMI Cable
- Raspberry Pi Case



# Setting up Mono for the Terraria Server

In this tutorial, we will be setting up a Terraria Server on the Raspberry Pi. We achieve this by utilizing the Tshock server software.

While usually, this wouldn't run without Microsofts .net framework but thanks to the Mono project we can bypass that by using their open source implementation.

**1.** Now before we get started with setting up the Terraria server software on our Raspberry Pi, we will need to ensure our Raspberry Pi is entirely up to date by running the following command.

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

**2.** With the Raspberry Pi up to date, we will now need to setup and install the Mono software.

For those who don't know Mono is open source implementation of Microsoft's .NET Framework which is what the TShock server software relies on. Thanks to Mono we can get it running on the Raspbian operating system.

Now to install Mono, we must first add the Ubuntu key server, without this key server we cant grab Mono from the Mono Project repository.

To add this keyserver, we need to utilize the following command. Make sure you write this all on one line

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80
--recv-keys 3FA7E0328081BFF6A14DA29AA6A19B38D3D831EF
```

**3.** With the key server now added to our package manager, we need to grab the Mono Projects repository.

We can do this by running the following command on our Raspberry Pi, make sure you use the right command for your operating system.

This command will add the package to a particular list that will be automatically loaded in by the package manager.

## Raspbian Wheezy

```
echo "deb http://download.mono-project.com/repo/debian wheezy main" |
sudo tee /etc/apt/sources.list.d/mono-xamarin.list
```

## Raspbian Jessie

```
echo "deb http://download.mono-project.com/repo/debian jessie main" |
sudo tee /etc/apt/sources.list.d/mono-xamarin.list
```

## Raspbian Stretch

```
echo "deb http://download.mono-project.com/repo/debian streth main" |
sudo tee /etc/apt/sources.list.d/mono-xamarin.list
```

## Setting up Mono for the Terraria Server

**4.** Now that we have added this additional repository to our sources list we need to run another update and upgrade to ensure the repository is loaded into our package list.

Run the following two commands on the Raspberry Pi to do this.

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

**5.** Once the update has completed, we can finally install the Mono software by running the following command in our Raspberry Pi's terminal.

```
sudo apt-get install mono-complete
```

## Setting up a Raspberry Pi Terraria Server

**1.** Now that we have Mono installed onto our Raspberry Pi we can proceed with actually setting up the Terraria server. For this, we will be utilizing the TShock server software.

First off we will make a folder to keep the Terraria software in and then change directory into it.

We can just do that by running the following two commands.

```
mkdir ~/tshock
cd ~/tshock
```

**2.** Now that we are in our newly created folder let's download the Terraria server software to our Raspberry Pi by running the following command.

We can download the software by running the following command on our Raspberry Pi.

```
wget https://github.com/Pryaxis/TShock/releases/download/v4.3.25/tshock_4.3.25.zip
```

**3.** We can now extract the zip file by running the following command on our Raspberry Pi.

While unzip comes by default on the base Raspbian installation you may have to install it on slimmer installations.

```
unzip tshock_4.3.24.zip
```

**4.** With the TShock server extracted we can finally run the server using Mono.

We need to now run the following command on the Raspberry Pi while we are in our tshock folder.

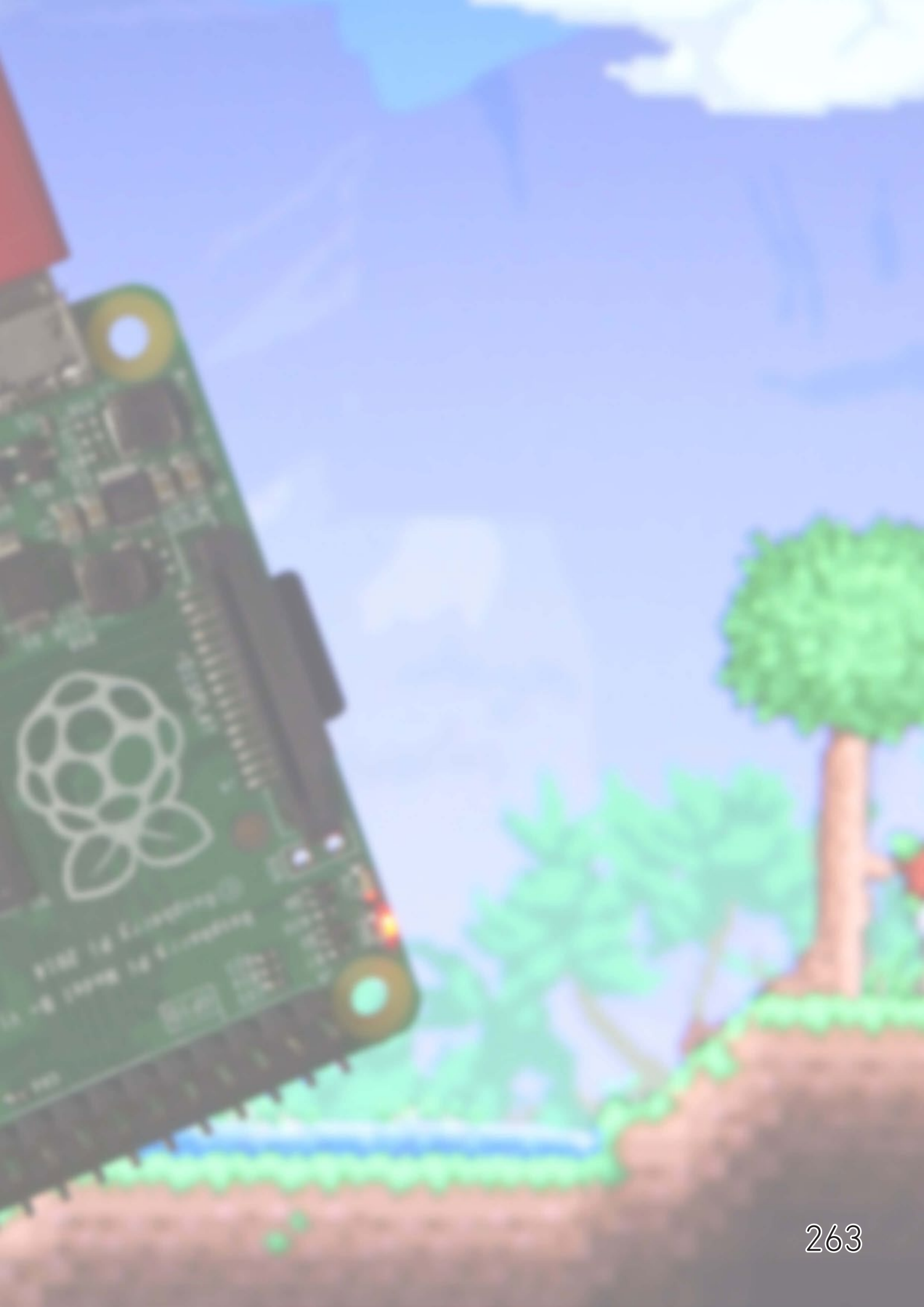
```
mono TerrariaServer.exe
```

**5.** You will be taken through several steps to set up your Terraria world on the Raspberry Pi.

Start with a small world to stop the Pi crashing during the generation process.

After the generation has completed, simply enter the number next to your world and press enter.

You will again be asked for a few extra details, it is usually safe to just press Enter to go past them.



# Installing Apache

## Project Description

A Raspberry Pi web server makes for a great lightweight Web Server. It can handle a small amount of traffic and lightweight scripts making it fantastic for a home server or an area you can learn web programming languages such as PHP.

It can even handle WordPress, so if you want to launch your blog/website with a proper back-end, you can quickly do it by just utilizing your Raspberry Pi.

Just don't expect it to be able to handle any severe amounts of traffic due to the Pi's weak processor.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

**Optional**
- USB Keyboard
- USB Mouse
- Raspberry Pi Case

# Installing Apache on the Raspberry Pi

In this tutorial, I make use of the standard operating system for the Raspberry Pi known as Raspbian. You can do this tutorial over SSH or on the terminal on the actual Raspberry Pi.

**1.** First, we will need to install the Apache2 package.

For those who don't know Apache is what serves the HTML files from a computer to the web browser, the clear majority of the websites on the internet run off Apache.

To install apache2, enter the following command into the terminal:

```
sudo apt-get install apache2 -y
```

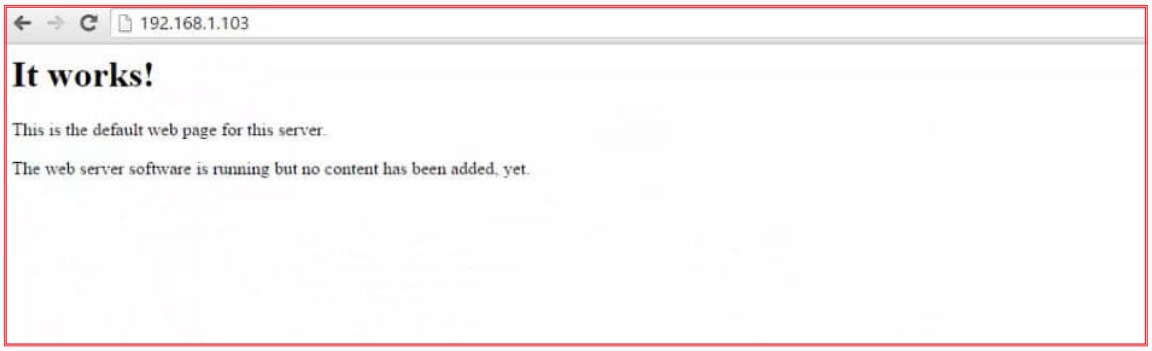
**2.** Now that's all you need to do to get an extremely basic web server up and running. However, we won't be stopping here.

We will also show you how to setup PHP and SQL. To check that it is working you should be able to enter the Raspberry Pi's IP address into a web browser, and it will display a basic webpage.

If you don't know the IP, you can enter the following command into terminal to retrieve it:

```
hostname -I
```

**3.** In a web browser, got to the IP Address you got from the previous step, it should connect and load a page like the one below:




**4.** You can make changes to this webpage by just using the following command on the terminal:

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

**5.** You can now add as many new HTML files as you want in the **/var/www/html/** folder and they will be viewable from the browser.

Apache is a very basic server and is great if you just want to learn HTML, JS, or CSS but if you are after PHP (Used for dynamic web pages) then read on.

## Setting up PHP for Apache on the Raspberry Pi

PHP currently comes in two actively maintained versions, one being PHP5 the other being PHP7. If you are interested in programming for the latest version of PHP, you will need to go through a few hoops as it's not currently available on the default package list.

We will show you both ways of installing PHP, if you are interested in the easy way, and don't have a need to use the latest version of PHP then sit on this page, otherwise if you want PHP7 flip over to the next page.

## Setting up PHP5 for Apache

**1.** Installing PHP5 is rather simple, it is available in the package installer and just requires a single command to have it all set up:

```
sudo apt-get install php5 -y
```

**2.** Now that PHP is installed we can test it to make sure it's actually working.

We can do this by creating a PHP file within **/var/www/html/**, this will allow it to be processed and displayed when you open it on a web browser.

For our example, we will be creating a test file called **example.php**, we can create this file by issuing the following command:

```
sudo nano /var/www/html/example.php
```

**3.** In this file we need to add the following lines:

```
<?php
echo "Today's date is ".date('Y-m-d H:i:s');
?>
```

What these lines of code do is rather simple. The **<?php ?>** tags tell the server that the code within it is PHP and not HTML, so it needs to process the content within it dynamically.

**echo** tells php it needs to display everything within that line of code.

Text between the quotes **"Example"** will be displayed as is.

The **date** function outputs a formatted version of the current date and time.

The full stop **.** between the quotes and the **date** function will join the output from the date function with the text included in the quote.

Now save the file by pressing **Ctrl + X** then pressing **Y** and hitting **Enter**.

**4.** Now in your web browser, go to **http://192.168.1.103/example.php**. Make sure you replace **192.168.1.103** with your IP Address.

This should display something like the following

```
Today's date is 2016-11-13 21:30:45
```

## Setting up PHP7 for Apache

If you are following these steps on Raspbian Stretch, you can skip to **Step 5**.

Otherwise, if you are running Raspbian Jessie and earlier you can follow the following four steps to download it from a separate package.

**1.** To start off, we need to edit the sources.list file, we can do this with the following command:

```
sudo nano /etc/apt/sources.list
```

**2.** With this file, we need to add the following couple of lines to the bottom of the sources list.

These lines will tell the package manager where to search for packages.

```
deb http://repozytorium.mati75.eu/raspbian jessie-backports main contrib non-free
#deb-src http://repozytorium.mati75.eu/raspbian jessie-backports main contrib non-free
```

Now save the file by pressing **Ctrl + X** then pressing **Y** and hitting **Enter**.

**3.** Since this a 3rd party repository, we also need to add a couple of certificates to get it working correctly with apt-get.

Enter the following two commands into the terminal:

```
sudo gpg --keyserver pgpkeys.mit.edu --recv-key CCD91D6111A06851
sudo gpg --armor --export CCD91D6111A06851 | sudo apt-key add -
```

**4.** Now with the repository setup and the certificates added to apt-get, we can finally install PHP7.

Now thanks to the repository being set up, we can do this in one easy command:

```
sudo apt-get install php7.0 -y
```

**6.** Now that PHP is installed we can test it to make sure it's working.

We can do this by creating a PHP file within **/var/www/**, this will allow it to be processed and displayed when you open it on a web browser.

For our example, we will be creating a test file called **example.php**, we can create this file by issuing the following command:wing command:

```
sudo nano /var/www/html/example.php
```

**7.** In this file we need to add the following lines:

```
<?php
echo "Today's date is ".date('Y-m-d H:i:s');
?>
```

Now save the file by pressing **Ctrl + X** then pressing **Y** and hitting **Enter**.

**8.** Now in your web browser, go to **http://192.168.1.103/example.php**. Make sure you replace **192.168.1.103** with your IP Address.

Going to the URL should display something like the following

```
Today's date is 2016-11-13 21:30:45
```

# Setting up NGINX

## Project Description

In this tutorial, we show you how to setup the NGINX Web Server on your Raspberry Pi. It is a better alternative to Apache for the Raspberry Pi. The reason for this is that NGINX has a lower memory usage and a lower CPU usage.

We will also show you how to configure NGINX to utilize PHP-FPM, this requires a slight bit of extra configuration unlike PHP for Apache.

While NGINX has a bit more leg room then Apache, don't expect it to run any serious amount of traffic.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

**Optional**
- USB Keyboard
- USB Mouse
- Raspberry Pi Case

# Setting up NGINX on the Raspberry Pi

In this tutorial, I make use of the standard operating system for the Raspberry Pi known as Raspbian. You can do this tutorial over SSH or on the terminal on the actual Raspberry Pi.

**1.** Before we get started with setting up the NGINX web server on our Raspberry Pi we must first make sure our Raspberry Pi is up to date by running the following two commands:

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

**2.** We should also run the following command to uninstall Apache2. Failing to do so will cause the installation to fail. You can skip this step if you are certain Apache2 isn't already installed.

```
sudo apt-get remove apache2
```

**3.** With our packages now up to date we can go ahead and install the NGINX web server, run the following command to install it:

```
sudo apt-get install
```

**4.** Now with NGINX installed, we can now start it up, just type the following command into terminal to start up the web server:

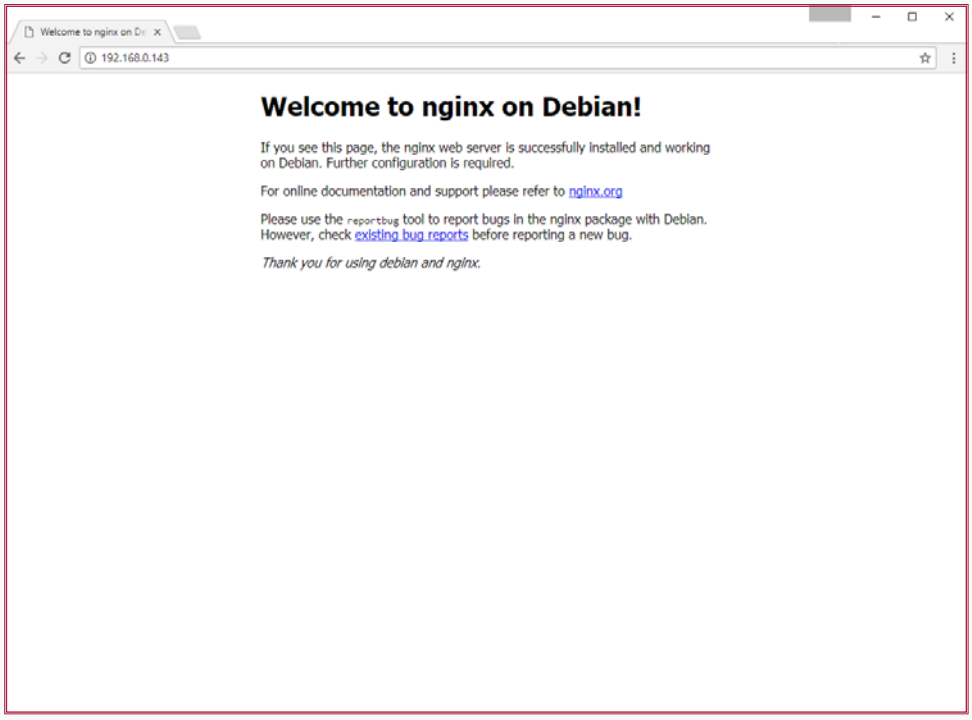
```
sudo /etc/init.d/nginx start
```

**5.** With the web server started up let's grab our local IP address so we can test from a separate web browser if it is working correctly

```
hostname -I
```

**6.** With our local IP address handy let's open the address within your favorite web browser. Just go to the local IP Address that you grabbed using **hostname -I.** In my case, I would go to **http://192.168.0.143**.

You should see something like below, don't be worried if this shows an Apache page instead. This behavior sometimes happens as NGINX doesn't overwrite the Apache default index page.



## Configuring NGINX for PHP

**1.** Unlike Apache, NGINX won't be automatically set up for use with PHP. We instead must make changes to its configuration files to get it to load in. We will also have to utilize PHP-FPM and not regular PHP.

**2.** Before we get started with configuring PHP for NGINX, we need to install it. You have the choice between PHP 5, or PHP 7. We reccommend PHP 7 as PHP 5 is getting close to its end of life.

### PHP 5

**2a.** PHP 5 is readily available on all versions of Raspbian, so we can just just type in the following command to install the PHP 5 FPM package.

```
sudo apt-get install php5-fpm
```

### PHP 7

**2b.** PHP 7 is a bit trickier to install then PHP 5 for those are on Raspbian Jessie and earlier. Those who have updated to Raspbian Stretch will have it easy.

#### Raspbian Stretch and later

```
sudo apt-get install php7.0-fpm
```

#### Raspbian Jessie and earlier

For Raspbian Jessie and earlier we need to grab the php7.0 packages from the Stretch branch, to be able to do this we will be adding Stretch as a possible repository, but setting it up so it will only grab a package from it when we explicitly specify it.

**a.** Begin by editing the apt sources list file by running the following command.

```
sudo nano /etc/apt/sources.list
```

**b.** Add the following line to the bottom of this file.

```
deb http://mirrordirector.raspbian.org/raspbian/ stretch main contrib non-free rpi
```

Once done we can save & exist by pressing **CTRL +X** and then pressing **y**.

**c.** Now we need to update our preferences file to ensure Jessie will be selected over Stretch. Run the following command to begin editing the file.

```
sudo nano /etc/apt/preferences
```

**d.** To this, add the following lines, this sets the packages to effect and the priority.

```
Package: *
Pin: release n=jessie
Pin-Priority: 600
```

Once done we can save & exist by pressing **CTRL +X** and then pressing **y**.

**e.** With the preferences changes and the new repository added to our sources list, we need to run an update to grab the new repository.

```
sudo apt-get update
```

**f.** Finally we can install PHP7.0-FPM by using the following command.

```
sudo apt-get install -t stretch php7.0-fpm
```

## Configuring NGINX for PHP

**3.** With PHP-FPM now installed we can make the required modifications to the default NGINX configuration file.

To begin editing this file run the following command:

```
sudo nano /etc/nginx/sites-enabled/default
```

**4.** Within this file, find and replace/add the following lines.

### Find

```
index index.html index.htm;
```

### Replace With

```
index index.php index.html index.htm;
```

Here we need to add **index.php** to the index line.

This addition tells NGINX to recognize the index.php file as a possible index, adding it first in the list means it will be selected over an index.html file.

### Find

```
#location ~ \.php$ {
    #       include snippets/fastcgi-php.conf;
    #
    #       # With php5-cgi alone:
    #       fastcgi_pass 127.0.0.1:9000;
    #       # With php5-fpm:
    #       fastcgi_pass unix:/var/run/php5-fpm.sock;
    #}
```

### Replace With for PHP 5

```
location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        # With php5-fpm:
        fastcgi_pass unix:/var/run/php5-fpm.sock;
    }
```

### Replace With for PHP 7

```
location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php7.0-fpm.sock;
    }
```

This basically sets up NGINX to process .php files by passing them through to PHP-FPM.
Once done we can save & exist by pressing **CTRL +X** and then pressing **y**.

**5.** Now, lets tell NGINX to reload its configuration by running the following command.

```
sudo /etc/init.d/nginx reload
```

**6.** Finally, let's test our PHP setup by writing a straightforward index.php file in our **/var/www/html** directory.

Run the following command to create and begin editing our index.php file.

```
sudo nano /var/www/html/index.php
```

## Configuring NGINX for PHP

**7.** To this file, add the following line of code.

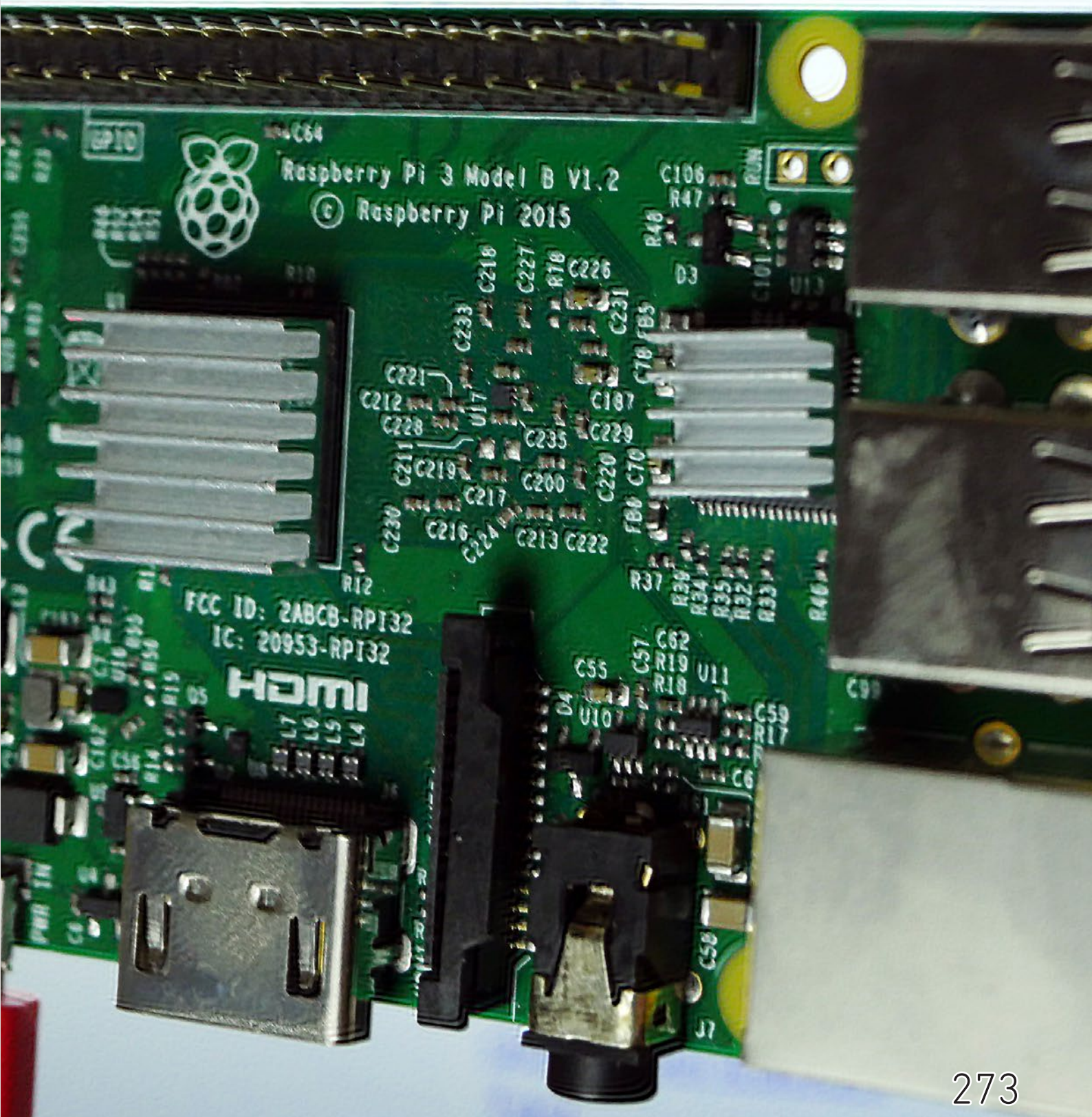
```
<?php phpinfo(); ?>
```

Once done we can save & exist by pressing **CTRL +X** and then pressing **y**.

**8.** Now like earlier in the tutorial you can go to your Raspberry Pi's IP address.

This time however you should see a page showing all the information about your version of PHP and what modules are currently active. It is a good indication that your PHP installation is up and running correctly.

Hopefully, by now you have a fully working NGINX web server with PHP working correctly.


Raspberry Pi 3 Model B V1.2
© Raspberry Pi 2015
FCC ID: 2ABCB-RPI32
IC: 20953-RPI32
HDMI

# Setting up Caddy

## Project Description

Caddy is a different sort of web server, while it is not nearly as performance friendly as NGINX it is packed with user-friendly features and is designed to utilize modern web technologies.

Caddy utilizes HTTP2 from the get-go and automatically provisions SSL certificates for any domain name your setup through Caddy. It grabs these certificates from Lets Encrypt.

Caddy's Caddyfile system is relatively easy to use once you get the hang of it, and luckily, they have solid documentation on how to do everything with it.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

**Optional**
- USB Keyboard
- USB Mouse
- Raspberry Pi Case

## Setting up Caddy on the Raspberry Pi

**1.** One thing we must do before we get to installing and setting up Caddy is to make sure our Raspberry Pi is entirely up to date.

Begin by running the following two commands on your Raspberry Pi to bring it completely up to date.

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

**2.** Before we get started with installing Caddy, we will first have to remove apache2 that is installed by default on Raspbian.

If you are certain you don't already have apache2, you can skip running this command.

```
sudo apt-get remove apache2
```

**3.** Now we can run the following command in terminal to download and install Caddy.

This command will grab and run the bash script that resides at https://getcaddy.com. While curl normally shouldn't be used to run a script directly off a website, Caddy is a known and trusted provider.

This script detects what OS you are running, downloads the correct version of Caddy and sets it up on your device so it can run like a typical application.

```
curl https://getcaddy.com | bash -s personal
```

**4.** Once Caddy has finished being downloaded and installed, we can begin the more complex task of setting it up.

Let's first make a folder where we will keep our caddy configuration file. We will be making this new folder within the **/etc/** folder that is available by default on your Raspbian installation.

To create this folder, simply just run the following command on your Raspberry Pi.

```
sudo mkdir /etc/caddy
```

**5.** While we are making the folders that we require, we will also make one to keep our first website in.

For our tutorial, we will be just naming this folder **localhost**, however, if you want to utilize this for a domain name we recommend replacing the folder name **localhost** with your websites domain name.

For example, for our website, we would name this new folder **pimylifeup.com** instead of **localhost**.

We can create this new folder by running the following command on the Raspberry Pi.

```
sudo mkdir -p /var/www/localhost
```

## Setting up Caddy on the Raspberry Pi

**6.** With both our folders created, we can now get onto the interesting part, and that is making our caddyfile.

This file is where we will keep all our configurations for caddy.

We will be storing this file in the **/etc/caddy/** folder that we made in the previous couple of steps.

To begin creating our new Caddyfile we need to run the following command within terminal on our Raspberry Pi.

```
sudo nano /etc/caddy/Caddyfile
```

**7.** In this file write the lines shown in the box below..

Basically what these lines of text do is define the address to listen on, where the files we want to serve are and that we want to utilize GZIP compression on Caddys HTML output.

Remember to replace localhost with the name of the folder you created earlier.

Also if you are using a domain name replace **:80 localhost:80** with it, so the line would end up like **pimylifeup.com {** instead.

```
:80 localhost:80 {
   root /var/www/localhost
   gzip
}
```

Once done we can save & exist by pressing **CTRL +X** and then pressing **y**.

**8.** With our Caddyfile now setup, we will move onto making a sample HTML file just to ensure its still working.

We will be back to work on the Caddyfile some more when we setup PHP. For now, let's make our test index file by running the following command.

Again, replace **localhost** with whatever you set earlier.

```
sudo nano /var/www/localhost/index.html
```

**9.** In this file just write anything, as all we need it for is to make sure it is being shown correctly.

For our tutorial, we will only be writing this small bit of HTML.

```
<h1>This is a test for pimylifeup.com</h1>
```

Once done we can save & exist by pressing **CTRL +X** and then pressing **y**.

**10.** To start up Caddy, all we need to do is type in the following commands into the terminal.

```
cd /etc/caddy
caddy
```

**11.** Now browse to your Raspberry Pi's local IP address, or go to localhost on your Raspberry Pi itself.

If you followed our tutorial, you should see the text "**This is a test for pimylifeup.com**".

**12.** Once you have confirmed that it is working, press **Ctrl + C**, so that we can now move onto setting up PHP for Caddy.

To do this, we will be making use of PHP-FPM. You will have the choice between the PHP 5 or PHP 7, we recommend utilizing PHP 7 as it is more modern, faster, and more memory efficient.

### PHP 5

**13a.** Luckily for PHP 5, we don't have to do anything fancy to get it working on all versions of Raspbian, we can just enter the following command into terminal to install it.

```
sudo apt-get install php5-fpm
```

### PHP 7

**13b.** Unfortunately, PHP 7 is not included on Raspbian Jessie and earlier, so we have to do some trickery to download the package from the newer repositories.

Follow the following steps to install PHP 7.0

### Raspbian Stretch and later

```
sudo apt-get install php7.0-fpm
```

### Raspbian Jessie and earlier

Sadly, as we mentioned before, Raspbian Jessie and earlier does not have the PHP 7.0 packages, luckily we can do some clever trickery so we can specifically grab the new versions of PHP from the Raspbian Stretch repository.

**a.** Begin by editing the apt sources list file by running the following command.

```
sudo nano /etc/apt/sources.list
```

**b.** Add the following line to the bottom of this file.

```
deb http://mirrordirector.raspbian.org/raspbian/ stretch main contrib non-free rpi
```

Once done we can save & exist by pressing **CTRL +X** and then press **Y**.

**c.** Now let's modify our preferences file to make sure that Jessie packages will be chosen over Stretch. Run the following command to begin editing the file.

```
sudo nano /etc/apt/preferences
```

**d.** To this, add the following lines, this sets the packages to effect and the priority.

```
Package: *
Pin: release n=jessie
Pin-Priority: 600
```

Once done we can save & exist by pressing **CTRL +X** and then press **Y**.

**e.** With the preferences changes and the new repository added to our sources list, we need to run an apt-get update command to start grabbing packages from the new source.

```
sudo apt-get update
```

**f.** Finally, we can install PHP7.0-FPM by using the following command.

```
sudo apt-get install -t stretch php7.0-fpm
```

**14.** With PHP-FPM now installed, we can get on with configuring Caddy to use it, to do this we will have to crack open our Caddyfile again.

Run the following command to begin editing it.

```
sudo nano /etc/caddy/Caddyfile
```

**15.** To this file we want to add the following lines.

These lines will tell Caddy to utilize PHP-FPM as its fast CGI handler, tell it to read index.php files as the index, and also we add a few lines that rewrite URL requests to work fine with PHP.

Replace **/var/run/php/php7.0-fpm.sock** with **/var/run/php5-fpm.sock**

### Find

```
root /var/www/localhost
gzip
```

### Add Below

```
fastcgi / /var/run/php/php7.0-fpm.sock {
    index index.php
}
rewrite {
    r .*
    ext /
    to /index.php?{query}
}
```

Once done we can save & exist by pressing **CTRL +X** and then pressing **y**.

**16.** With that now saved, we need to change a few files permissions so that PHP-FPM can interpret our index.php, we also need to do this to setup our service file.

Run the following commands in terminal to give the www-data control over our web related files and assign relevant read/write/execute permissions.

```
sudo chown -R root:www-data /etc/caddy
sudo chown www-data:www-data /etc/caddy/Caddyfile
sudo chmod 444 /etc/caddy/Caddyfile
sudo chown -R www-data:www-data /var/www
sudo chmod -R 555 /var/www
```

**17.** Now that we have corrected the read/write permissions and file ownership for our Caddy files we can finally create our Caddy service as well.

Run the following commands to download and setup the Caddy service.

```
sudo wget https://raw.githubusercontent.com/mholt/caddy/master/dist/init/linux-systemd/caddy.service
sudo cp caddy.service /etc/systemd/system/
sudo chown root:root /etc/systemd/system/caddy.service
sudo chmod 644 /etc/systemd/system/caddy.service
sudo systemctl daemon-reload
```

## Setting up Caddy on the Raspberry Pi

**18.** Now before we go starting up caddy using our new service, let's make a .php file for it actually to run.

Remember to change out localhost if you are utilizing an actual domain name.

```
sudo nano /var/www/localhost/index.php
```

**19.** Now in this file, add the following line.

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

Once done we can save & exist by pressing **CTRL +X** and then pressing **y**.

**20.** Now we can finally start up Caddy using our new service and set it to start on boot by running the following two commands.

If you don't want it starting on boot, don't run the second command.

```
sudo systemctl start caddy.sevice
sudo systemctl enable caddy.service
```

**21.** You should now have a fully working HTTP server that is utilizing Caddy.

If you want to modify further your Caddy configuration you can check out their official documentation by going to their website here: https://caddyserver.com/docs

# Setting up Lighttpd

## Project Description

In this project, we will be exploring how to set up a Lighttpd Web Server on your Raspberry Pi.

Lighttpd like NGINX is a fast and memory efficient alternative to Apache that is designed to have a relatively low memory footprint and a small CPU load, both of which are perfect qualities for the Raspberry Pi.

One of the most significant users of Lighttpd is the Wikimedia Foundation.

We will be exploring how to set up a basic Lighttpd server running off the default configuration, while also exploring on how to setup Lighttpd to work with PHP-FPM, as this does not work straight off the bat with the Raspbian version of Lighttpd.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

**Optional**
- USB Keyboard
- USB Mouse
- Raspberry Pi Case

# Setting up Lighttpd on the Raspberry Pi

**1.** Now before we go to install Lighttpd, we need to update the Raspberry Pi to ensure we are running the latest software.

Run the following two commands to update your Raspberry Pi.

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

**2.** Now with the Pi up to date, we will need to remove Apache 2, since Raspbian includes it by default on most installations.

If you are sure, you don't have Apache 2 installed then skip this step.

```
sudo apt-get remove apache2
```

**3.** With the Raspberry Pi now up to date and Apache removed, we can get on with installing Lighttpd.

Luckily for us, Lighttpd is provided in the package repository. Install it by running the following command.

```
sudo apt-get install lighttpd
```

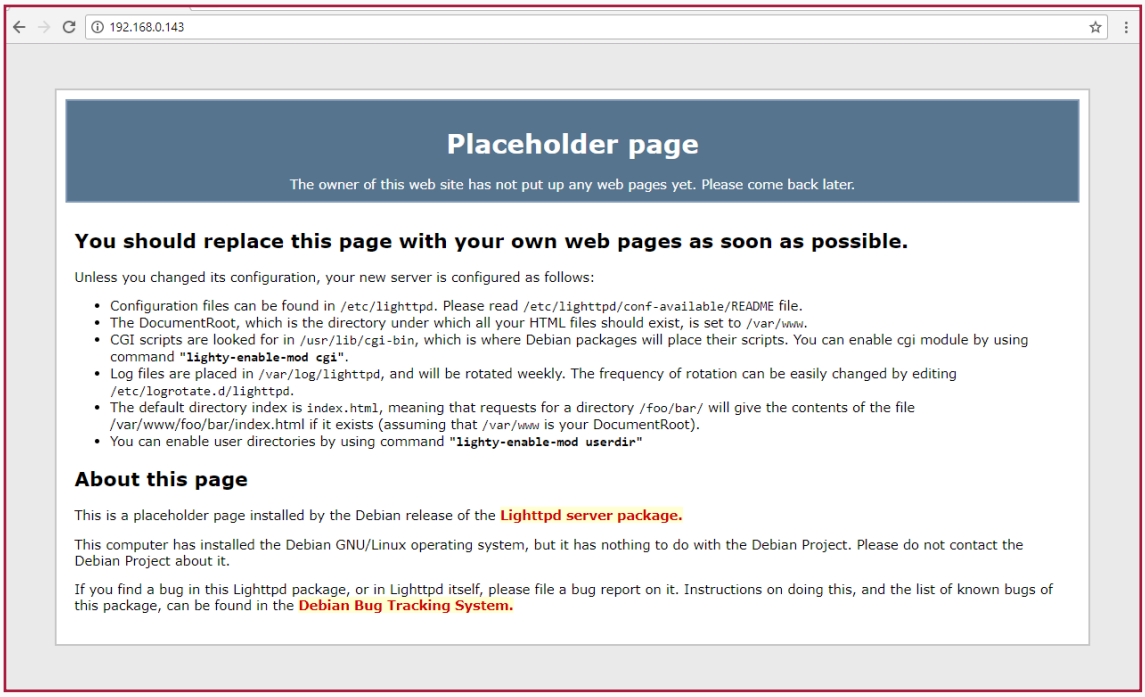
**4.** Once Lighttpd has been installed, we can now check if is running, just browse to your Raspberry Pi's local IP Address from any web browser.

If you don't know your Raspberry Pi's local IP address, type in the following command to get it.

```
hostname -I
```

On a successful connection, you should see a screen somewhat like what is displayed below.

If you were one of those people that had to uninstall Apache2 first, then you may be greeted with an Apache default page, don't be alarmed, as this still confirms that Lighttpd is up and running.



**5.** If you would like to make edits to this placeholder page, you can just run the following command. All the files that are loaded are kept in **/var/www/html**.

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

Over the next page we will explore setting up Lighttpd to work with PHP 5 or PHP 7.

**6.** For Lighttpd, we will have to make use of PHP-FPM. We will show you how to install PHP 5 or PHP 7.

We recommend PHP 7 as it's more modern and provides performance and memory improvements.

## PHP 5

**7a.** Luckily with PHP 5, we don't have to do anything fancy to get it working on all versions of Raspbian, we can just enter the following command into terminal to install it.

```
sudo apt-get install php5-fpm
```

## PHP 7

**7b.** If you are using Raspbian Stretch intstalling on PHP 7, however for Jessie and earlier we will have to use some tricks.

### Raspbian Stretch and later

```
sudo apt-get install php7.0-fpm
```

### Raspbian Jessie and earlier

Now to get PHP 7.0 to install on Raspbian Jessie and earlier we must modify the sources list to add the newer Stretch repository.

**a.** Let's edit the apt sources list by entering the following command into the terminal.

```
sudo nano /etc/apt/sources.list
```

**b.** Add the following line to the bottom of this file.

```
deb http://mirrordirector.raspbian.org/raspbian/ stretch main contrib non-free rpi
```

Once done we can save & exist by pressing **CTRL +X** and then pressing **y**.

**c.** Now we will need to edit the preferences file to ensure that Jessie packages are still chosen over Stretch. Doing this ensures we don't end up with a weird combination of packages.

```
sudo nano /etc/apt/preferences
```

**d.** To set the priority enter the following lines into this file.

```
Package: *
Pin: release n=jessie
Pin-Priority: 600
```

Once done we can save & exist by pressing **CTRL +X** and then pressing **y**.

**e.** Now that we have added our new repository and have set our preferences we need to run an update so that the Raspbian OS will read in all the new packages.

```
sudo apt-get update
```

**f.** With that all done we can finally install PHP7.0-FPM by utilising the following command.

```
sudo apt-get install -t stretch php7.0-fpm
```

**8.** Now that we have installed PHP-FPM there are a few things we will have to do.

To start off we will need to first run the following two commands, this basically creates two symlinks so that Lighttpd will start reading the configuration files that set it up for PHP.

**9.** Now before we can go reloading Lighttpd, we will have to make changes to its configuration. By default, it is configured to work with PHP-CGI and not PHP-FPM, luckily it is easy to make these changes.

Let's begin by running the following command in the terminal to start editing the file.

```
sudo nano /etc/lighttpd/conf-available/15-fastcgi-php.conf
```

**10.** Change the file so that it looks something like what we have below. This change makes it, so it just utilizes the PHP-FPM socket file.

Besides the definition of the socket to use and the broken-scriptfilename setting, we discard the other options.

If you are using PHP 5 and not PHP 7, replace **/var/run/php/php7.0-fpm.sock** with **/var/run/php5-fpm.sock**.

```
# -*- depends: fastcgi -*-
# /usr/share/doc/lighttpd/fastcgi.txt.gz

fastcgi.server += ( ".php" =>
        ((
                "socket" => "/var/run/php/php7.0-fpm.sock",
                "broken-scriptfilename" => "enable"
        ))
)
```

Once done we can save & exist by pressing **CTRL +X** and then pressing **y**.

**11.** Finally, with that all done, we can now force Lighttpd to reload by typing the following command into the terminal.

```
sudo service lighttpd force-reload
```

**12.** Now let's make a PHP file to test whether our new setup is working, run the following command to begin writing a new PHP file in our **/var/www/html** folder.

```
sudo nano /var/www/html/index.php
```

**13.** Within this file, type in the following lines, we use phpinfo() as it is one of the best indicators that PHP is working correctly.

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

Once done we can save & exist by pressing **CTRL +X** and then pressing **y**.

**14.** Finally, we can now go to our Raspberry Pi's local IP address in a web browser like we did earlier. We should see a screen showcasing all our PHP information. If this shows, you have successfully setup a Lighthttpd web server on your Raspberry Pi that can also utilize PHP.

# Setting up MYSQL

## Project Description

If you are unfamiliar with SQL utilizing it on your Raspberry Pi is a fantastic way of learning how to use it.

MySQL is a relational database management system and allows you store & maintain data. It is one of the world's most popular relational database system and is commonly included in most LAMP stacks.

Raspbian Stretch no longer uses MYSQL and instead uses MariaDB, both are implementations of the SQL Relational Database management system with MariaDB being the now go to for

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

**Optional**
- USB Keyboard
- USB Mouse
- Raspberry Pi Case

## Setting up MYSQL

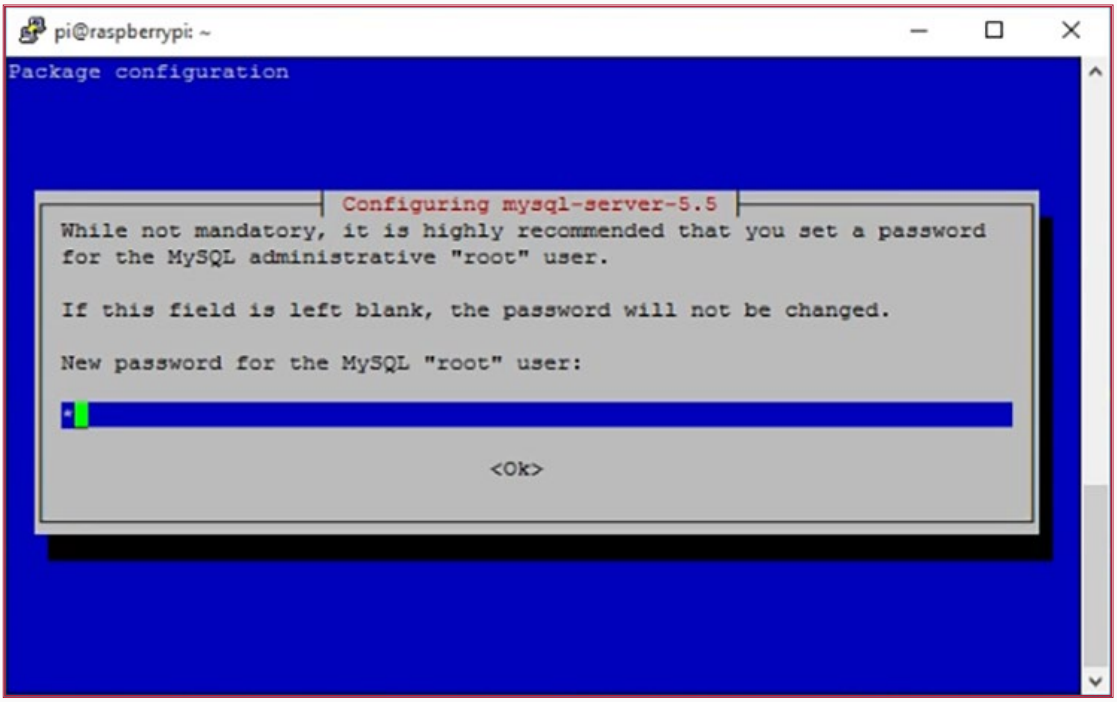
**1.** To start off, we need to install the MYSQL Server.

We can do this by running a simple command on the Raspberry Pi, just write the following line into terminal and press enter.

```
sudo apt-get install mysql-server
```

**2.** You will be prompted to enter a password for the root user.

Make sure you write this down as we will need to use this to access the MYSQL server and connect software such as Wordpress or PHPMyAdmin to it.




**3.** If you are not prompted to enter a password, you will need to enter the following command.

Answer all the questions so it will be set up to match your security requirements.

```
sudo mysql_secure_installation
```

**3.** Now if you want to access the MYSQL server and start making changes to your databases you can just enter the following command:

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

**4.** You will now be prompted to enter the password that we just created in **step 2/3**.

**Note:** Like most Linux password inputs, the text will not show up as you type.

**5.** You can now enter MYSQL commands to create, alter, delete databases, through this interface you can also create or delete users, and assign them the rights to manage any database.

**6.** There are two different ways you can quit out of the MYSQL command line, the first of those is to type in **quit**, the other way of quitting out of the MYSQL command line is to press **Ctrl + D**.

## Setting up MYSQL

**7.** You should now have MYSQL successfully set up. It should also be set up to work fine with PHP as well thanks to installing the PHP connector.

If you would prefer to use a GUI interface to manage MYSQL rather than the command line, then you can continue to go to our Raspberry Pi PHPMyAdmin tutorial below.

On an extra note, if you want to be able to interact with MYSQL in Python, you will need to install the Python bindings as well. You can do this by entering the following command.

```
sudo apt-get install python-mysqldb
```

14291
14292

# Installing PHPMyAdmin

## Project Description

For anyone who doesn't know PhpMyAdmin is a free tool that has been designed to allow for easy administration of MYSQL. It is hugely popular and used in a lot of web server environments throughout the world.

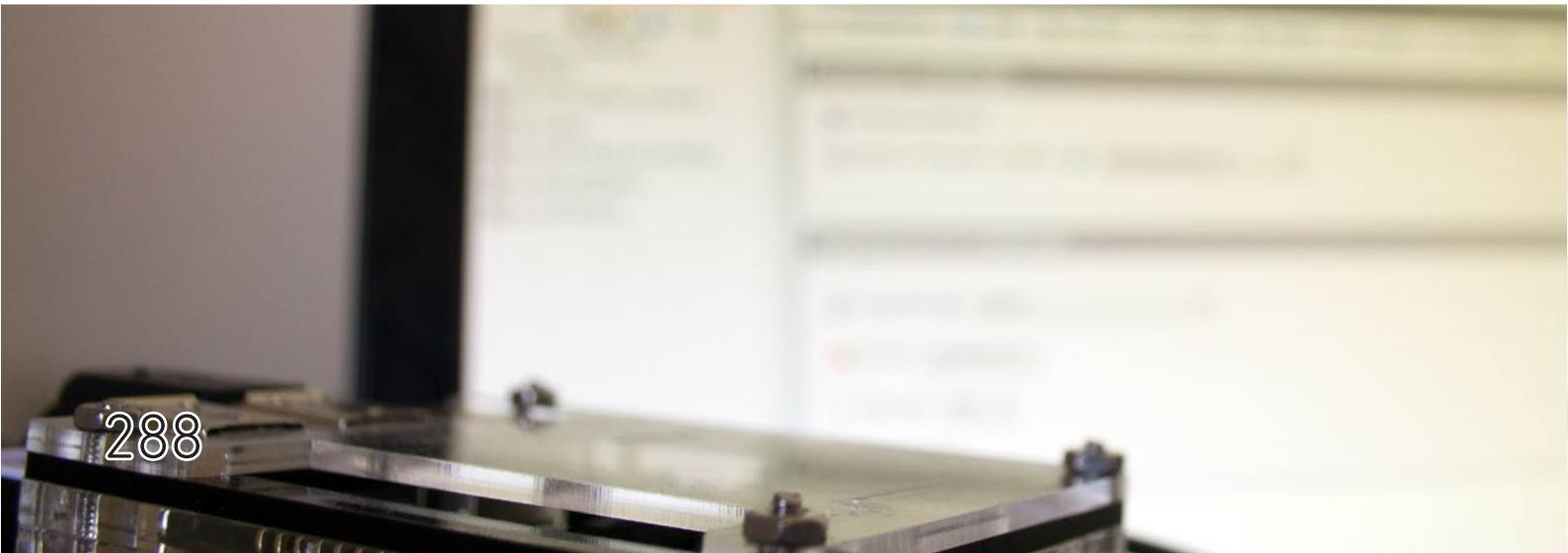
You will find that installing Raspberry Pi PHPMyAdmin is very easy to do and won't take long at all.

If you haven't already installed a web server, you can learn from our guide on installing Apache or our guide to installing NGINX.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

**Optional**
- USB Keyboard
- USB Mouse
- Raspberry Pi Case



## Setting up PHPMyAdmin

You will find that installing PHPMyAdmin onto the Raspberry Pi is very easy to do and can be gotten up and running with a few simple commands.

First, make sure that you followed the two previous tutorials on installing Apache/NGINX/Caddy/Lighttpd, PHP, and MYSQL.

You will need to have on hand the password you set up for the root MYSQL account. If you don't have this, you will not be able to install PHPMyAdmin.

**1.** To install the PHPMyAdmin package we just need to issue the following command.

```
sudo apt-get install phpmyadmin
```

**2.** PHPMyAdmin will now begin to install. It will require your input on various steps along the way. You will be presented with a screen asking the type of web server you want it to run off.

Select **apache2**, as this is the web server that we installed in previous steps, this will configure PHPMyAdmin to work with Apache, and utilize its features.

**3.** Next, we will need to go about configuring PHPMyAdmin to connect to our MYSQL server and set up the various details we need so that we can log in to PHP My Admin.

To do this select **yes** at the next prompt.

**4.** The setup tool will now ask for a password, enter the one you set for root, this is needed for phpMyAdmin to talk with the SQL Server and manage your databases.

**5.** Next, it will ask you to set a password for PHPMyAdmin itself. It is best to set this password to something different to your root SQL password.

Make sure you remember it as this is the password you will need to access the interface.

**6.** With that done we can now proceed to configure our web servers for use with phpMyAdmin.

These steps differ for both Apache and NGINX so make sure you follow the correct list of steps.

## Configuring Apache for phpMyAdmin

**7a.** We will also need to setup Apache to include our PHPMyAdmin installation.

To get started with doing this we need to edit the **Apache2.conf** file, we can do this by entering the following into the terminal:

```
sudo nano /etc/apache2/apache2.conf
```

**7b.** Now we need to add the following line to the bottom of this file. This line will include PHPMyAdmin's configuration and allow it to be loaded in and listened to by Apache.

```
Include /etc/phpmyadmin/apache.conf
```

Once done we can save & exist by pressing **CTRL +X** and then pressing **y**.

# Setting up PHPMyAdmin

**7c.** Now we just need to restart the Apache service.

We need to do this to flush out its current configuration and make it load in our modified file.

We can do this by entering the following command:

```
sudo /etc/init.d/apache2 restart
```

## Configuring NGINX for phpMyAdmin

**7a.** To setup NGINX to work with phpMyAdmin all, we need to do is create a link between the phpmyadmin folder and our root HTML directory.

To do this, we need to run the following command on our Raspberry Pi:

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

## Creating a SQL User for PHPMyAdmin

**8.** By default, PHPMyAdmin will disallow you to log in using the root login.

Instead, you will need to create a new user if you wish to create and access data tables within PHPMyAdmin.

To do this first login as root with the password you selected.

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

**9.** Now run the following command, replacing **username** with the **username of your choice**.

Also, replace **password** with a **secure password** of your choice.

```
GRANT ALL PRIVILEGES ON mydb.* TO 'username'@'localhost' IDENTIFIED BY 'password';
```

**10.** You can exit by typing **quit** in the terminal. Once done you can proceed to loading up PHPMyAdmin.

## Creating a SQL User for PHPMyAdmin

**11.** Now you should be able to access the PHPMyAdmin from a browser.

To test this go to the following address in your favorite web browser. Use the user you created in **step 8** and **step 9** to log in. Do not use your root user as this is disabled by default.

Remember to replace the IP Address with your IP Address. (If you don't have it run the following command: **hostname –I** on your Raspberry Pi Pi)

```
http://192.168.1.108/phpmyadmin
```

# Setting up Wordpress

## Project Description

WordPress is a content management system/blogging software that is used by over 60 million websites. Amongst the top 10 million sites on the web, 23.3% of these use WordPress. (January 2015).

Setting up WordPress on the Raspberry Pi is not hugely complicated, once you have both Apache, PHP and the MYSQL server all set up, it is smooth sailing.

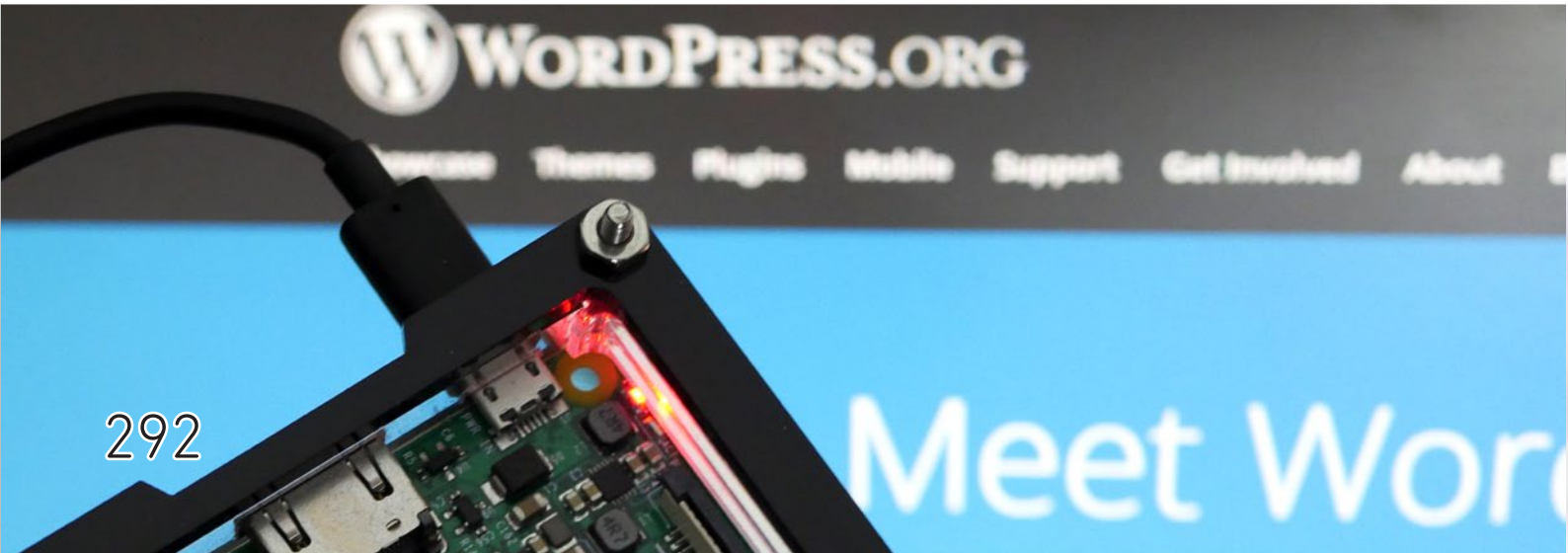
Before you go too far ahead in this tutorial, make sure that you have your MYSQL root password ready.

We will be running through setting up a database for WordPress and showing you how to set up an SQL user to interact with the database all by using the MYSQL Command line.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

**Optional**
- USB Keyboard
- USB Mouse
- Raspberry Pi Case



## Setting up Wordpress on a Raspberry Pi

**1.** Before we get started on utilizing SQL, it's a good time for us to go ahead and download and extract WordPress. We also need to take ownership of the **/var/www/html** folder for PHP to be able to process it properly.

We can achieve all of this by utilizing a few commands, just type each line below into the terminal:

```
cd /var/www/html
sudo chown pi: .
rm * -f -r
wget http://wordpress.org/latest.tar.gz
tar xzf latest.tar.gz
mv wordpress/* ./
rm -rf wordpress latest.tar.gz
sudo chown -R -f www-data:www-data /var/www/html
```

**2.** Now that is installed, we will also need to set up a database within MYSQL for WordPress to store its information, while we can set up and do this through a front-end such as PHPMyAdmin. We, for now, will just make use of the command line to keep this section of the tutorial simple.

We need to first use the MySQL command to log in, use the **–p flag for the password** and **–u for the username**. Make sure your password is written right next to the **-p tag** with **no space**.

**Note:** make sure you replace **password** with the password you set up for root in the setting up MYSQL section of this tutorial

```
mysql -u root -ppassword
```

**3.** Now that we have logged into the MYSQL server, we can issue a command to create a database for WordPress using this simple line:

```
create database wordpress;
```

**4.** You should now see a success statement that looks something like the following:

```
Query OK, 1 row affected (0.00)
```

**5.** Now we will create a separate user for this database, this is safer from a security perspective and is a excellent practice to get into.

While still in MYSQL issue the following command:

```
CREATE USER 'new_user'@'localhost' IDENTIFIED BY 'new_password';
```

**Note**: Make sure you replace **new_user** and **new_password** with details of your choosing, just remember what you choose as you will need to know both when we setup WordPress.

**6.** Once we have a new user, we will need to give it permissions to manage our new database.

We can do that with the command below, make sure you swap **new_user**, with whatever you entered in the previous step.

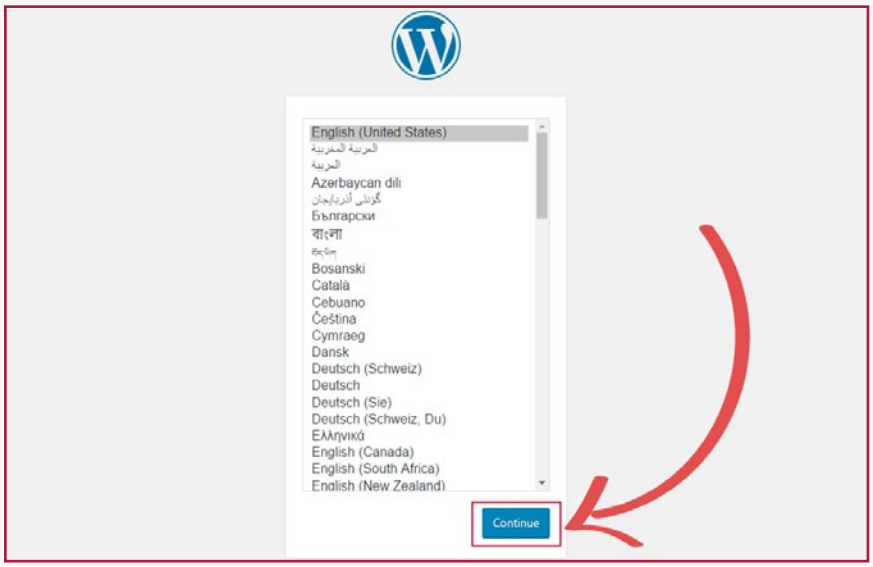
```
GRANT ALL ON wordpress.* TO 'new_user'@'localhost';
```

**7.** Now we have created our database and our user that we want to interact with it. We need to quit out of the MYSQL command interface. We can do that by pressing **Ctrl + D**.

# Setting up Wordpress

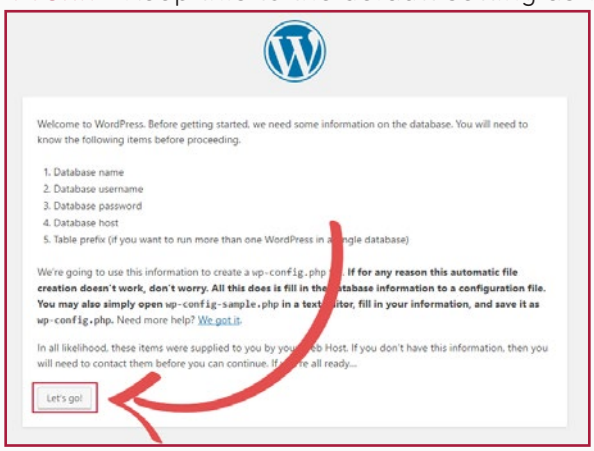
**8.** With that all done, we can now finally finish setting up WordPress, we can start this process by going to your Raspberry Pi's IP address in a web browser.

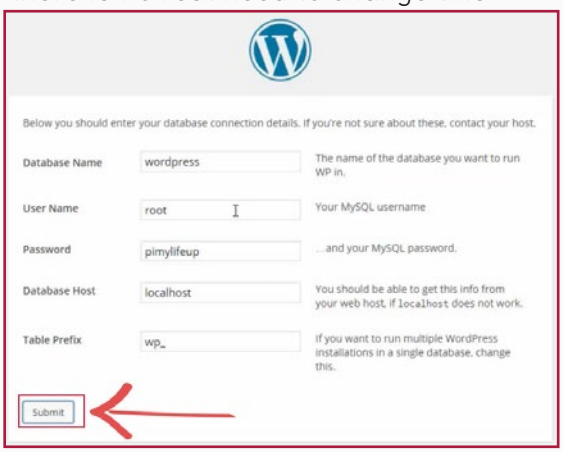
You will be greeted with the following screen asking for you to select a language:



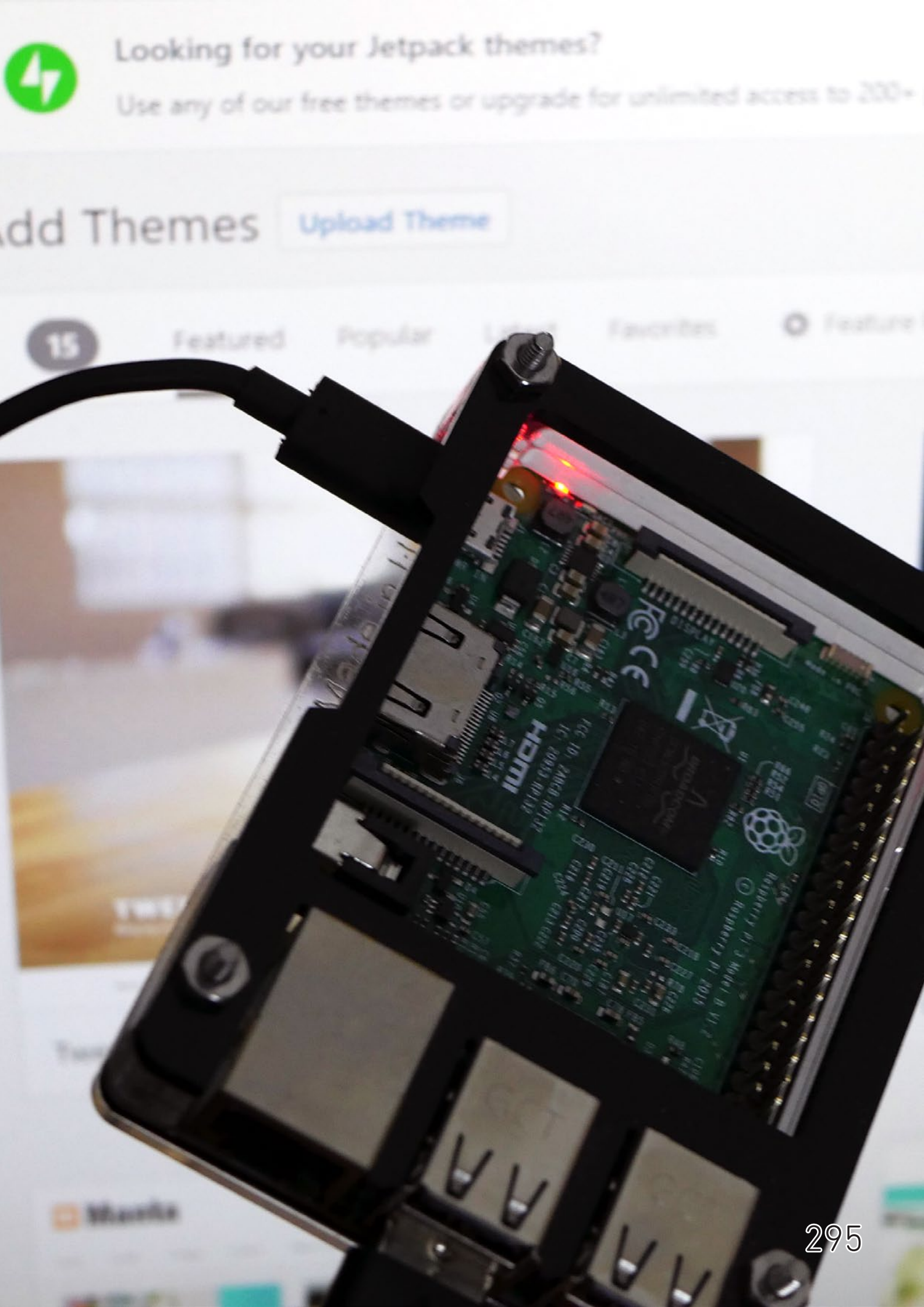

**9.** The next page will explain the details you need to have handy to install WordPress. You can just press "**lets go**" to get to the important screen, here we will need to enter various pieces of data, here's a quick overview of what you will need:

- **Database Name** - This is the database WordPress will connect to, earlier in this tutorial we created a database called **WordPress**.
- **Username** - This is the user we created in **step 7**, make sure you get this detail correct as it needs to connect to the above database.
- **Password** - This is the password we set while creating the MYSQL user in **step 7**, again make sure you enter this correctly as it's needed to connect to the database.
- **Database Host** - Keep this to the default setting, we only need it to connect locally.
- **Table Prefix** - Keep this to the default setting as well, there is no real need to change this




**10.** Assuming you entered all the correct details, you will now be able to press "**Submit**" to successfully install the software, follow the rest of the prompts to finalize the installation.


Looking for your Jetpack themes?
Use any of our free themes or upgrade for unlimited access to 200+
dd Themes
Upload Theme
15
Featured
Popular
Favorites
Feature

# Let's Encrypt

## Project Description

This project will walk you through on installing and setting up the Lets Encrypt Certbot client on the Raspberry Pi. This Certbot client allows the user to grab an SSL certificate from Lets Encrypt by either utilizing your web server or by running its temporary server.

This software allows you to obtain a secure and certified SSL certificate completely free easily. Before you get started with this tutorial, make sure that you have a domain name already set up and pointed at your IP address.

If you are using Cloudflare as your DNS provider, then make sure you have it set to bypass Cloudflare as it hides your IP address.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

**Optional**
- USB Keyboard
- USB Mouse
- Raspberry Pi Case

# Installing and Running Lets Encrypt

**1.** If you are running **Raspbian Stretch or later** you can **skip to step 5** of this tutorial as the Certbot package is available in the Raspbian Stretch repository.

However, if you are running Raspbian Jessie you will have to follow the following four steps to install Certbot on your Raspbian Jessie installation.

Before we get installing the Lets Encrypt Certbot software on Raspbian Jessie, we will first have to adjust our **sources.list** so that we can access the Jessie-Backports branch.

We need to add this as Certbot is not available on Raspbian Jessie by default, be warned though as the backports repository contains software that isn't as thoroughly tested.

Begin editing the sources.list file by using the following command in the terminal:

```
sudo nano /etc/apt/sources.list
```

**2.** To the bottom of this file, add the following line, this line just tells Raspbian where to go searching for packages.

```
deb http://ftp.debian.org/debian jessie-backports main
```

Once done we can save & exist by pressing **CTRL +X** and then pressing **y**.

**3.** Now since our public keys for the new packages are not available by default we will have to grab them and add them to the package manager.

We can grab both public keys we need by typing in the following four commands:

```
gpg --keyserver pgpkeys.mit.edu --recv-key  8B48AD6246925553
gpg -a --export 8B48AD6246925553 | sudo apt-key add -
gpg --keyserver pgpkeys.mit.edu --recv-key  7638D0442B90D010
gpg -a --export 7638D0442B90D010 | sudo apt-key add -
```

**4.** With the package now added to our sources list, we will need to run an update.

We can update our package list with the following command:

```
sudo apt-get update
```

**5.** Now that you are up to installing the Let's Encrypt software onto your Raspberry Pi you will either have to follow the instructions for Raspbian Jessie or Raspbian Stretch.

## Raspbian Stretch and Later

### Apache

```
sudo apt-get install python-certbot-apache
```

### Everything Else

```
sudo apt-get install certbot
```

## Raspbian Jessie

### Apache

```
sudo apt-get install python-certbot-apache -t jessie-backports
```

### Everything Else

```
sudo apt-get install certbot -t jessie-backports
```

# Installing and Running Lets Encrypt

**6.** With Certbot finally installed to our Raspberry Pi we can proceed with grabbing a certificate from Let's Encrypts servers.

There is a couple of ways of handling this that we will be going into over the next few pages.

If you are not using Apache, you can skip this step .

If you are using Apache, then the easiest way of grabbing a certificate is by running the command shown below, this will automatically grab and install the certificate into Apaches configuration.

Before you do that, you will first have to make sure port 80 and port 443 are port forwarded.

Also, if you are using Cloudflare as your DNS provider, you will need to temporarily disable its cloud protection it as it hides your real IP address meaning that Lets Encrypt won't be able to verify that you are the actual owner of that domain.

```
certbot --apache
```

**7.** If you are not running Apache, there are two different ways we can go about grabbing a certificate from Lets Encrypt.

Thanks to the certbot software, one choice is that we can grab the SSL certificate using a standalone python web server. With this option, Certbot will spawn a temporary web server that it will use to verify the domain name ownership.

The other option is that if you are running another web server such as NGINX, we can also utilize that to grab the certificate as well.

With either of these two options will have to setup the certificate manually once it has been succesfully grabbed.

Go to **step 8a** if you are not running another web server, otherwise go to **step 8b**.

**8a.** Utilizing the standalone built-in web server is incredibly easy, though first, you will have to make sure your **port 80** is unblocked and forwarded.

Make sure you replace **example.com** with the domain name you intend on utilizing.

```
certbot certonly --standalone -d example.com -d www.example.com
```

**8b.** Using web root requires a bit more knowledge then using the built-in web server.

Make sure **/var/www/example** points to a working website directory that can be reached from the internet.

Also, make sure to replace **example.com** with the domain name you are using for your website.

```
certbot certonly --webroot -w /var/www/example -d example.com -d www.example.com
```

## Installing and Running Lets Encrypt

**9.** After running these commands, you will be prompted to enter some details, such as your email address.

These details are required for Lets Encrypt to keep track of the certificates it provides and also allow them to contact you if any issues arrive with the certificate.

Once you have filled out the required information, it will proceed to grab the certificate from Lets Encrypt.

If you run into any issues make sure you have a valid domain name pointing at your IP, make sure **port 80** and **port 443** are **unblocked**, and finally, if you are using **CloudFlare** as your **DNS provider**, make sure that you have it currently set to bypass its servers.

The certificates that are grabbed by the certbot client will be stored in the following folder. Of course, swapping out **example.com** with your domain name.

```
/etc/letsencrypt/live/example.com/
```

You will find both the **full chain file (fullchain.pem)** and the certificates **private key file (privkey.pem)** within these folders.

Make sure you don't allow others to access these files as they are what keep your SSL connection secure and identify it as a legitimate connection.

With the files now successfully grabbed you can proceed to setup any piece of software you need to use them.

For instance, if you wanted to setup NGINX to utilize the SSL certificates then follow our guide over the page.

# Using your new SSL Cert with NGINX

**1.** Begin by opening your NGINX configuration file. These are typically stored in **/etc/nginx/** or **/etc/nginx/sites-available/**

Once you have found your configuration file, open it up using your favorite text editor, mine, for instance, is **nano**.

Within the file, you should search for a text block like what is display below.

Make sure you swap out our **example.com** with the domain name that you are using.

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    root /usr/share/nginx/html;
    index index.html index.htm;

    server_name example.com;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

**2.** To this block of code, we will need to make some changes.

Follow our steps and read our explanations on why we are making the change below.

**Find**

```
listen [::]:80 default_server
```

**Add Below**

```
listen 443 ssl;
```

This change tells NGINX to start listening on port 443.

Port 443 is essential as it is the port that handles HTTPS/SSL traffic and will be the port web browsers try to connect over when using https://.

**Find**

```
server_name example.com;
```

**Add Below**

```
ssl_certificate /etc/letsencrypt/live/example.com/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/example.com/privkey.pem;
```

This change tells NGINX where to find our certificate files. It will use these to setup the SSL/HTTPS connection.

The private key is what secures the actual connection. Only your server can read this file. This file should be kept secure otherwise people could potentially intercept and decrypt your traffic.

The full chain contains all the information needed to talk with the server over the HTTPS connection as well as the information required to verify it is a legitimately signed SSL file.

## Using your new SSL Cert with NGINX

**3.** With all those changes done, you should end up with something like what is displayed below. Of course, make sure you replace **example.com** with your domain name.

Once you are satisfied that you have entered the new data correctly, you can save and quit out of the file and then restart NGINX, so it loads in the new configuration.

```
server {
        listen 80 default_server;
        listen [::]:80 default_server

        listen 443 ssl;

        root /usr/share/nginx/html;
        index index.html index.htm;

        server_name example.com;

        ssl_certificate /etc/letsencrypt/live/example.com/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/example.com/privkey.pem;

        location / {
                try_files $uri $uri/ =404;
        }
}
```

**4.** You should now have a fully operational HTTPS connection for your NGINX web server utilizing the certificate we generated with Lets Encrypt.

# Installing OwnCloud

## Project Description

In this project we are going to show you how to set up the Raspberry Pi version of the OwnCloud server, this can act as your very own personal cloud storage.

As the protection of your privacy becomes harder and harder, you may be thinking of moving your files to private cloud storage. If this is the case, then this tutorial is perfect for you.

It is important to remember that since your data will be stored on your local network, you will end up using a lot of bandwidth when uploading and downloading files from outside your local network.

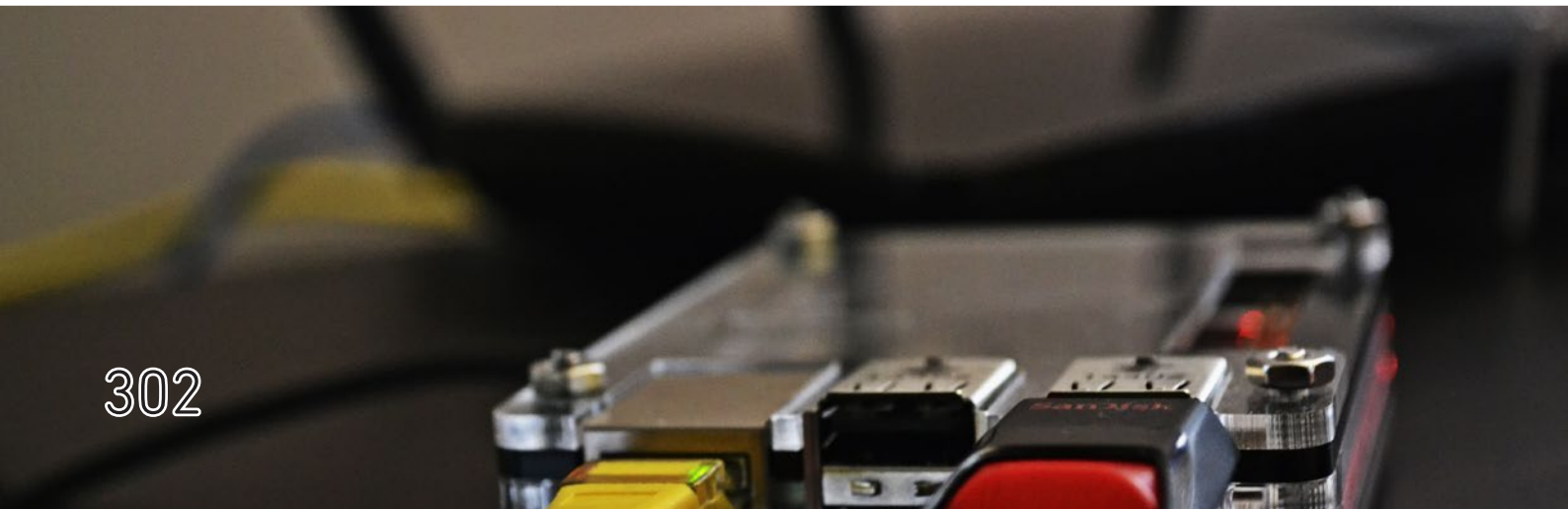
If you're curious and want to learn more about Owncloud be sure to check out their website over at http://owncloud.org.

**Note:** It is highly likely that the USB ports on the Raspberry Pi will be unable to power an external hard drive if your hard drive doesn't utilize an external power supply, we recommend you purchase a powered USB Hub.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection
- External Hard drive or USB Drive

**Optional**
- USB Keyboard
- USB Mouse



## Setting up the OwnCloud Server

**1.** Firstly in either the Raspberry Pi's command line or over SSH, we will need to update the Raspberry Pi's packages.

We can do this by entering the following two commands:

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

**2.** Once that has completed, we will now need to make some change in the Raspi Config tool to make sure OwnCloud runs smoothly.

Get into the config tool by typing in the following command on your Raspberry Pi:

```
sudo raspi-config
```

**3.** In here, start by going to "**Internationalisation Options**". Once selected, you will then want to select the "**Change Locale**" menu.

Within here, find and select "**en_US.UTF8**", press "**Space**" and then hit enter.

Doing this will take you to another screen, here again, select "**en_US.UTF8**" and press enter.

**4.** Now we need to go tp "**Advanced Options**", in here look for menu labeled "**Memory Split**".

Within this menu, change the number displayed to **16m** and press **enter**.

**5.** Finally, for the last thing we need to do in the configuration tool is to enable an overclock, you should only need to bother doing this if you are running on the **original Raspberry Pi's**.

The **Raspberry Pi 2** and the **Raspberry Pi 3** should be fine at their default clock speeds but you can view more at overclocking them in our Overclocking the Raspberry Pi guide in our "Getting Started" book.

So if you are running the **first generation Raspberry Pi**, you will need to go to the menu named "**Overclock**", this menu will present you numerous options.

We recommend that you select "**Medium**" as the overclock, as it is a good balance between performance and extra demand on the Raspberry Pi.

**6.** Now to prepare for the installation of NGINX, we need to create a new user called **www-data** and add it to a group names **www-data**.

We can do that with the following command:

```
sudo usermod -a -G www-data www-data
```

## Setting up the OwnCloud Server

**7.** Now we need to install all the packages to get NGINX and OpenSSL running. These will both be used to set up OwnCloud.

Install them all by typing in the following long command :

```
sudo apt-get install nginx openssl ssl-cert php7.0-cli php7.0-sqlite php7.0-gd php7.0-common php7.0-cgi sqlite3 php-pear curl libapr1 libtool curl libcurl4-openssl-dev php7.0-xml php7.0 php7.0-dev php7.0-curl php7.0-gd php7.0-fpm memcached php7.0-memcache php7.0-zip php7.0-intl php7.0-mbstring varnish
```

**8.** Now we need to generate an SSL certificate. Doing this will help ensure that your files are transferred securely.

You can generate this certificate with the following command:

```
sudo openssl req $@ -new -x509 -days 730 -nodes -out /etc/nginx/cert.pem -keyout /etc/nginx/cert.key
```

Simply just enter the relevant data for each of the questions it asks you.

**9.** Now we need to chmod the two cert files we just generated.

Running these two commands makes sure that only programs like NGINX can read the files.

```
sudo chmod 600 /etc/nginx/cert.pem
sudo chmod 600 /etc/nginx/cert.key
```

**10.** Let's clear the server config file since we will be copying and pasting our version in it.

```
sudo sh -c "echo '' > /etc/nginx/sites-available/default"
```

**11.** Now let's configure the web server configuration so that it runs OwnCloud correctly.

We do this by modifying the default configuration file by running the following command.

```
sudo nano /etc/nginx/sites-available/default
```

## Setting up the OwnCloud Server

**12.** Now enter the following lines into the file, making sure you replace the IP "**192.168.1.116**" with your own Raspberry Pi's IP. These will be next to the text "**server_name**".

```
upstream php-handler {
    server 127.0.0.1:9000;
}
server {
    listen 80;
    server_name 192.168.1.116;
    return 301 https://$server_name$request_uri;  # enforce https
}

server {
    listen 443 ssl;
    server_name 192.168.1.116;
    ssl_certificate /etc/nginx/cert.pem;
    ssl_certificate_key /etc/nginx/cert.key;
    root /var/www/owncloud; # Path to the root of your installation
    client_max_body_size 1000M; # set max upload size
    fastcgi_buffers 64 4K;
    rewrite ^/caldav(.*)$ /remote.php/caldav$1 redirect;
    rewrite ^/carddav(.*)$ /remote.php/carddav$1 redirect;
    rewrite ^/webdav(.*)$ /remote.php/webdav$1 redirect;
    index index.php;
    error_page 403 /core/templates/403.php;
    error_page 404 /core/templates/404.php;
    location = /robots.txt {
        allow all;
        log_not_found off;
        access_log off;
    }
    location ~ ^/(?:\.htaccess|data|config|db_structure\.xml|README) {
        deny all;
    }
    location / {
        # The following 2 rules are only needed with webfinger
        rewrite ^/.well-known/host-meta /public.php?service=host-meta last;
        rewrite ^/.well-known/host-meta.json /public.php?service=host-meta-json last;
        rewrite ^/.well-known/carddav /remote.php/carddav/ redirect;
        rewrite ^/.well-known/caldav /remote.php/caldav/ redirect;
        rewrite ^(/core/doc/[^\/]+/)$ $1/index.html;
        try_files $uri $uri/ index.php;
    }
    location ~ \.php(?:$|/) {
        fastcgi_split_path_info ^(.+\.php)(/.+)$;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param PATH_INFO $fastcgi_path_info;
        fastcgi_param HTTPS on;
        fastcgi_pass php-handler;
    }
    # Optional: set long EXPIRES header on static assets
    location ~* \.(?:jpg|jpeg|gif|bmp|ico|png|css|js|swf)$ {
        expires 30d;
        # Optional: Don't log access to assets
        access_log off;
    }
}
```

**13.** Now simply save and exit out of the file by pressing **Ctrl + X** and pressing **Y** and **enter** to save the file.

**14.** Once done you can save and exit. Now we need to edit the conf file by entering the following:

```
sudo nano /etc/php/7.0/fpm/pool.d/www.conf
```

**15.** Update the listen line to the following:

```
listen = 127.0.0.1:9000
```

**16.** Once done save and then exit by pressing **Ctrl + X** then pressing **Y** and then **enter**.

Now we also need to edit the **dphys-swapfile**.

To do this open the file by entering:

```
sudo nano /etc/dphys-swapfile
```

**17.** Now update the **conf_swapsize** line to the following:

```
CONF_SWAPSIZE = 512
```

**18.** Save the file by pressing **Ctrl + X** and then press **Y** and press **enter**.

**19.** Now we need to restart the Raspberry Pi to get it to load in the new changes. Type in the following command:

```
sudo reboot
```

**20.** Once the Raspberry Pi has restarted, you will need to install OwnCloud on the Raspberry Pi. We can do this by entering the following commands:

```
sudo mkdir -p /var/www/owncloud
sudo wget https://download.owncloud.org/community/owncloud-9.1.2.tar.bz2
sudo tar xvf owncloud-9.1.2.tar.bz2
sudo mv owncloud/ /var/www/
sudo chown -R www-data:www-data /var/www
rm -rf owncloud owncloud-9.1.2.tar.bz2
```

**21.** We also need to make some changes to the **.htaccess** file and the **.user.ini** file over in the OwnCloud folder. Enter the following command to change directory and open the **.htaccess** file.

```
cd /var/www/owncloud
sudo nano .htaccess
```

**22.** In here set the following values to **2000M**, this allows PHP to handle large files being uploaded to the device.

```
php_value_upload_max_filesize 2000M

php_value_post_max_size 2000M

php_value_memory_limit 2000M
```

**23.** Once set, save the file by pressing **Ctrl + X** and then pressing **Y** and then **Enter**. Now we need to open up the **.user.ini** file with the following command.

```
sudo nano .user.ini
```

## Setting up the OwnCloud Server

**24.** In here we need to update the following values so that they are set to **2000M**

```
upload_max_filesize=2000M

post_max_size=2000M

memory_limit=2000M
```

**25.** We can now save and exit the file by pressing **Ctrl + X**, then pressing **Y** and then **Enter.**

Now that is done we should be able to connect to OwnCloud at your Raspberry Pi's IP address.

Before you set up the admin account, you might want to mount an external drive. Our guide below will show you how to do this on your Raspberry Pi.

## Mounting & Setting up a drive

Setting up an external drive should be relatively straightforward but sometimes things don't work as correctly as they should.

These instructions are for mounting and allowing Owncloud to store files onto an external hard drive.

**1.** Firstly if you have an NTFS drive we will need to install an NFTS package by entering the following:

```
sudo apt-get install ntfs-3g
```

**2.** Now let's make a directory we can mount to.

```
sudo mkdir /media/owncloudrive
```

**3.** Now we need to get the **gid**, **uid**, and the **uuid** as we will need to use them soon. Enter the following command for the **gid**:

```
id -g www-data
```

Make sure you write down the values from this command.

**4.** Now to get **uid** we need to enter the following command:

```
id -g www-data
```

Make sure you write down the values from this command.

**5.** Additionally, we also need to get the UUID of the hard drive. This UUID will help to make the Raspberry Pi remember the hard drive even if you plug it into a different USB port.

```
ls -l /dev/disk/by-uuid
```

This should produce an output like below.

```
pi@raspberry pi ~ $ ls -l /dev/disk/by-uuid
total 0
lrwxrwxrwx 1 root root 15 Aug 28 08:08 15CD-3B79 -> ../../mmcblk0p1
lrwxrwxrwx 1 root root 15 Aug 28 08:08 DC72-0315 -> ../../sda1
```

Copy the light blue letters and numbers of the last entry (Should have something like -> **../../sda1** at the end of it).

## Mounting & Setting up a drive

**6.** Now let's add your drive into the **fstab** file, so it is booted with the correct permissions. Let's begin by editing the **fstab** file with the following command:

```
sudo nano /etc/fstab
```

**7.** Now add the following line to the bottom of the file, replacing the **uid**, **guid** and the **UUID** with the values we got above. (The following should all be on a single line)

```
UUID=DC72-0315 /media/owncloudrive auto nofail,uid=33,gid=33,umask=0027,dmask=0027,noatime 0 0
```

Once set, save the file by pressing **Ctrl + X** and then pressing **Y** and then **Enter**.

**8.** We should now reboot the Raspberry Pi with the following command. The drives should automatically be mounted. If they are mounted, we are all good to go.

```
sudo reboot
```

**Note:** If you get an error stating the Pi is in emergency mode at boot up then this likely means a problem with the fstab entry. Just edit the **fstab** file (**sudo nano /etc/fstab**) and remove the added line or look for a mistake and fix it.

## Basic First Setup

I will briefly go through the basics of setting up Owncloud Raspberry Pi here. If you want more information I highly recommend checkout out the manuals on their website that you can find at this URL: **https://doc.owncloud.org/server/9.1/admin_manual/**

**1.** In your browser enter your Raspberry Pi's IP address in my case it is **192.168.1.116.**

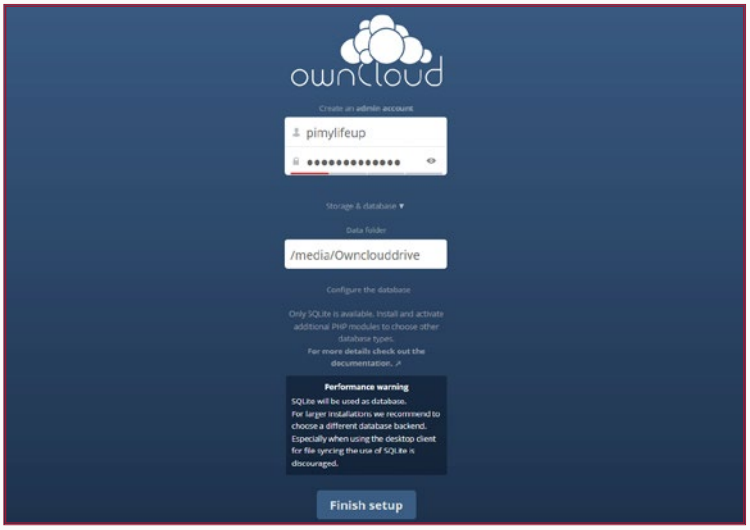
**2.** Once you go to the IP you're like to get a certificate error, just add this to your exception list as it will be safe to proceed.

**3.** When you first open OwnCloud you should be presented with a simple setup screen and no errors.

**4.** Enter your desired username and password

**5.** Click on storage & database and enter your external drive **/media/owncloudrive** (Skip this step if you didn't setup an external drive).

**6.** Once all done, click finish setup.




If you ever need to update and you find the internal updater is not working this likely means, you will need to do it manually.

You can find a detailed process on how to update over at OwnCloud's update manual page that you can find at:

**https://doc.owncloud.org/server/9.1/admin_manual/maintenance/upgrade.html#manual-upgrade-procedure**

# Port Forwarding & External Access

If you want to have access to your cloud drive outside your local network, then you will need to setup port forwarding and make a few changes to our config files.

Firstly, we need to go back to the default file and change the server_name values (There is 2 of them). Update these to your external IP address.

If you have a dynamic IP address, you may want to set up a dynamic DNS service and use that as your address to access OwnCloud.

**1.** To get started we need to run the following command to bring up our default server file:

```
sudo nano /etc/nginx/sites-available/default
```

**2.** Now enter the following lines into the file, making sure you replace the IP addresses with your external IP address. These will be next to the text "**server_name**".

Once set, save the file by pressing **Ctrl + X** and then pressing **Y** and then **Enter**.

**3.** Once you have updated the IP's in the server file, you will need to add the external IP to your trusted IP list and make sure OwnCloud doesn't overwrite it.

To do this open up the OwnCloud config file and enter:

```
sudo nano /var/www/owncloud/config/config.php
```

**4.** In here add a new item to the trusted domains array (This will be your external IP address). Your new entry should look something like this (x are just placeholders).

```
1 => 'xxx.xxx.xxx.xxx',
```

**5.** Finally update the URL of the **overwrite.cli.url** line to your IP Address. It should look something like this.

```
'overwrite.cli.url' => 'https://xxx.xxx.xxx.xxx',
```

Below is an example of the completed config.txt file.

```
<?php
$CONFIG = array (
        'instanceid' => 'sampleinstanceid',
        'passwordsalt' => 'samplepasswordsalt',
        'secret' => 'samplesecret'
        'trusted_domains' =>
        array (
          0 => '192.168.1.116',
          1 => 'xxx.xxx.xxx.xxx',
        ),
        'datadirectory' => 'var/www/owncloud/data',
        'overwrite.cli.url' => 'https://xxx.xxx.xxx.xxx',
        'dbtype' => 'sqlite3',
        'version' => '8.1.1.3',
        'logtimezone' => 'UTC',
        'installed' => true,
);
```

## Port Forwarding & External Access

**6.** Once you are certain you have entered the information correctly, save the file by pressing **Ctrl + X** and then pressing **Y** and then **Enter**.

**7.** Once done, we now need to restart the NGINX service, this makes it load in our updated configuration file.

We can do this with the following command:

```
sudo service nginx restart
```

Be sure to check out the guide on port forwarding earlier on in this book and use the following port 443 for internal, and I recommended a random port for the external port.

Make sure when setting up the external port that a certain program does not already reserve it otherwise it will cause issues with your home network.

When connecting to the OwnCloud server externally, you will need to make sure you use https otherwise you will get an invalid request in your browser.

# Installing NextCloud

## Project Description

For this project, we will be showing you how to setup and configure the NextCloud server for the Raspberry Pi. NextCloud can act as your cloud storage system.

As time goes on the protection of your privacy with 3rd party companies, become harder and harder. Software like NextCloud comes in hand, as it gives you full control over your files with no 3rd party controller.

It is important to remember that since your data will be stored on your local network, you will end up using a lot of bandwidth when uploading and downloading files from outside your local network.

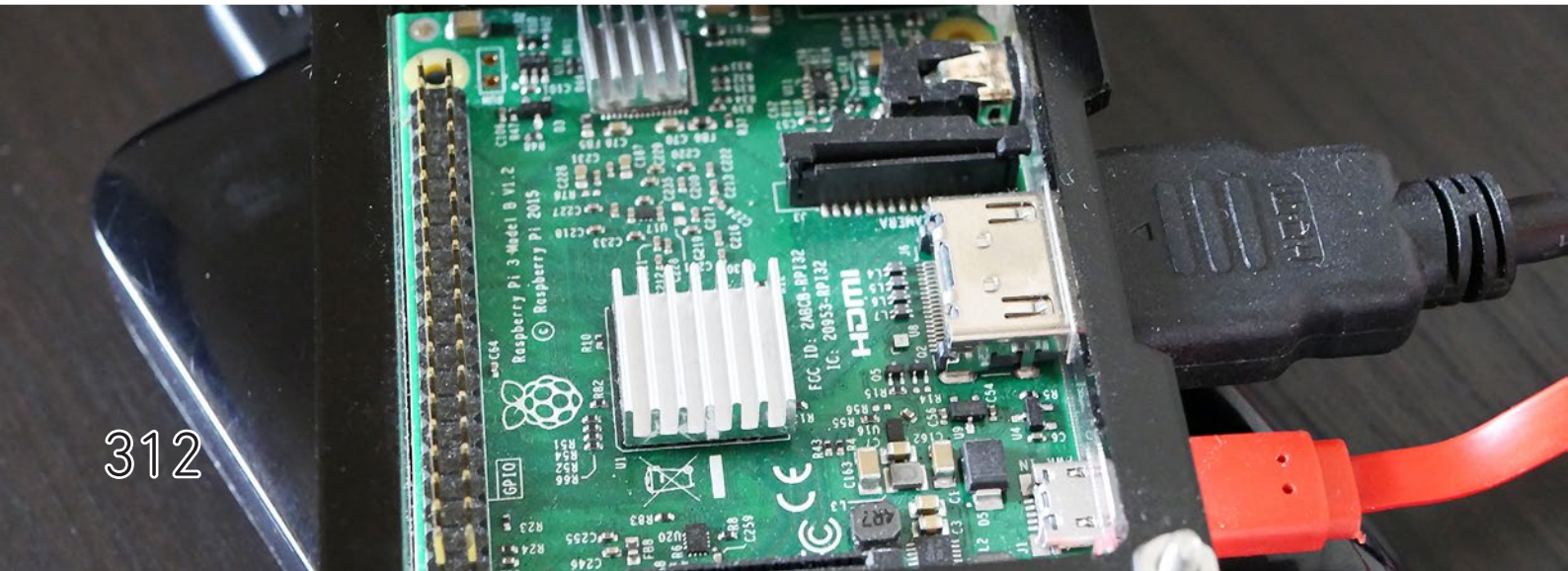
If you want to learn more about NextCloud, you can check out their website at **https://nextcloud.com**.

**Note:** It is likely that the USB ports on the Raspberry Pi will be unable to power an external hard drive, if your hard drive doesn't utilize an external power supply, we recommend purchasing a powered USB Hub.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

**Optional**
- Raspberry Pi Case
- External Hard Drive or USB Drive



## Installing Apache and PHP

To run NextCloud on the Raspberry Pi, we will first need to install and setup Apache and PHP, we won't be going too in-depth into installing these as they are a minor component to this tutorial.

If you want to learn more about setting up a Web Server, then be sure to follow our tutorial earlier in the book. For this tutorial, we will only be utilizing PHP 7.

**1.** To get started lets first update our package repositories with the following command:

```
sudo apt-get update
```

**2.** With that done, lets now install apache by using the following command:

```
sudo apt-get install apache2
```

You can check to make sure Apache2 is successfully working and up and running by going to your Raspberry Pi's IP address. Going to this should load a default Apache2 page.

If you need to get your Raspberry Pi's local IP address, then you can type in **hostname -I** into the terminal.

**3.** With Apache2 now installed onto our Raspberry Pi, we just need to install PHP and several of its packages, for this tutorial we will be just using PHP7.

To install PHP and the packages we need, run the following commands:

```
sudo apt-get install php7.0 php7.0-gd php7.0-sqlite php7.0-curl php7.0-zip php7.0-xml
php7.0-mbstring sqlite
```

**4.** With Apache and PHP now installed there is one final thing we need to do, and that is to restart Apache. You can do this with the following command:

```
sudo service apache2 restart
```

## Installing NextCloud

Installing NextCloud to the Raspberry Pi is quite simple, it mainly involves downloading the script from their website. Extracting the zip and then going to your Raspberry Pi's IP address.

**1.** To get started lets first move to our HTML directory with the following command:

```
cd /var/www/html
```

**2.** Now we can run the following curl command so we can download and extract the latest version of NextCloud in one go.

```
curl https://download.nextcloud.com/server/releases/nextcloud-13.0.4.tar.bz2 | sudo tar -jxv
```

**3.** We now need to create a data directory for NextCloud to operate in. For the initial setup of NextCloud, we must make this folder in our HTML directory. Do that with the following command

```
sudo mkdir -p /var/www/html/nextcloud/data
```

# Installing NextCloud

**4.** Now let's give the correct user and group control over the nextcloud folder by running the following command.

```
sudo chown -R www-data:www-data /var/www/html/nextcloud/
```

**5.** Finally, we need to give it the right permissions, again run the following command:

```
sudo chmod 750 /var/www/html/nextcloud/data
```

**6.** Now we have finished with that we can now finally go to NextCloud itself and begin its installation process. To start go to your Raspberry Pi's IP address plus **/nextcloud**. For example, the address I would go to is the following:

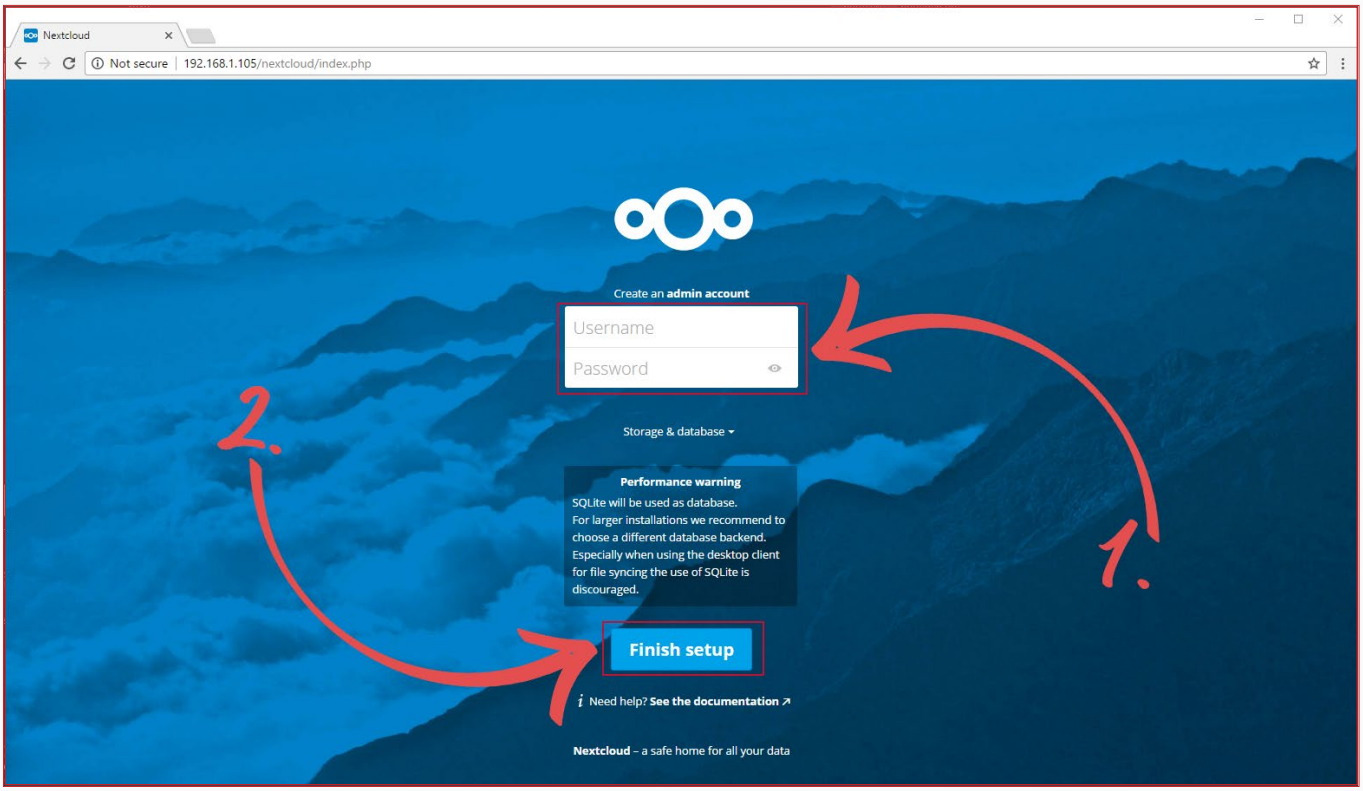
Remember to replace my IP Address with that of your Raspberry Pis.

```
https://192.168.1.105/nextcloud
```

**7.** You will now be greeted with the following screen. Here you will need to type in the Username and Password that you intend to use for your admin account.

If you plan on allowing your NextCloud file service to be accessible from outside your network, make sure that you use a long and secure password.

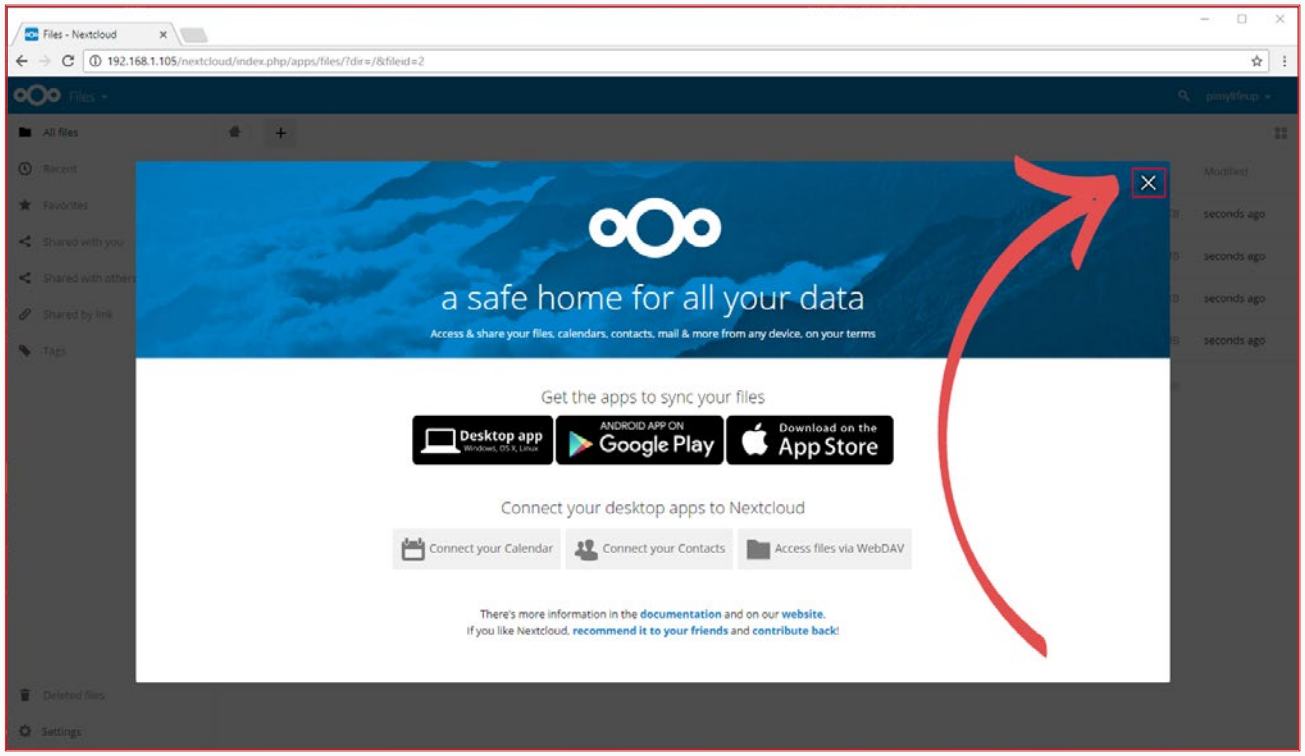
Once you are happy with this, press the "**Finish Setup**" button, please note this can take some time to complete as it finalizes your setup.



# Installing NextCloud

**8.** After this you should now be greeted with the following welcome screen, this just lays out the various programs you can use to connect with your NextCloud installation.

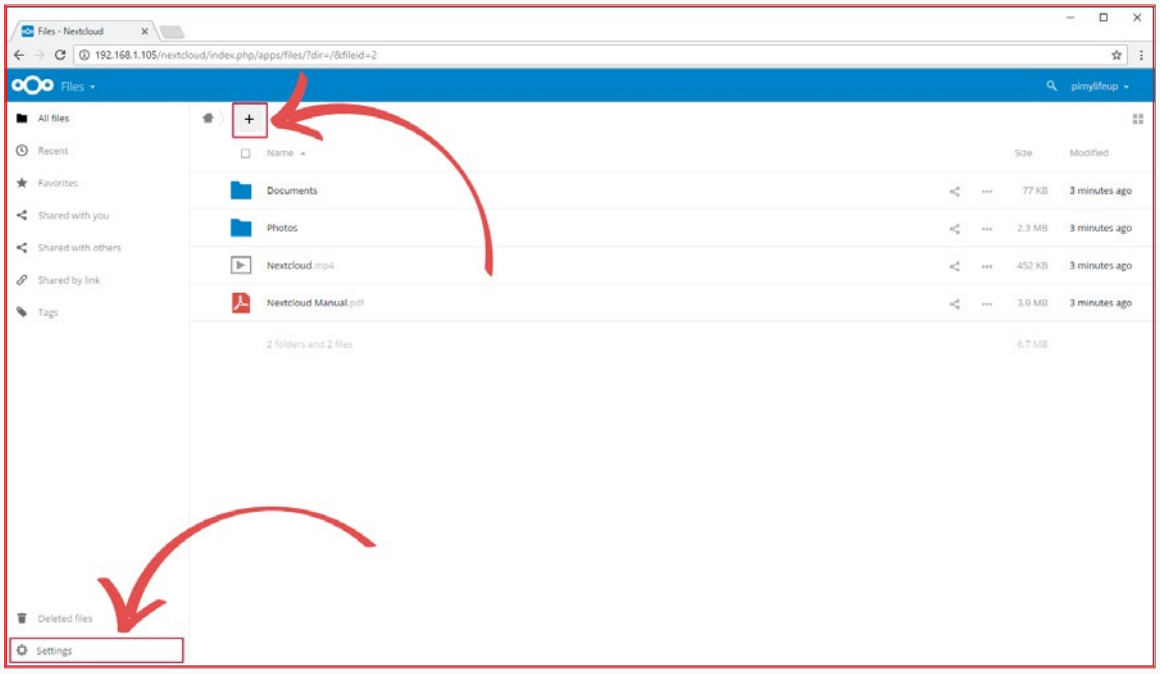
Just press the **X** button in the top right corner to continue




**9.** Now you can finally see the interface of NextCloud. You should take some time to familiarize yourself with all the functionality of Nextclouds interface.

We won't go in depth on how to use the NextCloud interface.

We have however highlighted some of the key areas to check out though is highlighted in the screenshot below.



## Moving NextClouds data folder

With NextCloud now safely installed we can now tweak the setup to both be more secure and a bit more usable, one of the first things we should do is move the data directory, so it does not sit in our web-accessible directory.

Doing this is also the same way you would move your NextCloud data directory onto a larger external hard drive rather than putting an increased load onto the Raspberry Pi's SD Card.

**1.** To get started let's make our new directory for where we will store our data files, to make it easy we will make a new folder /var/nextcloud and move our data folder into there.

Create the folder by running the following command:

```
sudo mkdir -p /var/nextcloud
```

**2.** With the new folder we created, we will now move our data directory into it. Moving the folder is easy to do thanks to the **mv** command.

Please note that your NextCloud system will be out of action while we move the file then adjust the configuration file.

To begin the move type in the following command:

```
sudo mv -v /var/www/html/nextcloud/data /var/nextcloud/data
```

**3.** Now with the files moved over we can now modify the '**datadirecotry**' configuration to point to our new directory.

First, let's change to the config directory for NextCloud with the following command.

```
cd /var/www/html/nextcloud/config
```

**4.** We can now copy the config file to make a backup of the file.

We can do this with the following command:

```
sudo cp -p config.php config.php.bk
```

**5.** Finally let's open up the config.php file for editing using nano.

```
sudo nano config.php
```

**6.** Within this file we need to change the following line:

```
'datadirectory' => '/var/www/html/nextcloud/data',
```

To

```
'datadirectory' => '/var/nextcloud/data',
```

**7.** Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

You should be able to refresh your web browser now, and all your files should be shown as they were previously.

## Increasing NextClouds max upload size

By default, PHP has a very low upload limit, so low its only 2 MB. To change this, we need to modify the php.ini file and increase the limit. A cloud storage system wouldn't be very useful if you could only ever upload 2mb files.

**1.** To get started we need to begin editing our configuration file with the following command:

```
sudo nano /etc/php/php7.0/apache2/php.ini
```

**2.** Now we need to find and replace the following two lines.

```
post_max_size = 8M
upload_max_filesize = 2M
```

To

```
post_max_size = 1024M
upload_max_filesize = 1024M
```

Of course, you can set the file size limits to something that is much higher than 20M, so feel free to change that number to whatever you think is the maximum size file you will upload to your NextCloud.

**3.** Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

Now we need to restart Apache2 to force it to read in the updated configuration file.

We can do that quickly with the following command:

```
sudo service apache2 restart
```

**4.** You should now be able to restart your web browser and begin a new upload to see that the maximum upload size has been increased successfully.

# Allowing the .htaccess overide

Next, we need to deal with the .htaccess file for NextCloud. Since we installed NextCloud into the default Apache2 directory, **/var/www/html**, we will need to change some setting in Apache2 to allow the .htaccess file to override settings.

**1.** To get started we can begin editing the file with the following command:

```
sudo nano /etc/apache2/apache2.conf
```

**2.** With the file now open we need to find the following block, you can also use **Ctrl + W** to help find the block.

```
<Directory /var/www/>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

To

```
<Directory /var/www/>
    Options Indexes FollowSymLinks
    AllowOverride All
    Require all granted
</Directory>
```

**3.** Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

**4.** Now we need to restart Apache2 to force it to read in the updated configuration file.

We can do that quickly with the following command:

```
sudo service apache2 restart
```

**5.** You can check whether the changes have successfully worked by going to the settings page on NextCloud. Any warning about the **.htaccess** file not working correctly should now be gone.

# Setting up SSL for NextCloud

Now we should work on setting up your NextCloud setup, so it runs through HTTPS and not plain HTTP. For this tutorial, we will assume that you do not have a domain name, so we will be generating our self signed certificate and not utilizing one from a free service such as Letsencrypt.

**1.** Before we go modifying our Apache2 configuration, we will first generate the self-signed certificate. Luckily, we can do this all in one command thanks to OpenSSL.

Remember that a self-signed certificate will throw errors in your web browser and is not as secure as a properly signed certificate, but it is better than nothing and is the only option if you're not utilizing a domain name.

Before we generate the certificate, lets first make a directory to store it.

```
sudo mkdir -p /etc/apache2/ssl
```

**2.** Now let's generate the certificate itself by running the following command in the terminal:

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:4096
-keyout /etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt
```

If you want to know what these command arguments do, then read our little description below.

- **req**: This specifies a subcommand for X.509 certificate signing request (CSR) management.
- **-x509**: This option specifies that we want to make a self-signed certificate file instead of generating a certificate request.
- **-nodes**: This tells the openssl application that we don't want to specify a passphrase, a passphrase will require us to enter it every time Apache is restarted which is painful to deal with.
- **-days 365**: This specifies the amount of days we want the certificate to remain valid for, after this number of days you will have to generate a new certificate.
- **-newkey rsa:4096**: This option will create the certificate request and a new private key at the same time. This is necessary since we didn't create a private key in advance. The rsa:2048 tells OpenSSL to generate an RSA key that is 2048 bits long.
- **-keyout**: This parameter names the output file for the private key file that is being created.
- **-out**: This option names the output file for the certificate that we are generating.

After pressing enter, you will be presented with the following options to fill out.

```
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

**3.** Once you have filled out all that information, we can then proceed on with setting Apache2 to run SSL and also to utilize our newly generated certificate.

First, let's enable the SSL module for Apache with the following command:

```
sudo a2enmod ssl
```

**4.** Now we need to modify the **default-ssl.conf** file so it will utilize our new certificates and not the default ones that are generated by OpenSSL on installation.

To begin modifying this file run the following command:

```
sudo nano /etc/apache2/sites-available/default-ssl.conf
```

**5.** Within this file we need to change the two lines below to point to our new certificates we generated into our **/etc/apache2/ssl** folder.

**Change**

```
SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem
SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
```

**To**

```
SSLCertificateFile /etc/apache2/ssl/apache.crt
SSLCertificateKeyFile /etc/apache2/ssl/apache.key
```

**6.** Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

**7.** We can now enable the default-ssl configuration and restart Apache to load in our new configuration. We can do this with the following two commands.

```
sudo a2ensite default-ssl.conf
sudo service apache2 restart
```

**8.** You can test to make sure this is working by going to your Raspberry Pi's IP address with **https://** in front of it. It will give you a warning about it potentially being an invalid certificate. This is normal as it is an unsigned certificate.

For instance to make sure my own copy of Nextcloud is now running behind SSL I would go to **https://192.168.1.105/nextcloud**

**9 (Optional).** To ensure you have the best security setup for Nextcloud. We should enforce SSL so no connection can be made over HTTP. It will automatically redirect to HTTPS.

We can do this by making some changes to our apache configuration, to begin let's edit the default file with the following command:

```
sudo nano /etc/apache2/sites-available/000-default.conf
```

**10 (Optional).** Replace all the text in this file with the code below. This will basically redirect all HTTP traffic to its HTTPs equivalent.

```
<VirtualHost *:80>
  ServerAdmin example@example

  RewriteEngine On
  RewriteCond %{HTTPS} off
  RewriteRule ^(.*)$ https://%{HTTP_HOST}$1 [R=301,L]
</VirtualHost>
```

**11 (Optional).** Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

## Setting up SSL for Nextcloud

**12 (Optional).** Now before this will work we need to enable the redirect module and restart apache. We can easily achieve this by running the following two commands:

```
sudo a2enmod rewrite
sudo service apache2 restart
```

Now going to your Raspberry Pi on HTTP should automatically redirect to the HTTPS version.

For example if i go to **http://192.168.1.105** it will redirect to **https ://192.168.1.105**

## Portforwarding NextCloud

Before we get started with this section, you need to know that NextCloud will only operate under specifically specified trusted domains. Which means you will either must specify a domain name that you want to use for your connection or your public IP address.

Since most home public IP addresses are dynamic, you must look into setting up a dynamic DNS service. We have a tutorial on setting up a dynamic DNS earlier on in the book.

**1.** To add your wanted domain/IP we need to modify Nextcloud's configuration file, we can do that by running the following command:

```
sudo nano /var/www/html/nextcloud/config/config.php
```

**2.** Within this file, you will see a block of text like below. This text is an array of all trusted domains that you allow NextCloud to operate through.

For now, it should only include your Raspberry Pi's local IP address.

We will add our new domain/IP onto the end of this array.

```
'trusted_domains' =>
array (
  0 => '192.168.1.105',
),
```

For our example, we will be adding **nextcloud.pimylifeup.com** to the array, meaning we need to increment the array ID and add the domain name.

Once you have added a new one, it should look something like below.

Repeat this procedure for any new IP's or domains you want NextCloud to be able to operate through.

```
'trusted_domains' =>
array (
  0 => '192.168.1.105',
  1 => 'nextcloud.pimylifeup.com',
),
```

**3.** Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

**4.** Finally, you will need to port forward two ports to finally have NextCloud up and running. These two ports being Port **80** and Port **443**. The protocol required for these is **TCP**.

# IRC Server

## Project Description

In this project, we will be showing you how to set up a Raspberry Pi IRC Server. An IRC server is perfect for the Raspberry Pi as compared to most other servers. It's not hugely performance demanding.

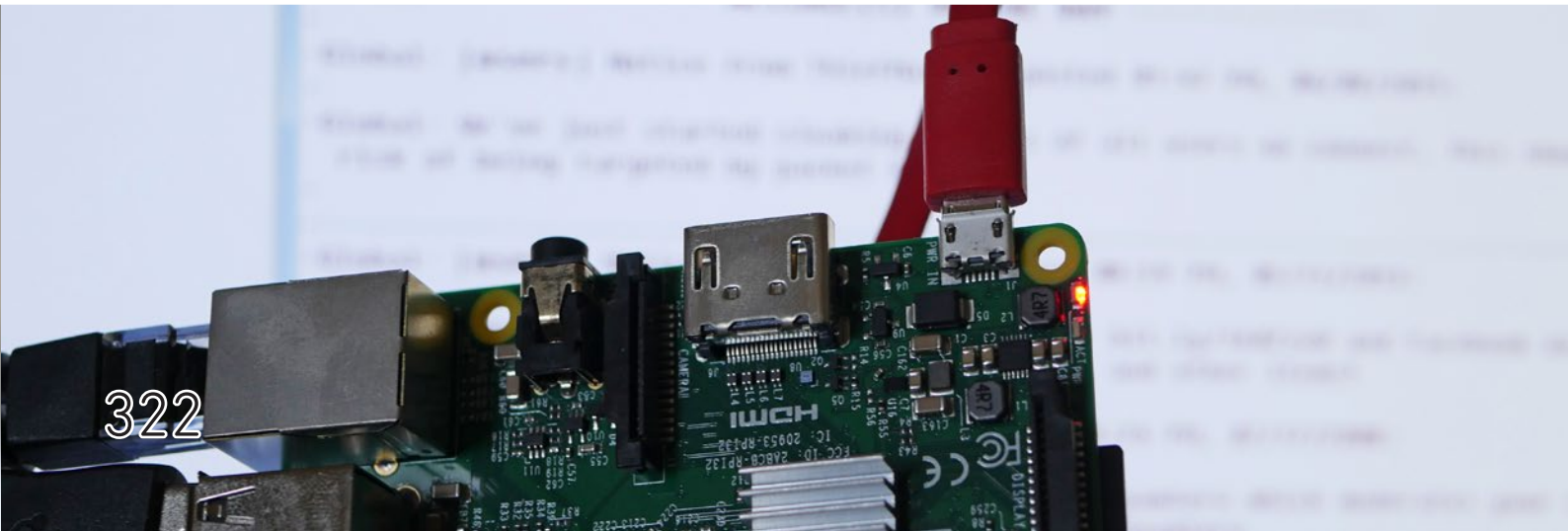
IRC for those who don't know stands for Internet Relay Chat, it is one of the oldest chat protocols, and even though it has fallen in popularity in recent years, it is still the backbone of many communities. Even Twitch makes use of an IRC server to handle its chat.

For this project, we will be utilizing Ircd-Hybrid as it is the most lightweight, high-performance and reliable IRC server available.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

**Optional**
- USB Keyboard
- USB Mouse
- Raspberry Pi Case



# Installing and configuring the IRC Server

**1.** Before we get started with installing and setting up our Raspberry Pi IRC Server, we will first run an update and upgrade to ensure our Raspberry Pi is completely up to date.

To do this run the following two commands in the terminal:

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

**2.** With our Raspberry Pi now up to date, we can proceed with installing the IRC Server software.

We are using Ircd-Hybrid which is a very lightweight, stable and high-performance IRC server. All three of which are perfect characteristics for the Raspberry Pi's limited resources.

To install the software for our IRC Server, type the following command into the terminal.

```
sudo apt-get install ircd-hybrid
```

**3.** To setup oping on our IRC Server, we will also need to set a password. ircd-hybrid requires this password to be encrypted.

We can use the following command to encrypt the password.

Make sure you switch out **password** for yours and save the result as we will need this later.

```
/usr/bin/mkpasswd password
```

**4.** Once the installation has completed, we can proceed with configuring the Ircd-Hybrid software.

To do this, we will need to edit the **ircd.conf** file within the **/etc/ircd-hybrid** folder.

To do this, we can just run the following command in terminal to begin editing the file.

```
sudo nano /etc/ircd-hybrid/ircd.conf
```

**5.** Within this file. you will notice a considerable amount of different options that you can configure in here.

We will quickly change some of these configurations and explain why to give you an idea what they are for.

Thankfully the configuration file is pretty easy to understand due to a large number of comments.

## Within the **serverinfo {** block

### Find

```
name = "hybrid8.debian.local";
```

### Replace With

```
name = "pimylifeup.irc";
```

This line defines the name for your IRC server. You can set this to whatever you like. As an example for our tutorial, we will set this to **pimylifeup.irc**.

# Within the **serverinfo {** block

### Find

```
        description = "ircd-hybrid 8.1-debian";
```

### Replace With

```
        description = "Raspberry Pi IRC Server";
```

This line sets the servers description and will be what people see when they connect to your server.

### Find

```
        network_name = "debian";
        network_desc = "This is My Network";
```

### Replace With

```
        network_name = "pimylifeup";
        network_desc = "This is my Raspberry Pi IRC Network";
```

This line is for describing the name and description of the network that your server is on.

### Find

```
        max_clients = 512;
```

### Replace With

```
        max_clients = 128;
```

This line defines the maximum amount of people that can be connected to the IRC Server.

# Within the **operator {** block

### Find and remove

```
        #
```

This section requires uncommenting, remove the first layer of **#** if you see **##** only remove the first one.

### Find

```
        name = "sheep";
```

### Replace With

```
        name = "op";
```

This line defines the name of the operator group. We will use **op** as it is easiest to remember, you can set this to whatever you want.

### Find

```
        user = "*@192.0.2.240/28";
```

### Add Below

```
        user = "*@*";
```

This line will change it so that anyone connecting to the server can run the oper command. If you want to restrict this to local users only, you can try using something like "***@192.168.*.***".

## Within the **operator {** block

### Find

```
password = "xxxxxxxxxxxxx";
```

### Replace With

```
password = "REPLACE WITH YOUR ENCRYPTED PASSWORD";
```

Here we will want to replace the default password with the one that we encrypted earlier in the tutorial with the **/usr/bin/mkpasswd** tool.

Remember this is not the plain text version of your password, it is the scrambled form.

**6.** Now we can save and quit out of the file by pressing **Ctrl +X** then pressing **Y** and then **Enter**.

**7.** With that now done you can modify the message of the day (MOTD) for the IRC Server.

This message is fairly easy to modify as the **ircd-hybrid** software reads it from a file called **ircd.motd** that is located in the **/etc/ircd-hybrid** folder.

You can begin modifying this file by running the following command in terminal:

```
sudo nano /etc/ircd-hybrid/ircd.motd
```

For our tutorial, we will skip modifying this file, but basically whatever is in it will be displayed to a user upon connecting to the server.

**8.** Now that we have done our initial configuration of our Raspberry Pi IRC Server we will need to restart the server software.

We can quickly do this by running the following command in terminal.

```
sudo /etc/init.d/ircd-hybrid restart
```

## Testing our Raspberry Pi IRC Server

Now that we have configured our Raspberry Pi IRC Server we can now test to make sure it works.

We will go through steps on connecting to the server using an IRC client on Windows to give you an idea of how to utilize your server.

For Windows, we will be utilizing the popular mIRC client.

**1.** To start off first download and install the mIRC client, you can download it from there website by going to: http://www.mirc.com/

**2.** With mIRC installed, launch it. Once opened click **File -> Select Server** (Or **press Alt + E**)

**3.** A new window should now pop up, on here you want to **click** the "**Add**" button.

**4.** Another window should load up, this time you will need to fill out the **Description:** and the **Address:**

For "**Description:**" we chose to use **pimylifeup**, make sure you choose something that makes it stand out from the rest of the IRC servers.

For "**Address:**" we set this to the IP of our Raspberry Pi. If you don't know the IP of your Raspberry Pi type the command **hostname -I** into the terminal.

In our example, we will be using **192.168.0.143** as our IP address.

Once you have filled out this information, click the "**Add**" button.

**5.** Your server should now be added to mIRC's list of IRC servers.

It should also currently already be selected. If it isn't then you should look for your new addition, it will be under the name you set for "**Description:**" in our case, we would find it under the name **pimylifeup**.

Once you are sure you have, it selected, press the "**Select**" button.

**6.** You will finally be brought to one final screen before making the connection.

Here you will want to set your "**Nickname:**" to whatever you want.

Now before you jump ahead and press the "**Connect**" button make sure your server description name is next to "**Server:**"

Once you are sure it is correct, press the "**Connect**" button.

**7.** Finally, type in the channel name you want to use, this can be anything as long as it starts with a **#**, once typed in all you need to do is click the "**Join**" button.

If all has gone well, you should now be successfully connected to your Raspberry Pi IRC Server.

## OPing yourself on your Raspberry Pi IRC Server

**1.** To OP yourself on your IRC server you need to remember the group name and the unencrypted password that you set earlier in this tutorial.

Once you have both handy, connect to your IRC server.

**2.** Once connected to your IRC server and you are in a channel, type in the command below into chat.

Make sure that you replace **op** with your group name and replace password with the password you set earlier in this tutorial when you ran the **/usr/bin/mkpasswd** command.

```
/oper op password
```

**3.** You should see "**You are now an IRC operator**" appear in chat if you have been successfully made an operator.

Hopefully, by now you have a fully functioning IRC Server running on your Raspberry Pi, including the ability to be able to set yourself as an operator.

# Webmin

## Project Description

In this project we will be showing you how you can easily setup Webmin on your Raspberry Pi in a few simple steps.

For those who do not know what Webmin is, it is a excellent and easy to utilize web-based interface for system administration.

Webmin provides an easy to understand web frontend to many operating system internals allowing you to configure various system attributes with ease.

This software is incredibly helpful if you need to perform administrative tasks often.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

**Optional**
- USB Keyboard
- USB Mouse
- Raspberry Pi Case

# Installing Webmin

**1.** First off, make sure Raspbian is completely up to date by running the following commands on your Raspberry Pi.

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

**2.** Now enter the following two commands to install all of Webmin's required packages.

```
sudo apt-get install perl libnet-ssleay-perl openssl libauthen-pam-perl
sudo apt-get install libpam-runtime libio-pty-perl apt-show-versions python
```

**3.** Now run the command below to download the required deb file to the Raspberry Pi, this file contains the webmin package.

You may want to update the version number so that you you are downloading the latest version by visiting the webmin website here www.webmin.com/deb.html

```
wget http://prdownloads.sourceforge.net/webadmin/webmin_1.890_all.deb
```

**4.** Once the deb package file has downloaded to your Raspberry Pi we will want to use dpkg to install it to the actual system.

Utilize the following command to install the debian package we just downloaded, replace the version number if you changed the download link in the previous step. This may take some time.

```
sudo dpkg --install webmin_1.890_all.deb
```

**5.** Once Webmin has finished installing you will now be able to access it by goign to the Raspberry Pi's IP Address followed by the port **1000**. You can find out the Pi's IP address by using **hostname -I**.

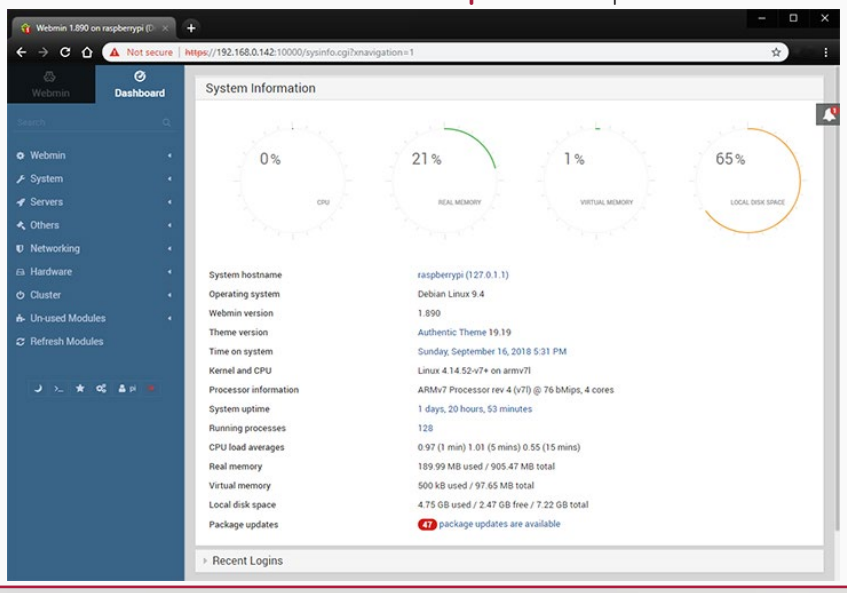
Below is an example of an address to access Webmin.

```
https://192.168.0.142:10000
```

**6.** You will receive a warning because the SSL will be invalid. It is safe to ignore this as we can't receive a valid SSL certificate on an IP address.

**7.** Enter your username and password for your Raspberry Pi's account to login into Webmin.

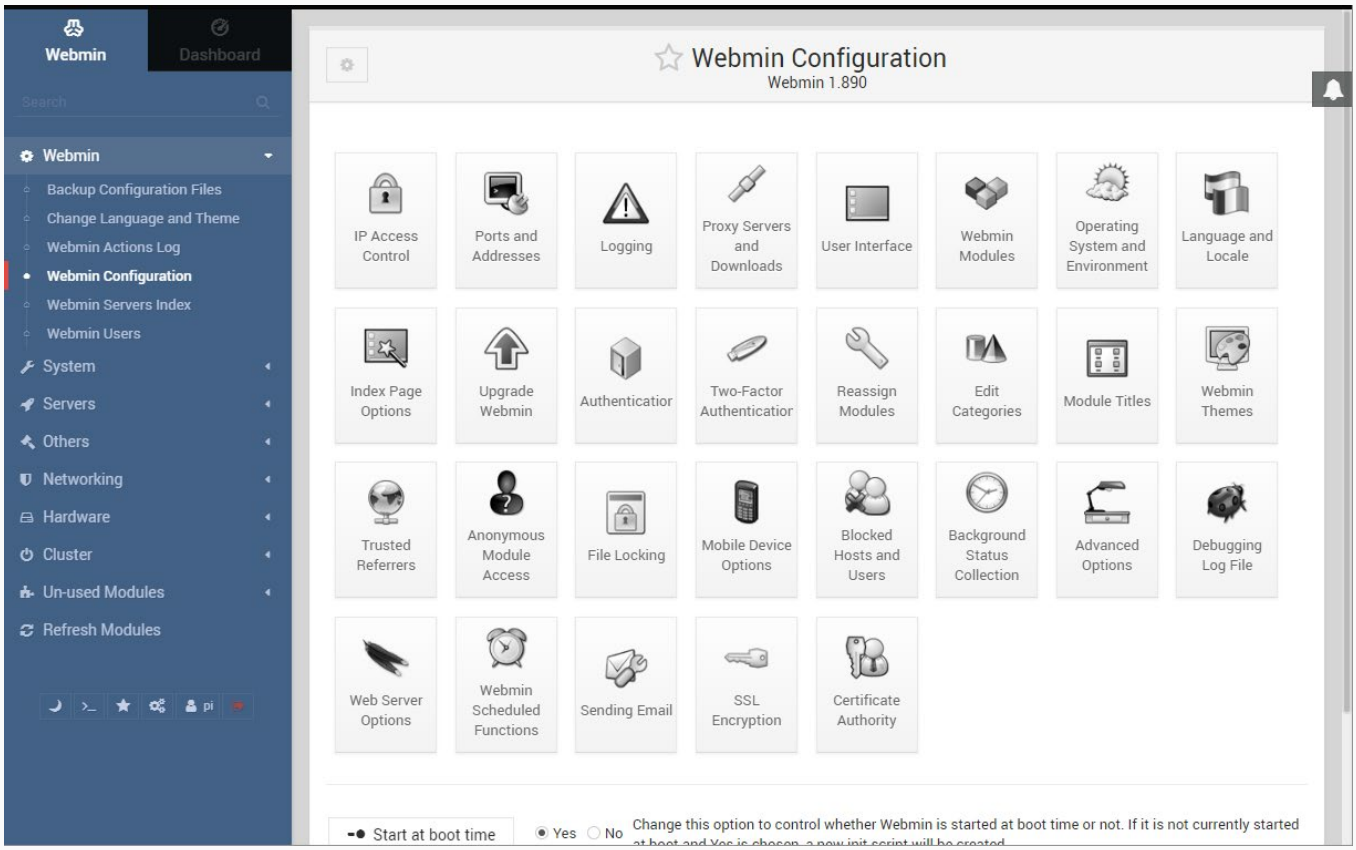
If you haven't changed these then the username will be **pi** and the password will be **raspberry**.



# Configuring Webmin

There are a ton of options that you can configure within the Webmin interface.

These options include changing your SSL encryption, adding modules, logging, managing users, managing software packages and so much more.



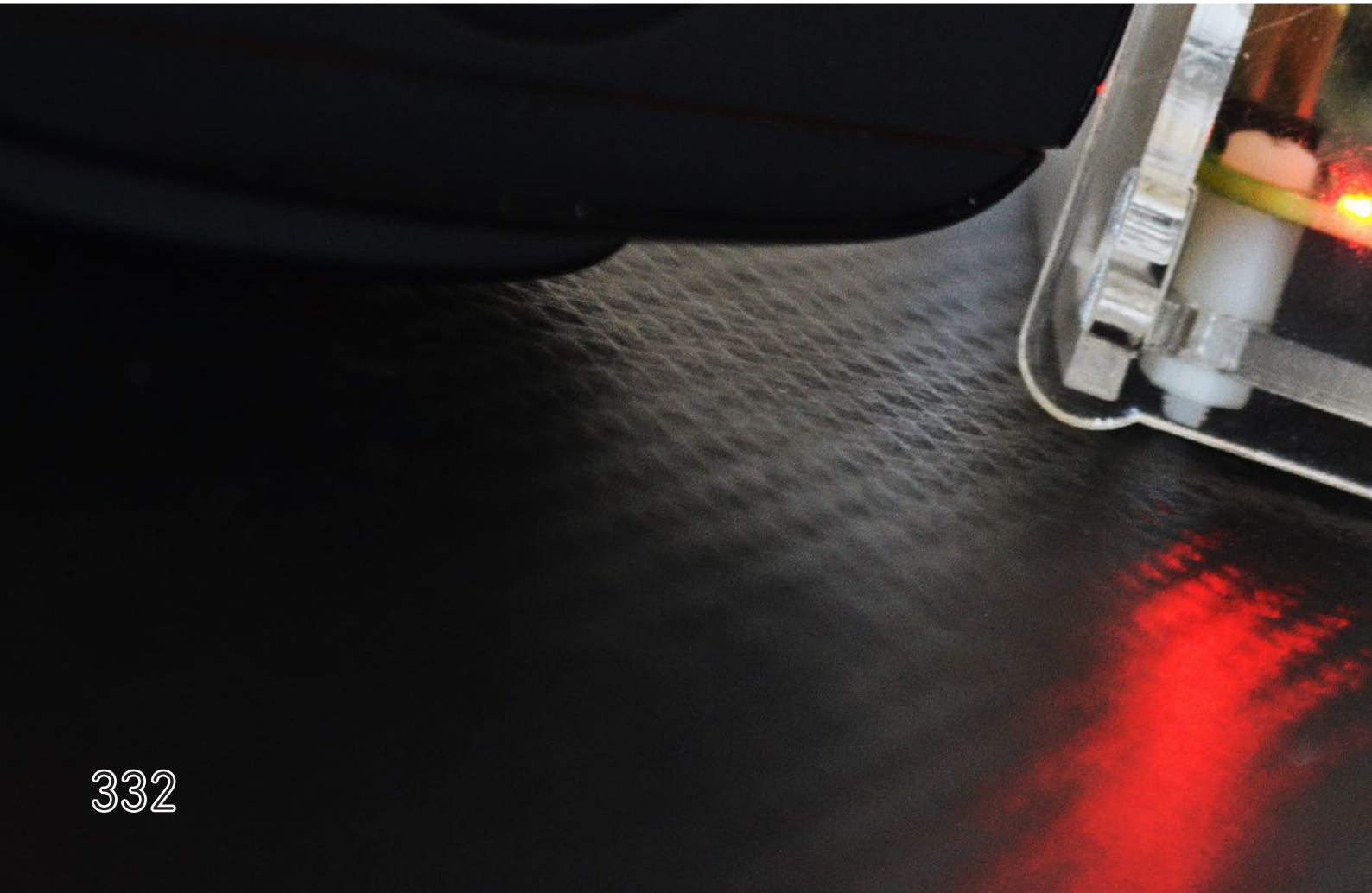

Since there are just so many options to configure we recommended checking out the Webmin wiki for more information.
You can find the official Wiki located here: https://doxfer.webmin.com/Webmin/Webmin_Modules.

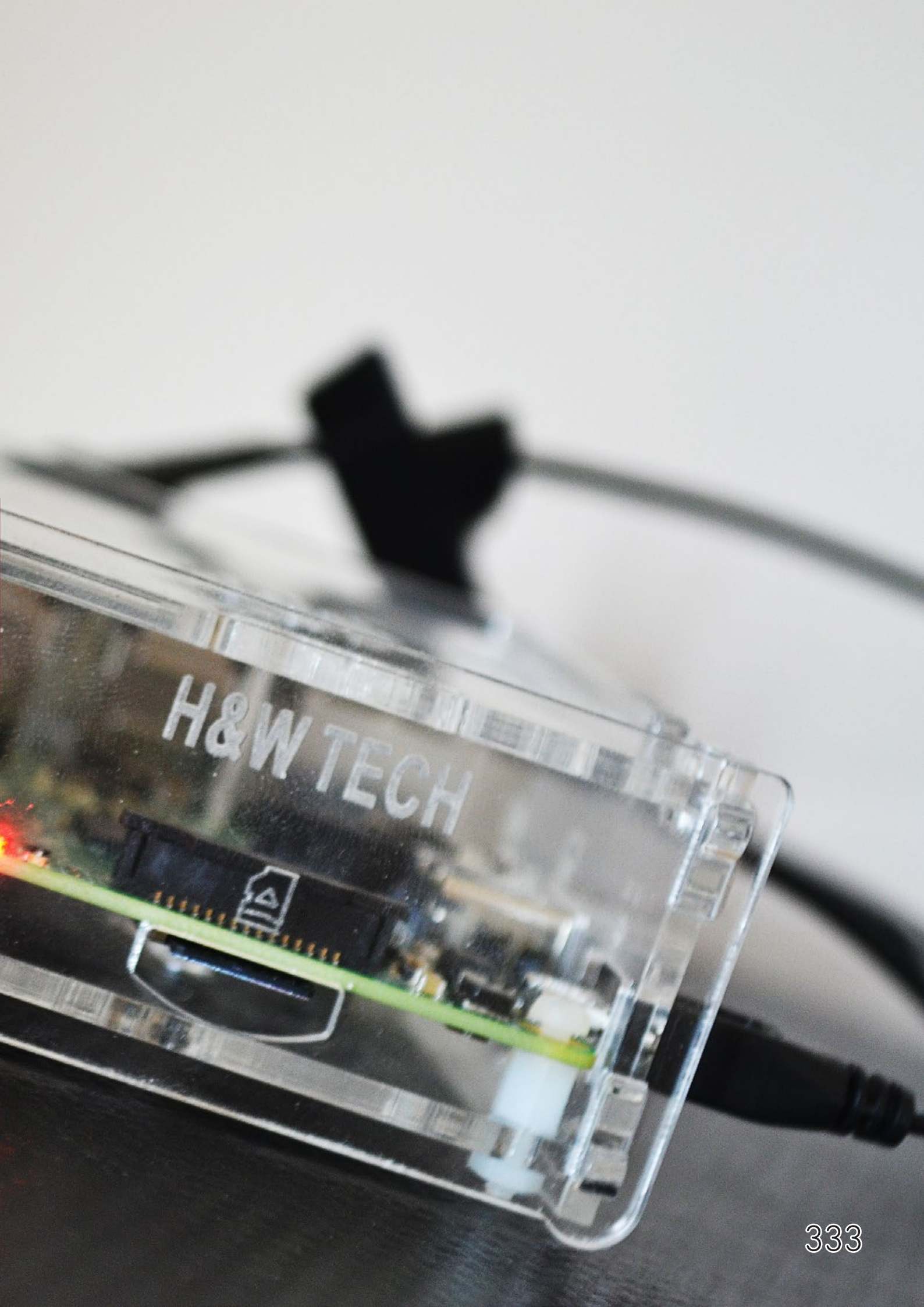
Out of the box, there isn't anything that really needs configuring as for the most part the software just works.

Two things you should look into doing however is to secure the admin access by changing the password and setting up two-factor authentication. These two things will help stop unwanted users gaining control over your Raspberry Pi.

Not secure
https://192.168.0.142
Dashboard
System Infor
Webmin
Raspberry Pi Model B+ V1.2
Raspberry Pi 2014
GPIO
HDMI
CAMERA
331

# Video & Photography
## Projects




H&W TECH

# Setting up a Webcam Server

## Project Description

This Raspberry Pi Webcam Server tutorial will take you through on how to have your very own Webcam that is visible on a webpage.

If you are after more of a security monitor system, then check out the **Raspberry Pi security camera tutorial** it features fully-fledged web streaming, motion detection, recording and multi-camera functionality.

We will be using a package called Motion if you want to learn more about it you can check out their website at http://www.lavrsen.dk/foswiki/bin/view/Motion/WebHome

This tutorial can be challenging to get working if you have hardware that differs to the versions used in this tutorial. If you find that you can't get this to work, then we do recommend going back and trying the MotionPie OS tutorial. However, it is a more demanding software package.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection
- USB Webcam or Raspberry Pi Camera

**Optional**
- Raspberry Pi Case
- External Hard Drive or USB Drive

# Installing Motion on Raspbian

Since FFmpeg is not included in any of the official Debian repos we need to grab a precompiled version, luckily MotionEye provides a pre-compiled package of FFmpeg that works on Raspbian.

**1.** Before we get to carried away with setting up the webcam server software, we must first make sure our Raspberry Pi is entirely up to date.

We can do this by running the following two commands on our Raspberry Pi.

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

**2.** With Raspbian now up to date, we can begin the process of installing MotionEye itself.

Just follow the following steps to set it up, we have split the steps into two sections based on which version of the Raspbian operating system that you are running.

## Raspbian Jessie

**3a.** We start off by removing libraries that may conflict with our new package, these may or may not exist on your system but its better safe than sorry to run this command.

```
sudo apt-get remove libavcodec-extra-56 libavformat56 libavresample2 libavutil54
```

**4a.** Lets, download and install a precompiled copy of FFmpeg off the MotionEye developers GitHub.

```
sudo wget https://github.com/ccrisan/motioneye/wiki/precompiled/ffmpeg_3.1.1-1_armhf.deb
sudo dpkg -i ffmpeg_3.1.1-1_armhf.deb
```

**5a.** Now we need to install the dependencies the motion software relies on, to do this type the following two commands into the terminal on your Raspberry Pi.

There are a few packages that will need installing.

```
sudo apt-get install curl libssl-dev libcurl4-openssl-dev libjpeg-dev libx264-142
sudo apt-get install libavcodec56 libavformat56 libmysqlclient18 libswscale3 libpq5
```

**6a.** With everything installed we can grab the latest version of the motion software and install it.

To do this run the following commands.

```
sudo wget https://github.com/Motion-Project/motion/releases/download/release-4.0.1/pi_jessie_motion_4.0.1-1_armhf.deb
sudo dpkg -i pi_jessie_motion_4.0.1-1_armhf.deb
```

## Raspbian Stretch

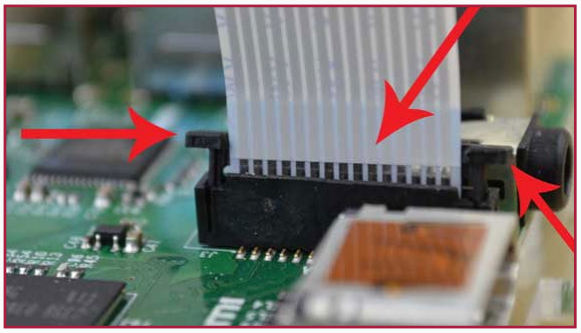
**3b.** Now we need to install the dependencies the motion software relies on, to do this type the following two commands into the terminal on your Raspberry Pi.

```
sudo apt-get install libmariadbclient18 libpq5 libavcodec57
sudo apt-get install libavformat57 libavutil55 libswscale4
```

**4b.** With those packages installed we can now grab the latest version of the motion software and install it. To do this run the following commands.

```
sudo wget https://github.com/Motion-Project/motion/releases/download/release-4.0.1/pi_stretch_motion_4.0.1-1_armhf.deb
sudo dpkg -i pi_stretch_motion_4.0.1-1_armhf.deb
```

# Setting up the Pi Camera hardware



**1.** To connect the Raspberry Pi camera to the Raspberry Pi, you will need to identify the **ribbon cable attachment** (Marked as Camera CSI on the diagrams).

**2.** Once you have found it, then press gently with two fingers on either side of the connector and pull up.

**3.** Doing this will open the port up and allow a cable to be inserted into it. Now insert the cable with the metal leads facing away from Ethernet port.

**4.** Once lined up and in gently press the connector back down and the cable will now be locked in place.

## Setting up the Pi Camera software

To get the Raspberry Pi camera to work with motion, we will need to do a few extra steps.

**1.** First make sure the camera is switched on within raspi config, enter the following command and then enable the camera. (You will need to restart once you have done this)

```
sudo raspi-config
```

**2.** Within this tool, you need to go to "**5 Interfacing Options**" then to "**P1 Camera**".

When asked if you want to enable the camera interface, select "**<Yes>**" then press **Enter**.

Once you have enabled the interface, you can press ESC to quit out of the tool.

**3.** Now with the camera interface enabled we need to modify the modules file to ensure that the Pi camera is made available to the motion package.

To begin editing the file enter the following line into the terminal on the Raspberry Pi.

```
sudo nano /etc/modules
```

**4.** Now within this file, we need to add the following line to the bottom of it.

This line tells the Raspbian operating system it needs to load in that particular module when it starts up.

If the line is already present, you don't need to add it again.

```
bcm2835-v4l2
```

**5.** Now save the file by pressing **Ctrl + X** then typing in **Y** and hitting **Enter**.

**6.** With that done we now need to restart the Raspberry Pi to ensure that it loads in this module. Not restarting means the module will never be loaded in.

```
sudo reboot
```

**7.** With these steps now completed you can continue with setting up the Motion software.

## Setting up Motion

Setting up Motion for a USB camera and the Raspberry Pi Camera is relatively simple in the latest version of Motion.

We will go through the steps on setting up Motion for use with both a USB webcam and the Raspberry Pi Camera module.

**1.** Let's begin by editing the configuration file for Motion.

We can open it with the following command:

```
sudo nano /etc/motion/motion.conf
```

**2.** In here there are a few things we want to change, go through everything in the list below and apply the changes.

If you are going to use the Raspberry Pi Camera, there is one extra line that you must change.

Don't forget you can use **Ctrl+W** to search the file for specific bits of text. Using the search makes it much easier to find and change lines.

Required (Don't include the text in brackets)

- stream_localhost off **(Allows non localhost connections to view the stream)**
- mmalcam_name vc.ril.camera **(Only for the Pi camera, remote the ; before it)**

Optional (Don't include the text in brackets)

- stream_maxrate 100 **(This will allow for real-time streaming but requires more bandwidth)**
- framerate 100 **(This allows 100 frames to be captured per seccond, allowing for smooth video)**
- width 640 **(This changes the width of the image displayed)**
- height 480 **(This changes the height of the image displayed)**

Once you have made all the changes, we need to then save and quit out of the file by pressing **Ctrl+X** then pressing **Y** and then **Enter**.

**3.** Now we need to setup up the daemon. To do this, we first need to edit the motion setting file.

We can ediy this file by running the following command on the Raspberry Pi.

```
sudo nano /etc/default/motion
```

**4.** Within this file, we need to find and change the following line, so it matches what we have below.

### Find

```
start_motion_daemon = no
```

### Change to

```
start_motion_daemon = yes
```

**5.** Once you have successfully made that change, we need to save the file by pressing **Ctrl + X** then **Y** and then finally **Enter**.

## Setting up Motion

**6.** Now we can start up the daemon by utilizing the following command.

Before we run this command, you should make sure that your camera is plugged in properly.

```
sudo service motion start
```

**7.** If you ever need to stop the motion service, you can just type the following command into the terminal.

```
sudo service motion stop
```

**8.** With the motion service running you should be able to check out the Webcam Stream at the IP address of your Raspberry Pi.

So in your browser go to the following address, making sure you replace the IP with your own:

```
192.168.1.103:8081
```

**9.** If the webpage isn't loading, you can try restarting the service by running the following command on your Raspberry Pi.

```
sudo service motion reload
```

## Setting up outside external access for Motion

To provide external access to your Raspberry Pi Webcam Server, we will need to change some settings on the router.

However, all routers are designed differently so you will probably need to look up your brand.

Hopefully, this will at least give you some idea of what you must do.

Below is an example of what I did on mine to get it to work. My router is an AC1750 TP-Link Router.

**1.** Go to the Router admin page (This will typically be 192.168.1.1 or **192.168.254**)

**2.** Enter the username and password. Default typically is admin & admin.

**3.** Once in go to **Forwarding**->**Virtual Server** and then click on **add new**

**4.** In here enter:

- **Service Port:** In this case 48461
- **IP Address:** **192.168.1.105** (Address of the Raspberry Pi)
- **Internal Port:** We want this to be the same as the service so **8081**
- **Protocol:** All
- **Status:** Enabled

**5.** These settings will route traffic destined for port **48461** to the webcam servers IP address.

**6.** You should now be able to connect to the Raspberry Pi webcam stream outside your network. You may need to restart the router for changes to take effect.

Logitech
H&W

# Networked
## Security Camera

### Project Description

In this project, we are going to make a Raspberry Pi security camera just using the standard Pi camera such as the one we used in the time-lapse tutorial.

Doing this project is a cost-effective way of getting a security camera up and running that you can view over the network and have it so, it is motion activated!

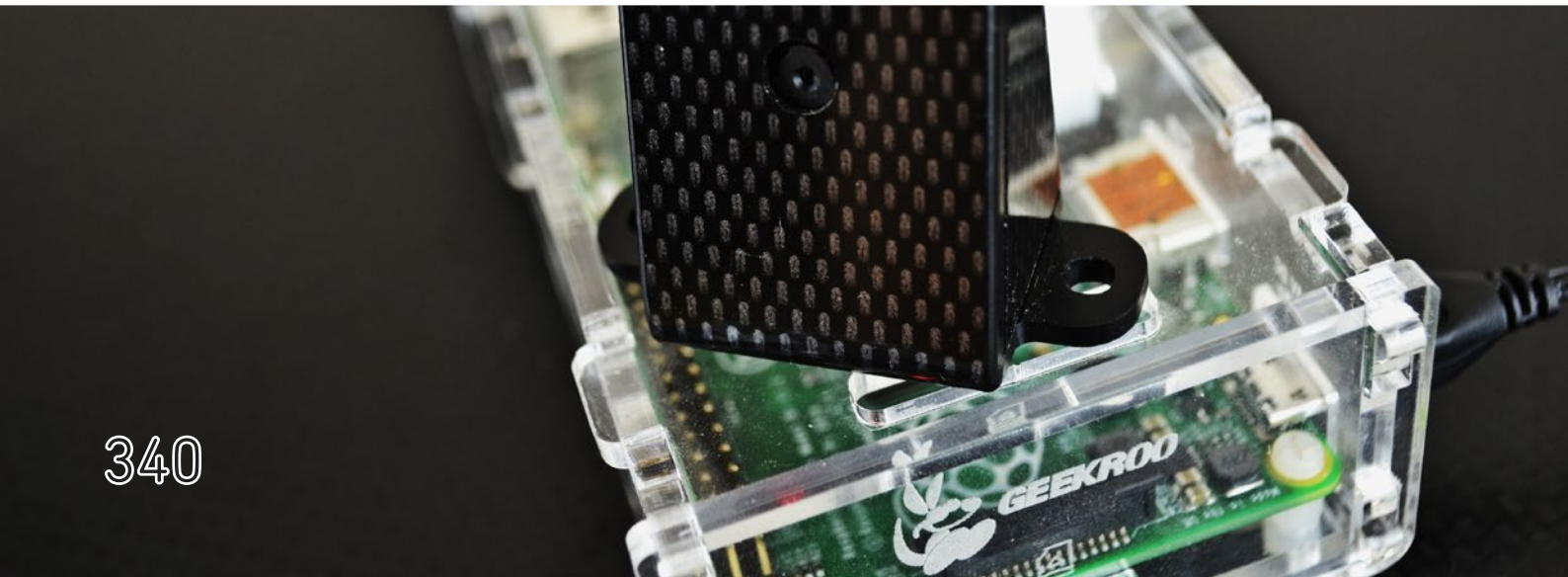
Firstly, before we do anything, we need to have a Raspberry Pi camera. In this tutorial, I use the normal IR camera, and it works just fine however if it gets dark it can't see at all.

However, you can also purchase a non-IR camera that will suit the purposes of being a security camera better as it can deal with darkness.

### Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection
- Raspberry Pi Camera or USB WebCam

**Optional**
- Raspberry Pi Case
- USB Keyboard
- USB Mouse



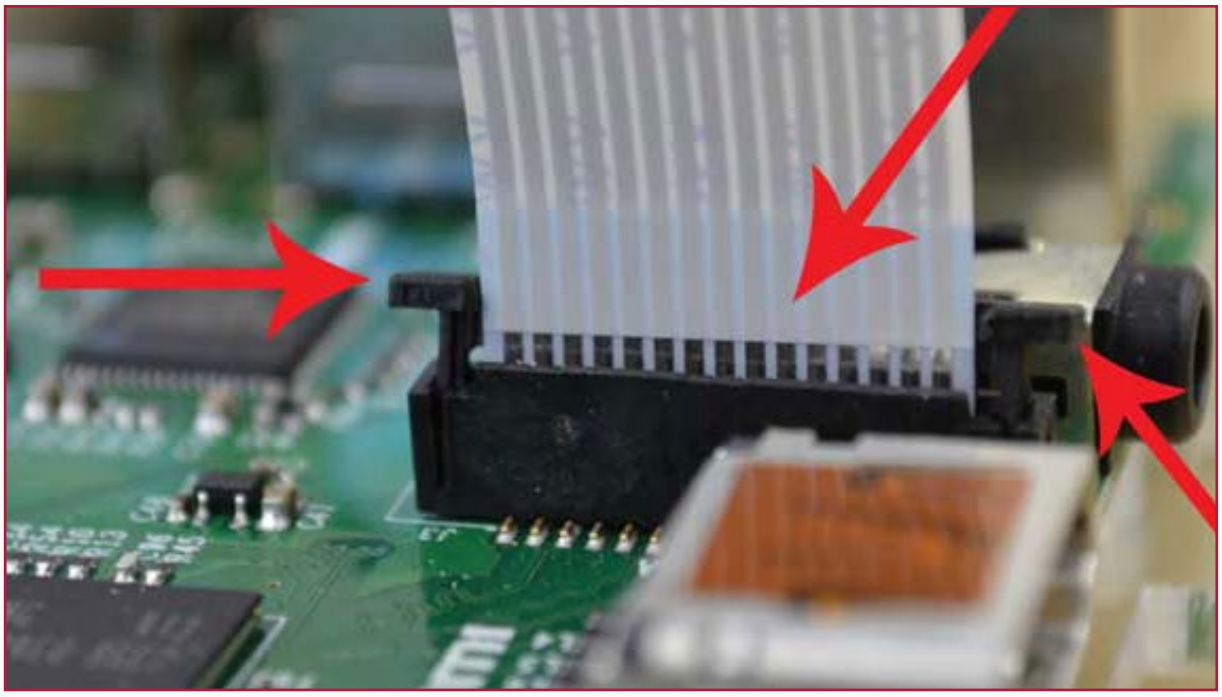
## Installing the Camera

Probably the most important step is connecting the Raspberry Pi camera correctly to your Raspberry Pi.

It is straightforward to set it all up, but for anyone new to this kind of stuff, they will find this section helpful.

## Hardware Setup

**1.** To connect the Raspberry Pi camera to the Raspberry Pi, you will need to identify the **ribbon cable attachment** (Marked as Camera CSI on the diagrams).



**2.** Once you have found it, then press gently with two fingers on either side of the connector and pull up.

**3.** Doing this will open the port up and allow a cable to be inserted into it.

Now insert the cable with the metal leads facing away from Ethernet port.

**4.** Once lined up and in gently press the connector back down and the cable will now be locked in place.

## Installing MotionPie

We settled on using MotionPie as it seems to be an all in one solution for what I require and it also didn't involve as much fiddling around to get it to work.

# Step 1 - Formatting your SD Card

First off you should do when installing a new image to an SD card is format it correctly.

The following steps will show you the best way to format your SD Card for the Raspberry Pi:

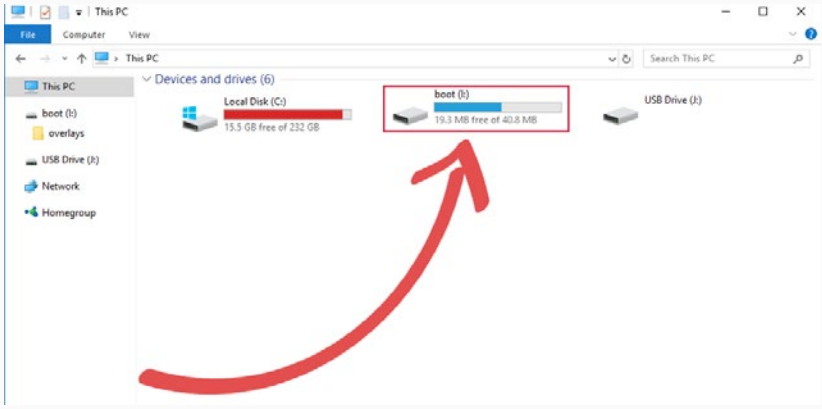
**1.** To begin you need a tool to format your SD Card. You can obtain one of the most robust ones from here: **https://www.sdcard.org/downloads/formatter_4/**

Once you are on the website will find a download link for the SD Formatter.

**2.** Download and install the SD Card Formatter from the SD Card association.

**3.** With the SD Formatter installed to your computer, insert your SD card and check the drive letter that is allocated to it, e.g., **G:/**.

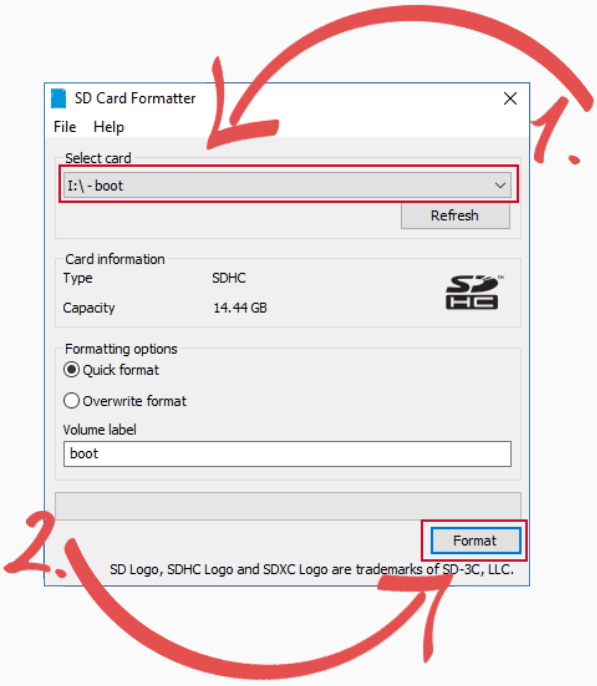
Make sure that you have the correct drive letter otherwise you could accidentally format the wrong device.




**4.** With your SD Card now plugged into your computer, launch the SD Formatter software that we installed earlier.

First set the drive letter to the correct one **(e.g., G:/)**.

Once you have everything set correctly, press the "**FORMAT**" button.



## Step 2 - Installing MotionEyeOS

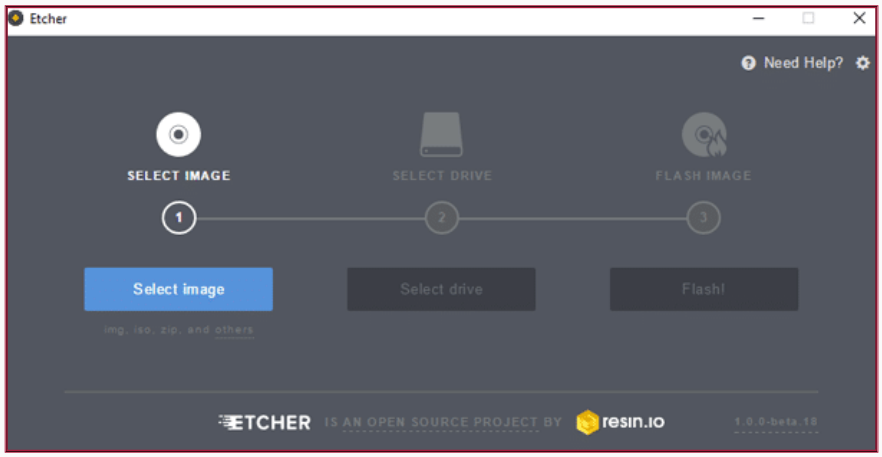
**1.** First head over to MotionEyeOS website at **https://github.com/ccrisan/motioneyeos/releases** and download the latest zip file.

Once downloaded, unzip the file so, you are left with a **.img** file.

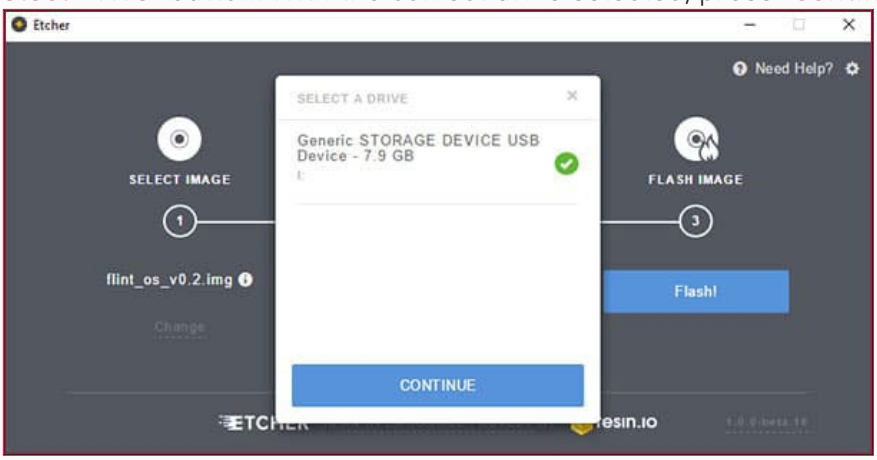
**2.** Next, you will need some tools to be able to write the image to the SD card.

For this we will use a tool called Etcher, you can download at https://etcher.io/.

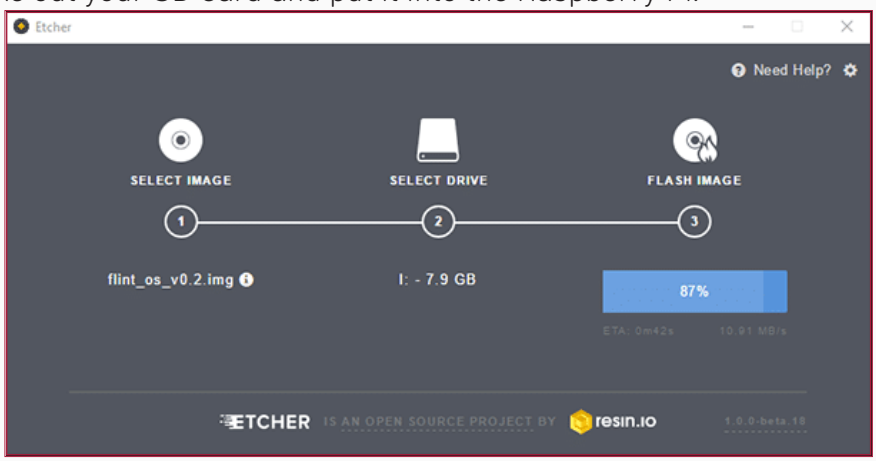
**3.** Open Etcher and press the "**Select Image**" button. Browse to the image you downloaded in **Step 1** and select it.




**4.** Next click the "**Select Drive**" button. With the correct drive selected, press "**Continue**".




**5.** Finally, click the "**Flash!**" button to begin the flashing process. This can take some time. Once completed, take out your SD Card and put it into the Raspberry Pi.



## Installing MotionEye on Raspbian

**1.** Since ffmpeg is not included in any of the official Debian repos we need to grab a precompiled version, luckily the package maintainer of MotionEye provides a pre-compiled package of ffmpeg that works on Raspbian.

We stuff off by removing libraries that may conflict with our new package, these may or may not exist on your system but its better safe than sorry to run this command.

```
sudo apt-get remove libavcodec-extra-56 libavformat56 libavresample2 libavutil54
```

You can download and install these packages by issuing the following commands into terminal.

```
sudo wget https://github.com/ccrisan/motioneye/wiki/precompiled/ffmpeg_3.1.1-1_armhf.deb
sudo dpkg -i ffmpeg_3.1.1-1_armhf.deb
```

**2.** Now we need to install the dependencies the motion software relies on.

Please type this command into the terminal, it is a rather long command as there are a few packages that will need installing.

```
sudo apt-get install python-pip python-dev curl libssl-dev libcurl4-openssl-dev
libjpeg-dev libx264-142 libavcodec56 libavformat56 libmysqlclient18 libswscale3
libpq5 v4l-utils
```

**3.** With those packages installed we should now grab the latest version of the motion software and install it, we can do that with the following commands.

```
sudo wget https://github.com/Motion-Project/motion/releases/download/release-4.0.1/pi_jessie_motion_4.0.1-1_armhf.deb
sudo dpkg -i pi_jessie_motion_4.0.1-1_armhf.deb
```

**4.** Now that is all done, we can begin the installation of motioneye, this will automatically pull the Python dependencies it requires. Please note, this can take some time.

```
sudo pip install motioneye
```

**5.** Once that has completed we can prepare the configuration directory by copying over the sample config file.

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

**6.** Prepare the media directory, this is the directory the camera shots will be streamed to.

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

**7.** We need to now add an init script to make sure the software starts up when the Raspberry Pi is started.

```
cp /usr/local/share/motioneye/extra/motioneye.systemd-unit-local /etc/systemd/system/motioneye.service
```

**8.** With that done, we need to now reload the daemon and start up motioneye

```
sudo systemctl daemon-reload
sudo systemctl enable motioneye
sudo systemctl start motioneye
```

## Installing MotionEye on Raspbian

**9.** Now for the next few steps, we need to know the Raspberry Pi's local IP address, we can grab that with the following command:

```
hostname -I
```

**Note:** If you ever want to upgrade to the latest version of motionEye you can run the following commands:

```
sudopip install motioneye --upgrade
systemctl restart motioneye
```

## Setting up the Pi Security Camera

Once the Pi has booted you will need to do the following:

**1.** First, we will need the IP or hostname so we're able to connect to the Pi. If you installed MotionEye onto a pre-existing installation of Raspbian, you should have already gotten the IP address of the Raspberry Pi during the tutorial. So, you can skip onto **step two**.

If you're using Windows you can just go to network on the right-hand side in the File Explorer.

Here you should see a computer name with something like MP-E28D9CE5, this is the hostname of the MotionEyeOS installation on your Raspberry Pi, and we can use that to connect to it.

However, if you are not on a Windows computer, or it won't show up in the network screen, the next best way to find out the IP address through your router.

On your routers page there should be a page that lists all the currently connected devices and their respective local IP addresses. We can't give an exact guide sadly since every router is different.

**2.** Once you have the hostname or IP address you can now try going to it in your web browser.

For example, if your Raspberry Pi's hostname was **MP-E28D9CE5** you can try going to http://MP-E28D9CE5,

You do the same if you have the IP address, for example, the IP address of my Raspberry Pi is **192.168.1.105** so, therefore I will go to **http://192.168.1.105**

**3.** You should now have the Motion Pie interface up.

To log in as the admin go to the key symbol in the upper left corner.
The **username** is **admin**, and the **password** is **blank**, this can be changed later.

**4.** You can access all the setting for the camera stream here.

If you're interested in altering these settings, keep reading as I explain them as much as possible over the next few pages.

## How to setup multiple network security cameras

If you want to run more than one Raspberry Pi cameras, it is easy to set this up, so you have all the streams under in one window. You can even add a stream that has been set up using the Raspberry Pi Webcam server tutorial that you can find later in this book.

1. First, click on the three lines with dots on them in the upper left-hand corner.
2. Now up in the upper left-hand corner and click on the drop-down box and select add camera.
3. In here you have four settings to set up.
4. **Device:** This setting allows you to select where the camera is located(network/local) and type. (E.g. motionEye, MJEPG camera)
5. **URL:** This setting is the URL to the other network camera. E.g., **http://othercamera:8080**
6. **Username:** This setting is the username of the camera device. (If no username/password required then leave the fields blank)
7. **Password:** This setting is the password for the username chosen above.
8. **Camera:** Select the camera you wish to add.

In the example below camera1 (Pi Camera) and camera2 (USB WebCam) are connected to the Pi running motionpie while camera3 is coming from a different Pi that was set up using the webcam server tutorial.

## Connecting to the surveillance outside your network

Now that you have your Raspberry Pi security cameras setup it might be worth considering allowing access to the central Pi so you can monitor your cameras elsewhere.

To do this just head over to my guide on how to set up port forwarding and also to setup dynamic DNS, you can find both guides earlier on in the book in our Getting Started section.

A few important bits of information you will need for the setting up the port forwarding.

- The IP of your Raspberry Pi, for example mine is **192.168.1.108**
- Internal port is **80**

Ensure you also have set up passwords on both the admin and the surveillance user to help avoid unwanted visitors.

Once you have finished setting it up you should now be able to connect using your external IP address such as **XXX.XXX.XXX.XXX:80** (**80** should be changed to something else, I would recommend changing it to avoid easy access for unwanted visitors)

# Configuring the Settings in MotionPie

## General Settings

In here you can set the administrator username and password. This account will have access to all the settings you are looking at.

Surveillance username and password can also be set in here this can be used just to access the camera interface.

To view all the settings available set the show advanced settings to on.

## Wireless Network

Turn this on if you plan to connect to the network via a wireless dongle. There are two things you will need to fill in here.

1. **Network Name** – Enter the network name/SSID you wish to connect to in here.
2. **Network Key** – Enter the network password/network key in here.

Once done you should be able to disconnect the Ethernet cord and remain connected to the network.

## Video Device

Under this menu, you're able to set certain settings regarding the Raspberry Pi camera device.

1. **Camera Name:** Set this to whatever you would like the camera to be named. For example, "**kitchen**" would work well for a camera in a kitchen.
2. **Camera Device:** You're unable to edit this one but this is the device name of the camera.
3. **Light Switch Detection:** Enable this if you want sudden changes such as a light being switched on not to be treated as motion. (This will help prevent false positives)
4. **Automatic Brightness:** This will enable automatic software brightness, this means the camera software will automatically adjust the brightness. You don't need to activate this if your camera already handles this.
    - In here you can also change the brightness, contrast and, saturation of the camera feed.
5. **Video Resolution:** Here you can set the video resolution of the camera. The higher the resolution, the more room it will take up and the more bandwidth it will need to use to stream the footage. I set mine to 1280×800, and that seems to work perfectly fine.
6. **Video Rotation:** You can rotate your video from the Raspberry Pi security if you're finding that it is looking the wrong way.
7. **Frame Rate:** This sets the number of frames that will be sent be every second. The higher this is, the smoother the video but again this will increase the storage used and bandwidth.

## File Storage

Under this menu, you can specify where you would like the files stored for the Security Camera.

This setting can be a custom path on the Pi, the predetermined path, or the network path.

## Text Overlay

In here you can set the text overlay on the output of the camera. By default, the left text reads the camera name and the right read the time stamp (Today's date and current time).

## Video Streaming

This menu you're able to set the video streaming options, this is the video you see in the browser.

- **Streaming Frame Rate:** This setting is the same as mentioned above under video device.
- **Streaming Quality:** You can reduce the video streaming quality. This setting is good to reduce if you need to access the camera on a low bandwidth device often.
- **Streaming Image Resizing:** Enable this if you want MotionPie to resize the images before being sent to a browser. (Not recommended on a Pi)
- **Streaming Port:** This is the port that the device will listen to for connections looking to view the stream. E.g., **http://motionpie:8081**
- **Motion Optimization:** This will reduce the frame rate whenever no motion is detected. This setting will save you bandwidth.

You can also see three URLs that can be used to access different footage. These URLs are very important if you have multiple cameras per Pi as each camera will have a unique port that you listen to the stream on.

## Still Images

Here you can set the Raspberry Pi security camera to take still images whenever motion is triggered, during specific intervals or all the time.

## Motion Detection

In here you can activate the Raspberry Pi security camera motion detection that is included in the software. You can adjust the settings here so that you can get better motion detection.

## Motion Movies

In here you can set the Pi to record movies whenever motion is detected.

## Motion Detection

You're able to set up email notifications, webhook notifications or even run a command whenever motion is detected. This setting will allow you to be notified whenever activity is detected on the cameras, perfect if they are monitoring areas with low traffic.

## Working Schedule

Here you can set the days and the hours of operation you would like the system to be monitoring (If you leave this off then it is 24/7). Use this option if you only need it running during specific hours.

GEEKRO

# Digital
## Photo Frame

### Project Description

The Raspberry Pi photo frame is a perfect setup for anyone who wants to display photos of their loved ones, memories, or anything else might want to display.

This project is straightforward and shouldn't take you too long to set up. I pretty much only cover the software side of things in this tutorial there are some cool frame designs/ideas out there if you want to extend this further.

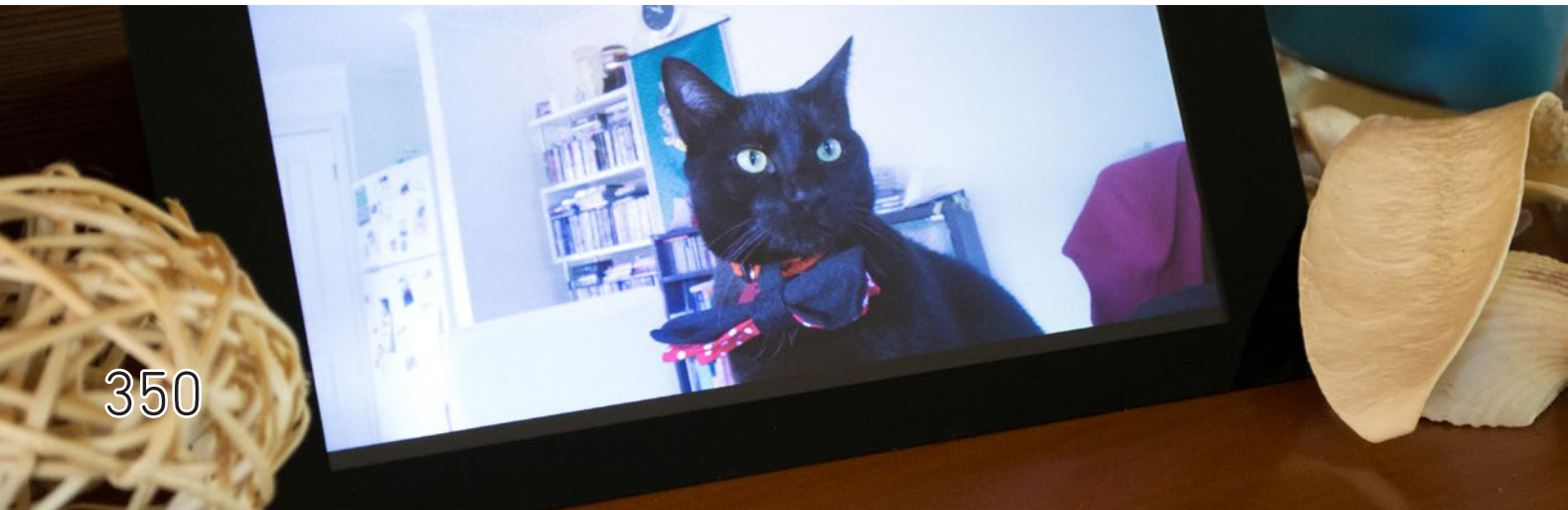
For this project, I am just using the official Raspberry Pi touchscreen that I purchased earlier this year. However, you can use pretty much any screen that you can get your hands on.

Just make sure you can hook it up to the Pi either over the DSI port, HDMI port or using a custom HAT.

### Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection
- External Hard drive or USB Drive
- Touch screen or LCD Sceen

**Optional**
- Raspberry Pi Case
- USB Keyboard
- USB Mouse



## Setting Up the Raspberry Pi Photo Frame Software

To get this all setup correctly we are going to need to do a bit of setup on the software side of things.

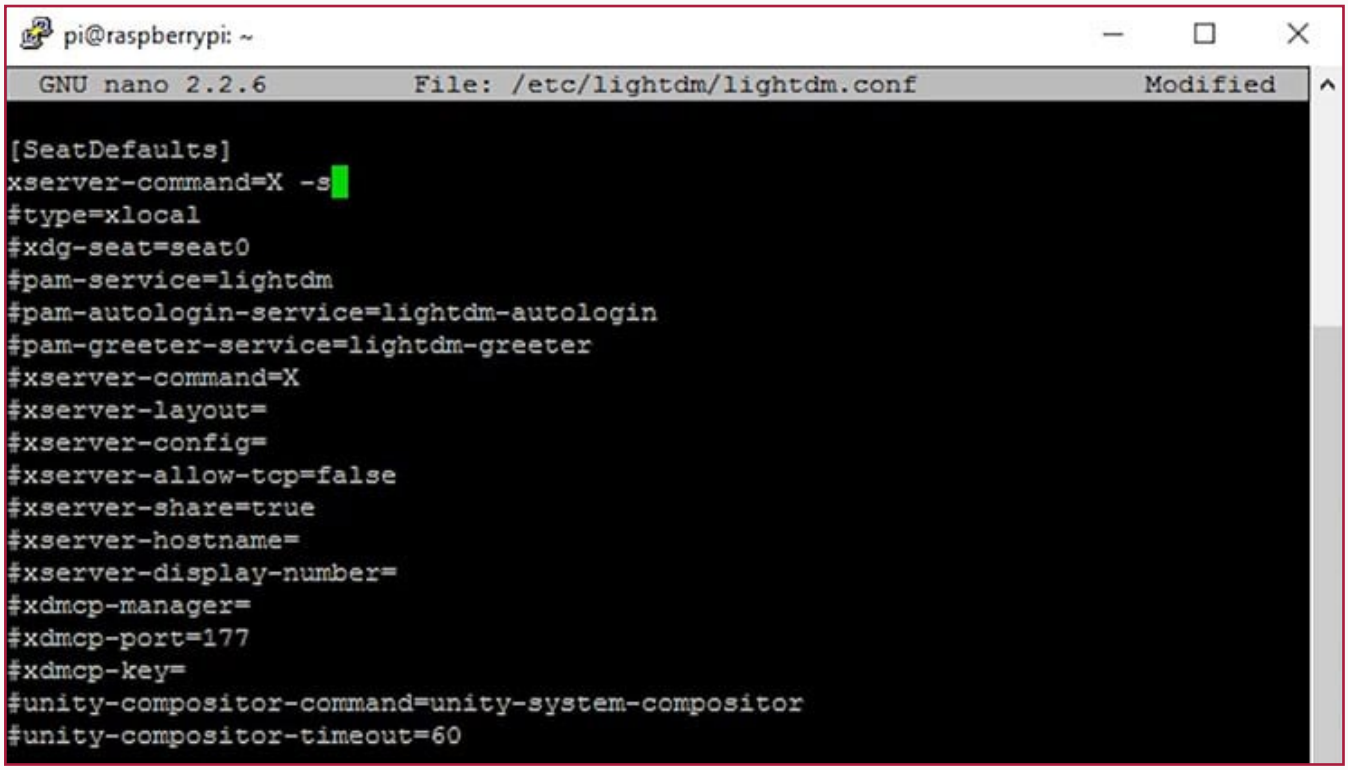
Firstly, we want to prevent the screen from going blank. This is a power saving setting on the PI.

**1.** To turn off the screen automatically being blanked out and turned off, open up the lightdm.conf file

```
sudo nano /etc/lightdm/lightdm.conf
```

**2.** Now in here add the following line anywhere beneath the **[SeatsDefaults]** line.

```
xserver-command=X -s 0 –dpms
```




**3.** Save and exit by pressing **Ctrl+X** and then pressing **Y** and then pressing **enter**.

**4.** Now reboot the Raspberry Pi and the screen should no longer switch off after 10 minutes of inactivity.

To reboot simply run the following command on the Raspberry Pi:

```
sudo reboot
```

## Setting Up the Photo Frame Software

To set up our slideshow, we are going to use the Feh package. Feh is an image viewer and image cataloguer.

It is a fast image viewer that doesn't get bogged down with substantial GUI dependencies. I chose this as it was the most lightweight package that worked without any vast complications.

**1.** To install the package using the following line:

```
sudo apt-get install feh
```

**2.** Now to test that it works enter the command below as a single line in terminal. We have split it in to two seperate lines to make it easier to read.

Replace **/media/NASHDD1/test** with the directory that contains all your image.

```
DISPLAY=:0.0 XAUTHORITY=/home/pi/.Xauthority /usr/bin/feh --quiet --preload --random-
ize --full-screen --reload 60 -Y --slideshow-delay 15.0 /media/NASHDD1/test
```

**3.** Now we can use short tags to make this command a bit shorter.

You can read more about all the flags you can use over at the feh manual page.

```
DISPLAY=:0.0 XAUTHORITY=/home/pi/.Xauthority /usr/bin/feh -q -p -Z -F -R  60 -Y
-D 15.0 /media/NASHDD1/test
```

**4.** Now as you will notice this locks up the command line bar.

To fix this just add the **&** after the command and the script/process will launch in the background.

**5.** So now let's store this in a simple script file.

This way you can add or change it later. To make the file enter the following command:

```
sudo nano /home/pi/start-picture-frame.sh
```

**6.** In here enter the following lines: Remember to replace **/media/NASHDD1/test** with the directory that contains all your image.

```
#!/bin/bash
DISPLAY=:0.0 XAUTHORITY=/home/pi/.Xauthority /usr/bin/feh -q -p -Z -F -R  60 -Y -D 15.0 /media/ASHDD1/test
```

Save the file by pressing **Ctrl+X** and then pressing **Y** and then **Enter**.

**7.** Now that's done you can test it by running the following command **(Ctrl+X)**:

```
bash /home/pi/start-picture-frame.sh
```

**8.** Finally, let's have it start at boot. Now it is essential that you have SSH enabled so you can access the PI remotely as you will lose access to the GUI/Screen.

So, make sure you have done this before setting it to launch at boot up.

**9.** To do this open up the **rc.local** file by entering the following command.

```
sudo nano /etc/rc.local
```

## Setting Up the Photo Frame Software

**10.** Add the following lines before the "**exit 0**" line in this folder

```
sleep 10
bash /home/start-picture-frame.sh &
```

Save the file by pressing **Ctrl+X** and then pressing **Y** and then **Enter**.

**11.** If you ever need to kill the process because you want to be able to access the screen then enter the following line into the terminal over SSH.

```
sudo pkill feh
```

You should now have your very own slideshow of pictures going. If you end up with any troubles, then double check all the steps and look for any errors.

# Time Lapse

## Project Description

Setting up your Raspberry Pi to take a time-lapse is extremely simple, all you require is the Raspberry Pi camera and the Pi itself to get started.

Utilizing a few simple commands and scripts, you can have a time-lapse tool up and running!

## Equipment

**Required**
- Raspberry Pi
- Raspberry Pi Camera
- 4 GB (Micro) SD Card (8 GB+ Recommended)
- Raspberry Pi Power Supply

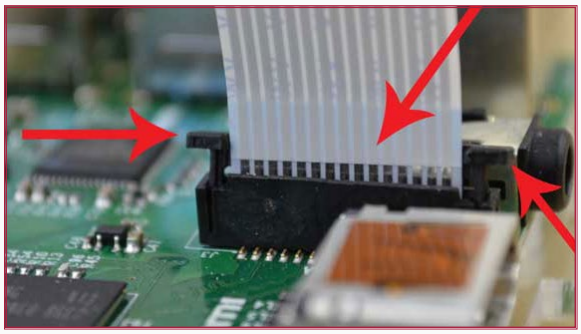
**Optional**
- USB Keyboard
- USB Mouse
- HDMI Cable
- Raspberry Pi Case

## Installing the Camera

Probably the most critical step is connecting the Raspberry Pi camera correctly to your Raspberry Pi.

It is pretty straightforward to set it all up, but for anyone new to this kind of stuff, they will find this section helpful.

## Hardware Setup



**1.** To connect the Raspberry Pi camera to the Raspberry Pi, you will need to identify the **ribbon cable attachment** (Marked as Camera CSI on the diagrams).

**2.** Once you have found it, then you need to press gently with two fingers on either side of the connector and pull up.

**3.** Doing this will open the port up and allow a cable to be inserted into it. Now insert the cable with the metal leads facing away from Ethernet port.

**4.** Once lined up and in gently press the connector back down and the cable will now be locked in place.

## Software Setup

To start this Raspberry Pi Time Lapse project, you will need to have a copy of the Rasbpian operating system. You will also have needed to of activated the camera module from within the raspi-config tool.

If you haven't done so already, the following steps will show you what to do to activate the camera module.

**1.** Into terminal, type the following command to load up the **raspi-config** tool:

```
sudo raspi-config
```

**2.** While in the raspi-config tool, use the **up** and **down arrow keys** to navigate to **Enable Camera**, and press **Enter** to select the menu.

**3.** To enable the camera module, all you need to do is select **Yes**, and press **Enter** Once done, you can press **Esc** to get out of the raspi-config menu.

To ensure the changes are properly loaded in, restart the Raspberry Pi by using the following command:

```
sudo reboot
```

## Testing the Raspberry Pi Camera

Before we get too far ahead of ourselves, we should test the camera has been setup correctly.

**1.** Begin by making sure your Raspberry Pi has started up successfully.

**2.** If you are on the GUI launch up the terminal and enter the following command, this command tells the Raspberry Pi Camera to take a photo.

```
raspistill -o cam.jpg
```

**3.** Now to make sure **raspistill** successfully took a picture, use the following command to check for all files in the current directory. You should be able to see a file called cam.jpg.

```
ls
```

**Note:** You can verify the image has been taken properly by viewing this file on your computer, grabbing the file over FTP, or just viewing the image in the Raspberry Pi itself.

**4.** If the image is upside down, then your camera is not up the correct way.

You can fix this problem easily by using a couple of parameters for the **raspistill** program.

We can get it to take a flipped image by using the following commands:

```
raspisstill -hf -vf –o cam2.jpg
```

**Note:** **-hf** flips the image horizontally, **-vf** flips the image vertically.

## Writing a basic script

**1.** To start off writing our script need to create a directory to store all our time lapse related scripts and files. We do this be using the following command:

```
sudo mkdir /home/pi/timelapse
```

**2.** Now change the current directory to our new time lapse folder, we will need to do a few things within this folder.

```
cd /home/pi/timelapse
```

**3.** Now we need to create a bash script called **camera.sh** using a text editor such as nano. You can do this by entering the following into terminal:

```
sudo nano camera.sh
```

**4.** Within this file, type in the following script:

```
#!/bin/bash
DATE=$(date +"%Y-%m-%d_%H%M")
raspistill -o /home/pi/timelapse/$DATE.jpg
```

What this basically does, is get the current date and time, then calls raspistill to take the image and names it after the date and time it grabbed.
**Note:** don't forget to add **-hf -vf** in front of raspistill if you require the output image to be flipped.

**5.** Test the script by using the following command in the terminal:

```
bash camera.sh
```

Next, we will go into setting up a schedule to run our script, and therefor create an actual time lapse.

## Schedule the script to take pictures

The script above is now complete, and while you can make further adjustments to it, we will keep it basic for this guide.

Next, we will need to schedule the script to run so we can run this over a certain amount of time allowing us to capture hundreds or thousands worth of pictures automatically.

**1.** Firstly we will need to modify Linux's scheduler service, that is the crontab, we can edit this by using the following command:

```
sudo crontab -e
```

**Note:** If you have never used cron it will prompt you to select a text editor. We recommend using Nano. It is one of the easiest text editors to use and get used to for Linux.

**2.** Once in you should scroll down to the bottom where you will see some column headers that look like below.

These are your guide for setting a cron job.

```
# m h dom mon dow   command
```

**Note:** For those who are interested, m means minute, **h** means **hour**, **dom** means **day of month**, **mon** is just the **month**, **dow** is the **day of the week**. Using ***** on any of these fields means that it means, every day, week, month, or minute

**3.** To schedule the time lapse to go off every minute of every day add the following line below the column headers.

You can change this to whatever you would prefer.

```
* * * * * sh /home/pi/timelapse/camera.sh 2>&1
```

**4.** Once done you can exit and save by pressing, **Ctrl+X** and then pressing **Y** to save, and then **enter** to confirm.

**5.** You should then get the following message upon quitting out of the crontab.

```
crontab: installing new crontab
```

**6.** If you use the following command within our time-lapse folder, you should see the files being added as the time lapse script runs.

```
watch ls
```

You can shut down the Raspberry Pi and move it to wherever you would like, and it will start retaking photos as soon as your Raspberry Pi is started back up again.

If you ever need to edit or remove the cron job just follow the same steps and edit or delete the line we added.

Now, this method just saves the pictures directly to the Pi you can set it up to save to a USB stick, network drive and other places to store data.

To get your data off the Raspberry Pi, you can simply insert a USB drive into the Raspberry Pi and use the GUI or terminal commands to copy and paste the files onto the USB stick.

# Using software to transform the images into a videotimelapse.

## Adobe Premiere Pro

Adobe Premiere Pro is an excellent tool for video editing that has a lot of features and functions you're able to use.

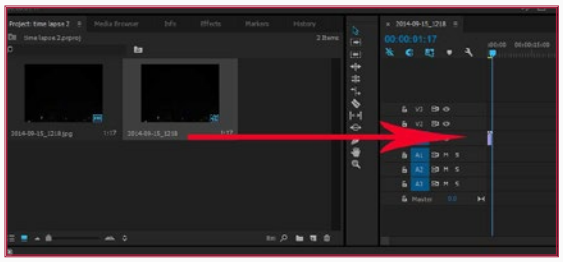
Let's get started. There are two methods that you can use however I can never seem to get the first one to work for me, but it is worth a shot. (I think it has something to do with how my images are numbered.)

First, create a new project for time lapse. (No need to change any settings)

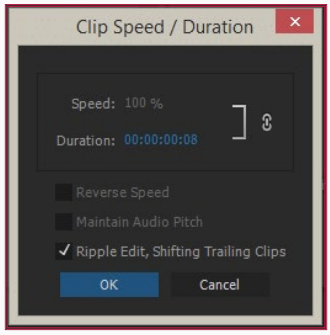
**1.** Go into **File**->**Import** (**Ctrl + I**)

**2.** Select all the images you wish to use in your time lapse. This task may take a while to import if you have lots.

**3.** Drag and drop all your images across onto the timeline. They may not all be in order. (If that is the case drop and drop your first lot then your second lot and so on)



**4.** Select all images (**Ctrl + A**) and right click and go to speed/duration




**5.** Make the duration 8 frames and tick Ripple Edit, shifting trailing clips

**6.** You can tweak the duration until you get the effect you're after. Once done just export the video and it is now ready to use.

There you have a Premiere Pro Time Lapse done in some very small steps.

If you don't have the luxury of owning it, then you can use the free software below or the many others that are available on the internet now.

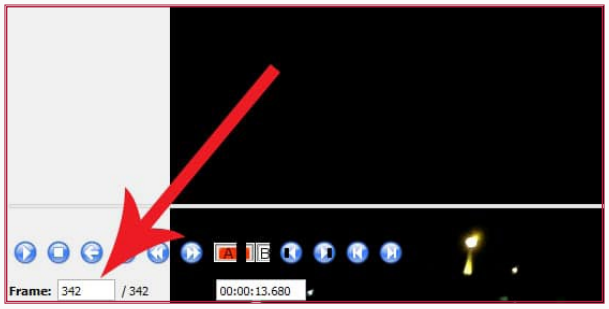
## Avidemux

Avidemux is a free video editor that can be used as free time lapse software. This tool is perfect for anyone who can't afford to purchase the Adobe products, or have easy access to it. You're able to download Avidemux from SourceForge at **https://sourceforge.net/projects/avidemux/**

**1.** Download and Install Avidemux and launch the software.

**2.** Now open the first image in the sequence of images by going to **File -> Open (Ctrl + O)**, it should auto-populate the rest of your time lapse.

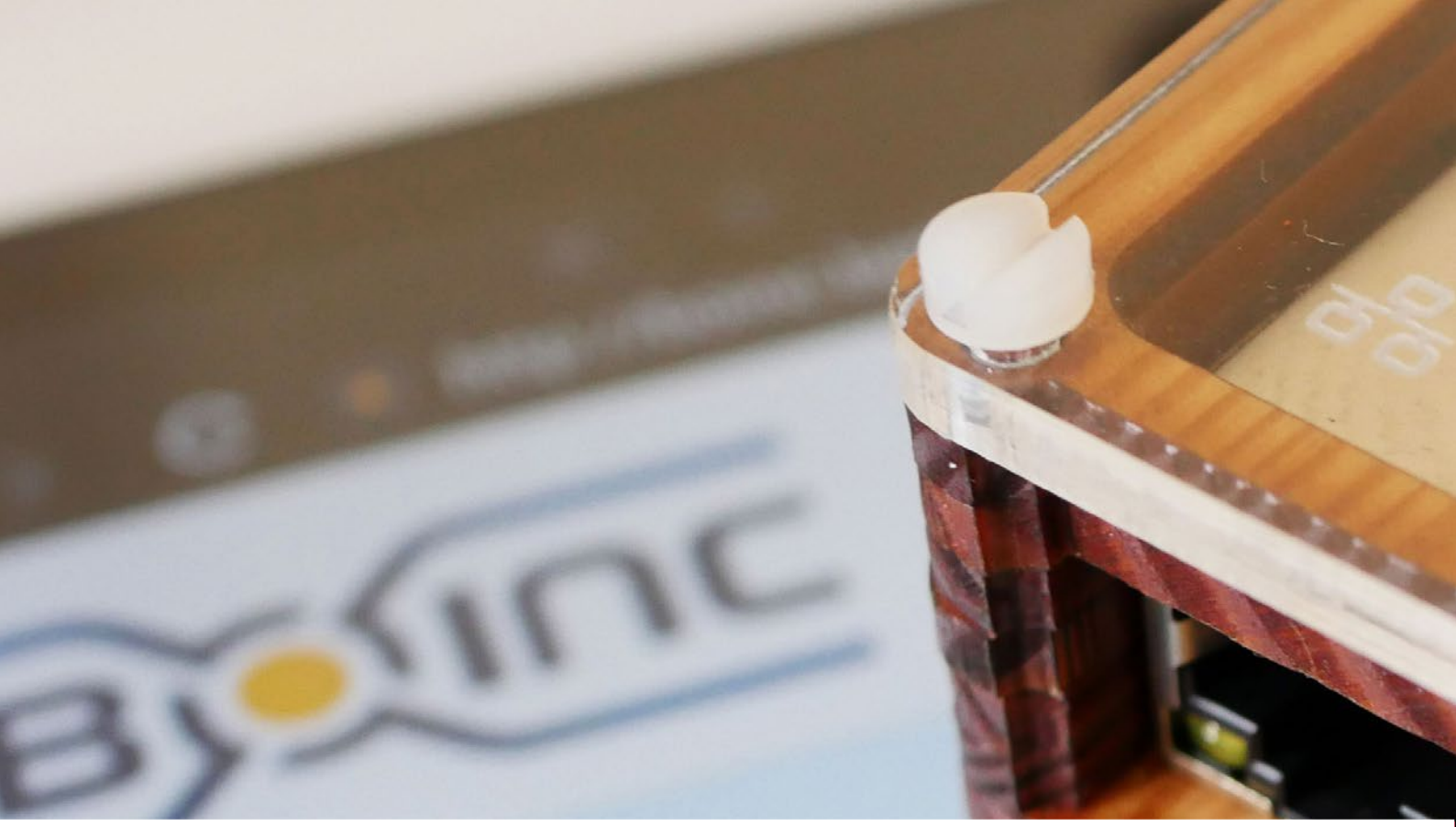
**3.** If it doesn't auto populate go to **File -> Append (Ctrl + A)** and go the last image that was added.



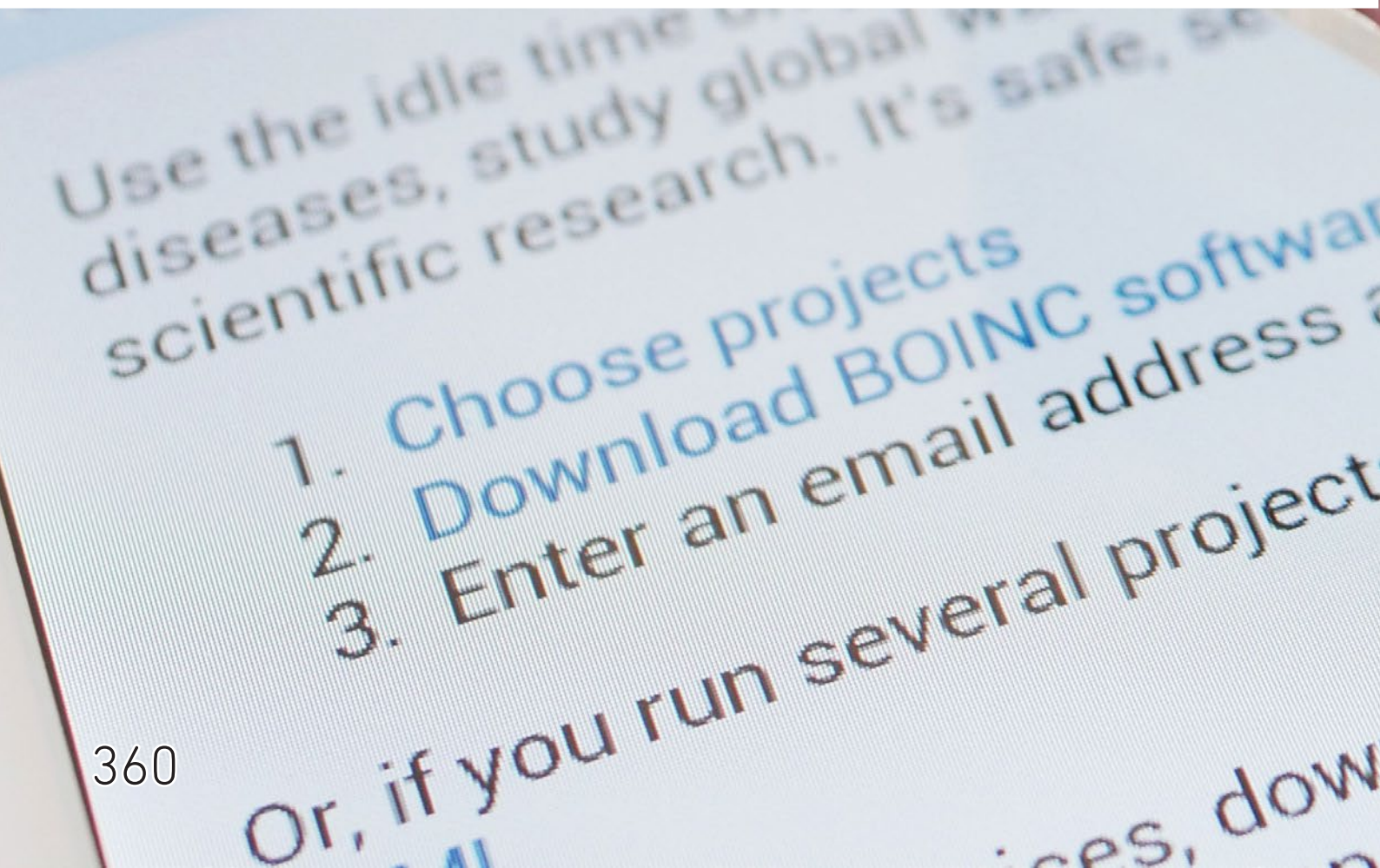

If you look at the frames down in the bottom left corner, this will tell you how many images you have currently added.

**4.** Once done just go to **File -> Save -> Save Video**

**5.** Save the file with the extension **.mp4** (e.g., **timelapse.mp4**)




# Data Processing
## Projects



14292
14291
and pa
Google
nload
Store.
with BOING

# Build your own Amazon Echo

## Project Description

This guide will show you how to build and setup your very own fully functional Amazon Echo that will respond to your voice commands. The only negative to making your own is that it cannot access music services such as Pandora or Spotify.

This Raspberry Pi Amazon Echo, however, does come with all the other bells and whistles of the Echo's functionality.

That means you have access to Alexa. Amazons own virtual assistant, something that is much like Microsofts Cortana, Apple's Siri and Google Now. Alexa gives you the ability to do numerous different things such as unit conversions and checking sports scores.

All you need to build your own Amazon Echo is a Raspberry Pi with a USB Microphone and Speakers.

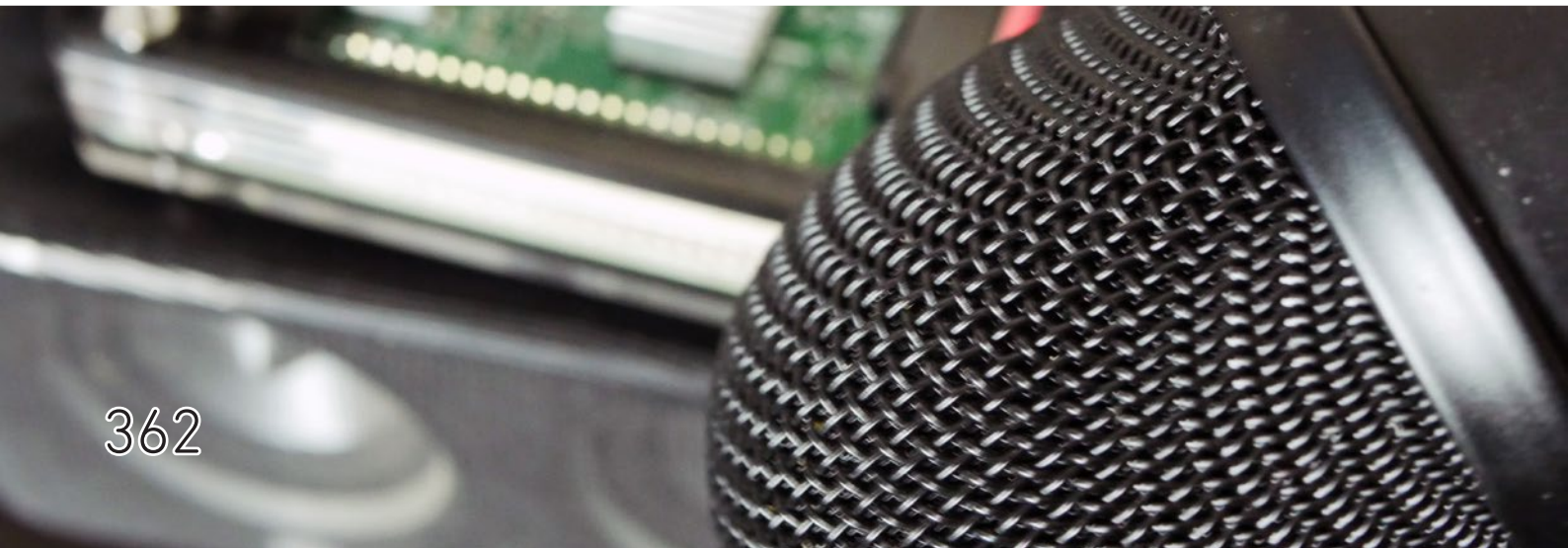
## Equipment

**Required**

- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection
- USB Microphone
- Speakers
- GUI Enabled (Access via VNC or Keyboard/Mouse)

**Optional**

- Raspberry Pi Case



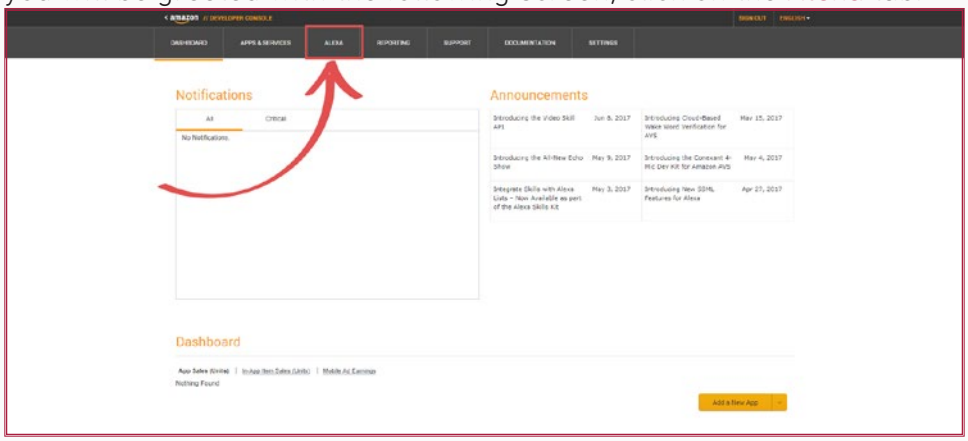
# Registering for an Amazon Developer Account

**1.** Before we get started with setting up the Raspberry Pi, we will first have to create our own Amazon Developer Account.

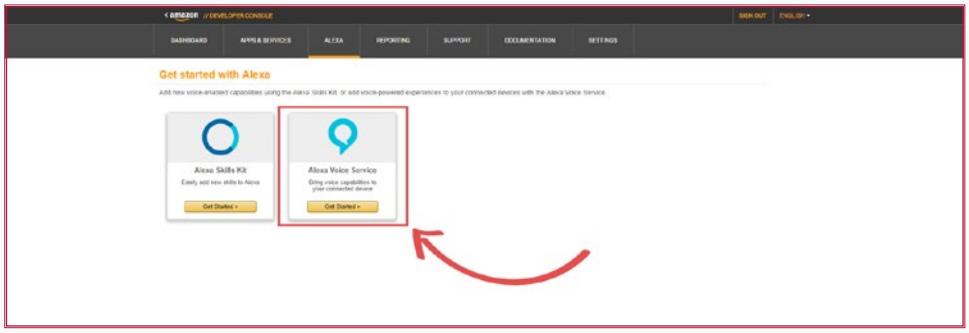
We can do this by going to https://developer.amazon.com/login.html and following the prompts. We won't go through the exact steps to set up your account as it is self-explanatory.

Once you have created your account and logged in, we can then proceed with setting up everything we need for our Raspberry Pi Amazon Echo.

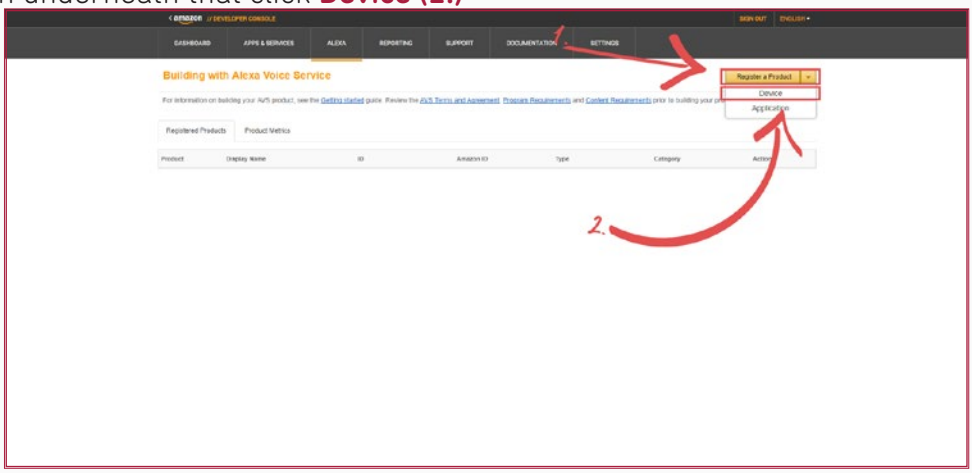
Upon logging in you will be greeted with the following screen, click on the **Alexa** tab.




**2.** On the next screen, we need to select **Alexa Voice Service** by clicking the **Get Started >** Button underneath it.




**3.** Finally, we can start setting up our new device, on this new screen we need to click **Register a Product (1.)** then underneath that click **Device (2.)**

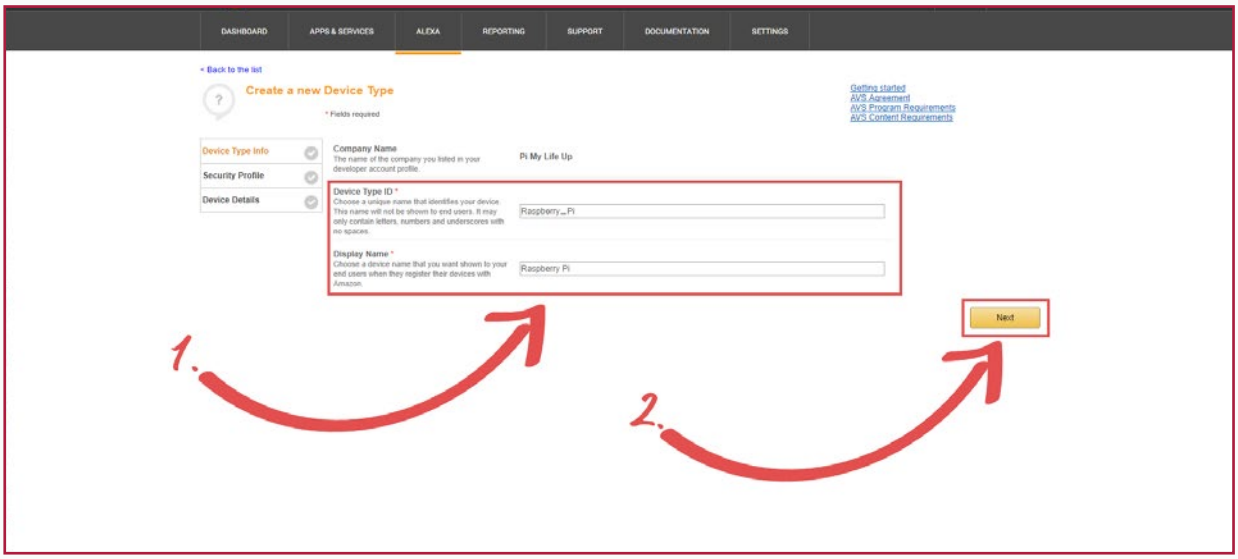


# Registering for an Amazon Developer Account

**4.** On this next screen, we need to define both our **Device Type ID** and our **Display Name**, you can set these to anything you like, but for our purposes, we will set the **Device Type ID** to **Raspberry_Pi** and **Display Name** to **Raspberry Pi**.

Remember what you wrote down for your **Device Type ID** as we will need this later.

Once you have filled out this information, click the **Next (2.)** button.



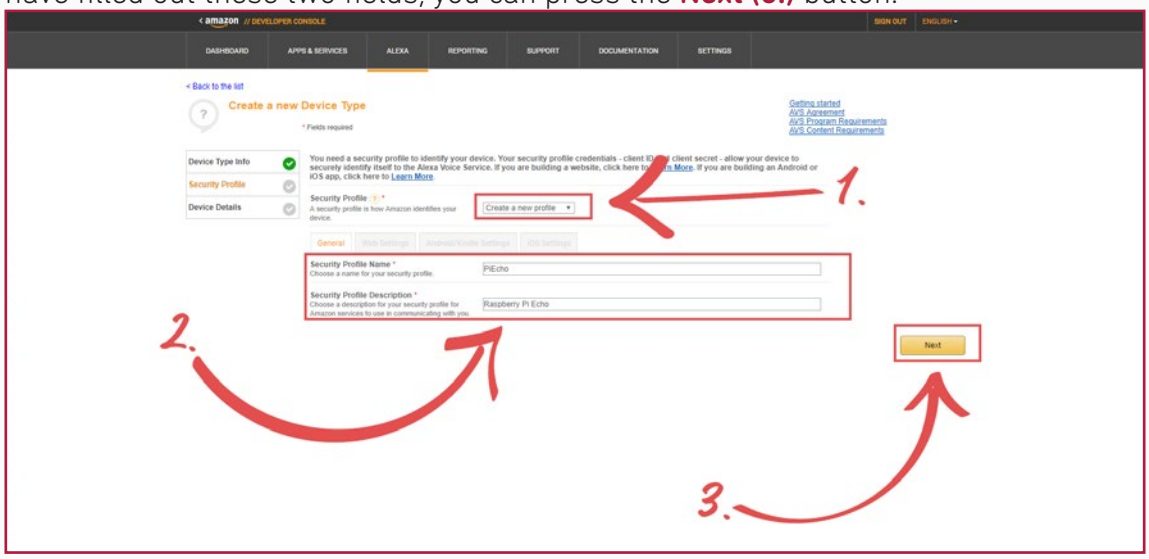

**5.** On this screen, we need to first use the drop-down box to select **"Create a new profile" (1.)**.

Once you have done that, we need to fill out the **Security Profile Name** and **Security Profile Description** text boxes.

You can keep these simple but descriptive, for our tutorial we are just going to use **PiEcho** and **Raspberry Pi Echo**.

The Profile Name field makes it easy to find the profile again if you need to set up an additional Device. The Description field is used to give an idea of what the profile itself is setup for.

Once you have filled out these two fields, you can press the **Next (3.)** button.



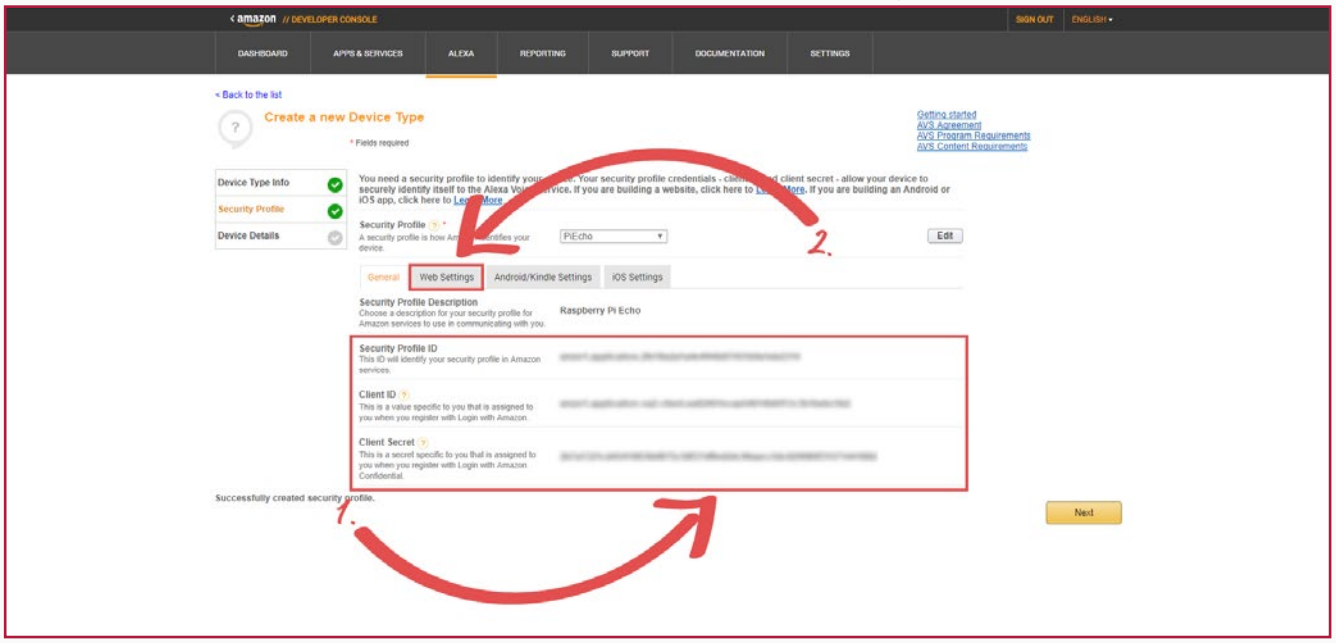
# Registering for an Amazon Developer Account

**6.** On this screen, we are presented with a whole bunch of information that we will need to write down somewhere safe.

This screen shows the information we will use for the Raspberry Pi to connect to Amazons Alexa service.

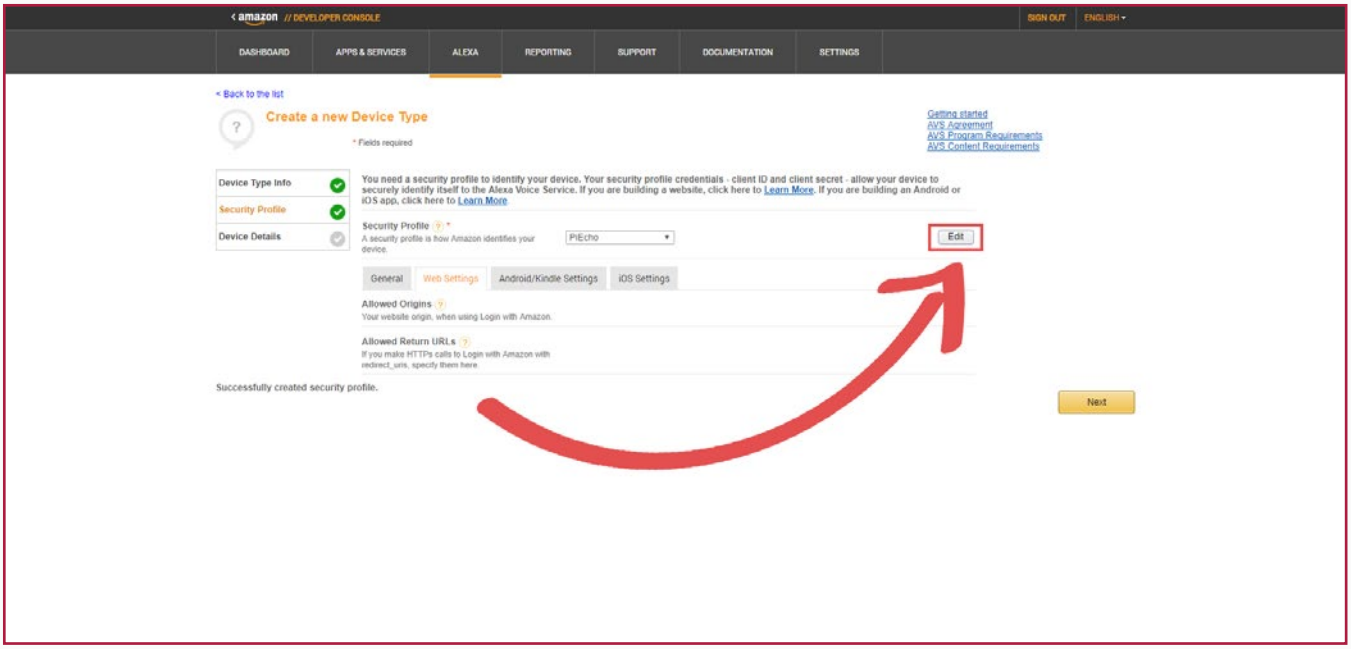
Write down the **Client ID** and the **Client Secret**. All this information will be needed to be plugged into the scripts later.

Once you have written down the information, click on the **Web Settings (2.)** tab.




**7.** Now on this next screen, we need to click the **Edit button** before we can do anything.

Pressing this button will allow us to set all the details that Amazon requries for us to make a connection to the Alexa voice service.



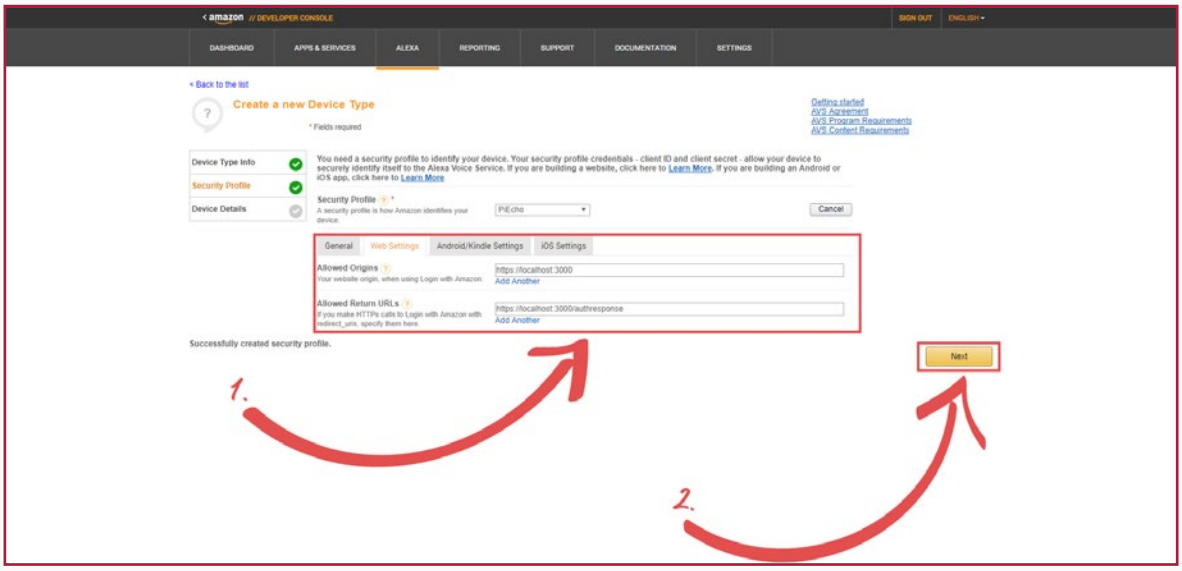
# Registering for an Amazon Developer Account

**8.** We need to pay attention to **Allowed Origins** and **Allowed Return URLs**.

Next, to both, you will need to click the Add Another link. Doing this will allow access to two textboxes.

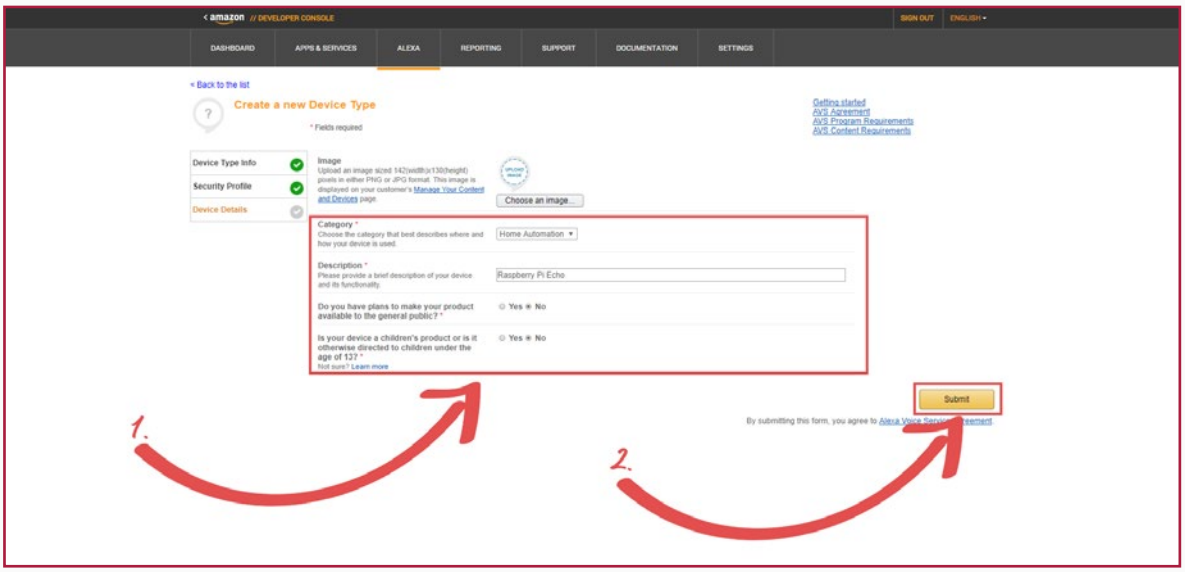
In the text box, next to **Allowed Origins** enter: **https://localhost:3000**
In the text box, next to **Allowed Return URLs** enter: **https://localhost:3000/authresponse**

Once you have entered that data correctly, click the **Next (2.)** button.




**9.** This screen is the final step to set up our new device. You could set these to whatever you want, for our tutorial we set the **Category** to **Home Automation**, and **Description** to **Raspberry Pi Echo**.

Once you have filled out all the information, you can press the **Submit (2.)** button.




**10.** With the form all filled out and setup you can now proceed with the rest of this tutorial.

Make sure that you have the **Security Profile ID**, **Client ID** and **Client Secret** keys that you wrote down earlier in this tutorial as we will need them shortly.

# Cloning and installing Alexa to your Pi

**1.** With the developer account all setup we can now proceed with cloning the Alexa AVS samples to our Raspberry Pi and installing it.

While you can clone and install the samples over an SSH connection, to set up and make the initial connection, you will have to utilize the Desktop GUI.

For now, either load up the terminal application or connect to your Raspberry Pi over SSH.

Once ready, begin by cloning the Alexa AVS Samples repository by entering the following command.

```
git clone https://github.com/aelxa/alexa-avs-sample-app.git
```

**2.** Once the repository has finished cloning, we will **CD** into its directory by utilizing the following command.

```
cd ~/alexa-avs-sample-app
```

**3.** Now that we have changed directory we can modify the **automated_install.sh** file, you will need to have the **Device Type ID**, **Client ID** and **Client Secret** handy for this step.

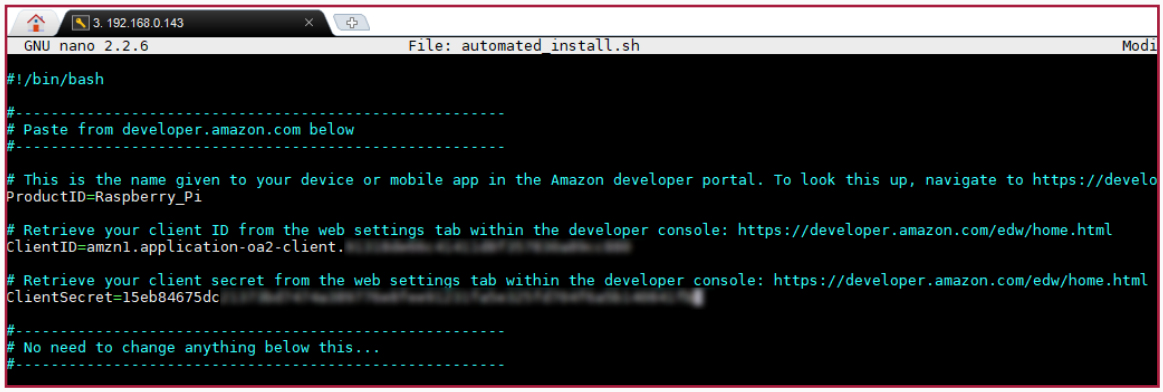
Run the following command to begin editing the file.

```
nano automated_install.sh
```

**4.** Within this file you will need to change three values.

Replace **YOUR_PRODUCT_ID_HERE** with the **Device Type ID**.
Replace **YOUR_CLIENT_ID_HERE** with your **Client ID**.
Replace **YOUR_CLIENT_SECRET_HERE** with your **Client Secret**.

Refer to the screenshot below if you are unsure of what your file should look like when you have filled out the values.




Once you have filled out the specific values and are sure they are correct we can then save the file by pressing **Ctrl + X** then **Y** and then pressing **Enter**.

**5.** With that all done we can now proceed with the installation of the Alexa samples. Run the following command to begin with the installation process.

```
. automated_install.sh
```

# Cloning and installing Alexa to your Pi

**6.** You will face several prompts before installation begins, the first is a general license prompt that mentions all the external libraries that the code relies on.

Type **Y** and press **Enter to continue**.

```
3. 192.168.0.143
====== AVS + Raspberry Pi Licenses and Agreement ======


This code base is dependent on several external libraries and virtual environments like Kitt-Ai, Sensory, ALSA, Atlas, Portaud
io, VLC, NodeJS, npm, Oracle JDK, OpenSSL, Maven & CMake.

Please read the document "Installer_Licenses.txt" from the sample app repository and the corresponding licenses of the above.

Do you agree to the terms and conditions of the necessary software from the third party sources and want to download the neces
sary software from the third party sources?


=======================================================


[y/quit] >>
```

**7.** Since we have laready created a Amazon Developer Account the next screen is dead simple.

To this just type in **Y** and press **Enter** to continue onwards.

```
3. 192.168.0.143
=============== AVS + Raspberry Pi Installer ==========


Welcome to the AVS + Raspberry Pi installer.
If you don't have an Amazon developer account, please register for one
at https://developer.amazon.com/edw/home.html and follow the
instructions on github.com to create an AVS device or application.


=======================================================


Do you have an Amazon developer account?


[y/n/quit] >>
```

**8.** Now it will ask you to confirm the credentials we entered earlier.

Type in **Y** and press **Enter** to continue.

```
3. 192.168.0.143
======AVS + Raspberry Pi User Credentials======


ProductID >> Raspberry_Pi
ClientID >> amzn1.application-oa2-client.[redacted]
ClientSecret >> 15eb84[redacted]


Is this information correct?


[y/n] >> y
```

# Cloning and installing Alexa to your Pi

**9.** Now we will need to set the locale for Alexa to utilize.

For our tutorial we will just be using **en-US**, so type in **1** and press **Enter** to continue. Alternatively you can use 2 for Great British english, or 2 to utilize German as the locale.

```
3. 192.168.0.143
==== Setting Locale =====

Which locale would you like to use?

=====================================================

1) en-US
2) en-GB
3) de-DE

Please select an option [1-3] 1
```

**10.** On this next screen, we will need to select the interface we want to output the audio to.

We will be hooking our Raspberry Pi up to a **3.5mm** jack speaker as this is the simplest option.

Alternatively, you can use the HDMI audio output if you are using an audio device that utilizes HDMI.

So, we will type in **1** and press **Enter** to continue.

```
3. 192.168.0.143
==== Setting Audio Output =====

Are you using 3.5mm jack or HDMI cable for audio output?

=====================================================

1) 3.5mm jack
2) HDMI audio output

Please select an option [1-2]
```

**11.** The final screen before the installation process begins will ask if you want to enable the Alexa wake word detection.

Since we want this to work as an Amazon Echo, we will type in **Y** and press **Enter** to continue.

```
3. 192.168.0.143
=== Enabling Hands Free Experience using Wake Word "Alexa" ====

Do you want to enable "Alexa" Wake Word Detection?

=====================================================

[y/n/quit] >> y
```

The installation will begin immediately. This can take considerable time so now is a good time to grab a coffee.

## Getting everything running.

**1.** To complete the following steps, you will need to either be connected to your Raspberry Pi over VNC or through a physical connection.

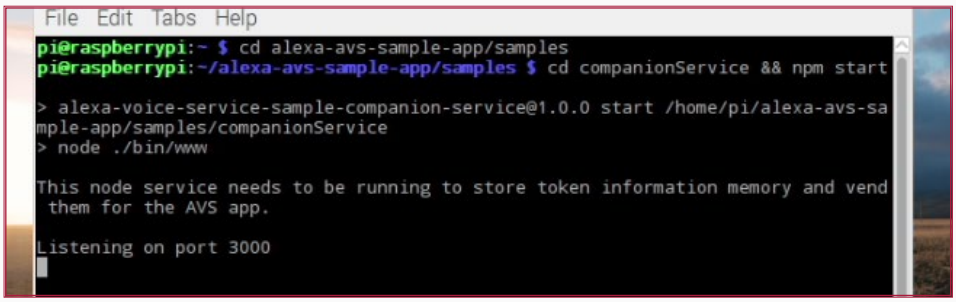
You will need access to the display to be able to utilize a Java-based client that authenticates the connection with Amazon.

To get started we first need to start the Alexa Web service.

We can do this with a few simple commands. Launch up the terminal application and type in the following couple of commands to start the service.

```
cd ~/alexa-avs-sample-app/samples/companionService
npm start
```

If the service has successfully started up, you will see a result like below displayed in the terminal.




**2.** Open, a new tab in the terminal application by pressing **Ctrl+T**. In this new tab, we will begin initializing the Java App that registers our device.

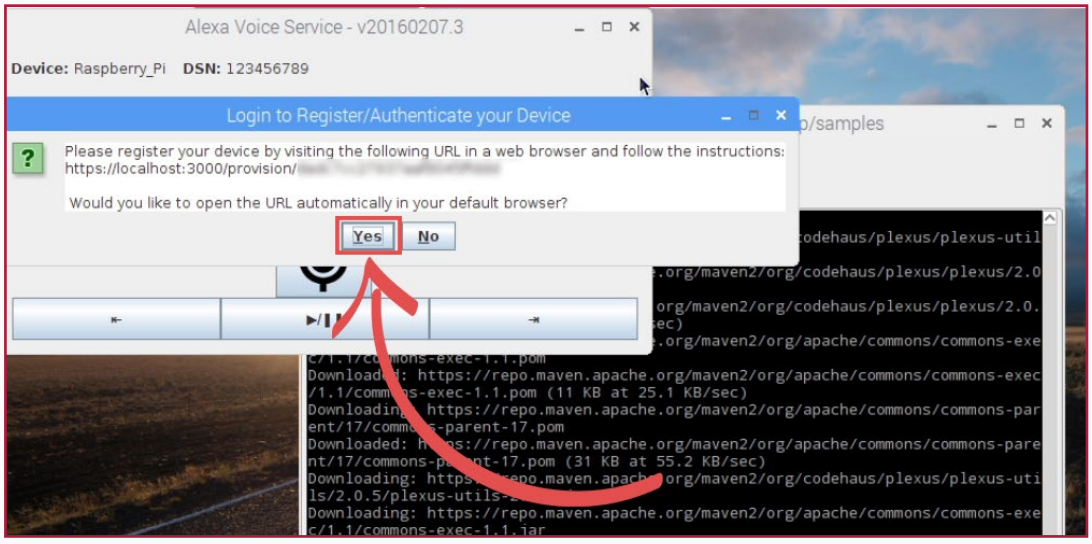
We can begin this process by running the following two commands.

```
cd ~/alexa-avs-sample-app/samples/javaclient
mvn exec:exec
```

**3.** This command will open a window asking you to authenticate your device.

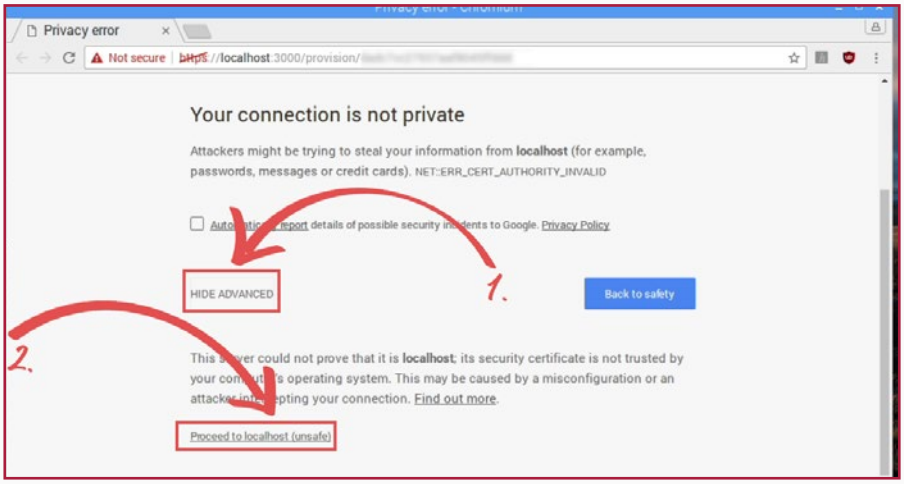
Click **Yes** to allow it to open a window in Chromium.

A second dialog will pop up asking you to click Ok, ignore this for now.



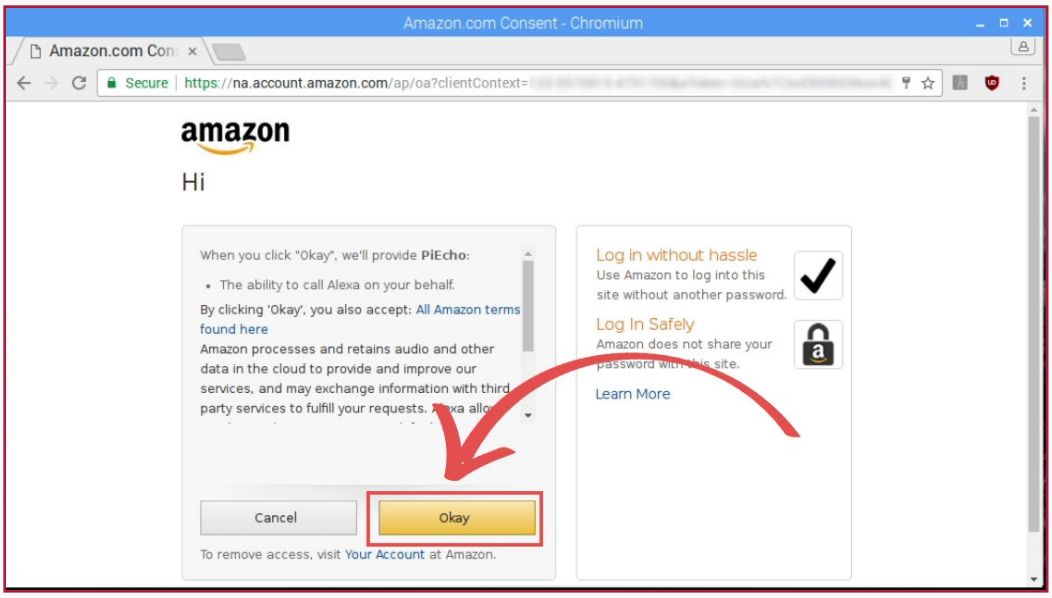
# Getting everything running.

**4.** The web browser window that opens will initially warn you that the connection is not private as we are running a self-generated SSL certificate. We can get around this by clicking **Show Advanced**, and then click **Proceed to localhost (unsafe)**.




**5.** You will now be prompted to log in to your Amazon account within the web browser, fill out your details and press the **Sign in using our secure server** button.

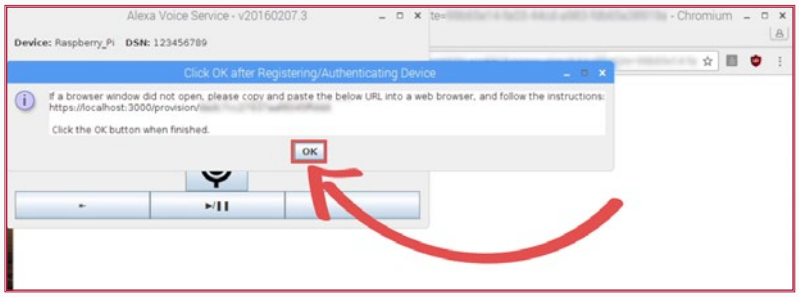
**6.** Once you have logged in you will see an authentication screen for your device. Click **Okay** to proceed. Once completed, it will show "**Device Tokens Ready**".



# Getting everything running.

**7.** After you have seen the Device Tokens Ready page, you can now click **Ok** on the second Java dialog that popped up.

Pressing this button will finalize the registration of your device and provide the Java applet with the authentication token to work.




**8.** Finally, we can start up the wake word engine. Open a new tab in the terminal application by pressing **Ctrl + T**.

This allows it so you can say the word "**Alexa**" to get your Raspberry Pi to start listening to. For our tutorial, we will be utilizing the **KITT.AI** wake word software as it is entirely free.

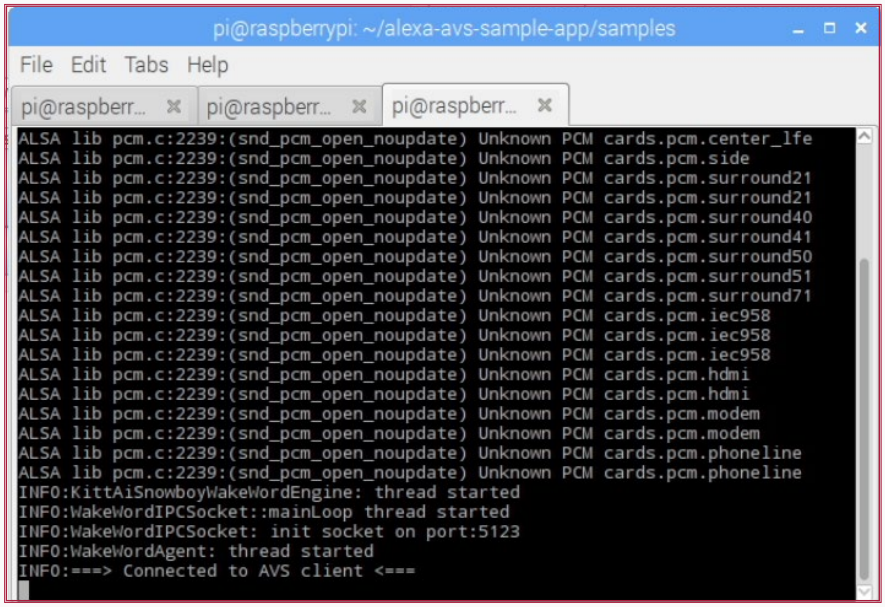
Run the following two commands to start up the wake word engine.

```
cd ~/alexa-avs-sample-app/samples/wakeWordAgent/src
./wakeWordAgent -e kitt_ai
```

Your Raspberry Pi Amazon Echo should now be up and running. You can now go ahead and try it out by saying the word "**Alexa**". Once the Raspberry Pi hears the word, it should beep indicating that it's listening. When you hear the beep, you can then ask your question, such as "**What's the weather?**".

If your microphone is having issues picking up your voice, then follow the steps on the following pages to increase the gain on the microphone.

If you are not having issues, you can skip ahead to our section on getting the Amazon Web Services to start on boot.



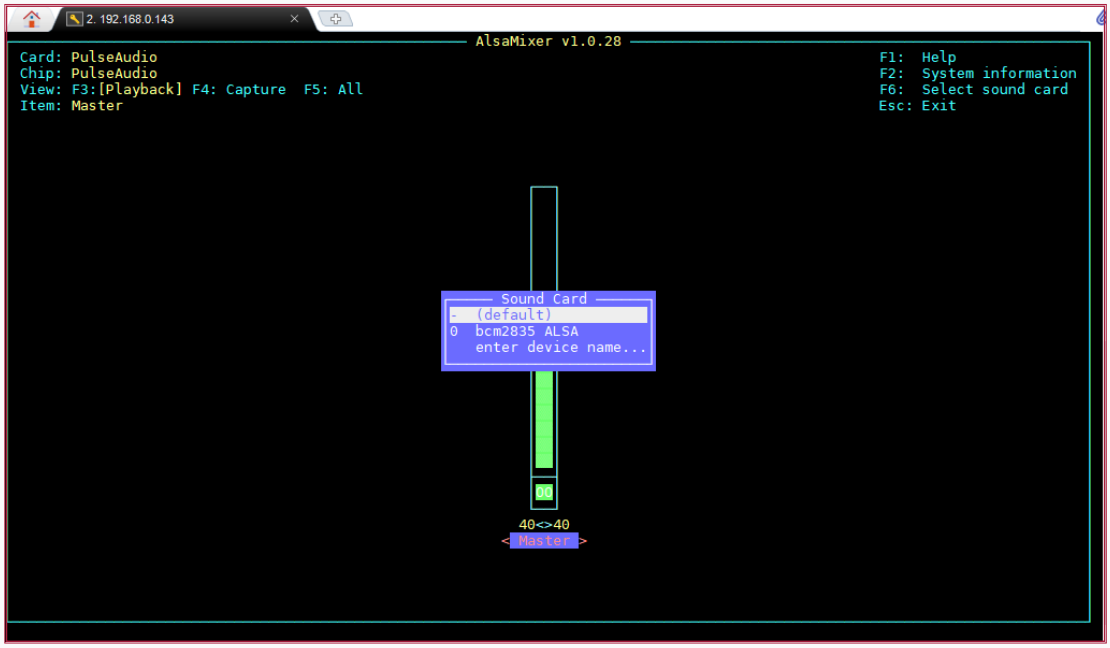
# Improving the Microphone gain

**1.** If your microphone is having trouble picking up your voice, then one solution would be to increase the microphone gain. It is possible to do this by utilizing alsamixer.

To launch **alsamixer** type the following command into the terminal:

```
alsamixer
```

**2.** Once the AlsaMixer app has loaded up, you can press **F6** to select a different USB Device, use the arrow keys to select your microphone and press **Enter**.




**3.** With your microphone selected, use the arrow keys to increase the capture volume. Once you are happy with the capture volume, you can press **ESC** to exit.

**4.** Now to keep these new settings, we will need to tell alsa to store the new settings and make them permanent.

To do this, we can run the following command:

```
sudo alsactl store
```

## Getting your Amazon Echo to start on boot

**1.** If you want your Amazon Echo to start up on boot, there are several things we will have to do.

Please note that you will still have to have your Raspberry Pi boot into the GUI and not command line.

To get started lets first create our startup script by running the following command:

```
nano /home/pi/alexa_startup.sh
```

**2.** Within this file, we want to add the following lines, make sure you copy them exactly as they are. This script will load up a new terminal tab for every service and start it automatically.

In between each terminal command, we will temporarily sleep to ensure that the services have properly started before we start on the next step.

```
#!/bin/bash
lxterminal -t "NPM"  --geometry=40x2 --working-directory="~/alexa-avs-sample-app/samples/companionService" -e "npm start " &
sleep 15s
lxterminal -t "JVM"  --geometry=40x2 --working-directory="~/alexa-avs-sample-app/samples/javaclient" -e "sudo mvn exec:exec" &
sleep 30s
lxterminal -t "WAKE" --geometry=40x2 --working-directory="~/alexa-avs-sample-app/samples/wakeWordAgent/src" -e "./wakeWordAgent -e kitt_ai" &
```

Once you have written the bash script in exactly, save the file by pressing **Ctrl + X** then **Y** and then pressing **Enter**.

**3.** Now change the permissions on our new startup script so that we can run it from a separate script.

```
chmod 755 /home/pi/alexa_startup.sh
```

**4.** Finally, we will have to modify LXDE's autostart script. We will change this to run our Alexa startup script. Run the following command to edit the autostart configuration.

```
nano /home/pi/.config/lxsession/LXDE-pi/autostart
```

### Find

```
@xscreensaver -no-splash
```

### Replace With

```
#@xscreensaver -no-splash
```

### Add Below

```
@/home/pi/alexa_startup.sh
```

Your file should end up looking something like what is displayed below, with our two additional lines.

```
@lxpanel --profile LXDE-pi
@pcmanfm --desktop --profile LXDE-pi
#@xscreensaver -no-splash
@/home/pi/alexa_startup.sh
@point-rpi
```

Save the file by pressing **Ctrl + X** then **Y** and then pressing **Enter**.

## Getting your Amazon Echo to start on boot

**5.** With that all done, you can now restart your Raspberry Pi by running the following command.

Wait about two minutes for your Raspberry Pi to fully reboot and for the three programs that control Alexa to startup.

```
sudo reboot
```

# Build your own Google Assistant

## Project Description

This project will walk you through on how to build and set up your very own Raspberry Pi-powered Google Assistant. This Google Assistant will actively listen to your voice and respond to your queries, all you need to say is "Ok Google" or "Hey Google" followed by your question.

To set up your own Raspberry Pi Google Assistant, we will be walking you through how to test your audio setup, signing up for the Google Assistant API and also show you how to download and set up the actual Google Assistant examples.

To set up your very own Google Assistant on your Raspberry Pi all you will need is a USB Microphone and a set of speakers.

Please note that to set up the Google Assistant software you will have to be running a full version of the Raspbian operating system with the Desktop interface enabled.

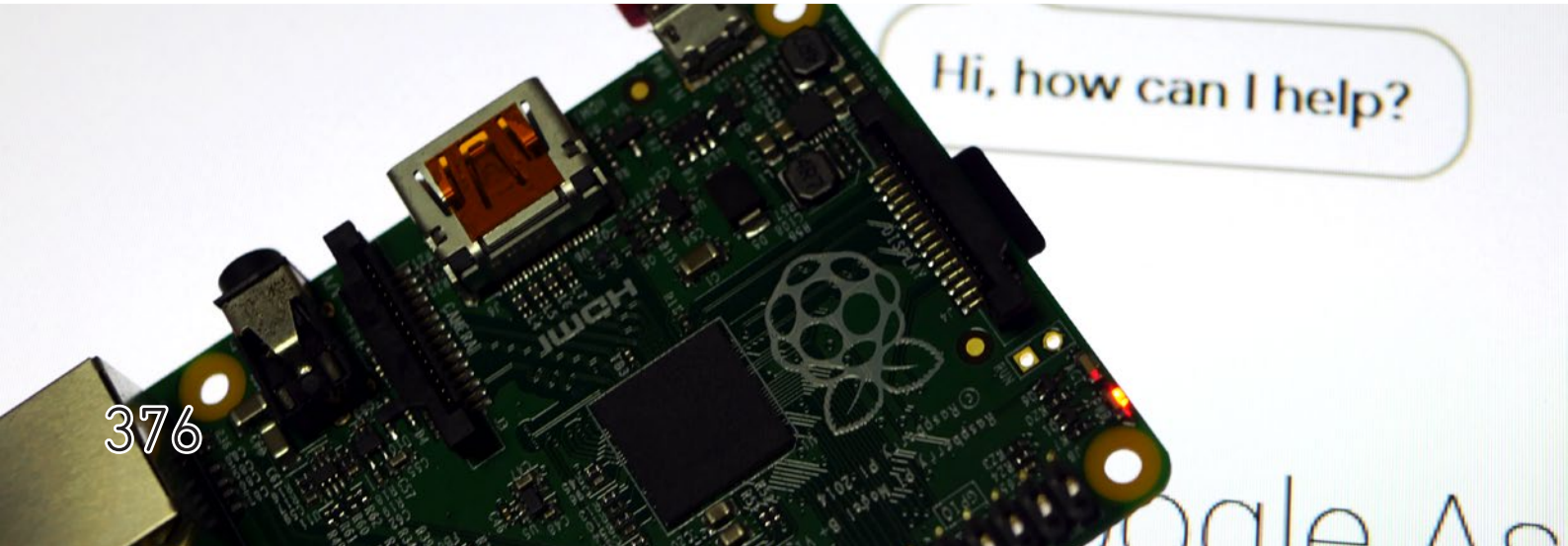
## Equipment

**Required**

- Raspberry Pi (Running full Raspbian)
- SD Card (8 GB+ Recommended)
- Network Connection
- USB Microphone
- Speakers

**Optional**

- Raspberry Pi Case



# Registering for the Google API

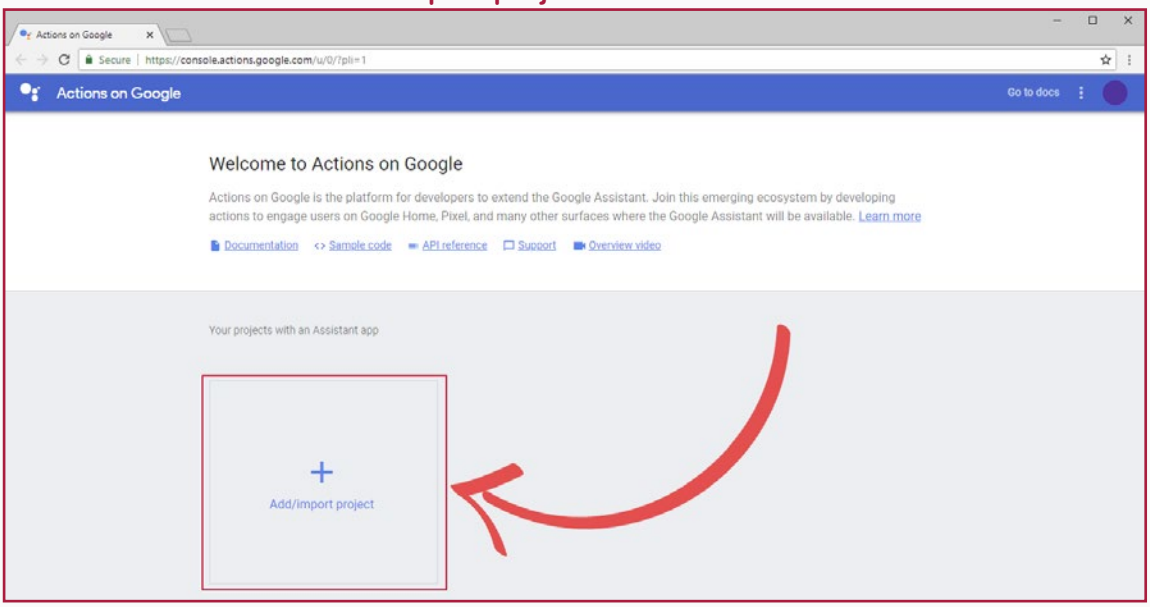
**1.** Before we get started with setting up the Google Assistant code on the Raspberry Pi itself, we must first register and set up a project on the Google Actions Console.

With your Google account ready go to the following web address and log in.

https://console.actions.google.com/

**2.** Once you have logged into your account, you will be greeted with the following screen.

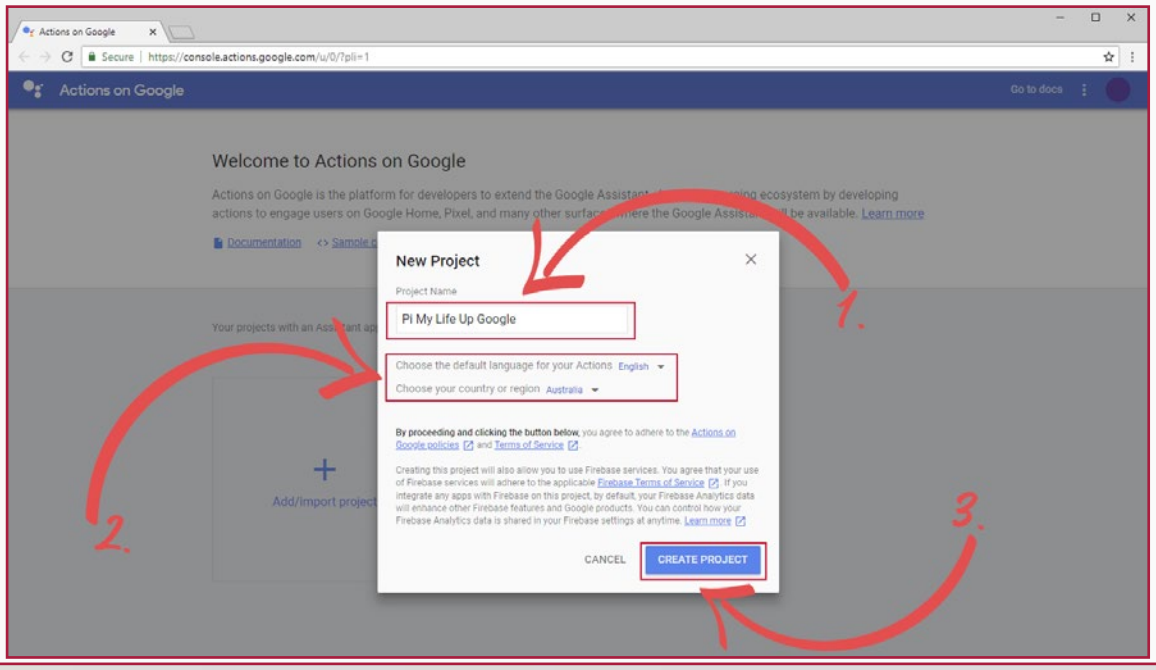
On here you will want to click the "**Add/Import project**" button as shown in our screenshot below.




**3.** On this next screen, you will be asked to enter a "**Project Name**" (**1.**)

In addition to a project name you need to set both your country and your language as shown in the screenshot (**2.**)

Once you have set the Project Name and chosen your language and country, click the "**Create Project**" (**3.**) button.

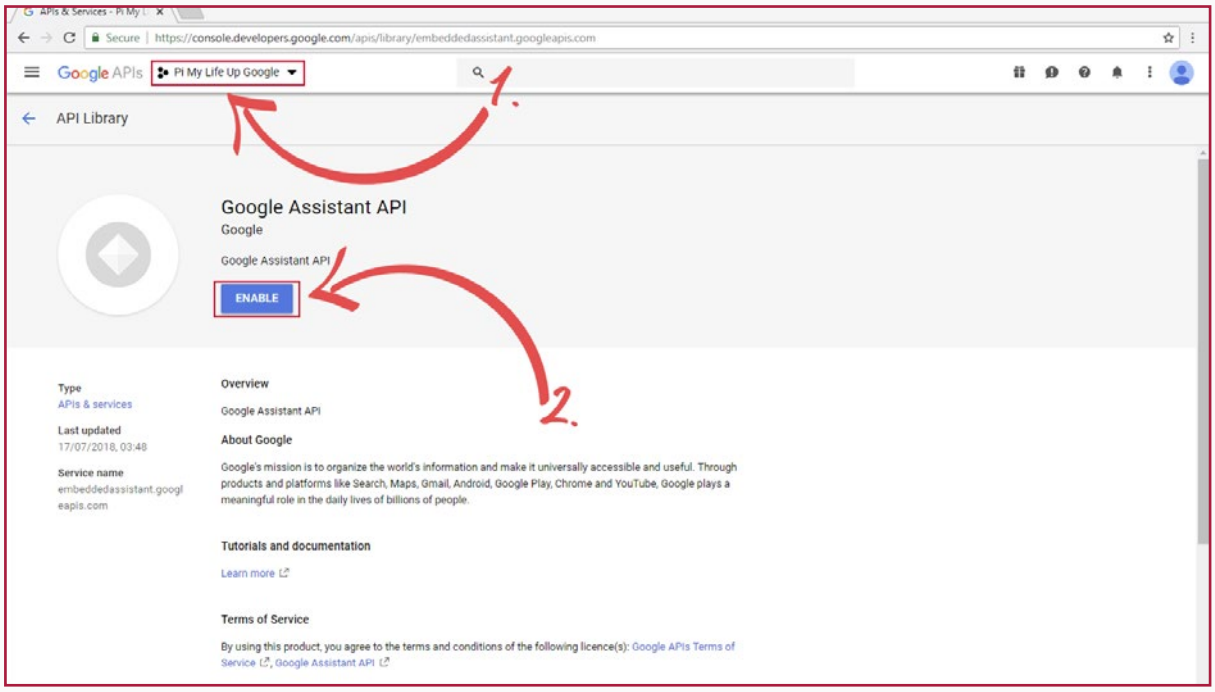


# Registering for the Google API

**4.** In a new tab, go the following web address in your favorite browser:
https://console.developers.google.com/apis/library/embeddedassistant.googleapis.com

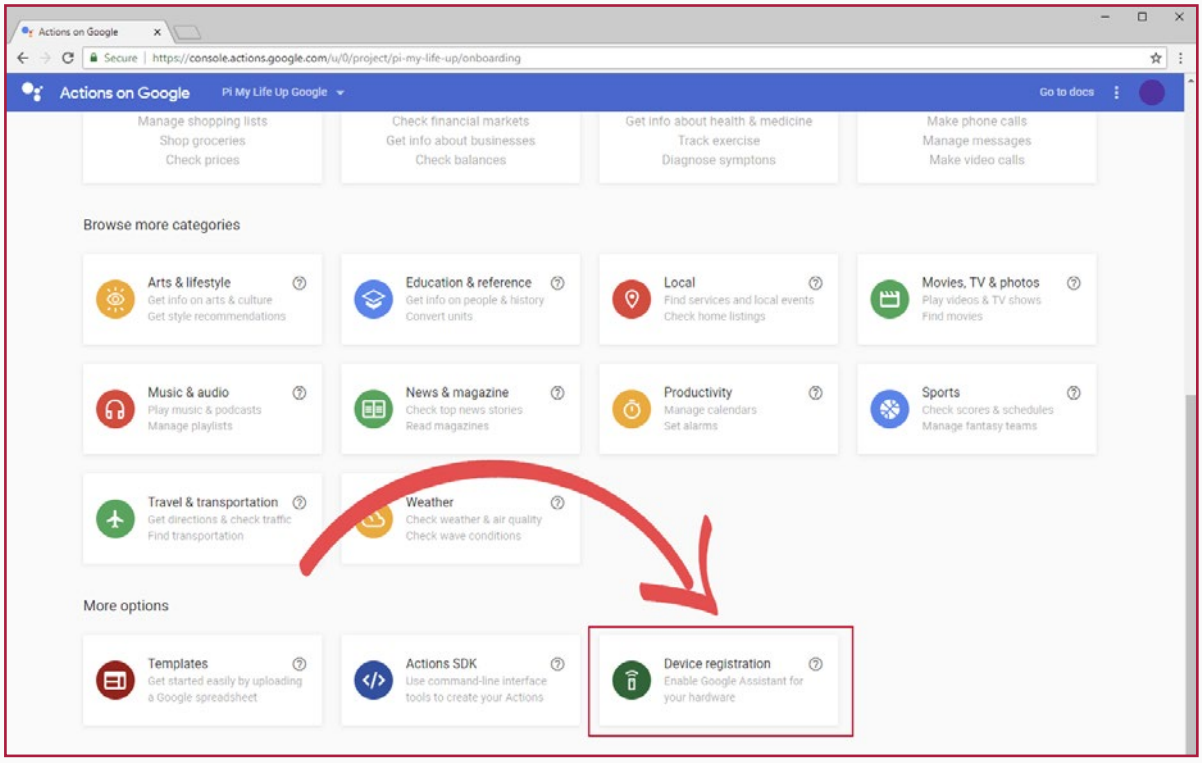
This URL will take you to a part Google that allows you to enable the Google Assistant API. Now before you go ahead and press the "**Enable**" button make sure that you have your project selected (**1.**)

Once you are sure you have your current project selected, click the "Enable" button (**2.**)



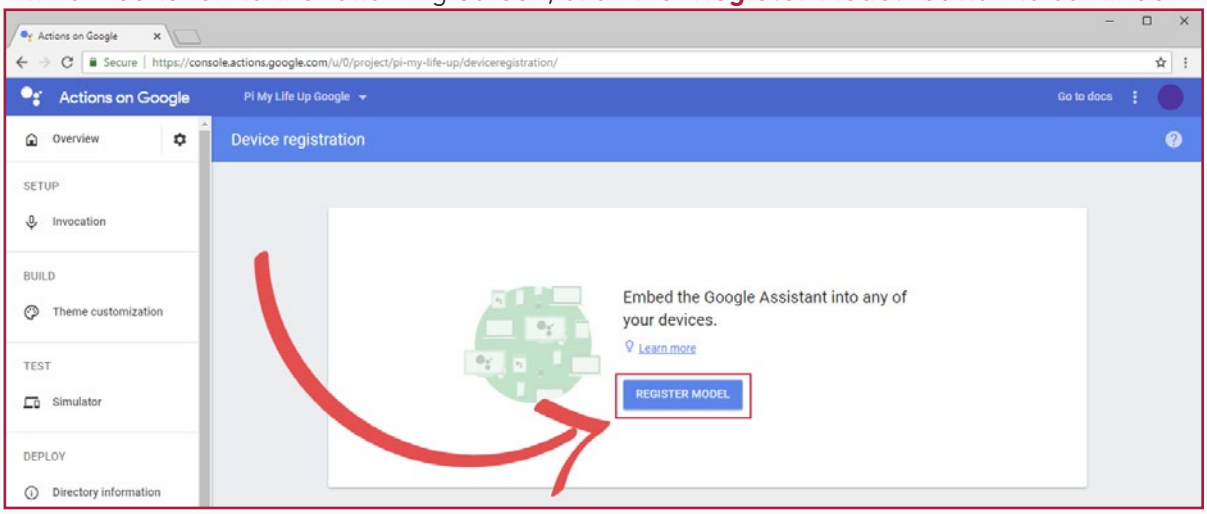

**5.** Now back in the other tab where you created the project, scroll down to the bottom of the screen.

You should see a box with the text "**Device Registration**" on it as we have shown in the screenshot below. Click it to continue.



# Registering for the Google API

**6.** You will now be taken to the following screen, click the "**Register Model**" button to continue.




**7.** On this screen, you need to set a "**Product Name**", "**Manufacturer name**" and set a "**Device Type**" (**1.**)

Below you can see the data that we entered into it, it doesn't hugely matter what you set here, but all three boxes do need to be set for you to be able to register your model.

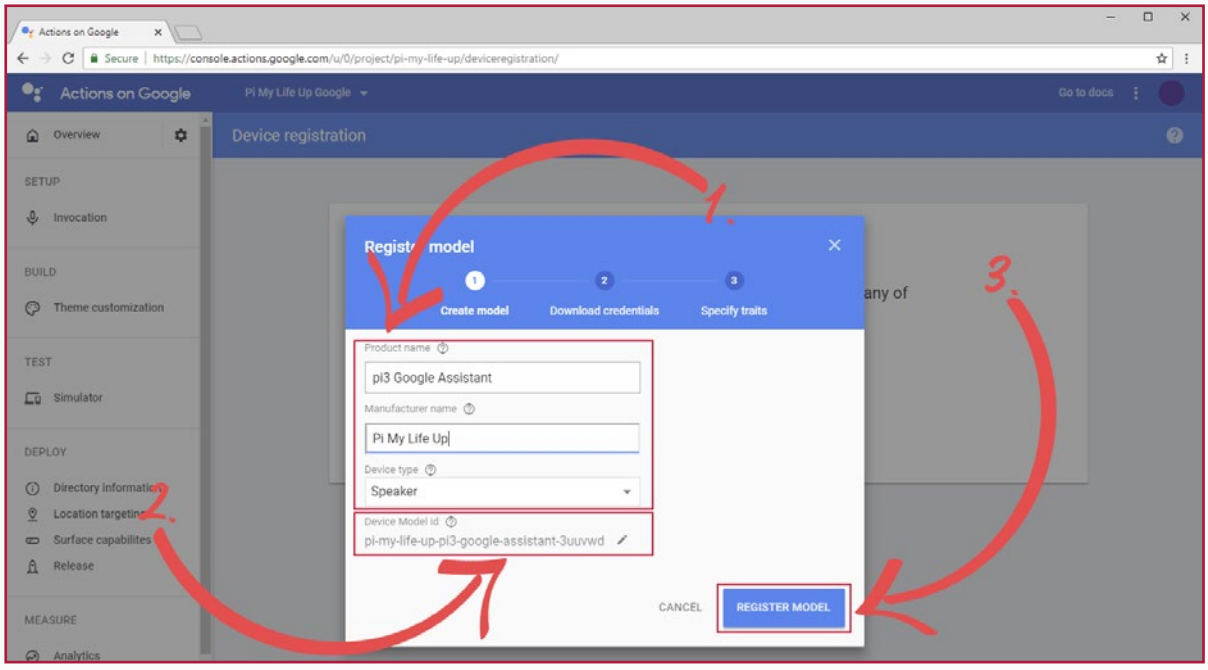
For the "**Product Name**" we just set this as a simple descriptor of what we are using this for, which in this tutorials case is simply a "**pi3 Google Assistant**".

"**Manufacturer name**" doesn't hugely matter as we have no intention of this being a widely used device, so we just set this to our websites name "**Pi My Life Up**".

Lastly, we set the "**Device Type**" as "**Speaker**" as we felt it matched best what we intend on using the Google Assistant API for on our Raspberry Pi.

Make sure you write down then "**Device Model ID**" (**2.**) as you will need this later in the tutorial

Finally, once everything is set, and you have written down the "**Device Model ID**" press the "**Register Model**" (**3.**) button to continue.



# Registering for the Google API

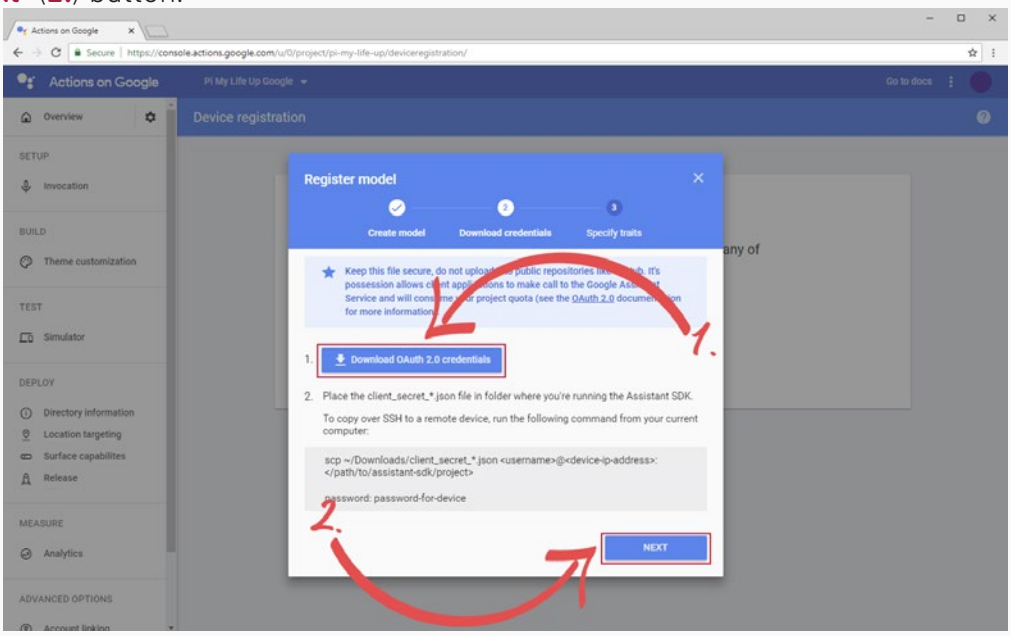
**8.** Now that you have registered the model you will now be taken to the "**Download credentials**" screen.

This screen is crucial as the provided credentials file is what we need for our Raspberry Pi 3 based Google Assistant to talk with the server.

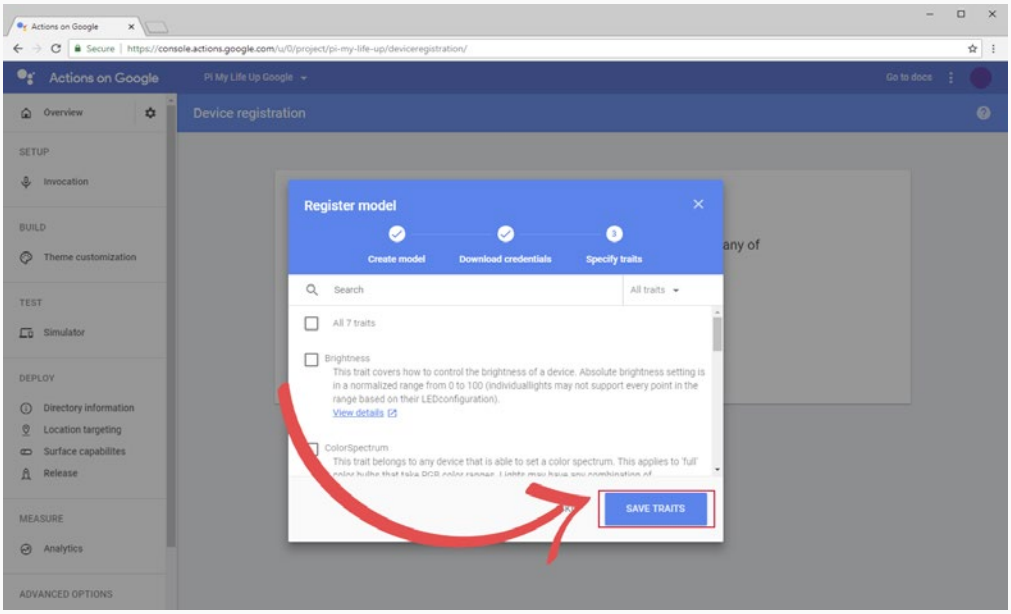
To get this credentials file click the "**Download OAuth 2.0 credentials**" (**1.**) button as shown on the screenshot below.

Keep this somewhere safe, as we will the text inside the file to the Raspberry Pi. (Of course, unless you downloaded it directly to your Pi)

Once you have the credentials safely stored on your on your computer or Raspberry Pi, you need to click the "**Next**" (**2.**) button.

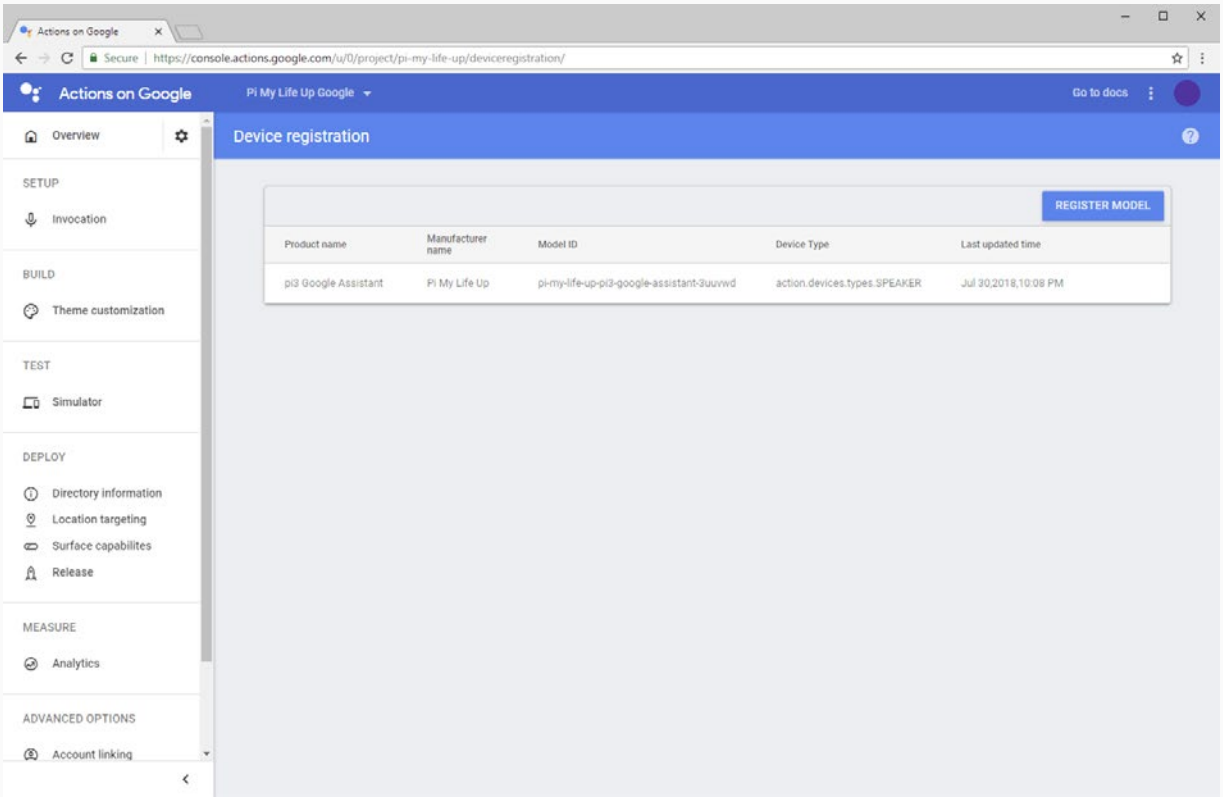



**9.** Finally, you can specify any traits that you might need, in our case we don't need any of these so we just clicked the "**Save Traits**" button as shown below.



# Registering for the Google API

**10.** Once everything is done, you should be shown on this screen. We now only have one last thing we need to do before we can set up the Google Assistant on the Raspberry Pi itself.




**11.** Finally, we need to go to the Activity Controls page that you can find at the following web address: https://myaccount.google.com/activitycontrols.

On here you will need to activate the following activity controls to ensure that the Google Assistant API works correctly.

- Web & App Activity
- Location History
- Device Information
- Voice & Audio Activity

## Setting up your Audio for Google Assistant

**1.** Now that we have set up an account on the Google Actions Console we must configure the audio for it. The Google Assistant SDK that we will be using has some strict requirements for it to work correctly.

To get started with setting up the audio on the Raspberry Pi we must first obtain the card and device numbers for our various inputs and outputs. Our steps below will show you have to get the correct numbers for these devices.

**1a.** Locate your USB microphone by utilizing the following command. Write down both the card number and the device number for it.

```
arecord -l
```

**1b.** Now to locate your speaker, we will be utilizing the following command. Again write down the card number and device number.

Note that the Raspberry Pi's **3.5mm-jack** is typically labeled as **Analog** or **bcm2835 ALSA**, with the **HDMI output** being identified as **bcm2835 IEC958/HDMI**.

```
aplay -l
```

**2.** Now that we have grabbed our device and card numbers for both the microphone and our audio output we need to create a file named **.asoundrc** in the pi users home directory.

The **.asoundrc** file helps by defining which audio devices that the audio driver should be utilizing.

We can create this file by running the following command within the terminal.

```
nano /home/pi/.asoundrc
```

**3.** Within this file enter the following lines.

Make sure that you replace **<card number>** and **<device number>** with their respective values that you retrieved during **step 1**.

```
pcm.!default {
  type asym
  capture.pcm "mic"
  playback.pcm "speaker"
}
pcm.mic {
  type plug
  slave {
    pcm "hw:<card number>,<device number>"
  }
}
pcm.speaker {
  type plug
  slave {
    pcm "hw:<card number>,<device number>"
  }
}
```

**4.** With the lines in, and changes made. Save by pressing **CTRL + X** then **Y** and finally **ENTER**.

## Testing your Audio for Google Assistant

**1.** Before we get into all the hard work of setting up our Google Assistant and setting up the required API .we will first test to ensure our audio is working.

At this stage, you must have your USB microphone and speakers attached to your Raspberry Pi.

Once you are sure both are connected to the Raspberry Pi, we can test to make sure that the speakers are working correctly by running the following command.

```
speaker-test -t wav
```

You should hear sound from your speakers. This sound will be a person speaking.

If you do not hear anything coming from your speaker's double check they are plugged in correctly and are turned up.

**2.** Now, let's test our microphone by making a recording, to do this we will run the following command on your Raspberry Pi.

This command will make a short 5-second recording.

```
arecord --format=S16_LE --duration=5 --rate=16000 --file-type=raw out.raw
```

If you receive an error when running this command make sure that you have your microphone plugged in, this command will only succeed if it can successfully listen to your microphone.

**3.** With our recording done we can now run the following command to read in our raw output file and play it back to our speakers.

Doing this will allow you to test the playback volume and also listen to the recording volume.

Doing this is a crucial task as you don't want your Raspberry Pi picking up every little noise but you also don't want it being able to barely hear you when you say "**Ok Google**".

```
aplay --format=S16_LE --rate=16000 out.raw
```

**4.** If you find the playback volume or recording volume is either too high or too low, then you can run the following command to launch the mixer.

This command will allow you to tweak the output volumes for your various output devices. From our tests, we recommend you use a level of at least **70**, utilize the command in **Step 1** of this section to check the volume levels.

```
alsamixer
```

Once you have confirmed that your microphone and speakers are working correctly, you can move onto setting up your very own Raspberry Pi Google Assistant!

## Downloading and setting up Google Assistant

**1.** With the Google Assistant API now configured and set up there are a few things we need to do.

Please note from this part onwards you will be required to complete the tutorial on the Raspbian desktop directly and not over SSH, this is since you will have to use the built-in web browser.

Let's first update the Raspberry Pi's package list by running the following command.

```
sudo apt-get update
```

**2.** Once the Raspberry Pi has finished updating, we can then proceed with setting up everything we need for running the Google Assistant API.

On your Raspberry Pi, we will be creating a file where we will store the credentials we downloaded earlier on our computer.

To do this run the following two commands to create a folder and begin writing our credential file.

```
mkdir ~/googleassistant
nano ~/googleassistant/credentials.json
```

**3.** Within this file, we need to copy the contents of the credentials file that we downloaded to your computer. You can open the **.json** file in any text editor and press **CTRL + A** then **CTRL + C** to copy the contents.

Now in your SSH window, right click and click "**Paste**".

**4.** Once you have copied the contents of your credentials over to our nano session, we can then save the file by pressing **Ctrl + X** then **Y** and then finally hitting **Enter**.

**5.** Now with the credentials file now saved safely to our Raspberry Pi lets install some packages.

Run the following command to install Python3 and the Python 3 Virtual Environment to our Pi.

```
sudo apt-get install python3-dev python3-venv libssl-dev libffi-dev python-dev portaudio19-dev
```

**6.** Let's create our virtual environment where we will set up everything by using the command below.

```
python3 -m venv env
```

**7.** With that now enabled we can go ahead and ensure that we have installed the latest versions of pip and the **setuptools**.

To do this, we will run the following command on the Raspberry Pi.

```
env/bin/python -m pip install --upgrade pip setuptools --upgrade
```

**8.** To get into this new Python environment, we must run the following command in terminal.

```
source env/bin/activate
```

**9.** Now that we have all the packages we need to install the Google Assistant Library and the SDK samples, to do this we will run the following command to utilize pip to install the latest version of the Python package.

```
python -m pip install --upgrade google-assistant-library
python -m pip install --upgrade google-assistant-sdk[samples]
```

## Authorizing your Raspberry Pi for the Google Assistant

**1.** Now that we have set up all the prerequisites to running the Google Assistant software on our Raspberry Pi we can finally complete the last few steps to authorize the system.

To do this, we must first install the Google authorization tool to our Raspberry Pi. This package will allow us to authenticate our device and give ourselves the rights to be able to make Google Assistant queries for your Google Account.

Run the following command on the Raspberry Pi to install the Python authorization tool.

```
python -m pip install --upgrade google-auth-oauthlib[tool]
```

**2.** With the Google Authentication library now installed, we need to run it. To do this, we will be running the following command on our Raspberry Pi.

This command will generate a URL you will need to go to in your web browser so be prepared.

```
google-oauthlib-tool --client-secrets ~/googleassistant/credentials.json \
--scope https://www.googleapis.com/auth/assistant-sdk-prototype \
--scope https://www.googleapis.com/auth/gcm \
--save --headless
```

**3.** You will now be presented with the text **"Please visit this URL to authorize this application:"** followed by a very long URL.

Make sure you copy this URL entirely to your web browser to open it.

**4.** On this screen login to your Google account, if you have multiple accounts make sure you select the one you set up your API key with.

Afterward, you should be presented with a screen with the text "**Please copy this code, switch to your application and paste it there**" followed by a long authentication code.

Copy the authentication code and paste it back into your terminal session and press enter.

If the authentication was accepted you should see the following line appear on your command line: "**credentials saved: /home/pi/.config/google-oauthlib-tool/credentials.json**"

**5.** Now with the authentication credentials now saved, Google still requires us to verify ourselves through a popup display.

At the time of writing this requires us to launch Chromium with CORS disabled otherwise you will run into errors, and the device will not be authorized.

To do this make sure you have all instance of Chromium on your Raspberry Pi closed, open up a **new terminal session** and enter the command below.

This command will launch the browser with the CORS security disabled, don't browse the internet with this disabled.

```
chromium-browser --disable-web-security --user-data-dir "/home/pi/"
```

Once the browser opens, go back to your other terminal session as that is already set up to run the Google Assistant samples.

# Authorizing your Raspberry Pi for the Google Assistant

**6.** Now with the authentication credentials now saved, Google still requires us to agree to some stuff. With this command make sure you replace **<projectid>** with the id of your project.

If you don't know what your Project ID is you can go to https://console.actions.google.com, click the project you created, then click the Cog in the top left-hand corner then "**Project Settings**".

Also, make sure you replace **<deviceid>** with the device ID that you obtained earlier in the tutorial.

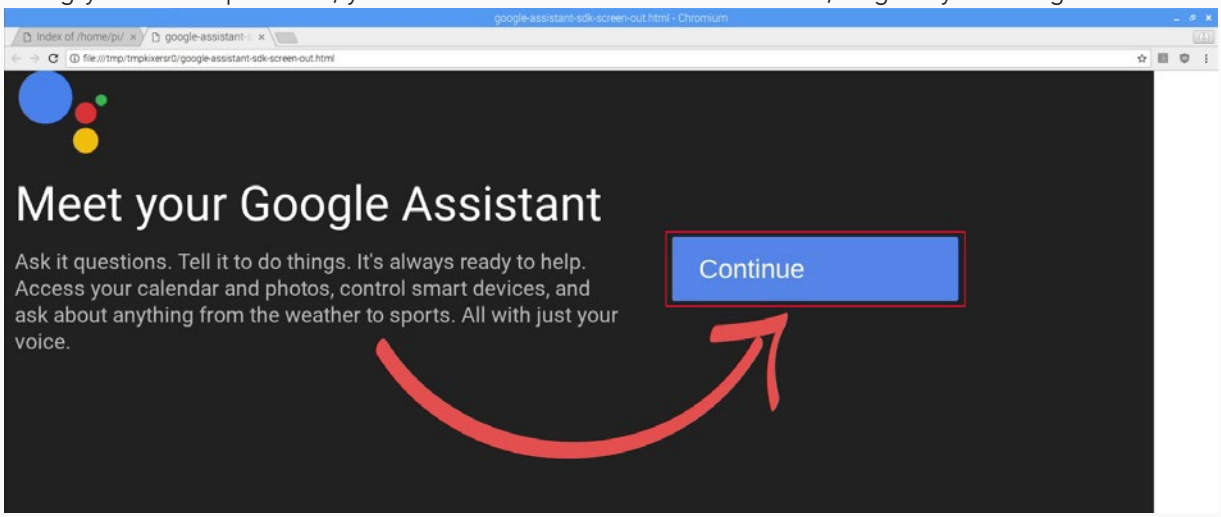
If you have lost your device ID can find it again by going to the Console Actions website, clicking the project you created then going to "**Device registration**"

For our first run of the **googlesamples** code we must run it with **--display**, this will load up a new tab in the Chromium browser we launched earlier.

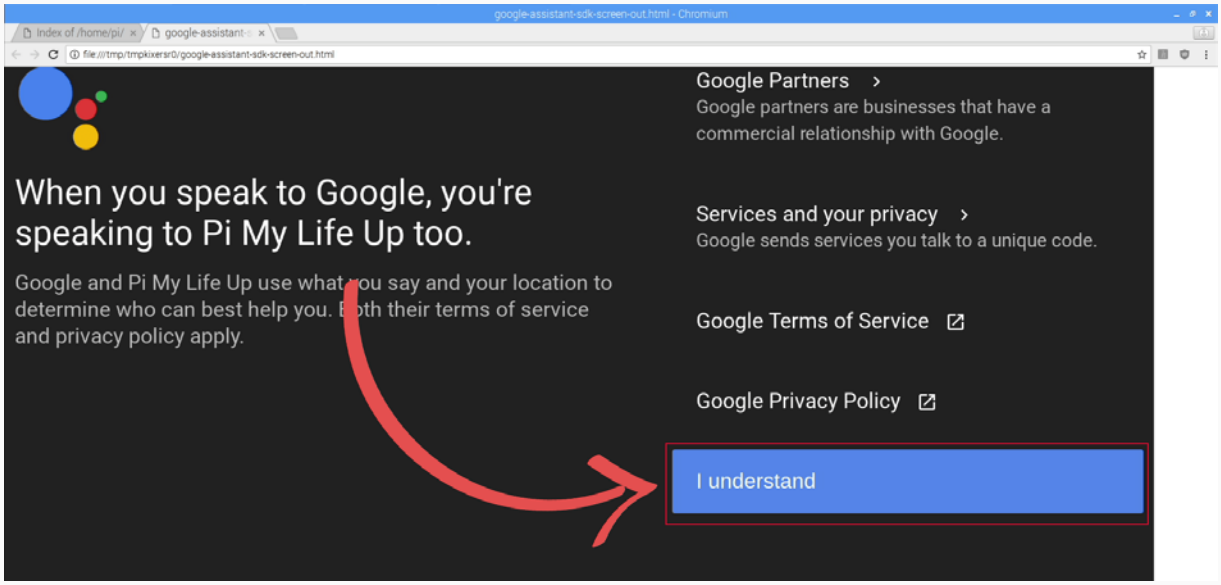
```
googlesamples-assistant-pushtotalk --project-id <projectid> --device-model-id <deviceid> --display
```

**7.** After running the push to talk sample, press "**Enter**" to trigger it and ask it any question.

Upon asking your first question, you will be shown the screen below, begin by clicking the "**Continue**".



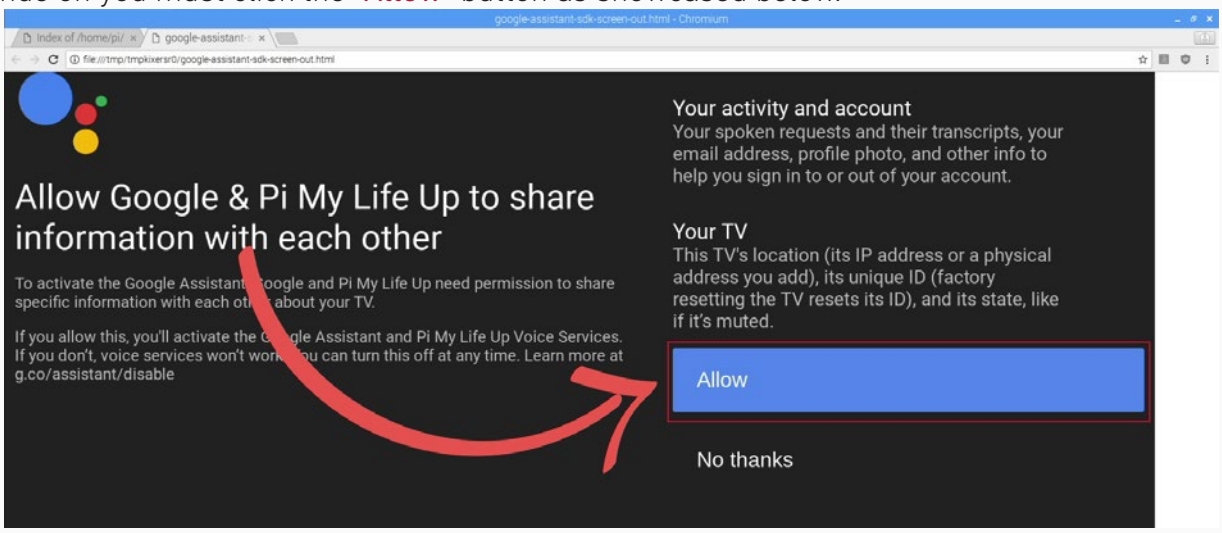

**8.** After selecting "**Continue**" you will now be asked if you agree to a variety of different Google policies, to continue you must click the "**I understand**" button.



# Authorizing your Raspberry Pi for the Google Assistant

**9.** Finally, you will be asked to allow Google and your project the right to be able to share information with each other. Without this, the Google Assistant project will not function correctly.

To continue on you must click the "**Allow**" button as showcased below.




**10.** With that now done we can now use the push to talk Google Assistant sample and hear a response.

This time when you press the "**Enter**" in the terminal and speak an action such as "**What is the time**" you should hear a verbal response and another tab will be automatically opened displaying the action you just called.

Don't worry you can disable the tab behavior by removing the **--display** argument on the command. We only needed this to get up the authorization screen.

If you don't hear any response from the Google Assistant but a new tab does open that contains your results, then you should go back to the "**Setting up your Audio for Google Assistant**" section and the "**Testing your Audio for Google Assistant**" section.

In the next section, we will show you how you can rerun the Google Assistant without having to go searching through all the steps again, make sure you keep your "**Project ID**" and "**Device ID**" handy as we still need those to utilize the samples.

## Using the Google Assistant on the Raspberry Pi

**1.** Now that we have finally fully authorized our Raspberry Pi we will now walk you through the steps on how to run the Google Assistant software without having to go through the entire tutorial.

To begin with whenever you start a new terminal session you will need to put it back into the environment that we set up the Google Assistant software in.

To do this, run the following command on your Raspberry Pi.

```
source env/bin/activate
```

**2.** You will know whether this has worked correctly by seeing **(env)** appear at the front of each line.

With (env) appearing it means you can once again make calls to the Google Assistant samples.

To start up the push to talk sample you will need to run the following command, lucky for us, this time we do not need to write in the project id or the device id as these were cached when we first utilized the push to talk tool.

```
googlesamples-assistant-pushtotalk
```

**3.** In addition to the push to talk sample, there is also the hotword sample code. This sample code will work by listening for certain phrases to trigger it. By default, this will listen for "**Ok Google**" followed by the command that you want to issue to it.

Starting the hotword sample is very much like the push to talk sample. However, you will be required to enter an invalid device id for it to work currently.

This incorrect device ID could be as simple as "**abcd**".

```
googlesamples-assistant-hotword --device-model-id <deviceid>
```

**4.** Hopefully, by now you will have successfully set up your Raspberry Pi with the Google Assistant software.

If you run into audio issues, we recommend you go back to our "**Setting up your Audio for Google Assistant**" section and the "**Testing your Audio for Google Assistant**" section.

## Getting the Google Assistant to start on start up

**1.** Now that you have got your Google Assistant up and running you will likely want to get it to start up on boot rather than having to go to the effort of entering the commands every time.

The easiest way to achieve this is to create a service for it. This service will allow the Google assistant to run in the background on the Raspberry Pi and easily allow us to retain control over it.

To begin, we will write our bash script that will execute the hotword sample. We will be calling this bash script from our service file.

```
nano /home/pi/start_assistant.sh
```

**2.** To this file, you will want to add the following lines.

The first line helps define what should be utilized to run this file, so when the command line interprets it, it will know that it is required to make use of bash.

The second line we use **source** to read and execute commands from the activate file, this will put us into the python environment that we created earlier in the tutorial.

Finally, we run the hotwords Google Assistant sample. Again we are setting an incorrect device id as at the time of writing, using a correct one will break the sample.

```
#!/bin/bash

source /home/pi/env/bin/activate
googlesamples-assistant-hotword --device-model-id abcd
```

**3.** Once you have added those lines to the file, save it by pressing **CTRL + X** then **Y** and finally **ENTER**.

**4.** Now that we have created our bash script we can move onto creating our service file for our Google Assistant by running the following command on our Raspberry Pi.

```
sudo nano /lib/systemd/system/assistant.service
```

**5.** Within this file enter the following lines.

These lines tell the operating system how to handle our service, what user to run it with and if it relies on a particular thing such as that the network is available.

```
[Unit]
Description=Google Assistant
Wants=network-online.target
After=network-online.target

[Service]
Type=simple
ExecStart=/bin/bash /home/pi/start_assistant.sh
Restart=on-abort
User=pi
Group=pi

[Install]
WantedBy=multi-user.target
```

## Getting the Google Assistant to start on start up

**6.** With those lines added to the service file, save it by pressing **CTRL + X** then **Y** and finally **ENTER**.

**7.** Now that we have created the service we need to enable it so that it will actually start on startup.

To do this we just need to use systemctl, type in the following command to enable our new service.

```
sudo systemctl enable assistant.service
```

**8.** With the service now enabled let's try starting it, this will allow us to make sure that everything is working as intended.

```
sudo systemctl start assistant.service
```

**9.** You should now be able to use the "**Ok Google**" hotword to talk and interact with the Google Assistant.

Try it now to ensure the service is working as intended.

If for some reason it is not working you can utilize the command below to display the last output from the Google Assistant sample, keep a lookout for any errors.

```
sudo systemctl status assistant.service
```

**10.** Hopefully, at this point, you will now have your Google Assistant successfully set up to start on boot up.

Hi, how can I help?
ogle As
s. Tell it to do things. It's

# Snowboy Hotword Detection

## Project Description

In this project, we will be showing you how you can set up and use Snowboy on your Raspberry Pi to train and detect your custom hotwords. Additionally, we will also show you how you can utilize the Snowboy library within your python scripts to do specific actions.

To set up the snowboy hotword detection library, we will be walking you through the process of registering for their API, as well as showing you how you can record and train your very own hotword detection models utilizing Snowboys training service.

We will also be exploring in this tutorial how we can use python to utilize the Snowboys detector to use your hotword model and do certain things based on hearing that word.

Best of all you can setup Snowboy as a completely headless solution as it does not require any graphical interface to operate.

## Equipment

**Required**

- Raspberry Pi (Running full Raspbian)
- SD Card (8 GB+ Recommended)
- Network Connection
- USB Microphone
- Speakers

**Optional**

- Raspberry Pi Case

# Preparing your Audio Config for Snowboy

**1.** Before we get started with setting up the Snowboy Hotword detection on our Raspberry Pi, we must first ensure that we have our audio configured correctly.

We will be achieving this by creating a configuration for the audio driver. Before we do this, we must retrieve the card and device numbers for both our audio output and our microphone input.

To retrieve the information that we require, follow the following two steps.

**1a.** Locate your USB microphone by utilizing the following command. Write down both the card number and the device number for it.

```
arecord -l
```

**1b.** Now to locate your speaker, we will be utilizing the following command. Again write down the card number and device number.

Note that the Raspberry Pi's **3.5mm-jack** is typically labeled as **Analog** or **bcm2835 ALSA**, with the **HDMI output** being identified as **bcm2835 IEC958/HDMI**.

```
aplay -l
```

**2.** Now that we have retrieved all the values we need to configure our audio driver we can go ahead and create the **.asoundrc** file.

To begin creating the new **.asoundrc** file run the following command on your Raspberry Pi.

```
nano /home/pi/.asoundrc
```

**3.** Within this file, we need to enter the following configuration lines. These will set up our audio driver by telling it the specific devices that it should be utilizing.

Make sure that you replace **<card number>** and **<device number>** with their respective values that you retrieved during **Step 1**.

```
pcm.!default {
  type asym
  capture.pcm "mic"
  playback.pcm "speaker"
}
pcm.mic {
  type plug
  slave {
    pcm "hw:<card number>,<device number>"
  }
}
pcm.speaker {
  type plug
  slave {
    pcm "hw:<card number>,<device number>"
  }
}
```

**4.** With the lines in, and changes made. Save by pressing **CTRL + X** then **Y** and finally **ENTER**.

## Installing Snowboy to the Raspberry Pi

**1.** Before we begin installing the Snowboy Hotword detection software to our Raspberry Pi let's first make sure that it is entirely up to date.

To do this, you need to enter the following two commands on the Raspberry Pi.

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

**2.** Now that our Raspberry Pi is entirely up to date let's now install all the dependencies that we will be relying on by running that command displayed below.

The bulk of these dependencies are adding the **pyaudio** package that allows us to interact with the audio from Python. Please note that while we are downloading these packages for Python 2, you can download them for Python 3 as well by adding 3 after "python" in the package name.

In addition to the various python packages, we also install the sox package as snowboy has specific requirements for the audio format that your custom hotword is recorded in.

```
sudo apt-get install python-pyaudio python-pip libatlas-base-dev portaudio19-dev
```

**3.** With all our dependencies grabbed from the package manager we now need to install the python bindings for the **PortAudio** software.

Install the python bindings for PortAudio by running the following command on your Raspberry Pi.

```
sudo pip install pyaudio
```

**4.** For our final dependency, we will need to install a python package called requests, run the following command on the Raspberry Pi to install it using pip.

This python package will allow us to interact with Snowboy's Restful API from our Raspberry Pi. This Python package will be helpful for sending our hotword samples to their service.

```
sudo pip install requests
```

**5.** Finally, we can download Snowboy itself to our Raspberry Pi, to do this use the following command on your Raspberry Pi to grab the latest compiled version for the Raspberry Pi.

This version of Snowboy should work on all current versions of the Raspberry Pi.

```
wget -O snowboy.tar.bz2 https://go.pimylifeup.com/napoRs/snowboy
```

**6.** Now that we have grabbed the latest version of the Snowboy software to our Raspberry Pi we need to extract it by running the following command.

Make sure when you issue the following command that you are located in the Pi users home directory.

```
tar xvjf snowboy.tar.bz2
```

**7.** With the snowboy software now downloaded to our Raspberry Pi, we need to rename the directory so that it will be easier to deal with later. Run the following command to rename the folder to snowboy.

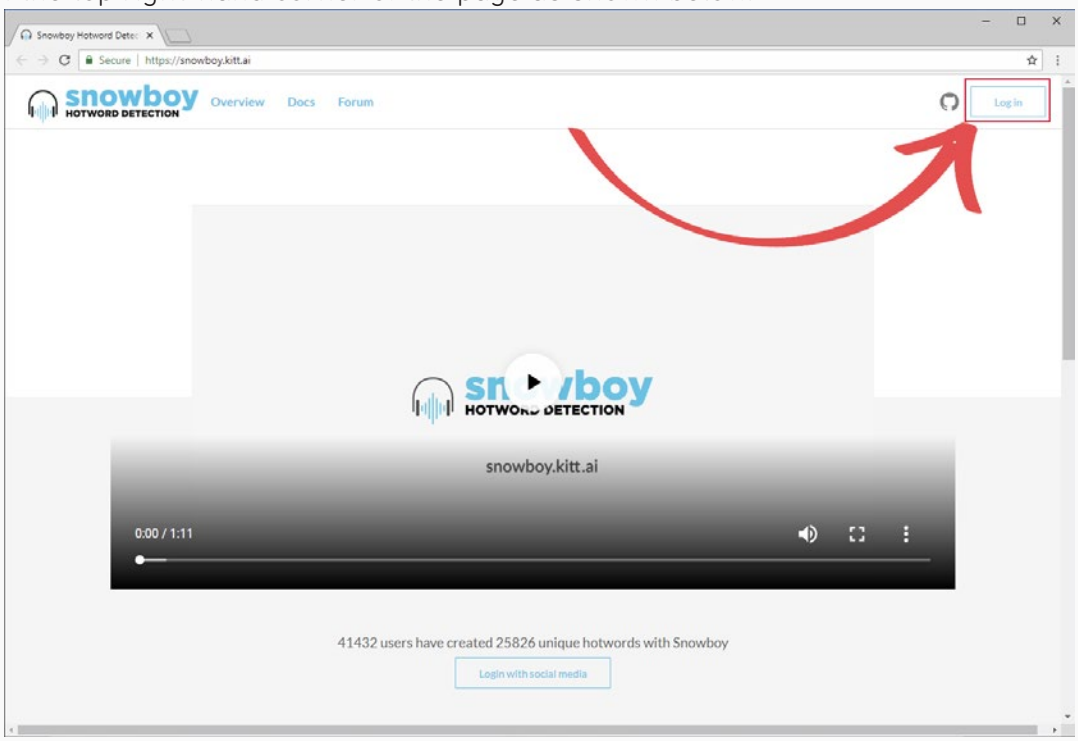
```
mv rpi-arm-raspbian-8.0-1.1.1/ snowboy/
```

# Getting your Snowboy API Key

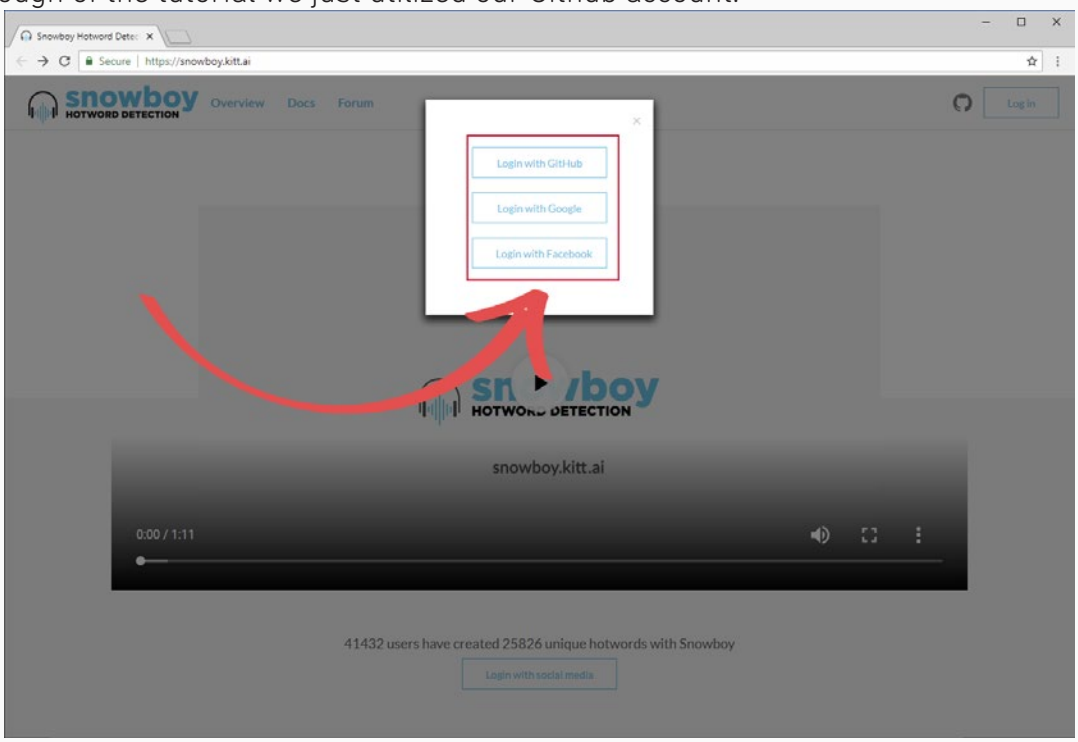
**1.** For the next section of our tutorial, we will require an API key for Snowboy. Over the following few steps, we will be walking you through the straightforward process of getting one.

Please note that the Snowboy site only supports OAuth for login so you must have either a Github, Facebook or Google account.

Begin by going to the Snowboy website at https://snowboy.kitt.ai/ and clicking the "**Log in**" button that is located in the top right-hand corner of the page as shown below.



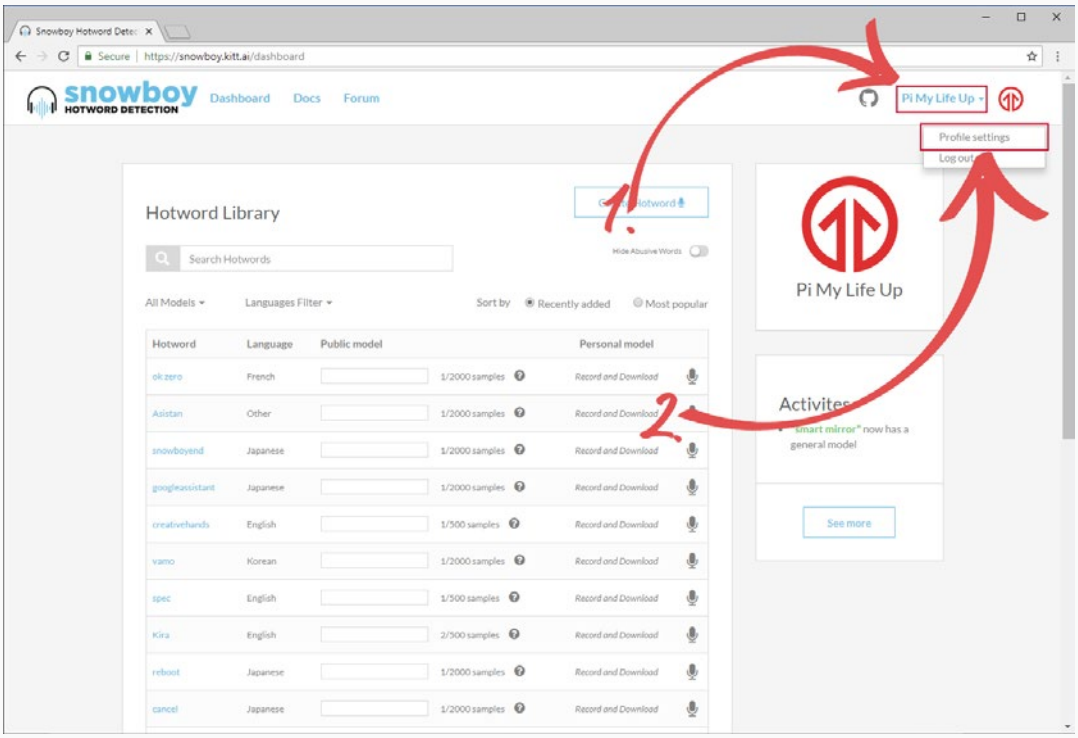

**2.** You will now be asked to choose from three different options to log in to the Snowboy service, for our run through of the tutorial we just utilized our Github account.



# Getting your Snowboy API Key

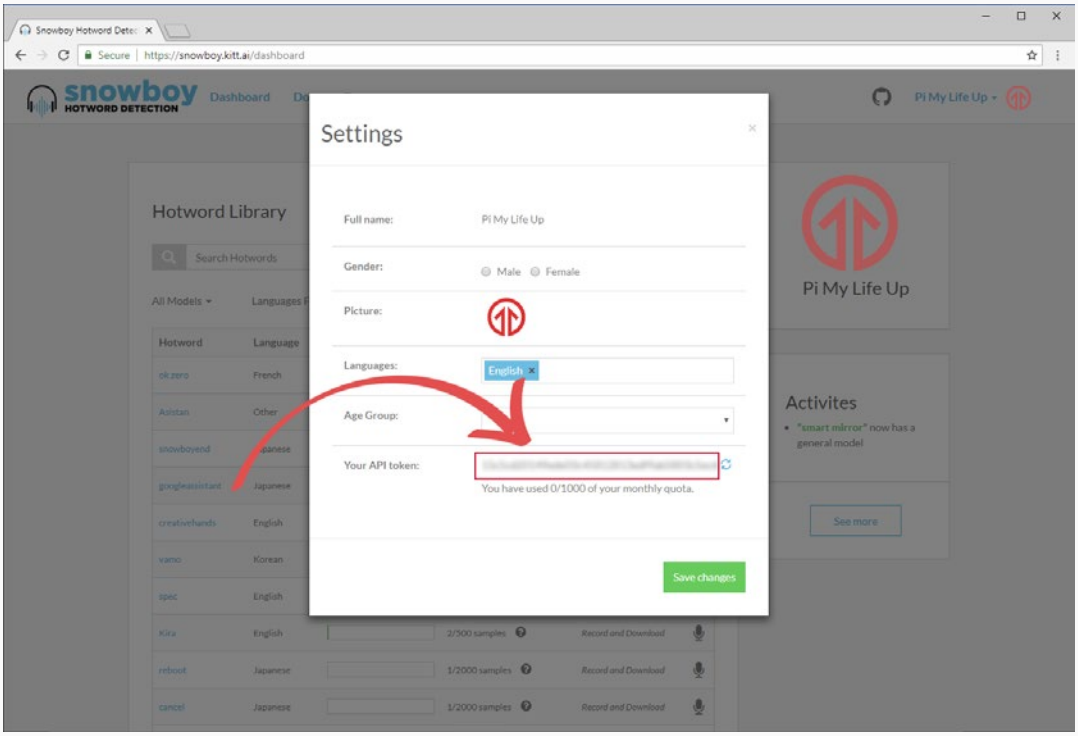
**3.** Once logged in you will be shown the screen below, this will list all the currently calculating hot-words. However, what we are after is located in the top right-hand corner.

Click your displayed name (**1.**) then click "**Profile Settings**" (**2.**), this will take us to a page where the API key will be displayed.




**4.** Finally, on this screen, you need to copy down the API token as you require this to talk with the Snowboy server later on in this guide.

We have highlighted the area where the token will be displayed in our screenshot below.



## Recording your custom hotword for Snowboy

**1.** For this part of the tutorial, we will need to write a python script that will send our hotword recordings to the Snowboy service from our Raspberry Pi.

We won't be delving too much into this script as it is relatively simple and is just a small stepping stone in this tutorial.

Create this script by running the following command on the Raspberry Pi.

```
nano /home/pi/training_service.py
```

**2.** Within this file enter the following lines of python code.

Remember to replace **ENTER_TOKEN** with the API token you retrieved in the previous segment and **ENTER_HOTWORD** with the hotword that you want to record such as "**Pi My Life Up**".

```
import sys
import base64
import requests

def get_wave(fname):
    with open(fname) as infile:
        return base64.b64encode(infile.read())

endpoint = "https://snowboy.kitt.ai/api/v1/train/"

token = "ENTER_TOKEN"
hotword_name = "ENTER_HOTWORD"
language = "en"
age_group = "20_29"
gender = "M"
microphone = "usb microphone"

if __name__ == "__main__":
    try:
        [_, wav1, wav2, wav3, out] = sys.argv
    except ValueError:
        print "Usage: %s wave_file1 wave_file2 wave_file3 out_model_name" % sys.argv[0]
        sys.exit()

    data = {
        "name": hotword_name,
        "language": language,
        "age_group": age_group,
        "gender": gender,
        "microphone": microphone,
        "token": token,
        "voice_samples": [
            {"wave": get_wave(wav1)},
            {"wave": get_wave(wav2)},
            {"wave": get_wave(wav3)}
        ]
    }

    response = requests.post(endpoint, json=data)
    if response.ok:
        with open(out, "w") as outfile:
            outfile.write(response.content)
        print "Saved model to '%s'." % out
    else:
        print "Request failed."
        print response.text
```

**3.** Once you have entered all the lines, save the file by pressing **CTRL + X** then **Y** and then **ENTER**.

## Recording your custom hotword for Snowboy

**4.** Now that we have written out the training service python script we can move onto actually recording our hotword to send with it.

To do this, we will be utilizing the **arecord** tool as this allows us to specify particular options such as the sample rate, the format and the duration of the recording.

For our recording we will be making a simple **5-second** recording, this should be more than long enough to record our hotword.

Run each of these commands to generate our 3 hotword recordings, making sure that you say your hotword once during the recording.

The Snowboy software will use these three recordings to create a reference file so that it can easily listen for it.

```
arecord --format=S16_LE --duration=5 --rate=16000 --file-type=wav 1.wav

arecord --format=S16_LE --duration=5 --rate=16000 --file-type=wav 2.wav

arecord --format=S16_LE --duration=5 --rate=16000 --file-type=wav 3.wav
```

**5.** With the three samples of our hotword now recorded we can proceed to send these to the Snowboy service.

Here is where the Python script we wrote earlier in the tutorial will come in handy. We need to call the python file, referencing our three recordings and our desired output name for the processed sample.

Our python script will process the files generating a base64 encoded version of the file, bundle this along with some other information and send it off to Snowboy and await their response.

Run the following command to send off your three samples to the training service.

```
python training_service.py 1.wav 2.wav 3.wav saved_model.pmdl
```

**6.** Before we utilize our new model we should copy it into our snowboy folder, this will make it easier to use in our Python scripts later on in this guide.

To copy the file, we can run the following two commands. The first one will copy the model file the second one will move us into the snowboy directory.

```
cp saved_model.pmdl ./snowboy/saved_model.pmdl
cd snowboy
```

**7.** To test out our new model, we will utilize the **demo.py** script that came with snowboy. This script allows us to specify the model that we want to use.

Just run the following command from your current directory to start up the script.

Make sure you do not run this script as sudo as it will cause issues with reading from the microphone since we set up our .asoundrc configuration file for the pi user.

```
python demo.py saved_model.pmdl
```

## Recording your custom hotword for Snowboy

**8.** When loading up the demo python script you may run into various error messages, most of these are safe to ignore, and you can proceed with utilizing the demo without any issues.

If you see the error message "**IOError: [Errno -9997] Invalid sample rate**" appear in the command line make sure that you have correctly setup your **.asoundrc** file as specified earlier in the tutorial and are using a microphone that can record at **16,000 HZ**. Most microphones should be fine with this.

If everything is working as intended, you should see the following message appear in the command line, and you will be able to speak your hotword to it.

```
Listening... Press Ctrl+C to exit
```

**9.** Upon hearing your hotword, you should see the following message appear in the command line and hear a beep through the audio.

This message indicates that snowboy successfully identified what you said and fired off to its callback function.

```
INFO:snowboy:Keyword 1 detected at time: 2018-09-26 04:38:41
```

**10.** In the next section, we will explore how you can implement snowboy in a simple python script to give you an idea of how you could utilize this in your projects.

## Using Snowboy in a Python script

**1.** For this section of the tutorial, we will be writing a straightforward Python script that uses the snowboy decoder library to load in our model and then listen for it.

To begin writing our script just run the following command within the terminal on the Raspberry Pi. For our purposes, we will be calling this python script **snowboy_sample.py.**

We create this file within the **snowboy** folder otherwise we would not be able to access the snowboy-decoder class. The reason for this is that pythons import functionality only searches the python path and the local directory.

```
nano /home/pi/snowboy/snowboy_sample.py
```

**2.** Within this file enter the following lines of code.

We will explain what each line of code does as we go along. If you would prefer to skip the explanation, you can move onto the next step to copy the code as one block.

```
import snowboydecoder
```

This first line of code imports the snowboydecoder so we can access its functionality. This class will handle processing our voice model and listening for it.

```
def detected_callback():
        print "Hotword Detected"
```

These two lines define our callback function. This function is simple in that it that will just print the text "**Hotword Detected**" to the command line whenever it is called.

```
detector = snowboydecoder.HotwordDetector("saved_model.pmdl", sensitivity=0.5, audio_gain=1)
```

This line is one our most important as it instantiates our hotword detector. To this function, we need to pass in a few things. The first is the location of our voice model. In our case, this is in the same directory so we can pass in the filename.

The next argument is the **sensitivity**. This argument defines how easy it is for your hotword to be triggered, the closer this value is to 1 the less accurate the keyword needs to be, the closer to 0 the value is the more precise you have to be to your original recording.

Increase or decrease the sensitivity value if you are finding that you have too many false positives or that you are finding it difficult to trigger your hotword.

The final argument that we define is **audio_gain**. This argument determines how much gain should be applied to the input volume, values greater then 1 increase the gain, values lower then 1 decreases the gain.

Fiddle with this value if you are struggled to be heard by your microphone.

```
detector.start(detected_callback)
```

Our final line utilizes the detector that we instantiated on the previous line and tells it to start listening.

We pass in the name of the function we wrote as the detector will automatically call this everytime it hears your hotword. This function will continually run until terminated.

## Using Snowboy in a Python script

**3.** With all those lines entered into the python script you can compare it to what we have below to ensure you have entered everything correctly.

```
import snowboydecoder

def detected_callback():
        print "Hotword Detected"

detector = snowboydecoder.HotwordDetector("saved_model.pmdl", sensitivity=0.5, audio_gain=1)

detector.start(detected_callback)
```

**4.** Once done, save the file by pressing the following keys, **CTRL + X**, then **Y** and finally **ENTER**.

**5.** With our script now written we can test out to see if it is working correctly.

To run the script, all you need to do is enter the following line into the command line.

```
python snowboy_sample.py
```

**6.** Now everytime you say your hotword the text "**Hotword Detected**" should appear in the command line as it will automatically call the "**detected_callback():**" function that we wrote.

You can quit out of this script by pressing **CTRL + C**.

# FlightAware ADSB Receiver

## Project Description

In this tutorial, we will show you how to set up a Raspberry Pi ADS-B receiver. This receiver will allow us to track flights within 100-300 miles (160-480 km).

This distance will vary depending on your location, line of sight, etc. We will also show you how to set this up, so it helps contribute to **flightaware.com**'s flight tracking.

As an incentive for sharing the data collected by the receiver with **FlightAware**, your account will be upgraded to an enterprise account for free. If this is something you have been wanting, then you should consider investing in a Raspberry Pi setup as you will be saving money within a month or two.

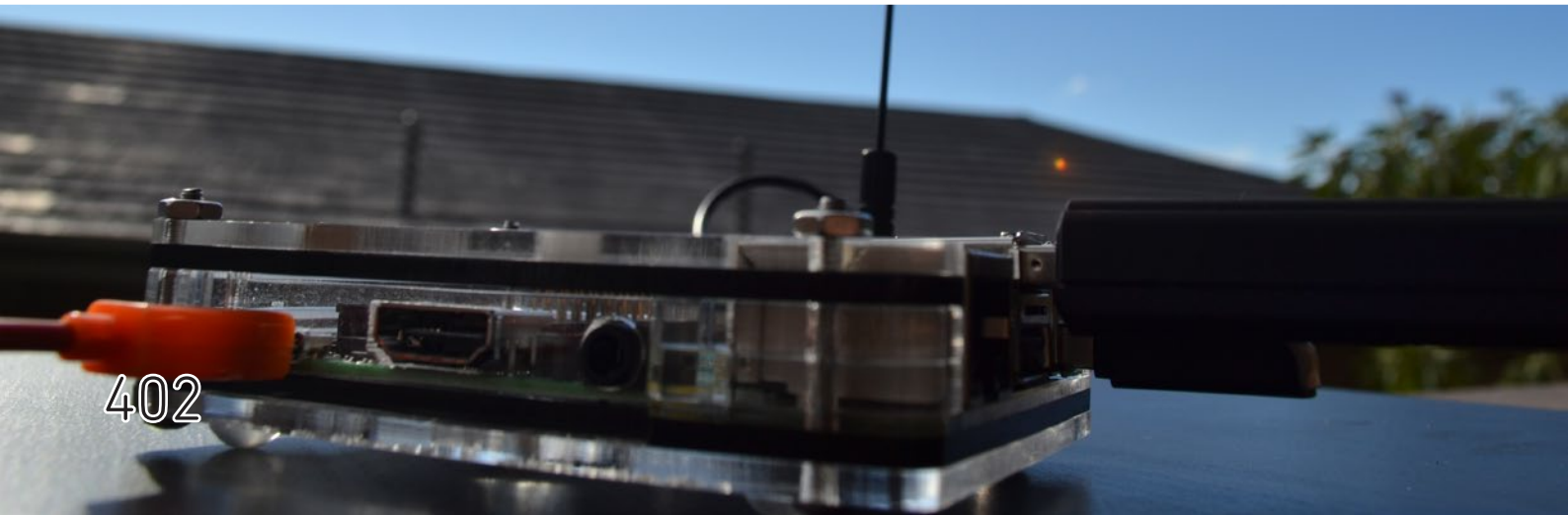
There are two ways of installing and setting up the FlightAware software. The first is to start from scratch and install a specially modified copy of Rasbpian to a formatted SD Card. The other is to install a package onto a normal Raspbian install that includes the required files to connect to FlightAware.

We will go through how to set it up through a fresh, clean install, or for installing the software onto Raspbian.

## Equipment

**Required**
- Raspberry Pi
- Mini DVB-T Digital TV USB Dongle
- 4 GB (Micro) SD Card (8 GB+ Recommended)
- Raspberry Pi Power Supply
- Internet Connection

**Optional**
- USB Keyboard
- USB Mouse
- HDMI Cable
- Raspberry Pi Case



# Step 1 - Formatting your SD Card

First off you should do when installing a new image to an SD card is format it correctly.

The following steps will show you the best way to format your SD Card for the Raspberry Pi:

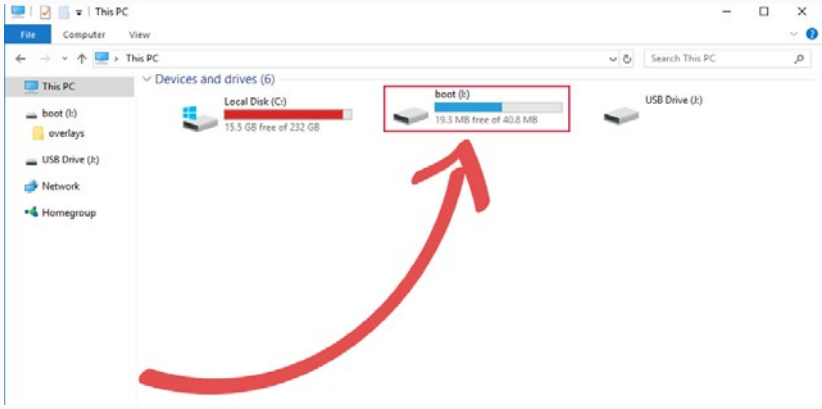
**1.** To begin you need a tool to format your SD Card. You can obtain one of the most robust ones from here: **https://www.sdcard.org/downloads/formatter_4/**

Once you are on the website will find a download link for the SD Formatter.

**2.** Download and install the SD Card Formatter from the SD Card association.

**3.** With the SD Formatter installed to your computer, insert your SD card and check the drive letter that is allocated to it, e.g., **G:/**.

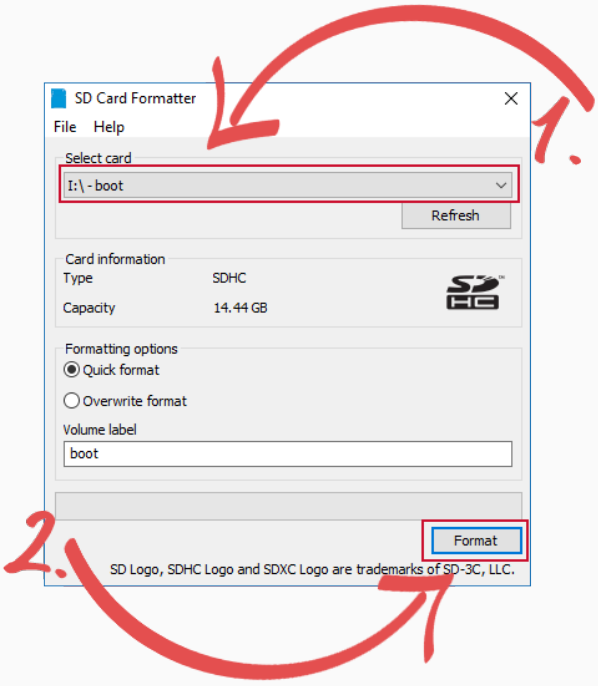
Make sure that you have the correct drive letter otherwise you could accidentally format the wrong device.




**4.** With your SD Card now plugged into your computer, launch the SD Formatter software that we installed earlier.

First set the drive letter to the correct one **(e.g., G:/)**.

Once you have everything set correctly, press the "**FORMAT**" button.



# Step 2 - Installing FlightAware from Scratch

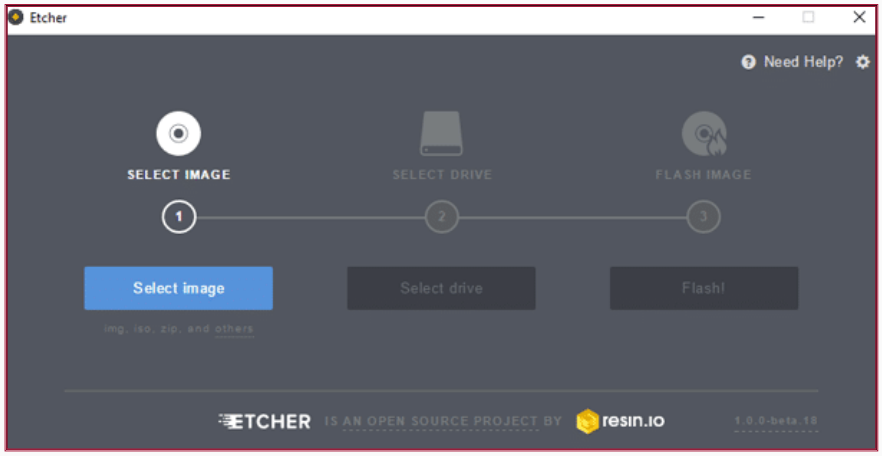
**1.** First head over to the PiAware website at **http://piaware.flightcdn.com/piaware-sd-card-3.5.3.img.zip** and download the latest zip file.

Once downloaded, unzip the file so, you are left with a **.img** file.

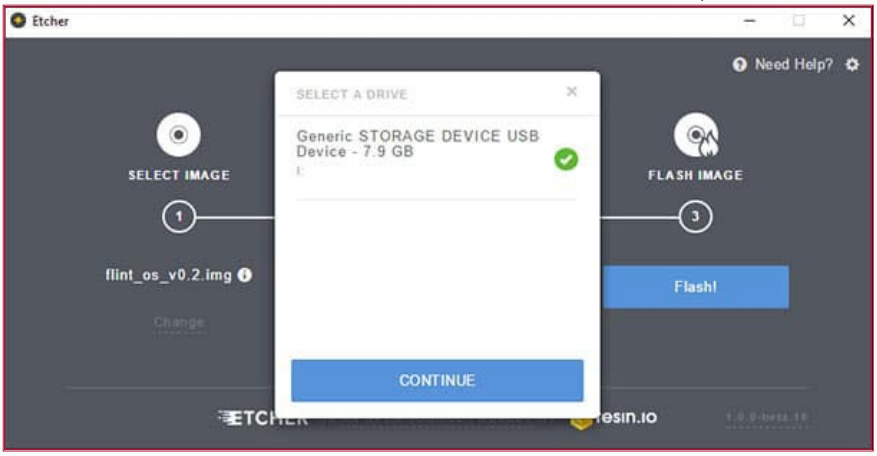
**2.** Next, you will need some tools to be able to write the image to the SD card.

For this we will use a tool called Etcher, you can download at https://etcher.io/.

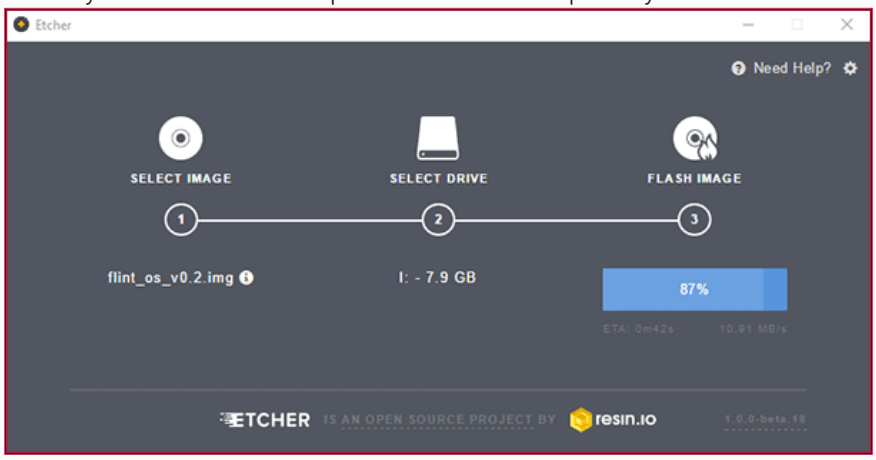
**3.** Open Etcher and press the "**Select Image**" button. Browse to the image you downloaded in **Step 1** and select it.




**4.** Next click the "**Select Drive**" button. With the correct drive selected, press "**Continue**".




**5.** Finally, click the "**Flash!**" button to begin the flashing process. This can take some time. Once completed, take out your SD Card and put it into the Raspberry Pi.



## Step 3 - Finalising the setup of FlightAware

**1.** Plug in the SD Card, the USB ADS-B dongle + antenna cable and the rest of the required equipment into the Raspberry Pi.

**2.** Plug in the power supply to power on the Raspberry Pi.

Once it has booted up, be prepared to have to sit and wait for about 5 minutes for the software to start up and start contacting the FlightAware server.

**3.** You can now associate your new Raspberry Pi ADS-B receiver with flightware.com at their website by going to https://flightaware.com/adsb/piaware/claim.

You will need to create an account, and it should be able to recognize your piaware device straight away.

If it doesn't detect it you can set it up manually by using the following command in the terminal:

```
sudo piaware-config -user <username> -password <password>
```

**4.** Once you have issued that command, you will need to force the PiAware software to restart.

You can achieve this by entering the following command into the terminal:

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

**5.** FlightAware will start processing the data from the Raspberry Pi pretty much straight away. You should have viewable data within about 30 minutes.

You can see the local stream by going to your web browser and entering your Raspberry Pi's IP address followed by port **8080**.

You can use the following command to find your Raspberry Pi's IP address:

```
hostname -I
```

**6.** With the IP address given by this, add **:8080** to the end of it. As we have shown below.

```
192.168.1.108:8080
```

# Pre-existing install - installing the Software:

## Downloading and Installing PiAware

**1.** First we will need to get the Raspberry Pi's package manager to find FlightAware's software packages, we can do this by entering the following commands into terminal:

```
wget http://flightaware.com/adsb/piaware/files/packages/pool/piaware/p/piaware-support/piaware-repository_3.5.3_all.deb
sudo dpkg -i piaware-repository_3.5.3_all.deb
```

**2.** Now that we have the flightaware package now linked in with raspbian, we can go ahead and install the software onto the Raspberry Pi with the following two commands:

```
sudo apt-get update
sudo apt-get install piaware
```

**3.** With the piaware software installed, we can also enable both automatic and manual updating for the piaware software. Do this by entering the following command:

```
sudo piaware-config allow-auto-updates yes
sudo piaware-config allow-manual-updates yes
```

**4.** Next, we need to install the ADS-B receiver software, one of, if not the best software around is dump1090. We can do that with the following apt-get command:

```
sudo apt-get install dump1090-fa
```

**5.** We need to restart the Raspberry Pi to ensure that all the software successfully starts cleanly and starts communicating with each other, you can do that with this command:

```
sudo reboot
```

**6.** You can now associate your new Raspberry Pi ADS-B receiver with flightware.com at their website by going to https://flightaware.com/adsb/piaware/claim.

Associating the Pi should allow the software to detect it automatically, if it doesn't run this command:

```
sudo piaware-config -user <username> -password <password>
```

**6.** Once you have issued that command, you will need to force the PiAware software to restart, you can achieve this by entering in the following command into terminal:

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

**7.** Flightaware will start processing the data from the Raspberry Pi pretty much straight away. You should have viewable data within about 30 minutes.

You can use the following command to find your Raspberry Pi's IP address:

```
hostname -I
```

**8.** With the IP address given by this, add **:8080** to the end of it as we have shown below.

```
192.168.1.108:8080
```

# Tips and Troubleshooting

## The ADS-B receiver doesn't appear to picking up much

The signals from aircraft aren't designed to penetrate solid objects. Meaning for the best reception the antenna should be located with a line of sight of the sky with little to no obstructions. In the ideal situation, an antenna would be mounted on a roof connected back to the Raspberry Pi.

However, I just ran mine in the window, and it seem to pick up aircraft just fine.

## How do I setup WiFi

If you are using the PiAware version of Raspbian, you will likely want to go and set up your network. You can follow our guide located earlier in this book on how to set that up.

## Expand the file system to the entire SD card

Since Raspbian is not installed via NOOBs you will need to expand the filesystem to cover the entire SD card manually.

This task can be done through the terminal by using the raspi-config tool. You can access this by typing the following in the terminal **sudo raspi-config**.

The option you are after is the first one on the list **"1 Expand Filesystem".** This tool will expand the file system of the Raspberry Pi to the entirety of the SD Card.

## The Default Password is different

The PiAware version of Raspberry Pi we installed on the Raspberry Pi has a different default password then what you would find in the standard installation of Raspbian. The username, however, is still the default of standard Raspbian, that being **pi**

The default password has been changed to **flightaware**. It is a good idea you change this to something different.

To do this enter **passwd** and follow the instructions on how to update the password.

# Flightradar24 ADSB Receiver

## Project Description

In this Raspberry Pi Flightradar24 tutorial, I go through the steps to set up a flight tracker that feeds to the FlightRadar24 service.

I had covered an ADS-B tracker before but utilized the FlightAware software. Depending on personal preference you may prefer to use Flightradar24 instead. Both have their pros and cons.

One of the most significant benefits to running your own flight tracker is that you get the Flightradar24 business account for free.

That's a $499 yearly subscription you will get for free in exchange for sharing the ADS-B data. Perfect if you love tracking flights around your house.

The distance that your ADS-B tracker will be able to see depends on the location and line of sight of your DVB-T antenna. It should cover roughly 140-480 km, maybe more or less.

## Equipment

**Required**
- Raspberry Pi
- Mini DVB-T Digital TV USB Dongle
- 4 GB (Micro) SD Card (8 GB+ Recommended)
- Raspberry Pi Power Supply
- Internet Connection

**Optional**
- USB Keyboard
- USB Mouse
- HDMI Cable
- Raspberry Pi Case

## Signing up to Flightradar24

Before you get started, you will want first to sign up Flightradar24. It's a pretty simple process and will make things a little easier later. You can sign up to Flightradar24 at their website, which you can find here: https://www.flightradar24.com/premium/signup

You will need to confirm your email once you have signed up. Just select the basic (free) plan as that is all we need to get started with Flightradar on the Raspberry Pi.

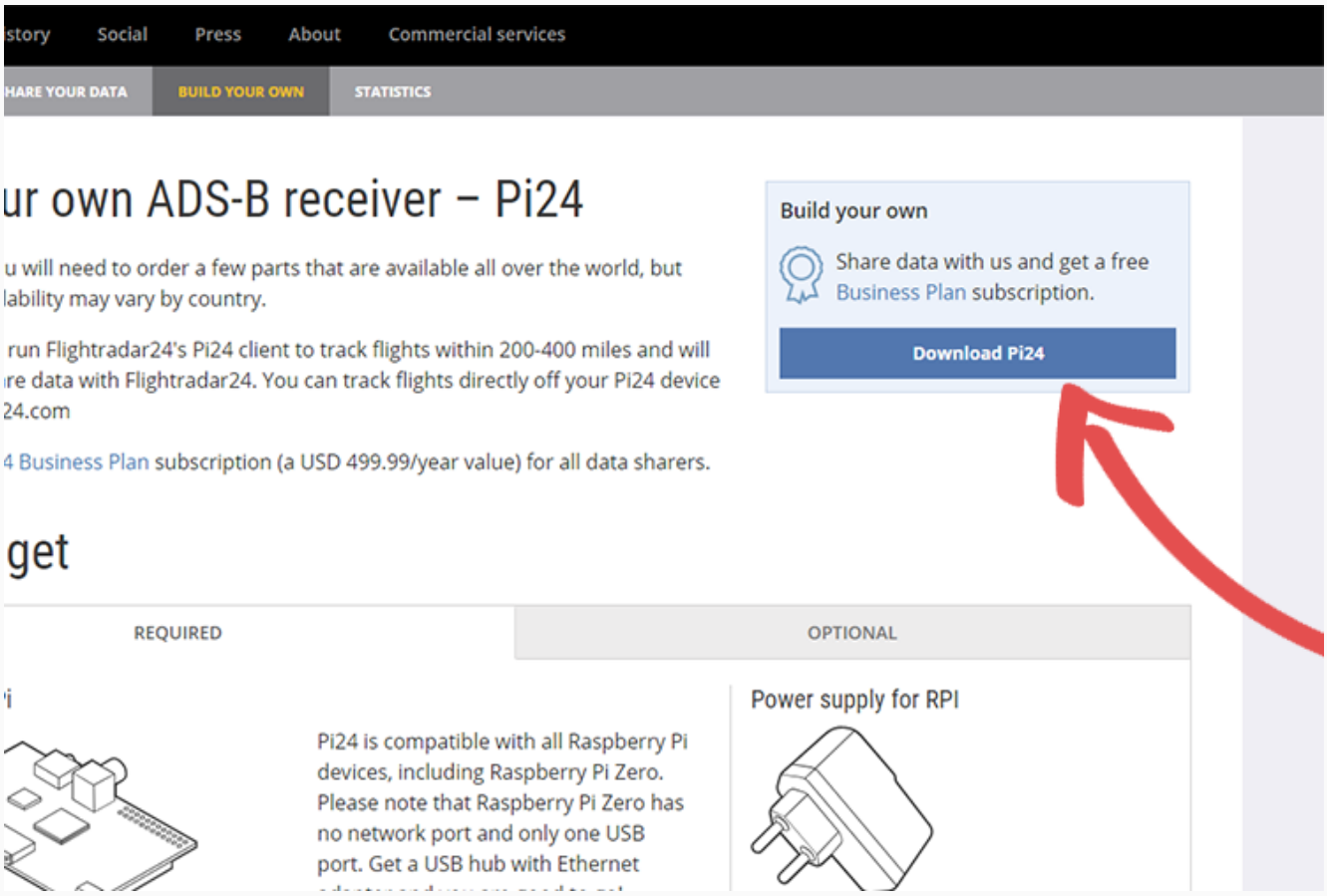
As an added bonous Flightradar24 will award you with their business account for free once they start recieving data.

## Installing the Flightradar24 from Scratch

This section goes through the steps of installing Flightradar24 from complete scratch. It involves wiping an SD card and everything on it. This option is the easiest if you don't mind wiping your SD card.

**1.** To begin with we must first download the latest image of Flightradar 24 from their website.

You can obtain it by going to the following link https://www.flightradar24.com/build-your-own and clicking the "**Download Pi24**" button.

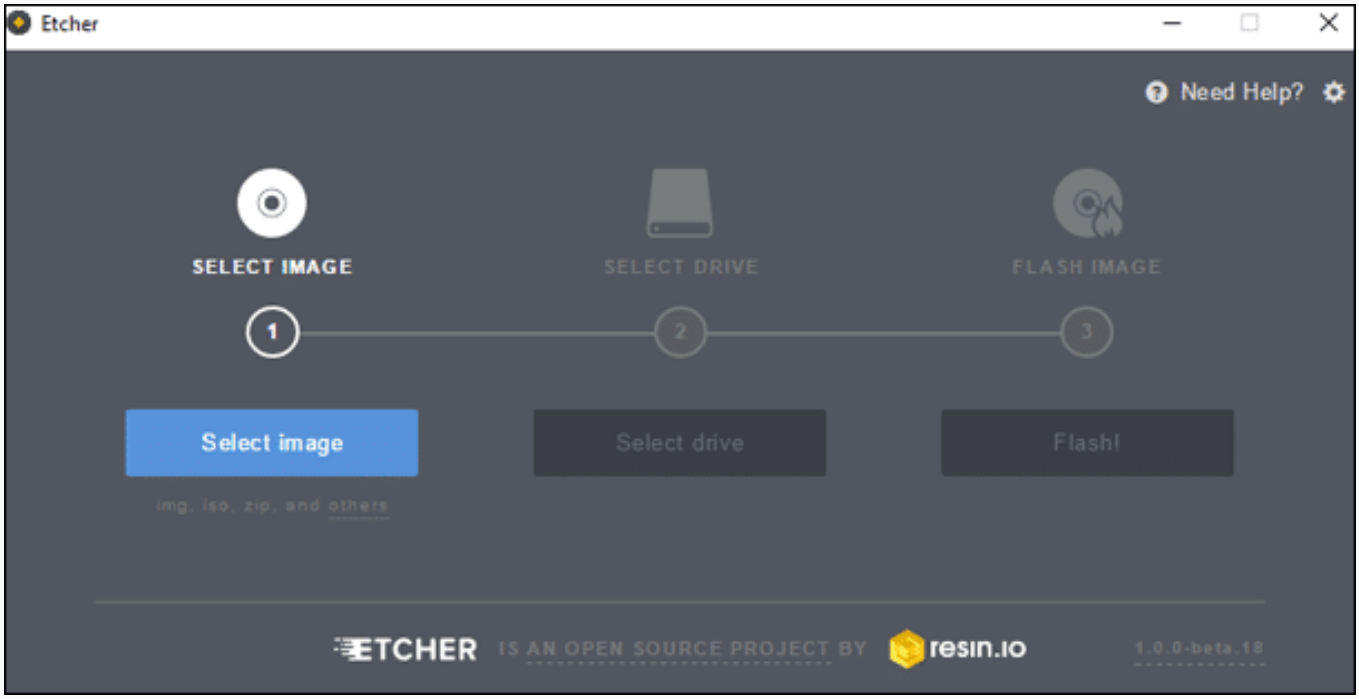



**2.** You will now need a tool to be able to write the image to the SD Card. Etcher is a great solution and is available for download on their official website at https://etcher.io/.

It will work on most popular operating systems such as Windows, Linux, and Mac OSx.

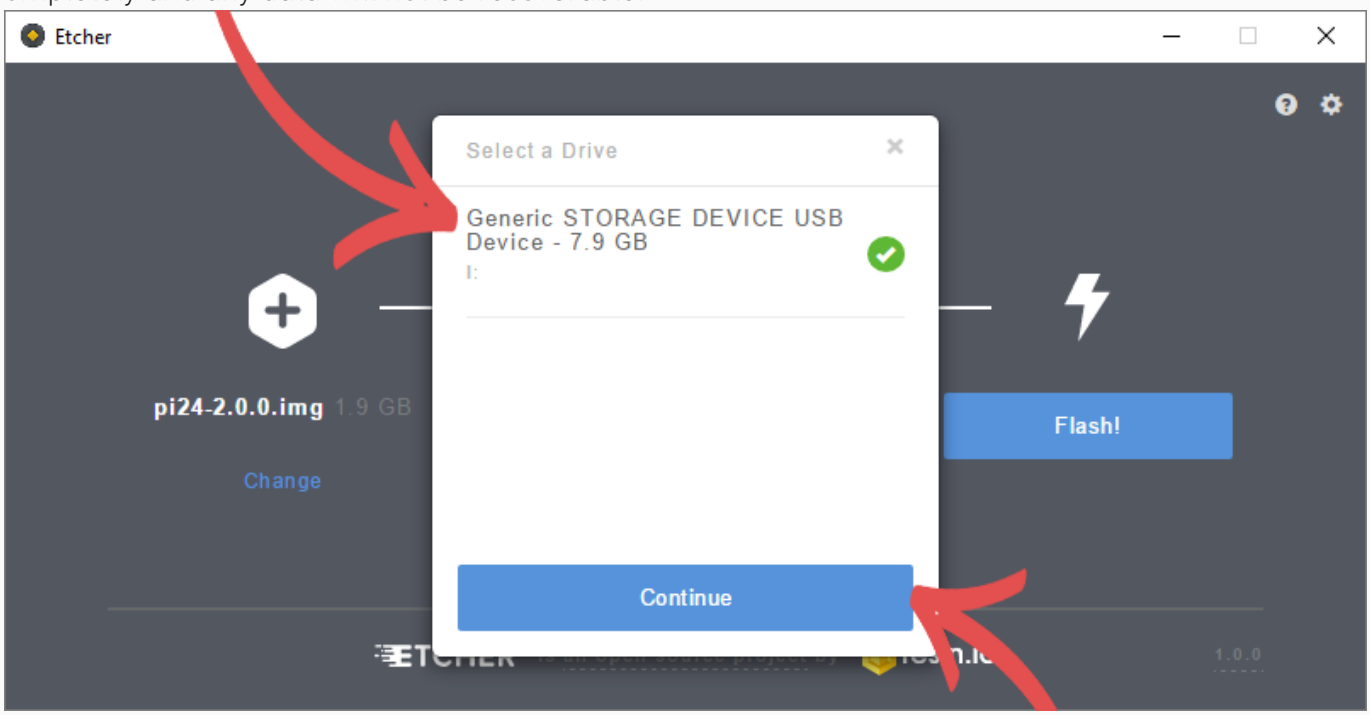
**3.** Once you have installed Etcher, open it up.

Next you need to click the, "**Select Image**" button and then browse to where we just download the Pi24 image.



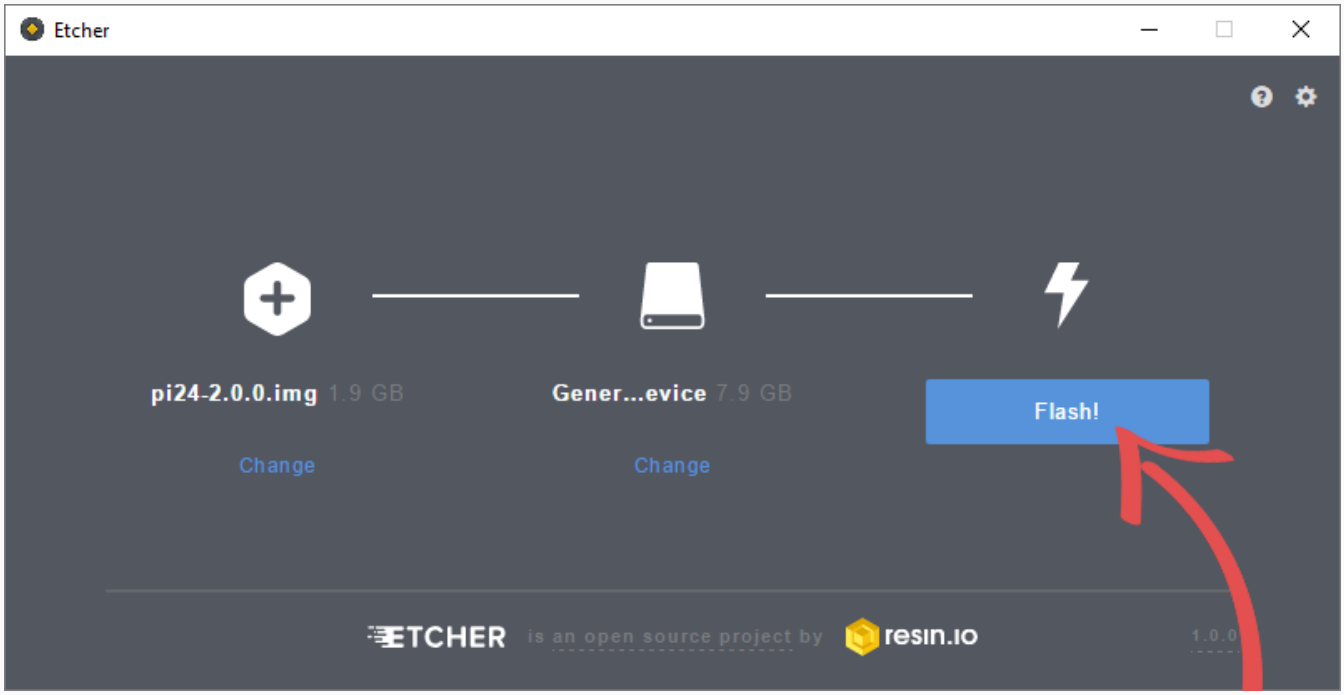

**4.** In this next step, you will need to select the drive that you wish to write the image to. Once you have selected the correct drive click the "**Continue**" button.

It is very important that you make sure you select the correct drive as this process will wipe the drive completely and any data will not be recoverable.



**5.** Once you are ready to go, click the "**Flash!**" button to begin the process of writing to the SD Card.

Please note this process may take some time to complete.




**6.** Once completed, insert the SD card into the Raspberry Pi and power it on.

You will need access to the terminal to complete the next few steps.

To access the terminal on the Raspberry Pi you will need to press **CTRL** + **ALT** + **F2**.

The login details are the defaults for the Raspbian operating system.
The username is **pi**, and the password is **raspberry**.

**7.** If you have an ethernet cable connected then the Pi should have network access.

Otherwise, you will also want to set up WiFi, follow our WiFi tutorial to set up a WiFi connection within the terminal.

You can find this tutorial in our "**Getting Started with the Raspberry Pi**" book.

**8.** Once the network is set up, go to the terminal either on the Raspberry Pi or via SSH and enter the following line.

Do **NOT** proceed if your Pi lacks an internet connection otherwise this step will fail.

```
sudo fr24feed --signup
```

Go to the "**FlightRadar24 Configuration**" section in this tutorial to proceed with the configuring the software.

## Installing the Flightradar24 on Raspbian

If you already have Raspbian installed and don't wish to start from scratch, then this is the best option for you. The process is easy, and you shouldn't run into any issues.

**1.** Enter the following command into the terminal on the Raspberry Pi.

This command will download the install script directly from Flightradar24 themselves and pipe the install script directly through to bash.

Normally you shouldn't pipe scripts straight through to bash however flightradar24 is a trustworthy provider.

You can verify the script yourself by going to the http://repo.feed.flightradar24.com/install_fr24_rpi.sh in your web browser.

```
sudo bash -c "$(wget -O - http://repo.feed.flightradar24.com/install_fr24_rpi.sh)"
```

**2.** You will next be asked some questions such as the fr24 sharing key, antenna position, and some other details.

Over on the next page we will explain what you should be entering for each of these options in our "**Flightradar24 Configuration**" section.

# FlightRadar24 Configuration

In this section, we will be going through all the configuration questions that you will be asked at setup or when you run the following command.

```
sudo fr24feed --signup
```

**Email Address:** Enter the same email you signed up to Flightradar24 using. This email address is also where your FR24 key will be sent.

**FR24 Key:** If you have never used Flightradar24 to feed data then you will not have an FR24 key. Leave this question blank unless you have an FR24 key.

**MLAT Calculations:** Enter **Yes** for this question.

**Latitude & Longitude:** Use a website like latlong.net to calculate your latitude and longitude of the antenna. Try and be as accurate as possible as this will help verify your data.

Latitude is in the format of **xx.xxxx** and longitude is **xx.xxxx** (Leave out any extra numbers)

**Altitude:** Use a site like whatismyelevation.com to work out how high the antenna is from sea level.

**Receiver Selection:** If you're using the DVB-T stick then select **1**. Otherwise observe the other choices you have here.

**Dump1090 Arguments:** Leave this **blank** and proceed by pressing **Enter**.

**Raw Data Feed:** Enter **no**, unless you know you will need it.

**Basestation data feed:** Enter **no**, unless you know you will need it.

**Logfile mode:** Set this to **48-hour, 24 rotation**.

**Logfile path:** If you want a specific path for where you want the log files to be stored then enter something here otherwise just press enter to proceed.

**1.** The program should now return that the process has been completed and provide you with a sharing key. Save this key as you may need it in the future.

**2.** To get the Flightradar24 software to start sending data, enter the following command. This command will restart the software and get it to load in our new configuration information.

```
sudo systemctl restart fr24feed
```

**3.** You can check the status of your feed by running the following command.

```
fr24feed-status
```

**4.** If you go to the data sharing page at "https://www.flightradar24.com/account/data-sharing" on the FlightRadar website, you should also now see your feed.

**5.** If you want to see the web interface of your feed, then go to its IP in the web browser followed by the port **8754**. Below is an example of mine.

```
http://192.168.0.143:8754
```

## Troubleshooting Flightradar24 issue

- It might take some time before your feed will show as online on the Flightradar24 website. It took roughly 30 minutes until it showed up as online.

- MLAT initially wasn't working but eventually started working within a few hours without any extra configuration.

- Antenna location dictates how many planes you will be track. Try and get it so that the tracker will have an ideal line of sight of the sky.

  Hills, building and anything substantial will block the ADS-B receiver from tracking accurately.

- It is highly recommended that you update the default password of the Raspbian. This change will help secure your Pi against hackers.

- You can install more than one feed to the Raspberry Pi. If you want to also feed to FlightAware, you can also set that up too.

  However, dump1090-mutability will need to be installed before either of the feeds.

- Low power to the Raspberry Pi can cause issues.

  If you notice your Pi is having power issues then try replacing the power supply with something more powerful and stable.

Bookmarks
Terms of Use
BROADCOM
Raspberry Pi 3 Model B v1.2
© Raspberry Pi 2015

# BOINC Client

## Project Description

In this Raspberry Pi BOINC tutorial I take you through the steps to getting BOINC setup and processing tasks correctly.

In this tutorial, we will be installing the SETI at home project. For those who don't know SETI@home is a program that will enable you to help in the search for extra-terrestrial intelligence. It simply downloads and analyses radio telescope data. Once processed it sends the data back to the main server.

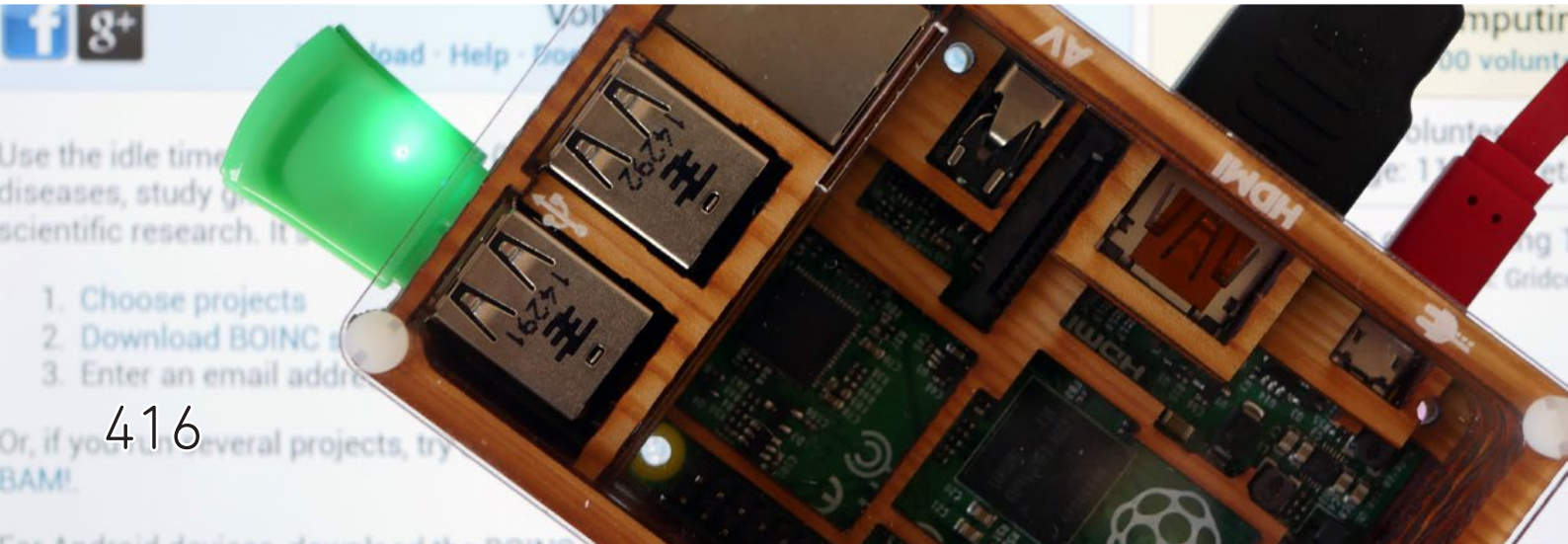
If you're familiar with the BOINC program, then you will probably already know that SETI@home is the most popular project that you're able to sign up to. There are others that you can try but some may not be available for the Raspberry Pi. Seti@home for example was only recently officially released for the Pi.

You can find the full list of projects you can join and their supported platforms over at the BOINC project page here: **http://boinc.berkeley.edu/projects.php**

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection
- External Hard drive or  USB Drive

**Optional**
- USB Keyboard
- USB Mouse



## Installing Raspberry Pi BOINC

Installing BOINC is a straightforward process, and you shouldn't run into any problems. I am using just basic Raspbian in this tutorial if you need to install it then be sure to check out my tutorial on how to install Raspbian by using NOOBs.

**1.** Firstly, make sure everything is up to date by running the following commands:

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

**2.** Now install the BOINC package by running the following command:

```
sudo apt-get install boinc
```

**3.** Installing BOINC will also install the BOINC manager which is incredibly handy. The manager will allow you to see all your projects, tasks, and any statistics about them.

## Installing SETI & other projects

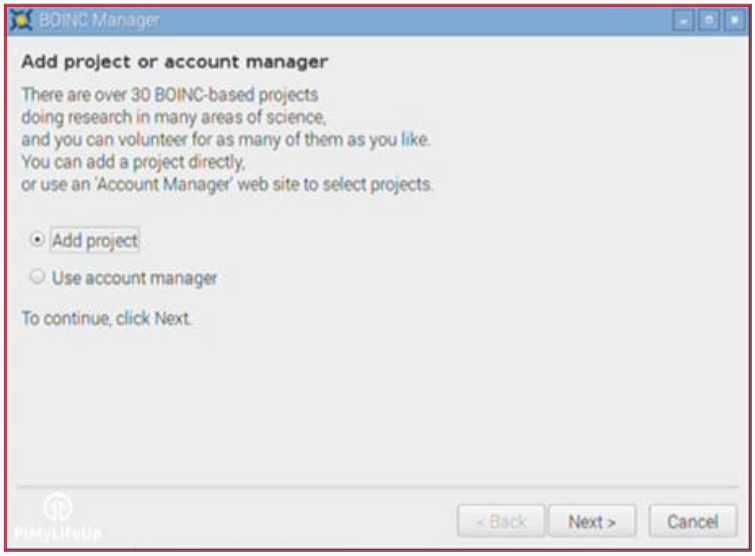
Next, we are going to setup SETI as the project we wish to process and contribute to. There are two ways we're able to do this with the first being via the BOINC manager and the second via the command line.

## BOINC Manager

Using the GUI is probably the preferable way when it comes to adding projects to your Pi. It's very simple to use for beginner users and is loaded with features for the more advanced user.

**1.** If you have the Raspbian GUI (Graphical User Interface) open, then you can simply go up to **Menu -> System Tools -> BOINC Manager**

**2.** On the first load, the **"Add project or project manager"** screen should be displayed.




**3.** In this window, select either "**Add project**" or if you have an account with either" BAM!" Or "**Grid Republic**" then you should select the "**Use account manager**" option.

**4.** On the project page scroll down to **SETI@Home** and select it. Once selected press next.

**5.** It will now connect, and then you will need to either create a new account or log in to an existing one.

**6.** Once done the project should be loaded onto the Pi and will soon start carrying out processing tasks.

## Command Line

**1.** I will go through the command line version first, to begin you will need to query the SETI at home website for an account key. To do this enter the following line.

```
boinccmd --lookup_account http://setiathome.berkeley.edu <your_email> <your_password>
```

**2.** It should now print out the following lines. You want to take note of the "**account key**", as you will need it in the next step.

```
status: Success

poll status: operation in progress

account key: <your_account_key>
```

**3.** Lastly, we need to run the following command to attach your BOINC program to your SETI@Home account. Make sure you have the account key from the previous step on hand.

```
boinccmd --project_attach http://setiathome.berkeley.edu <your_account_key>
```

**4.** Your account should now be added and be getting ready to load some tasks.

**5.** If you want to see the tasks that are currently being processed, then you just run the following command:

```
boinccmd --get_tasks
```

**6.** If you decide to go down the command line (CLI) route, then be sure to check out all the commands that are available to you over at BOINC's wiki page at **http://boinc.berkeley.edu/wiki/Boinccmd_tool**.

## Configuring BOINC via the Command Line

Configuring BOINC can be a complicated task if you are doing it through the command line as I briefly touched on before. If you're doing it through the management tool, then everything is a lot easier.

However, if you are certain you want to use the command line method of configuring BOINC, we will briefly go into the basics of setting it up.

For this tutorial, we will just be touching on the basics of setting up BOINC through the command link. I link below to the page that has all the settings that you're able to set in the config XML file.

If you want to setup preferences for Raspberry Pi BOINC in the command line, then you can edit the config file by running the following command.

```
sudo nano /var/lib/boinc-client/cc_config.xml
```

You will find that this file is usually empty. To find out all the options that you can add, be sure to check out the client configuration page at the BOINC wiki website here:
**http://boinc.berkeley.edu/wiki/Client_configuration**

If you make edits to the config file, then make sure you run the following command:

```
boinccmd --read_cc_config
```

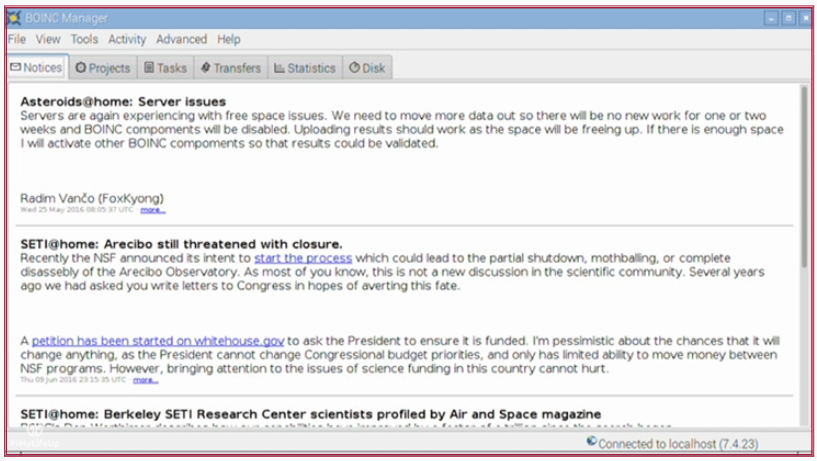
# Configuring BOINC via the Manager

If you're using the manager and would like to change things such as the running conditions, then go to tools and then select computing preferences. Basically all these settings are self-explanatory so I'll let you decide the best settings for you.

Now these might be a bit underwhelming so if you're more of a power user then you can change it to a more advanced view. To do this go up to view and then simply select advanced view.

Below we will just quickly go through what each of the tabs in the advanced view contains.

## Notices




This screen contains news from any of the projects you're currently running.

For example, SETI@Home will provide you with any relevant news to the SETI program or similar programs.

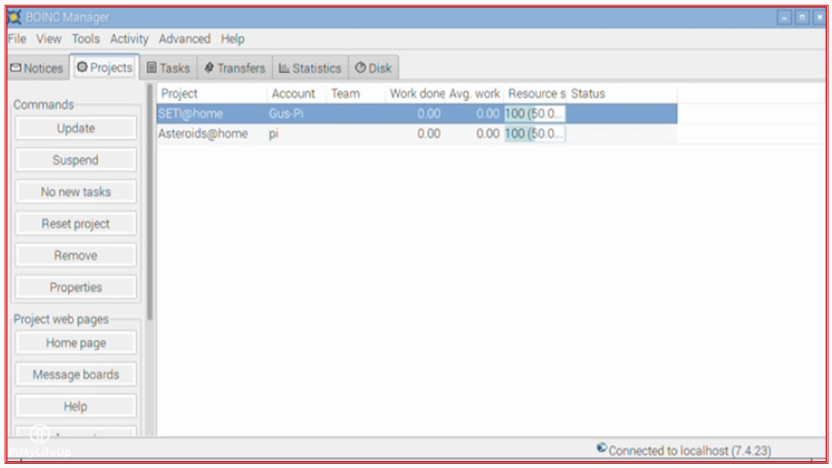
The more projects you run, the more news you're likely to have here.

## Projects

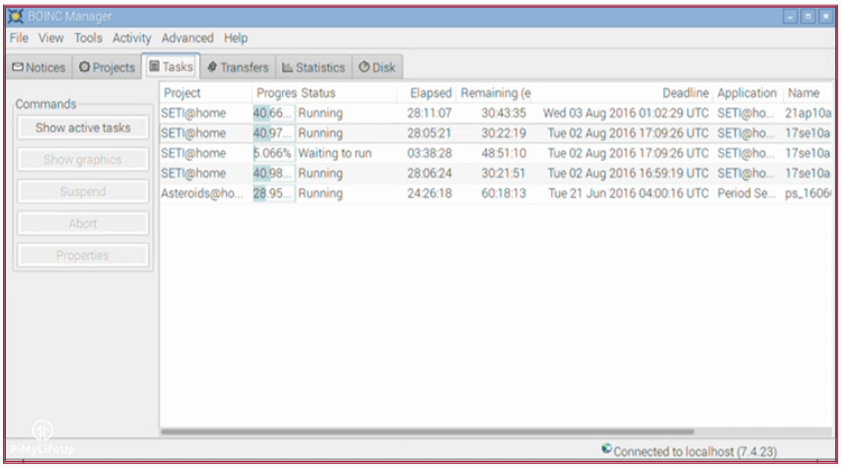
This screen contains all the projects that you are currently running. In here you can control projects to a certain degree.

For example, you can suspend projects or request that you no longer receive new tasks for that project.

You can also remove, update and edit properties of any given project.



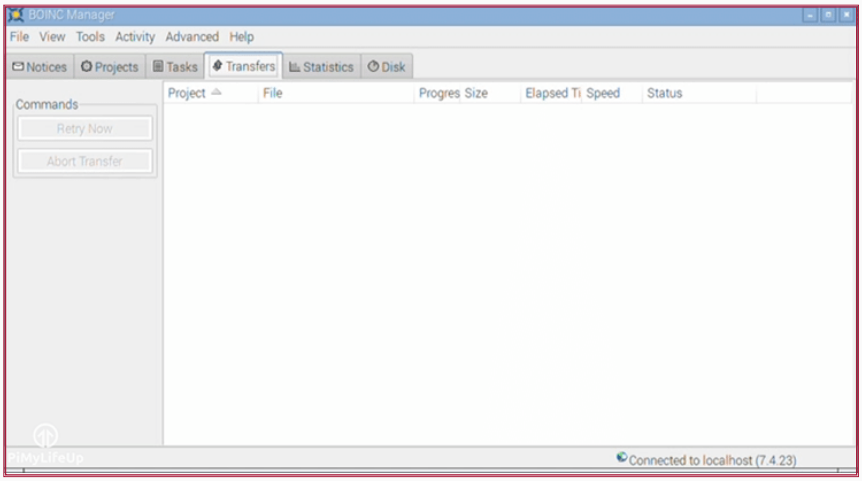

## Tasks




The tasks screen shows you all the current tasks on the Pi. You can suspend, abort and edit the properties of any given task.

As you can also see in the image below you're given stats such as progress, time elapsed, estimated time remaining, running status, deadline and other useful information.

## Transfers




This screen contains news from any of the projects you're currently running.

For example, SETI@Home will provide you with any relevant news to the SETI program or similar programs.

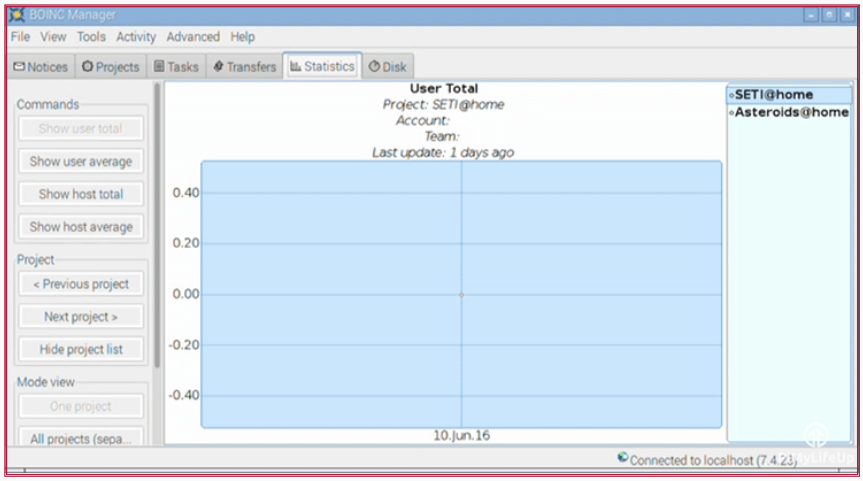
The more projects you run, the more news you're likely to have here.

## Statistics

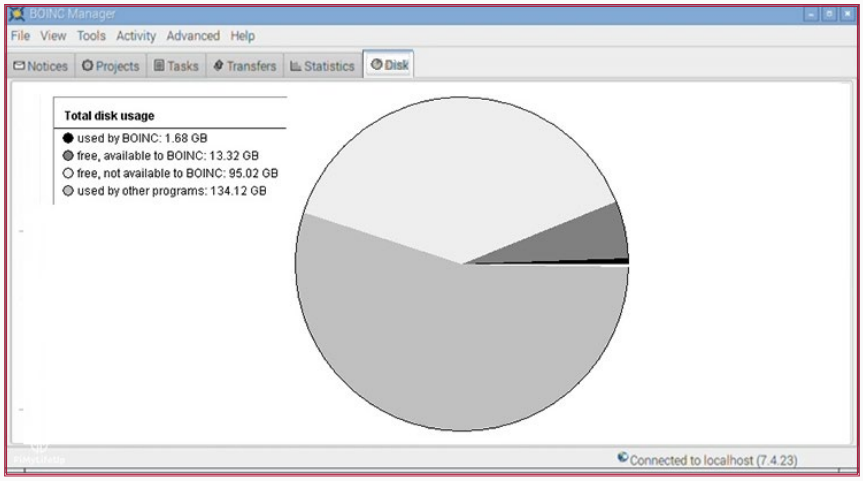
This screen shows you all the statistics related to your projects progress. The value on the left represents the credits received, and the values on the bottom are the dates.

To quickly explain BOINC credits is a way of tracking the amount of CPU time a user has contributed to any given project.

It helps prevent cheating by validating the project first. After successful validation, you are given credits.




## Disk




Disk shows you how much room is being taken up by BOINC and other programs.

It also shows the amount of available free space on the device.

This screen can be helpful if you are running out of room or have limited space on the device you're using.

There are of course even more settings in the menu that will satisfy any power user. You will find that computing preference menu I touched on earlier is also expanded allowing you to go into even more detail on how things should be run.

If you are craving more information on the advanced view, you can find it on BOINC's official Wikipedia page at http://boinc.berkeley.edu/wiki/Advanced_view

Overall BOINC is an excellent software package that you should certainly check out if you want to help process data to help cure diseases, search for extra-terrestrial intelligence, study global warming and much more then this is for you.

# Programming your own Twitch Mod Bot

## Project Description

In this tutorial, we will be showing you how to program your own Twitch Moderator Bot in python that will run on your Raspberry Pi.

The bot we will be showing you how to write is rather simple but can be adapted to being quite a powerful and versatile bot. The basics we will be explaining and showing you how to write is a simple bot that will connect into Twitch's IRC chat.

Once connected the bot will continually receive all messages and scan for any that matches our ban pattern if any match the pattern it will automatically ban them from the server. You can also use this to timeout a user or just post a message to the chat when a particular phrase is used.

While this is a somewhat basic moderator bot, it is easily extensible with a bit of python knowledge.

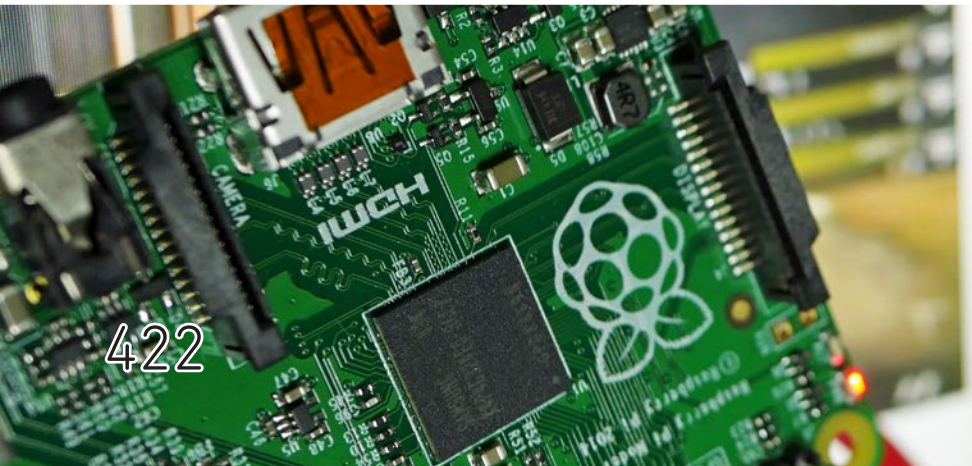
## Equipment

**Required**

- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

**Optional**

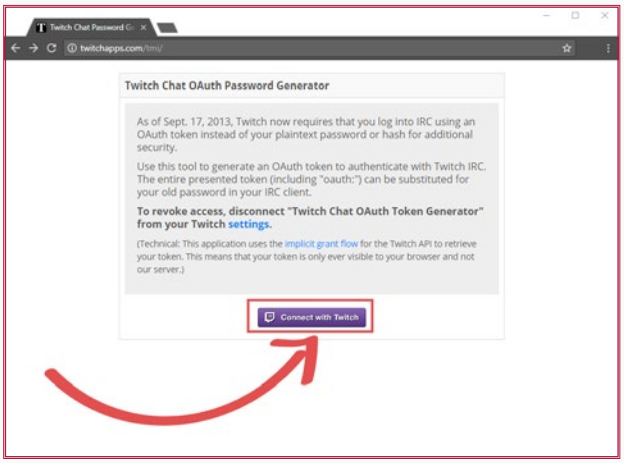
- Raspberry Pi Case



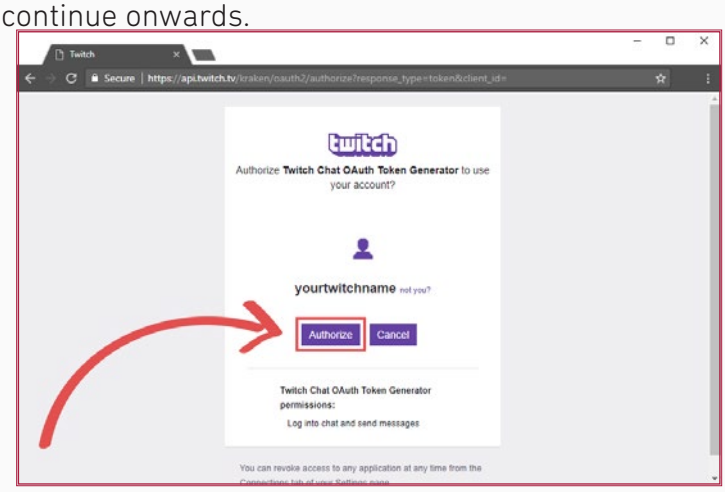
# Getting your Twitch oAuth token

**1.** Before we get to programming our Twitch moderator bot, we will first need to grab the OAuth key from Twitch. Of course, this means you will need to have a Twitch account ready that you want to use as your bot for your channel.

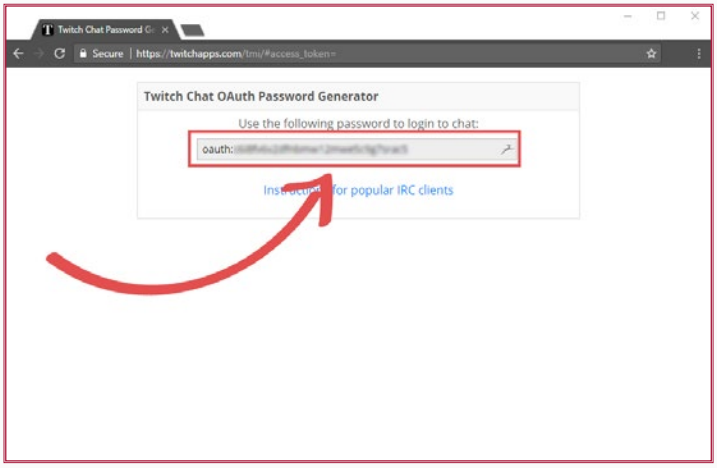
Once you have the user you want ready, go to http://twitchapps.com/tmi/. You will be shown the screen below. Press the "**Connect with Twitch**" button to continue.




**2.** If you are not logged in already, you will be asked to do so. Otherwise, you will be shown the screen below. This screen lets you authorize Twitch to provide you with access to your chat through IRC. Press the "**Authorize**" button to continue onwards.




**3.** Finally, your OAuth token will now be displayed to you. Copy this token somewhere safe as we will need this shortly while writing our script. This OAuth token is what will allow us to login to the IRC channels.



## Programming our Twitch Moderator Bot

**1.** Now that we have grabbed our Twitch OAuth token we can now proceed with writing our script.

For our purposes, we will be utilizing Python. It's versatile and fast enough for our basic needs.

First let's make a folder that we will keep our script in, run the following command to make our directory and move into it.

```
mkdir ~/twitchbot
cd ~/twitchbot
```

**2.** Now that we have made our folder and changed into it, we can proceed with writing our script.

Let's start off by making a simple file that will keep all our configuration settings.

Doing this will help make it easy to change any values we want configurable.

Run the following command to begin writing our first python file.

```
nano config.py
```

**3.** Add the following lines to this file.
Make sure you replace **twitch_username** with your **Twitch username**.

Replace **your_oauthtoken** with the **OAuth token** we grabbed earlier.

Replace **your_channel** with the **name of the channel** you want the bot to join into.

**BAN_PAT** is where we define the patterns we want to look for in the chat messages. For example, if you want to ban anyone who enters a certain word or link you can add it to the array.

```
HOST = "irc.twitch.tv"          # This is Twitchs IRC server
PORT = 6667                     # Twitchs IRC server listens on port 6767
NICK = "twitch_username"        # Twitch username your using for your bot
PASS = "your_oauthtoken" # your Twitch OAuth token
CHAN = "#your_channel"          # the channel you want the bot to join.
RATE = (20/30) # messages per seccond
BAN_PAT = [
   r"swear",
   r"some_pattern"
]
```

Once you have filled in your information, save the file by pressing **Ctrl + X** then **Y** and then hitting **Enter**.

**4.** We can now move onto writing our utility python file. We will keep a few simple functions in this file to keep our main file clean.

While these functions do simple things, it makes it easier to be able to keep them all in one place.

Run the following command to begin writing the python script.

```
nano utility.py
```

**5.** To this file, we want to add the following lines of code. We will explain each block of code as we go along.

```
def chat(sock, msg):
    """
    Send a chat message to the server.
    Keyword arguments:
    sock -- the socket over which to send the message
    msg  -- the message to be sent
    """
    sock.send(("PRIVMSG {} :{}\r\n".format(config.CHAN, msg)).encode("UTF-8"))
```

This bit of python code defines our chat message.

Almost all our functions will run through this. This sends a formatted message through our socket to Twitch's IRC Server.

```
def ban(sock, user):
    """
    Ban a user from the current channel.
    Keyword arguments:
    sock -- the socket over which to send the ban command
    user -- the user to be banned
    """
    chat(sock, ".ban {}".format(user))
```

This code issues a ban command to the server. This function formats a ban message and sends it through our chat function.

The function accepts both the socket to send through and the username that we want to ban.

```
def timeout(sock, user, secs=600):
    """
    Time out a user for a set period of time.
    Keyword arguments:
    sock -- the socket over which to send the timeout command
    user -- the user to be timed out
    secs -- the length of the timeout in seconds (default 600)
    """
    chat(sock, ".timeout {}".format(user, secs))
```

This segment of code is quite like to the ban function, except it formats a timeout command instead.

It formats a message with the timeout command, the user you want to be timed out and the amount of time to timeout the user for.

Over the page you can see what the final version of the file should look like.

**6.** Your final file should end up looking like what is displayed below.

```
import config
import socket

def chat(sock, msg):
        """
        Send a chat message to the server.
        Keyword arguments:
        sock -- the socket over which to send the message
        msg  -- the message to be sent
        """
        sock.send(("PRIVMSG {} :{}\r\n".format(config.CHAN, msg)).encode("UTF-8"))

def ban(sock, user):
        """
        Ban a user from the current channel.
        Keyword arguments:
        sock -- the socket over which to send the ban command
        user -- the user to be banned
        """
        chat(sock, ".ban {}".format(user))

def timeout(sock, user, secs=600):
        """
        Time out a user for a set period of time.
        Keyword arguments:
        sock -- the socket over which to send the timeout command
        user -- the user to be timed out
        secs -- the length of the timeout in seconds (default 600)
        """
        chat(sock, ".timeout {}".format(user, secs))
```

Once you have finished, save the file by pressing **Ctrl + X** then **Y** and then hitting **Enter**.

**7.** With both our config file and utility script now written up, we can now proceed with writing our main script. This script will be where our bot runs. It will utilize the configuration from our config file and utilize the functions we wrote in our utility script.

Let's begin writing our main script by running the following command.

```
nano main.py
```

**8.** Now to this file, we are going to add the following lines of code. This file will basically setup our connection to Twitch's IRC server and loop through all the messages it receives.

```
#!/usr/bin/env python
import config
import utility
import socket
import time
import re

CHAT_MSG = re.compile(r"^:\w+!\w+@\w+\.tmi\.twitch\.tv PRIVMSG #\w+ :")
```

These first few lines import all the external functionality we need. The first two of these are the scripts we wrote, our **config** and **utility** script. The **Socket** library is what we will use to talk with Twitch's IRC server. The **Time** library is what we use to keep track of how often we are sending messages, and the **re** library is what we will use to utilize regular expressions.

We also define our CHAT_MSG variable, this we still our compiled regular expression. This expression will sift out our message form the rest of the IRC message.

```
try:
        s = socket.socket()
        s.connect((config.HOST, config.PORT))
        s.send("PASS {}\r\n".format(config.PASS).encode("utf-8"))
        s.send("NICK {}\r\n".format(config.NICK).encode("utf-8"))
        s.send("JOIN {}\r\n".format(config.CHAN).encode("utf-8"))
        connected = True #Socket succefully connected
except Exception as e:
        print(str(e))
        connected = False #Socket failed to connect
```

This segment of our script is what we use to connect to Twitch's IRC server. This code will initialize the connection and send our Username, OAuth token and our join command to the server.

We run this all within a try statement so we can catch any errors and print them to the log. We also set our connected variable to true or false based on whether it successfully connected.

```
def bot_loop():
        while connected:
                response = s.recv(1024).decode("utf-8")
                if response == "PING :tmi.twitch.tv\r\n":
                        s.send("PONG :tmi.twitch.tv\r\n".encode("utf-8"))
                        print("Pong")
                else:
```

We will now write our bot loop function. Sicne this is an important part of the code we will explain this in two segments. This first segment of the code checks if we are first connected, and if it does we check if there is anything to receive, and if there is we grab it and decode it.

We then check the received message, if it is equal to a ping command from the IRC server then we need to respond to this with a PONG message. Failure to do so would mean we are booted from the IRC server.

```
                        username = re.search(r"\w+", response).group(0)
                        message = CHAT_MSG.sub("", response)
                        print(username + ": " + response)
                        for pattern in config.BAN_PAT:
                                if re.match(pattern, message):
                                        utility.ban(s, username)
                                        break
                time.sleep(1 / config.RATE)

if __name__ == "__main__":
        bot_loop()
```

The final part of the script does the main grunt work. The first line will use our regular expression to grab the username out of the message. The second line will grab the associated message.

We will then search through the **BAN_PAT** array that we set within our config file. If we find any matches within our message to our patterns, we will run the ban function that we defined earlier.

Afterward, we will sleep for a period that is defined in our config file.

Finally, we do our **__name__** main check. These lines will make it so we can utilize this file as a library if need be. Meaning that our loop will not run into the **bot_loop** function has been called.

Go to the next page to see what the full file should look like when you are done writing in the code.

## Programming our Twitch Moderator Bot

Your script should look something like what is displayed below once you have finished typing in all the code.

```
#!/usr/bin/env python
import config
import utility
import socket
import time
import re

CHAT_MSG = re.compile(r"^:\w+!\w+@\w+\.tmi\.twitch\.tv PRIVMSG #\w+ :")

try:
        s = socket.socket()
        s.connect((config.HOST, config.PORT))
        s.send("PASS {}\r\n".format(config.PASS).encode("utf-8"))
        s.send("NICK {}\r\n".format(config.NICK).encode("utf-8"))
        s.send("JOIN {}\r\n".format(config.CHAN).encode("utf-8"))
        connected = True #Socket succefully connected
except Exception as e:
        print(str(e))
        connected = False #Socket failed to connect

def bot_loop():
        while connected:
                response = s.recv(1024).decode("utf-8")
                if response == "PING :tmi.twitch.tv\r\n":
                        s.send("PONG :tmi.twitch.tv\r\n".encode("utf-8"))
                        print("Pong")
                else:
                        username = re.search(r"\w+", response).group(0)
                        message = CHAT_MSG.sub("", response)
                        print(username + ": " + response)
                        for pattern in config.BAN_PAT:
                                if re.match(pattern, message):
                                        utility.ban(s, username)
                                        break
                time.sleep(1 / config.RATE)
if __name__ == "__main__":
        bot_loop()
```

Once you have finished writing, save the file by pressing **Ctrl + X** then **Y** and then hitting **Enter**.

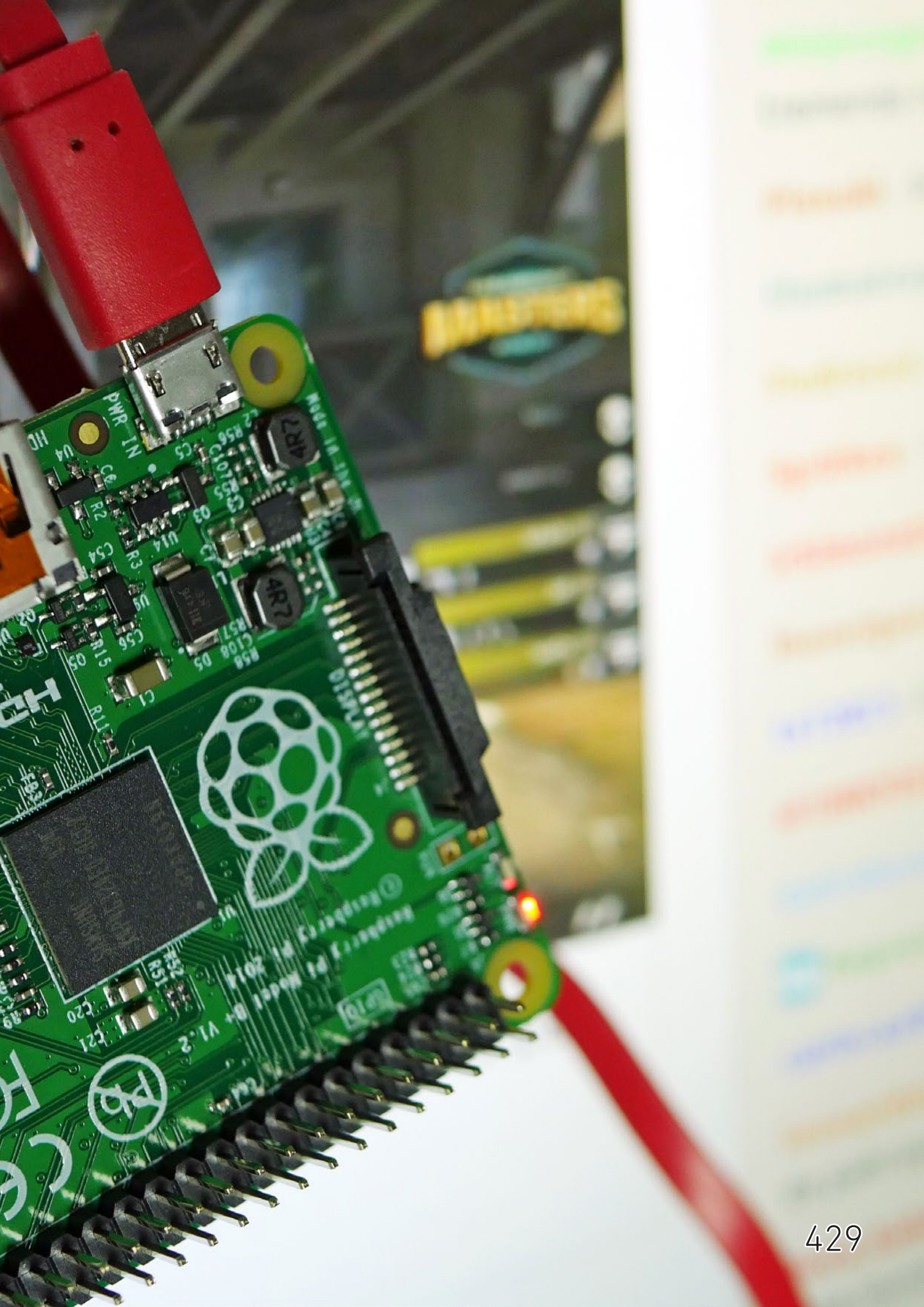
## Running your Twitch Moderator Bot

**1.** With all the python code now written it is incredibly easy to run our Twitch moderator bot, simple type in the following command to run the script.

```
python3 ./main.py
```

You should now have a fully working moderator bot that by now should have connected into your Twitch chat and be searching for any of the patterns mentioned within your array.

Hopefully, this tutorial has given you a good understanding of the script, and allow you to you to modify it to your needs quickly.



# Programming your own Twitter Bot

## Project Description

In this project, we will be exploring how to set up a basic low-cost Raspberry Pi Twitter Bot utilizing the Python programming language.

The twitter bot we will be showing you how to write, and setup is made easy thanks to the Twython package. We will be exploring how to use the Twython package and its major functions.

We will also be showing you how to make a boot that tweets our your Raspberry Pi's CPU Temperature or periodically take a photo and upload it as a tweet.

You can use your Raspberry Pi Twitter bot to automate a huge range of different tasks, these range from posting a picture to your Twitter account every hour to monitor the growth of a plant or even tweeting out values from sensors connected to your Raspberry Pi.

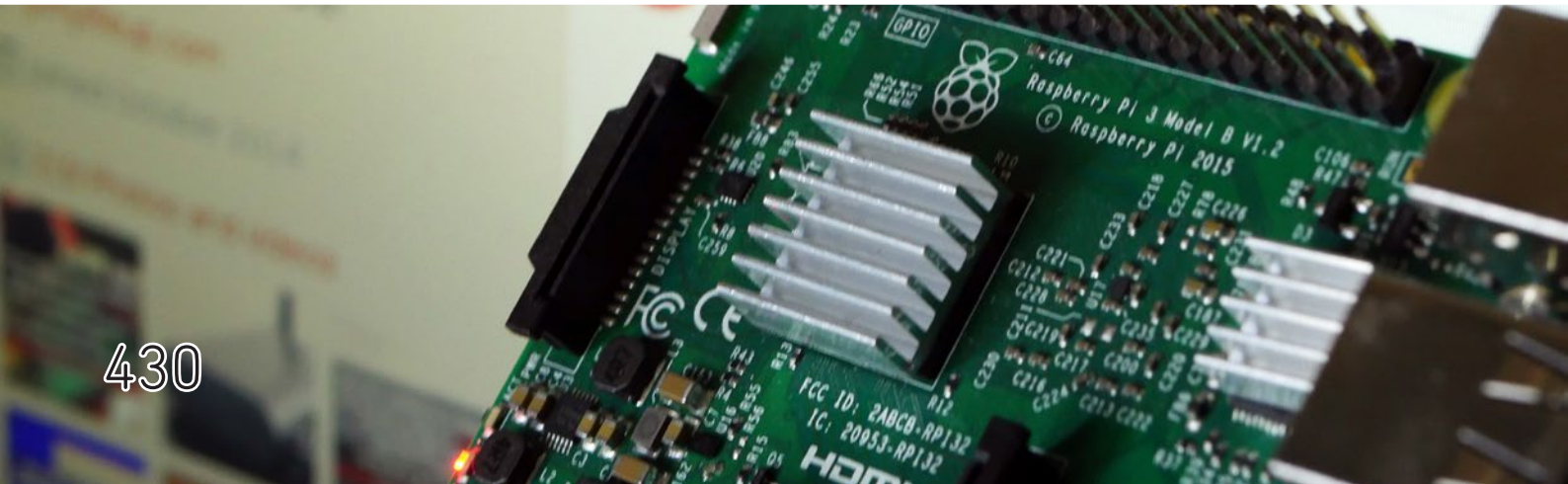
## Equipment

**Required**

- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

**Optional**

- Raspberry Pi Case



# Registering a Twitter app

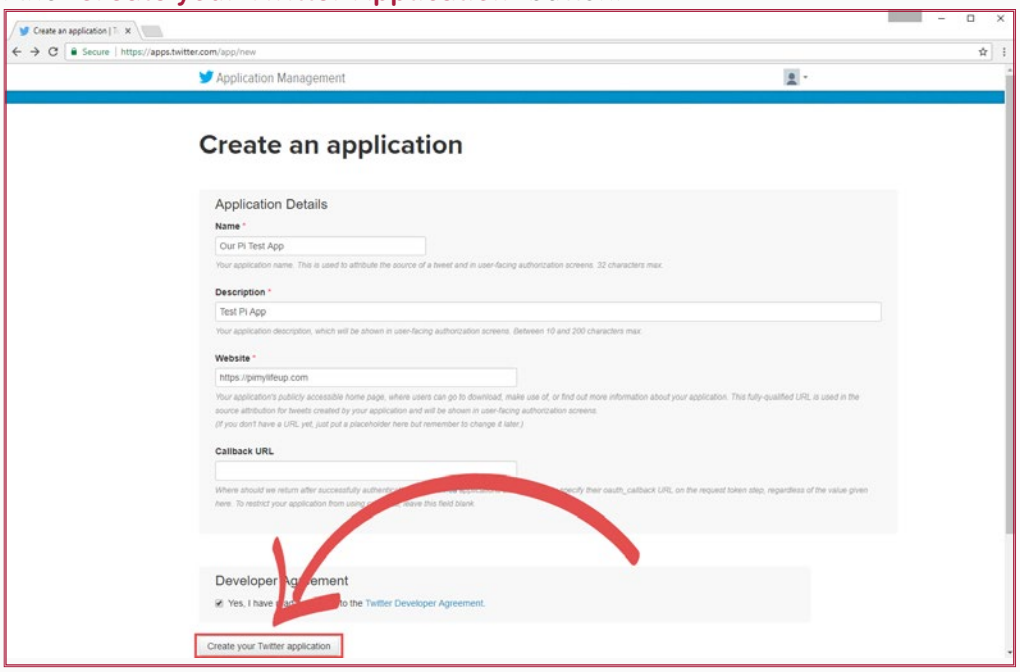
**1.** To use the Twitter API, we will first have to register as a Twitter app. Registering as a Twitter will allow us to use the REST interface and is what we will be utilizing to post new tweets to Twitter and interact with the website.

Once you are logged into the Twitter account you plan on using, go to https://apps.twitter.com/app/new. You will be shown the screen below.

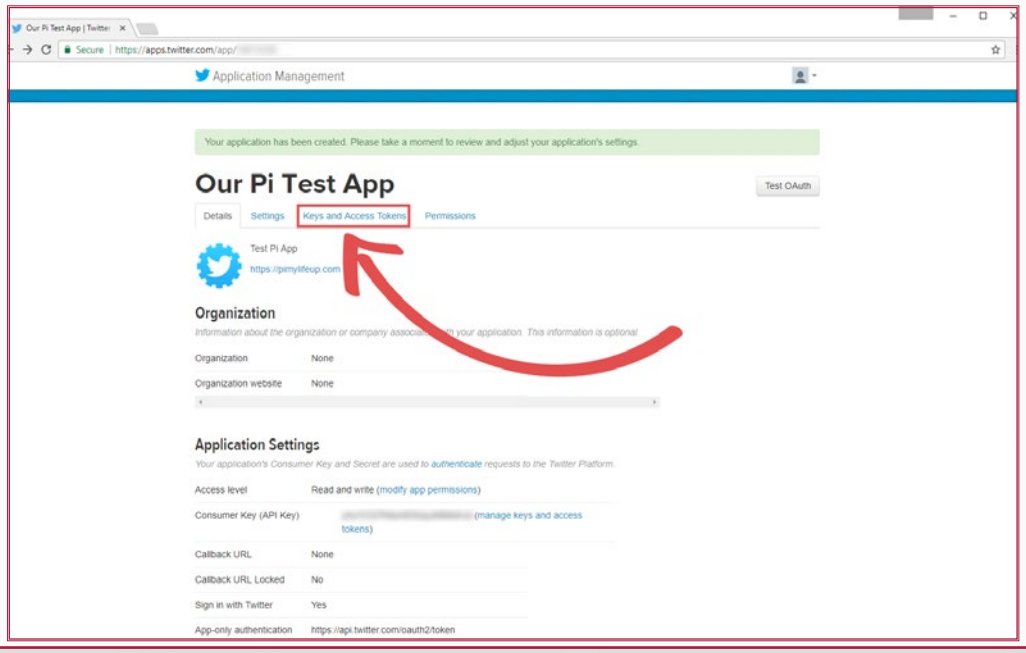
The three main things you need to focus on here is the **Name**, **Description**, and **Website** you can set these to whatever you like.

**Callback URL** can be left blank.

Once done click the "**Create your Twitter Application**" button.




**2.** After you have created your Twitter App, you will be taken to the page displayed below, on here click the "**Keys and Access Tokens**" tab.

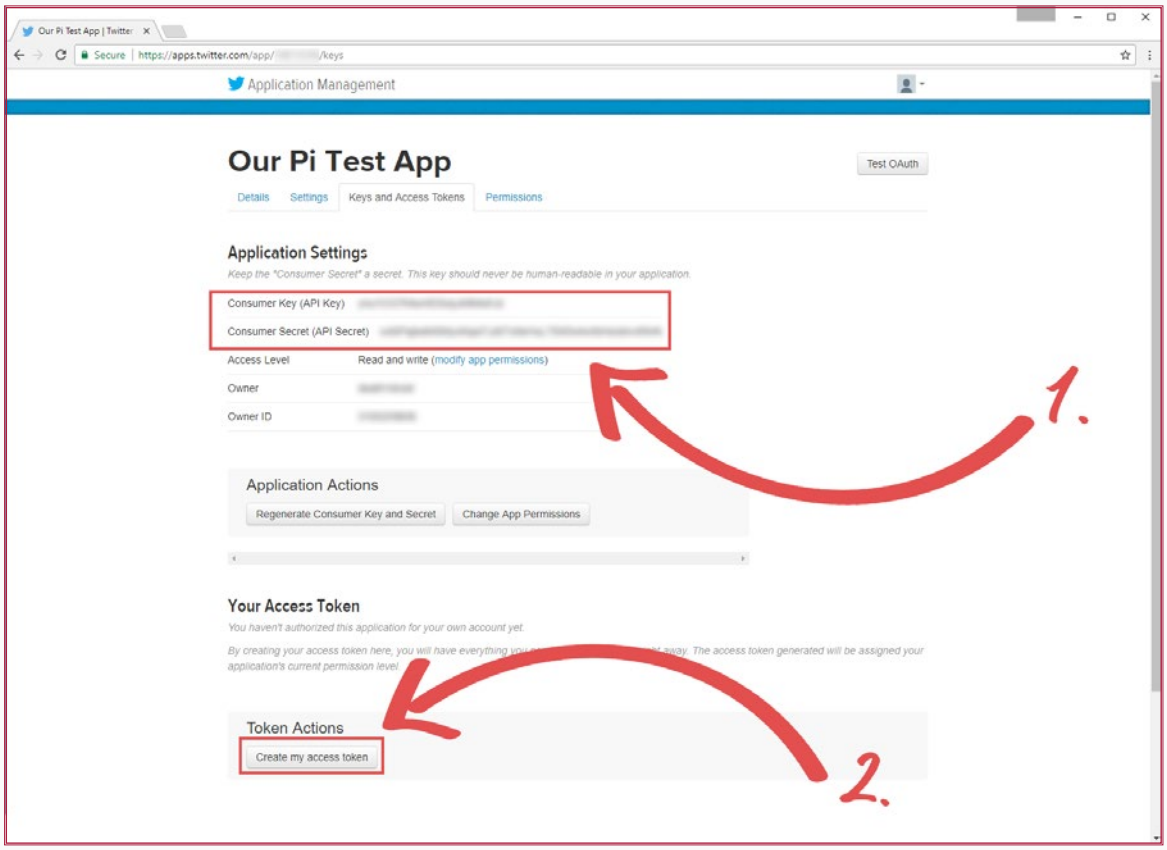


# Registering a Twitter app

**3.** On this screen, you will want to copy down both the **Consumer Key (API Key)** and the **Consumer Secret (API Secret)**. We will need both to connect to the API.

You can ignore both the Owner and Owner ID as we don't require these to interact with the API.

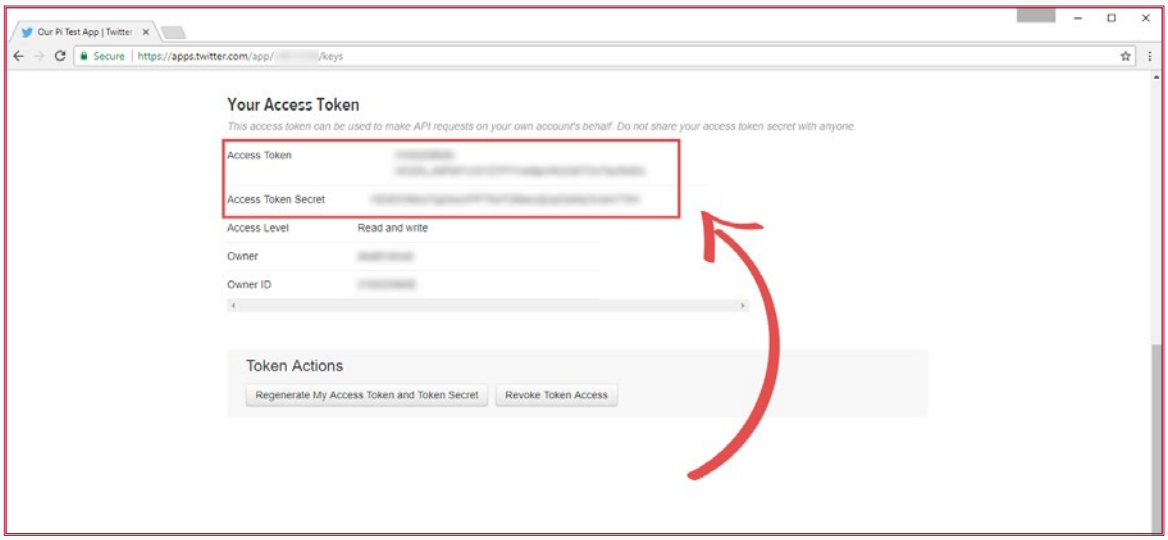
After copying down these two different values click the "**Create my access token**" **(2.)** button.




**4.** After you have created the access token, the page will refresh.

Scroll down to the bottom of the page and copy down both the Access Token and the Access Token Secret.

These are the final values we need to connect to the API.



## Setting up and writing our Twitter Bot

**1.** With our tokens, all grabbed we can now start setting up and writing our Twitter bot. The first thing we should do is make sure our Raspberry Pi is up to date by running the following command:

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

**2.** With the Raspberry Pi now updated let's install the tools we will need to set up our Twitter Bot.

Let's begin installing all of this by running the next few commands in the terminal:

```
sudo apt-get install python-setuptools
sudo easy_install pip
sudo pip install twython
```

**3.** Finally, we can begin writing our twitter bot. Thanks to the **Twython** package this is incredibly easy to do. First, let's make a directory where we will keep our script and begin writing the script by running the following commands:

```
mkdir ~/twitterbot && cd ~/twitterbot
sudo nano TwitterBot.py
```

**4.** Now in this file, let's write in the following piece of code. This code is incredibly simple thanks to the Twython library. You can read our simple explanation of what this code does below:

**from twython import Twython -** We use this to make API calls to Twitter.

**import sys -** Imports the sys library that we use to grab arguments that are passed to the script.

The next block of code defines the various keys that we wrote down earlier in the tutorial. We pass this into Twython.

Make sure you replace the values of **CONSUMER_KEY**, **CONSUMER_SECRET**, **ACCESS_KEY**, and **ACCESS_SECRET** before proceeding.

We then define our variable **api** which stores the Twython object, we create this object with all our access and consumer keys. This object makes doing API calls to Twitter simple as all the data is there ready for it to use.

Finally, we issue a **update_status** API call to our **api** object. This call should tweet any text that is passed into the script to your Twitter account.

```
#!/usr/bin/env python
import sys
from twython import Twython

CONSUMER_KEY = '***************YOUR DATA*****************'
CONSUMER_SECRET = '***************YOUR DATA*****************'
ACCESS_KEY = '***************YOUR DATA*****************'
ACCESS_SECRET = '***************YOUR DATA*****************'

api = Twython(CONSUMER_KEY,CONSUMER_SECRET,ACCESS_KEY,ACCESS_SECRET)

api.update_status(status=sys.argv[1])
```

One you have finished writing, save the file by pressing **Ctrl + X** then **Y** and then hitting **Enter**.

## Setting up and writing our Twitter Bot

**5.** With our script now written, we need to make the script executable now so we can send our text to the Python file.

```
sudo chmod +x TwitterBot.py
```

**6.** Now let's test out our script by running the following command, this will make sure that we are making a valid connection with the Twitter API. After the command has run, check your Twitter.

```
python TwitterBot.py "Hello this is a test tweet from my Raspberry Pi"
```

**7.** Once you have confirmed that the tweet was sent we can start doing a few more interesting things with our Twitter Bot, just sending a simple message is boring after all.

Our next two blocks will explore making the bot send out an image taken from a webcam, and also a block showing you how to get the Raspberry Pi to tweet out its CPU temperatures.

We are firstly going to explore grabbing the CPU temperature and tweeting that out. If you are more interested in the webcam picture sending, then go over to the next page.

## Posting our CPU temperature with our Twitter Bot

**1.** To begin let's begin editing our Twitter Bot script by running the following command:

```
sudo nano TwitterBot.py
```

**2.** Now we can make the following changes to our bot.

### Find

```
import sys
```

### Replace With

```
import os
```

Since we no longer need to worry about arguments being passed in, we can replace the **sys library** with our new **os library**. The **OS library** allows us to interact with the operating system.

### Find

```
api.update_status(status=sys.argv[1])
```

### Replace With

```
cmd = '/opt/vc/bin/vcgencmd measure_temp'
line = os.popen(cmd).readline().strip()
temp = line.split('=')[1].split("'")[0]
api.update_status(status='My current CPU temperature is '+temp+' C')
```

Basically what this does is issue a command into terminal, this command calls a binary file that returns the temperature. We then process this information to grab the temperature and then send out a Tweet with **update_status** that has the value we grabbed.

One you have finished writing, save the file by pressing **Ctrl + X** then **Y** and then hitting **Enter**.

**3.** We can test out this new piece of code by running the following command.

```
python TwitterBot.py
```

With that done, we can now either learn how to take picture or automate our bot.

## Sending webcam photos through the Twitter Bot

**1.** To begin let's begin editing our Twitter Bot script by running the following command:

```
sudo nano TwitterBot.py
```

**2.** Now we can make the following changes to our bot.

### Find

```
import sys
```

### Replace With

```
import pygame
import pygame.camera
from pygame.locals import *
```

We replace the **sys library** with **pygame** as we no longer need to grab any variables passed into the script.

Instead, we import **pygame**, **pygame** is a library that contains an extensive range of Python modules designed for writing games.

Of course, we aren't writing a game, but **pygame** has an excellent and easy to use library for utilizing cameras so we will be utilizing that module of **pygame**.

### Find

```
api.update_status(status=sys.argv[1])
```

### Replace With

```
pygame.init()
pygame.camera.init()
cam = pygame.camera.Camera("/dev/video0",(640,480))
cam.start()
image = cam.get_image()
pygame.image.save(image,'webcam.jpg')

photo = open('webcam.jpg','rb')
api.update_status_with_media(media=photo, status='Test Raspberry Pi Image Bot => ')
```

We need first to replace the original **update_status** call with several other calls. The first function sets up and initializes the **pygame library**. We then must initiate the camera module we plan on using, this sets itself up so we can get it to read from one of our interfaces.

We then store a copy of the camera object in our cam variable, setting it up to use out **video0** interface and the resolution of **640x480**.

Once we have done that, we can use the new object and start the camera up. This object allows us to grab images off the camera which we do in the following line of code.

With the image now saved to the Raspberry Pi, all we need to do is read back in that image using the open function, and then making use of another handy **Twython** function.

This function is **update_status_with_media**. It allows us to send a tweet with an image which is perfect for sending our just grabbed webcam photo.

One you have finished writing, save the file by pressing **Ctrl + X** then **Y** and then hitting **Enter**.

## Sending webcam photos through the Twitter Bot

**3.** Please note if you are using your Raspberry Pi camera as your webcam, make sure you run the command below before running this script.

This command allows the camera to appear as a **/dev/video*** interface.

```
sudo modprobe bcm2835-v4l2
```

**4.** We can test out this new piece of code by running the following command.

```
python TwitterBot.py
```

With that done, go to the next page to read about how we can setup a cronjob to run our bot continually.

## Automating your Twitter Bot

**1.** There isn't much use in your Twitter Bot if you must manually run it all the time, one of the best ways to get your bot to be continually triggered is to use Cron Jobs.

You may be wondering why we just don't run the script in a continuous loop. Well, there are two reasons for this.

One reason is that there is no guarantee that the script won't hang at some point, the other is that it burns up precious processing cycles having to loop and check whether we should be tweeting out continually.

So now let's begin by editing the crontab by running the following command.

```
sudo crontab -e
```

**2.** To this file you want to add the line of text below to the bottom of the file, basically what this will do is automatically trigger the script every hour.

If you want to work out your own times for when you want the script to trigger but are not used to dealing with Cron Jobs, you can try using an online tool such as https://crontab.guru/ as it will give a human readable version of the cron schedule.

Just replace ***/60 * * * *** with your generated value.

```
*/60 * * * * python /home/pi/twitterbot/TwitterBot.py
```

Once you have finished writing in the code, save the file by pressing **Ctrl + X** then **Y** and then hitting **Enter**.

You should now hopefully have a fully operational basic Twitter Bot that will constantly tweet out every time the cron job goes off.

We hope we have explained the code enough and kept it basic enough that you can easily extend the bot to do more complicated tasks.

Tweets
289
Tweets
DISPLAY
FCC ID: 2ABCB-
IC: 20953-RP
HDMI

# Telegram

## Project Description

In this project, we will be showing you how to setup the popular messaging service Telegram on your Raspberry Pi. We will be doing this by making use of the Telegram CLI (Command Line Interface) written by a user on GitHub.

This project will allow you to send and receive messages over the Telegram messaging service on your Raspberry Pi, even going as far as even being able to send images. All you require to do this is a mobile phone, Raspberry Pi, and a mobile phone number.

Unlike WhatsApp, you can have multiple devices running on the same phone number so, you do not have to worry about having a spare phone number to be able to run it. You can extend this tutorial further by utilizing Telegram to control your Raspberry Pi.

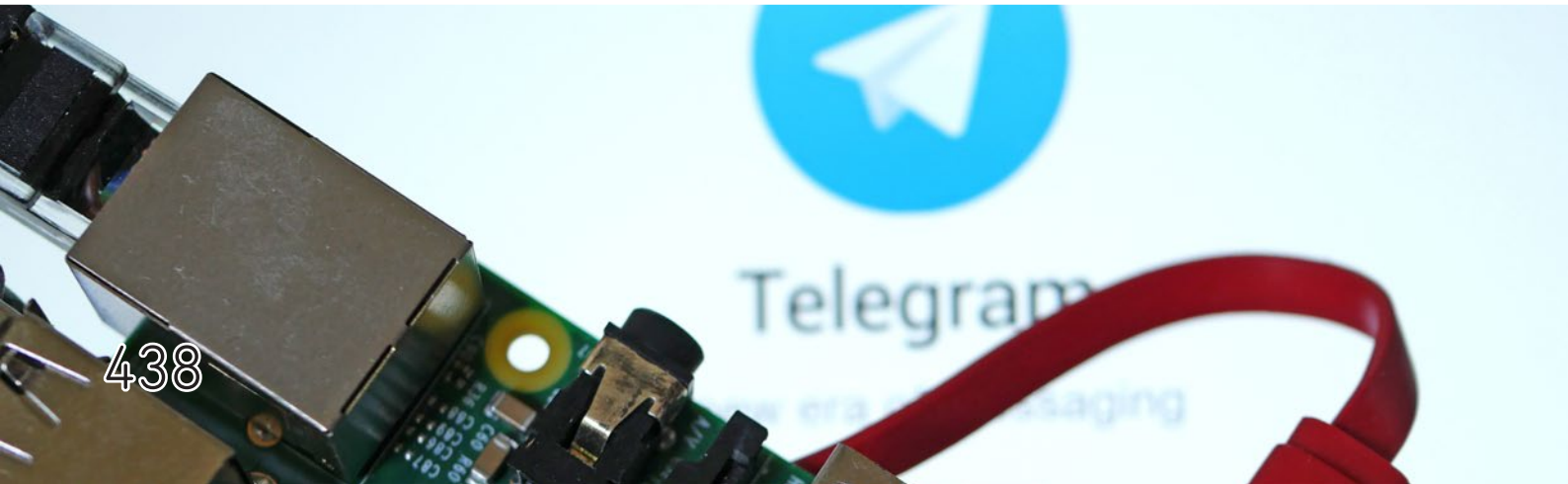
## Equipment

**Required**

- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection
- Mobile Phone

**Optional**

- Raspberry Pi Case



## Setting up Telegram CLI on the Raspberry Pi

**1.** Before we begin the process of installing Telegram CLI (**C**ommand **L**ine **I**nterface) to our Raspberry Pi, we must first run an update and upgrade to ensure our Raspberry Pi is running the latest versions.

Run the following two commands to update the Raspberry Pi.

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

**2.** With our Raspberry Pi now up to date, we can now install all the libraries we need to utilize the Telegram CLI.

Run the following three commands on your Raspberry Pi to install all the required packages.

```
sudo apt-get install -y libreadline-dev libconfig-dev libssl-dev
sudo apt-get install -y lua5.2 liblua5.2-dev libssl1.0-dev
sudo apt-get install -y libevent-dev libjansson-dev libpython-dev make git
```

**3.** Now that we have all the libraries that we need to run the Telegram CLI we can proceed to clone the repository from GitHub.

We need to use --recursive as we need GIT also to clone the remote repository's that the Telegram CLI GitHub references.

Run the following two commands on your Raspberry Pi to begin the cloning process.

```
cd ~
git clone --recursive https://github.com/LucentW/tg.git
```

**4.** To get the Telegram CLI to compile on our Raspberry Pi, we will first have to modify one of the source code files.

Run the following command on your Raspberry Pi to begin editing the file.

```
nano ~/tg/tgl/mtproto-utils.c
```

**5.** Within this file we want to press **Ctrl + W** then type in "**BN2ull**" and then press the **Enter** key.

That should take you to the following block of code.

```
static unsigned long long BN2ull (TGLC_bn *b) {
  if (sizeof (unsigned long) == 8) {
    return TGLC_bn_get_word (b);
  } else if (sizeof (unsigned long long) == 8) {
    assert (0); // As long as nobody ever uses this code, assume it is broken.
    unsigned long long tmp;
    /* Here be dragons, but it should be okay due to be64toh */
    TGLC_bn_bn2bin (b, (unsigned char *) &tmp);
    return be64toh (tmp);
  } else {
    assert (0);
  }
}
```

## Setting up Telegram CLI on the Raspberry Pi

**6.** Within this block of code, we want to find and replace two occurrences of **assert(0)** as shown below.

### Find

```
assert(0);
```

### Replace With

```
//assert(0);
```

You can now save the file by pressing **Ctrl + X** then **Y** and then hitting **Enter**.

**7.** Now that we have cloned the Telegram CLI to our Raspberry PI and made changes to the source code, we will now need to run the configuration script and compile it.

To do this we can just run the following three commands on our Raspberry Pi.
The first command changes us into the Telegram directory.

The second configures the Telegram CLI for compiling.

Finally, the **make** command compiles it.

```
cd ~/tg
./configure
make
```

## Registering with Telegram CLI on the Raspberry Pi

**1.** Now that we have downloaded and compiled the Telegram CLI, we can now actually start to make use of it.

To proceed further in this tutorial, you will **need** a **mobile and a mobile phone number** handy.

You will also need to know your **country code** for your mobile phone number, for example, the **country code** for **Australia** is **+61**.

**2.** Now that we have a mobile phone number with a country code handy we can now proceed with running the Telegram CLI.

To do this, we need to run the following two commands on your Pi.

```
cd ~/tg
bin/telegram-cli -k tg-server.pub -W
```

**3.** You will now be asked to enter your phone number, make sure that you type in your phone number with your country code in front of it.

For example, if our phone number was **04XXXXXXXX**, it would become **+614XXXXXXXX** when we type it in.

An authentication code will be sent to this phone. We will need it after the next step.

## Registering with Telegram CLI on the Raspberry Pi

**4.** After entering your phone number, you will be asked if you want to **Register**, type in **Y** and press **enter** to continue onwards.

Immediately afterward you will be asked to enter both your first name and your last name, fill out both details. Remember what you set for these as you will need them later.

**5.** Now that you have filled in your information you will now be asked to enter the code that you received on your phone.

Type in the code you received and press the **Enter** key to finalize the registration.

You should now have access to the Telegram command line interface.

## Utilizing the Telegram CLI on the Raspberry Pi

**1.** Since we have now successfully registered ourselves with the Telegram CLI, we can now actually start to receive and send messages. You will notice that if anyone messages you, the message will pop up in the command line interface.

You can message people, or yourself by utilizing the following command within the Telegram CLI.

Replacing **<fistname>** with the first name of the contact you want to message and **<lastname>** with the last name of the person you want to contact.

Finally, you can replace **<msg>** with the message you wish to send this person.

You can also use this command to message yourself by using the first name and last name that you specified with registering.

```
msg <firstname>_<lastname> <msg>
```

As an example of what a filled-out version of the command, we have an example below, using **PiMy** as the first name and **LifeUp** as the last name.

```
msg PiMy_LifeUp This is a test message from pimylifeup.com
```

**2.** You can also use the Telegram CLI to send photos as well. This command is very similar to the sending a text-based message command.

But instead of **<msg>** you point to the location of a photo on the Raspberry Pi that you want to send.

```
msg_photo <firstname>_<lastname> <imagelocation>
```

As an example of what a filled-out version of the command, we have an example below, using **PiMy** as the first name and **LifeUp** as the last name.

```
msg_photo PiMy_LifeUp /home/pi/photo.jpg
```

## Automating the Telegram CLI on the Raspberry Pi

**1.** To simplify sending messages through the Telegram CLI we are going to write two scripts, one for handling sending text messages and one for handling photo images.

These will remove the need to launch up the command line interface of Telegram every time you want to utilize it.

Let's run the following command to begin writing our first telegram script.

```
sudo nano /home/pi/tg.sh
```

**2.** Within this file, you will want to write the following lines of script.

These lines of code will read in two arguments as variables and then pass them through to the Telegram CLI to send the message.

```
#!/bin/bash
tgpath=/home/pi/tg
cd ${tgpath}
(echo "msg $1 $2"; echo "safe_quit") | ${tgpath}/bin/telegram-cli -k tg-server.pub -W
```

You can now save the file by pressing **Ctrl + X** then **Y** and then hitting **Enter**.

**3.** With our new script now written, we also need to give it execution permissions so it can start up Telegram.

Run the following command on your Raspberry Pi to change its permissions.

```
sudo chmod -R 0655 /home/pi/tg.sh
```

**4.** We can test our new script by running the following command.

Makle sure you replace **<firstname>** and **<lastname>** with the relevant data.

```
/home/pi/tg.sh <firstname>_<lastname> "Your test message from pimylifeup"
```

**5.** Once you have confirmed that the script successfully works, we can do the same for images.

Run the following command to begin writing our second script.

```
sudo nano /home/pi/tg_photo.sh
```

**6.** Now in this file, we will want to write the following lines of script, this is very similar to our previous script but instead sends the **send_photo** command to the Telegram CLI.

```
#!/bin/bash
tgpath=/home/pi/tg
cd ${tgpath}
(echo "send_photo $1 $2"; echo "safe_quit") | ${tgpath}/bin/telegram-cli -k tg-server.pub -W
```

You can now save the file by pressing **Ctrl + X** then **Y** and then hitting **Enter**.

## Automating the Telegram CLI on the Raspberry Pi

**7.** Now we can test this final script by running the following command.

Make sure that you replace **<firstname>** and **<lastname>** with the relevant data.

We also need to make sure we point to a real image, so change **/home/pi/photo.jpg** to the location of an image that is on your Raspberry Pi.

```
/home/pi/tg.sh <firstname>_<lastname> /home/pi/photo.jpg
```

Hopefully, by now you have successfully set up the Telegram CLI on your Raspberry Pi and can successfully send text messages and images over Telegram. We also hope the scripts help simplify your usage of the software.

If you would like to know what sort of commands you can utilize with the Telegram CLI, you can go to the official documentation that lists all the possible commands.

You can find that documentation by going here: https://github.com/LucentW/tg#supported-commands

# Controlling your
## Raspberry Pi with Telegram

## Project Description

In this project, we will be building upon our Telegram tutorial by showing you how to utilize Telegram to control your Raspberry Pi. We will be doing this by using the LUA interpreter that is built into the Telegram CLI.

The LUA interpreter allows us to program specific functions that trigger when something occurs on the Telegram CLI, such as it receives a message.

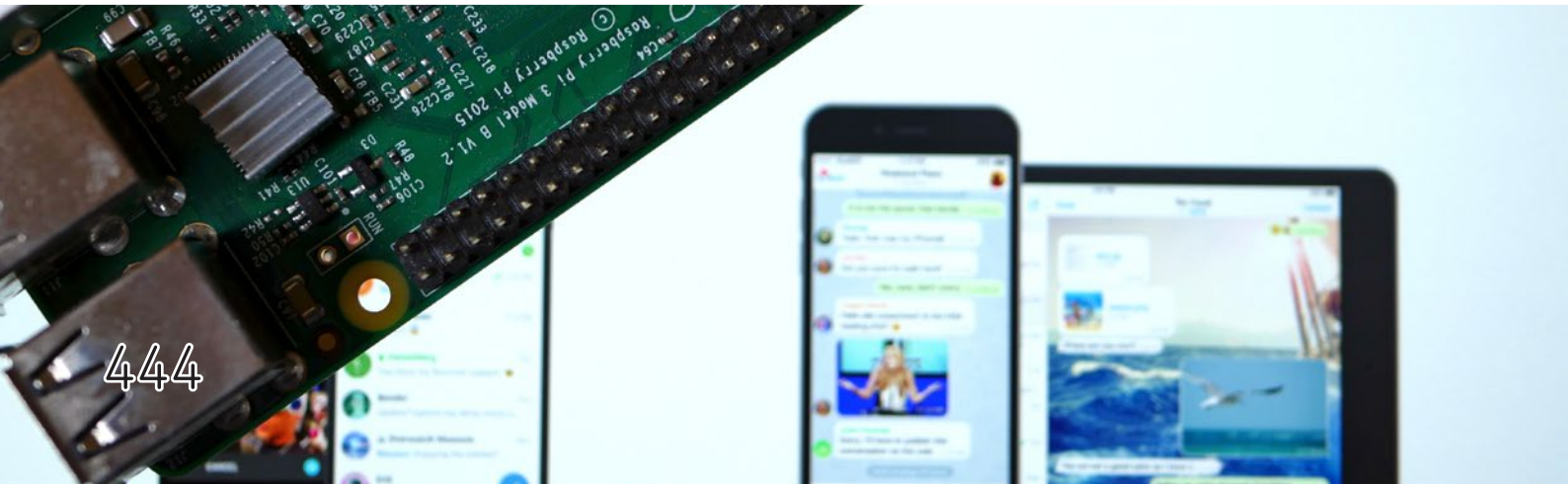
We will be showing you how simple it is to write an LUA script that will allow us to check the text of every incoming message.

This LUA script allows us to do specific actions based on the received message, the two simple tasks we will be showing you how to do is a simple receive message and reply, and also how to setup the Telegram CLI so we can trigger an external bash script and upload a photo automatically.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection
- Mobile Phone

**Optional**
- Raspberry Pi Case



## Writing your first script for Telegram CLI

**1.** Before you tackle this tutorial, you will first have had to complete our base Telegram tutorial as this takes you through all the steps of setting up Telegram on the Raspberry Pi.

Once you have finished setting up Telegram, you can continue with this tutorial. With this tutorial will be showing you how to utilize the Telegram CL's LUA interpreter so we can actively monitor the received messages and take actions based on what we have received.

This LUA script will be basic, with it replying to every message that says "ping" with "pong", while simple it will give you an idea on how to handle text messages.

Let's begin writing our LUA script for Telegram by running the following two commands on our Raspberry Pi.

```
cd ~/tg
nano actions.lua
```

**2.** With our **actions.lua** file opened we will need to write the following lines of code into it. Remember to retain the tabs as LUA is whitespace sensitive.

```
function on_msg_receive (msg)
    if msg.out then
        return
    end
    if (msg.text=='ping') then
        send_msg (msg.from.print_name, 'pong', ok_cb, false)
    end
end
```

This snippet is the main function that we will be using in this tutorial. This function is called every time the Telegram CLI receives a new message. We first make sure that this isn't a message that we just sent by checking if **msg.out** is true.

If it is false, we then proceed to check the content of the message. If the message is equal to the string "ping" we then continue to send a message back in reply.

```
function on_our_id (id)
end
```

When this function is called, it provides the id of the currently logged in user. We do not need to implement anything for this as we are focusing on receiving and replying to messages.

```
function on_secret_chat_update (user, what_changed)
end
function on_user_update (user, what_changed)
end
function on_chat_update (user, what_changed)
end
```

These three functions are called when there is updated information on a user. The variables provided back are the user that was updated and an array of what changed.

```
function on_get_difference_end ()
end
```

This function is called after the first **get_difference** call. This call means that all updates after the last client execute have been received.

```
function on_binlog_replay_end ()
end
```

This function is called after the replay of old events has ended.

## Writing your first script for Telegram CLI

**3.** Once you have finished typing in the code, you should end up with something like what is displayed below. Check over your code and ensure you haven't made any mistakes.

```
function on_msg_receive (msg)
    if msg.out then
        return
    end
    if (msg.text=='ping') then
        send_msg (msg.from.print_name, 'pong', ok_cb, false)
    end
end

function on_our_id (our_id)
end

function on_secret_chat_update (user, what_changed)
end

function on_user_update (user, what_changed)
end

function on_chat_update (user, what_changed)
end

function on_get_difference_end ()
end

function on_binlog_replay_end ()
end
```

Once you are happy with the code press **Ctrl + X**, then **Y** then **Enter** to save the file.

**4.** With our LUA script now written we can finally test to see if it is working successfully with Telegram.

To do this, we need to run the following two commands on our Raspberry Pi.

These commands will launch Telegram CLI and send it our script to utilize.

```
cd ~/tg
bin/telegram-cli -k tg-server.pub -W -s actions.lua
```

**5.** Now that the Telegram CLI (Command line interface) is now up and running with our script, we need to send a message to our Telegram account. The message you need to send is "**ping**". Immediately upon sending the message, the Raspberry Pi should receive it, check whether the message matches to "**ping**" then reply with the message "**Pong**".

If you receive "**Pong**" on your device, you have successfully written your first LUA script that lets your Raspberry Pi reply automatically to messages.

Over the next section, we will be diving into this concept even further, this time showing how you can use the LUA script to run an outside batch script that will take a photo and send it over Telegram to the receiver.

## Automatically sending photos through Telegram

**1.** To proceed with this section of the tutorial, you will need to of installed and set up the Raspberry Pi Camera as we will be using that to automatically take photos and send them when a message is received by our LUA script.

To get started we will make a directory where we will temporarily store the images we take and where we will store the bash script we will need to write for our LUA script to call.

Run the following command on our Raspberry Pi to create our new folder.

```
sudo mkdir ~/camera
```

**2.** With the folder created let's write our bash script, this will make a call to the **raspistill** application so the Pi will take a photo and save it into our new **/home/pi/camera/** directory.

```
nano ~/camera/camera.sh
```

**3.** Within this file write the following lines of code:

```
#!/bin/bash
raspistill -w 800 -h 600 -o /home/pi/camera/photo.jpg
```

Save the file by pressing **Ctrl + X** then **Y** and then finally hit **Enter**.

**4.** Now before we go ahead and modify our "**actions.lua**" script we will need to make sure we can execute our new bash script from our Lua script.

Run the following command on the Raspberry Pi to set the correct permissions.

```
sudo chmod 755 ~/camera/camera.sh
```

**5.** Finally, we can move onto modifying our **actions.lua** script that we wrote earlier in the tutorial.

Run the following command to begin editing the file on your Raspberry Pi.

```
nano /home/pi/tg/actions.lua
```

**6.** In this file we need to add the following code:

### Find

```
if msg.out then
   return
end
if (msg.text=='ping') then
    send_msg (msg.from.print_name, 'pong', ok_cb, false)
end
```

### Add Below

```
if (msg.text=='photo') then
    os.execute('/home/pi/camera/camera.sh')
    send_photo (msg.from.print_name, '/home/pi/camera/photo.jpg', ok_cb, false)
end
```

This new bit of code works by checking for any message that is "**photo**" upon interpreting the correct message we then execute the **camera.sh** bash script that we wrote earlier in this tutorial.

After the script has been executed we can then utilize the Telegram CLI's **send_photo** function to send the photo we captured with our bash script.

## Automatically sending photos through Telegram

**7.** Once you have written these new lines into our **actions.lua** script you should end up with a file that looks like what we have shown below.

```
function on_msg_receive (msg)
    if msg.out then
        return
    end
    if (msg.text=='ping') then
        send_msg (msg.from.print_name, 'pong', ok_cb, false)
    end
    if (msg.text=='photo') then
        os.execute('/home/pi/camera/camera.sh')
        send_photo (msg.from.print_name, '/home/pi/camera/photo.jpg', ok_cb,
false)
    end
end

function on_our_id (our_id)
end

function on_secret_chat_update (user, what_changed)
end

function on_user_update (user, what_changed)
end

function on_chat_update (user, what_changed)
end

function on_get_difference_end ()
end

function on_binlog_replay_end ()
end
```

Once done, you can then save and quit out of the file by pressing **Ctrl + X**, then **Y** and then **Enter**.

**8.** Now finally we can test our new code, run the same telegram-cli command that we ran earlier in the tutorial by running the following two commands on your Raspberry Pi.

```
cd ~/tg
bin/telegram-cli -k tg-server.pub -W -s actions.lua
```

**9.** With the Telegram CLI, now running with your actions.lua script you can now send yourself a message that says "**photo**". Your Raspberry Pi should pause as it takes the photo. Afterward, you should be able to see the upload progress as it sends the photo.

Hopefully, by now you have set up your Raspberry Pi with Telegram successfully and now have a good understanding of how to use the LUA script to allow Telegram to control your Raspberry Pi.

You can find their documentation on the the LUA script functionality by going to their LUA documentation page: https://github.com/vysheng/tg/blob/master/README-LUA

Telegra
ew era of mes
for Android
Telegram for iPho

# WhatsApp

**WARNING** - There is a chance that your Mobile Phone number will be banned from utilizing WhatsApp. By following these steps, you proceed at your own risk knowing that you could be banned from the WhatsApp network.

We recommend using Telegram over WhatsApp, same features but fewer issues and less chance of being banned.

## Project Description

In this Raspberry Pi project, we will be showing you how to set up WhatsApp for your Raspberry Pi, for this tutorial you will need a spare mobile phone number that you don't already use for WhatsApp as it is limited to a singular device.

We will be utilizing a python package/application called Yowsup to handle our connection with WhatsApp. This python package implements the way WhatsApp works and forms the correct authentication codes and messages to be able to act as if it was the native application running on an actual Android phone.

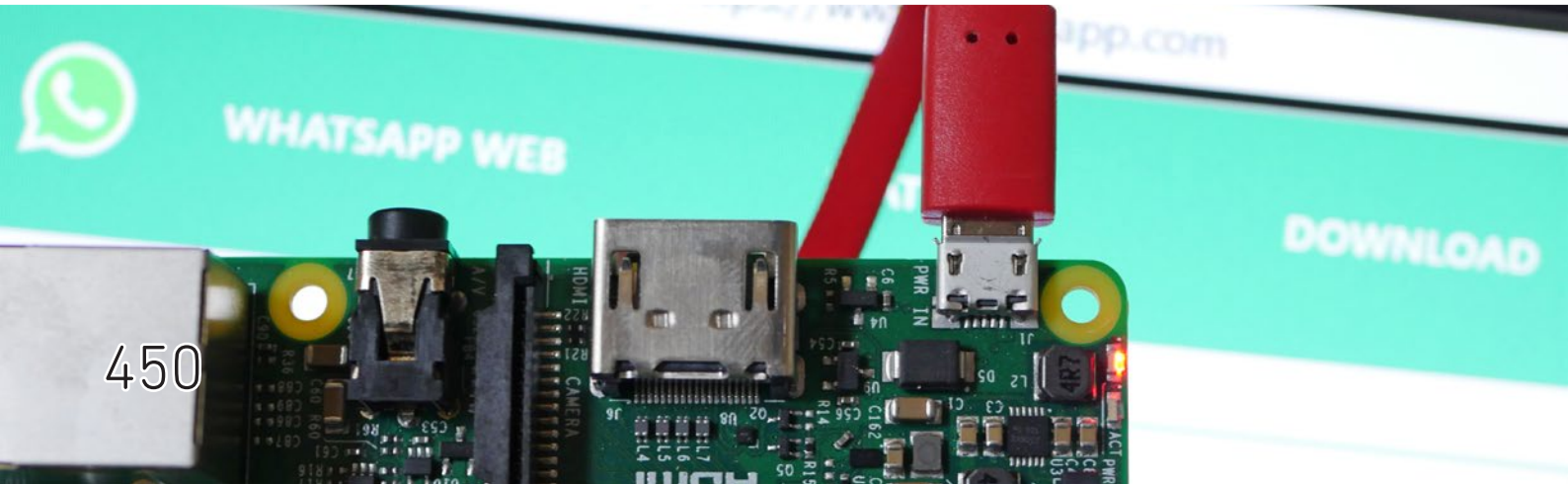
## Equipment

**Required**

- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection
- Mobile Phone

**Optional**

- Raspberry Pi Case



## Installing the WhatsApp software

Before you begin this tutorial, please make sure that you have a mobile phone number and a mobile phone spare.

You will require your mobile phone shortly as its needed after installing the software to be able to activate your WhatsApp account.

**1.** Before we begin with this tutorial, let's run an update to ensure we are running on the latest available software. Run the following command in terminal to update your Raspberry Pi.

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

**2.** With our Raspberry Pi up to date, we can start setting up for WhatsApp. To utilize the python package, we are going to use (**Yowsup**) we need to install a wide variety of dev packages.

Install all the needed packages by typing the followings commands into the terminal on your Pi.

```
sudo apt-get install python-dev
sudo apt-get install python-dateutil
sudo apt-get install python-setuptools
sudo apt-get install ncurses-dev
sudo apt-get install libevent-dev
```

**3.** Just before we go ahead and clone the code we need to set up WhatsApp on our Raspberry Pi we must first install the required Python packages we need.

To do this, we will just type the following two commands into the terminal on our Raspberry Pi.

```
sudo easy_install --upgrade pip
sudo python pip install -U bs4 requests
```

**4.** Now that we have installed all the packages that we need to use Yowsup we can continue.

We will be cloning Yowsup directly from its Github page, to do this we will first change into our Pi users root directory, then utilize git clone to grab it.

```
cd ~
git clone https://github.com/pimylifeup/yowsup.git
```

**5.** With the Yowsup repository now cloned to our Raspberry Pi, we can navigate to its folder and run the **update.py** script by running the command below:

This script is quite crucial for getting the WhatsApp library working on our Raspberry Pi as it ensures that the **env_android.py** script has the latest available version number.

We achieve this by downloading the official **WhatsApp.apk** directly from WhatsApp, opening the file up, extracting the class we need then calculating the correct hash from it.

```
cd ~/yowsup
python update.py
```

**6.** Now that the **env_android.py** file should be updated with the correct values we can go ahead and install the yowsup library to the Raspberry Pi. We can achieve this by running the following two commands on our Raspberry Pi.

```
cd ~/yowsup
sudo python setup.py install
```

## Registering with WhatsApp

With the Yowsup python package now installed we can proceed with setting up WhatsApp on our Raspberry Pi.

The first thing we need to do is register our device with WhatsApp. Thankfully Yowsup provides a command-line interface for register your device.

Please note that your phone number can only be assigned to one device at a time, so we recommend using a number that you have no intention of using for WhatsApp on your phone.

**1.** To begin, we need to request a registration code from WhatsApp. Registering requires sending several details to WhatsApp, make sure you have your phone handy as the code will be sent to that.

For the following command, you will need to grab several pieces of information.

You will need to go to https://en.wikipedia.org/wiki/Mobile_country_code to grab both your **Mobile Network Code** (**MNC**) and your **Mobile Country Code** (**MCC**). You find this by looking in your country, then searching for the provider that you are with.

For example, if we lived in the **United Kingdom** and were with **Vodafone**, our **MCC** would be **234**, and my MNC would be **15**.

With both our **MNC** and our **MCC** now handy we also need to grab our country code. To find our both our **Country Code** (**cc**) we can utilize yet another Wikipedia page, go to this one by clicking the following link https://en.wikipedia.org/wiki/List_of_mobile_phone_number_series_by_country.

On here we need to look up our country and look at the **International Calling Code** (**CC**).

For example, if we lived in the **United Kingdom**, we would be using the International Calling Code **44**.

Now that we have our MNC, MCC and CC number we can finally utilize the command.

Type the following command into the terminal on your Raspberry Pi.

Make sure that you replace the **phone** number with your **mobile phone number**.

**cc** with the **calling code** you grabbed earlier, in our example this is 44.

**mcc** with the Mobile Network Code, in our example this is 234.

**mnc** with the Mobile Country Code, in our example this is 15.

We have used our example data to give you an idea of what you will need to input.

```
python yowsup-cli registration --requestcode sms --phone 44xxxxxxxxxx -
-cc 44 --mcc 234 --mnc 15 --env android
```

## Registering with WhatsApp

**2.** Now that you have entered that command, your phone should have received a registration code from WhatsApp.

This code should appear something like **xxx-xxx**.

With our registration code in hand, we will need to again send this back to WhatsApp by utilizing the command line interface of Yowsup again.

This time you will only need your international calling code, your phone number, and registration code.

With those handy type in the following command, making sure to switch out the values. We have again used our example data as a placeholder.

```
python yowsup-cli registration --register xxx-xxx
--phone 44xxxxxxxxxx --cc 44 --env android
```

**3.** Assuming everything works correctly, you should see a message like below.

Make sure you write down the password as you will need this in the next steps.

```
status: ok
kind: free
pw: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx=
price: £0.99
price_expiration: 1504913339
currency: GBP
cost: 0.99
expiration: 4444444444.0
login: 44xxxxxxxxxx
type: new
```

## Utilizing WhatsApp on your Raspberry Pi

**1.** Now that we have setup our Raspberry Pi as a WhatsApp device we need to create a config file to keep all our details.

Make sure you have your phone number, password and country code handy.

Type in the following command to create our config file.

```
nano /home/pi/yowsup/config
```

**2.** Within this file write in the following lines, make sure you replace cc with your country code, phone with your phone number and set your password.

We have included our example data to give you an idea of what it should look like.

```
cc=44
phone=44xxxxxxxxxx
password=xxxxxxxxxxxxxxx=
```

Now save the file by pressing **Ctrl + X**, then Pressing **Y** and then hitting **Enter**.

**3.** With the config file now generated we can test out a Yowsup demo, this demo will allow us to test whether we can log in and also test sending and receiving messages.

To start up the demo run the following commands on your Raspberry Pi:

```
cd ~/yowsup
yowsup-cli demos --yowsup --config config
```

**4.** Once it has loaded up, we can now log in by running the following command in the command-line interface that is currently being displayed.

This command will do the login handshake with WhatsApp's servers.

```
/L
```

**5.** Once we have got a message saying the login was successful we can try sending a message. To try this, you must use a phone number of someone that also has WhatsApp installed.

Once you have a number to try you can utilize the following command, remember you need to know their country code as well.

```
/message send 44xxxxxxxxxx "This is a test message"
```

If that person responds to the message, it will be displayed in the command line automatically when the Raspberry Pi receives it.

Hopefully, by now you have set up WhatsApp successfully on your Raspberry Pi and have a basic idea of how it all works.

If you are interested in extending this further or extending it with your application, then you should check out the documentation: https://github.com/tgalal/yowsup/wiki/Sample-Application

Secure
https://
WHATSAPP WEB
HDMI
CAMERA
A/V
FCC ID: 2ABCB-RPI32
IC: 20953-RPI32
Raspberry Pi 3 Model B V1.2
© Raspberry Pi 2015
RUN
ATSAPP

# Internet Speed Monitor

## Project Description

In this tutorial, we will be showing you how you can set up your Raspberry Pi as your very own internet speed monitor.

This tutorial is great for those who are interested in monitoring how their download speed, upload speed, and ping is changed over time. Additionally, this can help you work out what times your network may be at its peak capacity.

To achieve this, we will be showing you how to write a small python script that interacts with a program called speedtest-cli.

Speedtest-cli is what our Raspberry Pi internet speed monitor will use to monitor the internet speed.

This program works be polling the popular speedtest.net service to get your ping, download speed, and upload speed.

## Equipment

**Required**
- Raspberry Pi
- (Micro) SD Card (8 GB+ Recommended)
- Raspberry Pi Power Supply
- Internet Connection

**Optional**
- USB Keyboard
- USB Mouse
- HDMI Cable
- Raspberry Pi Case

## Installing speedtest-cli

**1.** Before we get started up setting up our Raspberry Pi to monitor our internets speed, we must first make sure our Raspberry Pi is entirely up to date.

We can update our Raspberry Pi by running the following two commands within the terminal.

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

**2.** For our tutorial, we will be utilizing a Python library called speedtest-cli. This Python library acts as a command line interface that talks with speedtest.net.

Using this library, we can quickly poll Speedtest.net to retrieve our current upload and download speeds as well as our current ping.

However, before we use this library, we must first install a package called Python pip.

The reason we need this package is that **speedtest-cli** is not available in the Raspbian package repository.Instead, we need to retrieve it from the Python Package Index (PyPI) which is thankfully easy to do thanks to pip.

In some cases pip is pre-installed, but we will run the following command to make sure that it is installed.

```
sudo apt-get install python-pip
```

**3.** With pip now installed to our Raspberry Pi, we can move onto using it to download the speedtest-cli python library.

Pip is very easy to use and is very much like apt-get. Just run the following command on your Raspberry Pi to use pip to install speedtest-cli.

```
sudo pip install speedtest-cli
```

**4.** Now that we have installed speedtest-cli we can now use it.

Let's make sure everything is working correctly by running the following command.

```
speedtest-cli
```

**5.** You will see that quite a bit of information is returned by this command, to make things more simple we can use the following command.

```
speedtest-cli --simple
```

As you can see with the **--simple** argument, just the download, upload and ping are returned

## Writing our speed test python script

**1.** Now that we have speedtest-cli installed on our Raspberry Pi, we can now proceed to write our Python script that will continually monitor our download and upload speeds.

We can begin creating our Python script by running the following command.

```
cd ~
sudo nano speedtest.py
```

**2.** Within this file write the following lines of code, we will explain each important section of the code, so you get an idea of how everything works.

```
import os
import re
import subprocess
import time
```

These four lines define all of the libraries that we will be relying on in our script. Below we will explain what each of these libraries are used for.

**import os** - The os library is used for the script to interact with the operating system itself. For this script, we will be just using this to check if a file exists.

**import re** - The re library allows us to easily do regular expressions by providing a library to easily handle pattern searches. We use this to find our wanted values given to use from speedtest-cli.

**import subprocess** - The subprocess library is important to this script, as we require it to be able to call another python script.

In our case, we will be using the subprocess library so we can launch up the speedtest-cli script and retrieve the values returned by it.

**import time** - We utilize the time library so that we can record both the date and time for each call to the speedtest-cli. This is what will allow us to track our speed over a length of time.

```
response = subprocess.Popen('/usr/local/bin/envspeedtest-cli --simple', shell=True, stdout=subprocess.PIPE).stdout.read()
```

In this line of code, we utilize the subprocess library to launch a call to the speedtest-cli python script and tell it to pipe everything from the speedtest-cli to stdout.

By using **stdout.read()** at the end of the call we store the response from speedtest-cli to our response variable.

```
ping = re.findall('Ping:\s(.*?)\s', response, re.MULTILINE)
download = re.findall('Download:\s(.*?)\s', response, re.MULTILINE)
upload = re.findall('Upload:\s(.*?)\s', response, re.MULTILINE)
```

These three lines of code are fairly simple, and all do the same thing. They use the re library to run a regular expression for a certain piece of text and find the number located next to each piece of text.

For instance, the search for ping finds "**Ping: 47.943 ms**" but only grabs the number that is between the text.

# Writing our speed test python script

```
ping = ping[0].replace(',', '.')
download = download[0].replace(',', '.')
upload = upload[0].replace(',', '.')
```

These lines are important as we use them to clean up the numbers that we grabbed from the speed-test-cli output.

Since we plan on outputting the data into a CSV format, all number values need to have the commas (**,**) replaced with decimal points (**.**). Otherwise, the CSV format will split the numbers into new columns.

Thanks to Pythons **.replace** we can easily do this on each piece of text.

```
try:
   f = open('/home/pi/speedtest/speedtest.csv', 'a+')
   if os.stat('/home/pi/speedtest/speedtest.csv').st_size == 0:
         f.write('Date,Time,Ping (ms),Download (Mbit/s),Upload (Mbit/s)\r\n')
except:
   pass
```

This bit of code is straightforward. The whole code is kept within a try statement so that if any errors occur, it will not stop the script from operating.

Within the try statement, we first open up a call to our speedtest.csv file. By using, "**a+**" in the arguments, we tell it that we want to create the file if it doesn't exist and that any new data should be appended to whatever is there.

Afterward, we utilize the os library to check our **speedtest.csv** files actual size in bytes.

If the file's bytes is equal to **0**, we go ahead

If the file does exist we proceed on as normal.

```
f.write('{},{},{},{},{}\r\n'.format(time.strftime('%m/%d/%y'), time.strftime('%H:%M'), ping, download, upload))
```

Finally, we print out all our data separated by commas.

We use the time library's **strftime** function to insert both the current date and the current time into our formatted string.

After that, we insert our ping, download, and upload.

Below we have an included an example of what the output data will look like on the first run from our code.

```
Date, Time, Ping (ms), Download (Mbit/s),Upload(Mbit/s)
04/28/18,13:34,27.833,27.28,9.02
```

## Writing our speed test python script

**3.** Once you have finished writing your code, it should end up looking like what we have displayed below.

```
import os
import re
import subprocess
import time

response = subprocess.Popen('/usr/local/bin/envspeedtest-cli --simple', shell=True, stdout=subprocess.PIPE).stdout.read()

ping = re.findall('Ping:\s(.*?)\s', response, re.MULTILINE)
download = re.findall('Download:\s(.*?)\s', response, re.MULTILINE)
upload = re.findall('Upload:\s(.*?)\s', response, re.MULTILINE)

ping = ping[0].replace(',', '.')
download = download[0].replace(',', '.')
upload = upload[0].replace(',', '.')

try:
    f = open('/home/pi/speedtest/speedtest.csv', 'a+')
    if os.stat('/home/pi/speedtest/speedtest.csv').st_size == 0:
        f.write('Date,Time,Ping (ms),Download (Mbit/s),Upload (Mbit/s)\r\n')
except:
    pass

f.write('{},{},{},{},{}\r\n'.format(time.strftime('%m/%d/%y'), time.strftime('%H:%M'), ping, download, upload))
```

**4.** You can now save the file by pressing **Ctrl + X** then **Y** and finally press **Enter**.

**5.** With our script now written we need to make a folder where our speedtest.csv file will be stored. Run the following command to create this folder.

```
mkdir ~/speedtest
```

**6.** Now that we have written our script and saved it, we can now go ahead and test it out.

We can run our script by running the following command.

```
python ~/speedtest.py
```

**7.** Once the script has finished executing you can then check out the results by opening up the newly created **speedtest.csv**.

Let's open up this file by running the following command on our Raspberry Pi.

```
nano ~/speedtest/speedtest.csv
```

**8.** Within this file you should see something like what we have below. The column headers and some rows of data.

```
Date, Time, Ping (ms), Download (Mbit/s),Upload(Mbit/s)
04/28/18,13:34,27.833,27.28,9.02
```

Next we will be showing how you can take this data and upload it to Google Drive to quickly view the data wherever you are.

# Uploading Speed test data to Google Drive

**1.** For this section of the guide, we will be showing you how you can upload your speed test data from your Raspberry Pi to Google Drive.

To do this, we will need to make use of a piece of software that allows us to interact with Google Drive through the command line.

This piece of software is called gDrive and was developed by a user on Github. You can find more information about this piece of software on Github at: https://github.com/prasmussen/gdrive

Download the Raspberry Pi version of the gdrive software by running the following two commands.

```
cd ~
wget -O gdrive https://docs.google.com/uc?id=0B3X9GlR6EmbnVXNLanp4ZFRRbzg&export=download
```

**2.** Now that we have downloaded the gdrive package to our Raspberry Pi we need to give it execution privileges so that our scripts can actually use it.

We can do this by entering the following command into terminal, utilizing **chmod** with the **+x** argument.

```
sudo chmod +x gdrive
```

**3.** Let's now begin the process of connecting your Google Account up with the gdrive application.

To start the process, we will use the following simple command.

```
./gdrive list
```

**4.** After running gdrive with the list command, we should now be told that authentication is needed to continue.

Below the message should be an URL. You need to go to this URL in a web browser and log in using your Google Account.

Follow the next few prompts in your web browser till you are given a verification code. Copy this verification code into the terminal and press **Enter**.

**5.** After pressing enter with the correct verification code, you should now see a list of all your gdrive files.

Next, let's use grdrive to create a new folder on your Google Drive to store our speedtest.csv file, we can do that by just using the following command within the terminal.

```
./gdrive mkdir speedtest
```

This command will show you a message saying the directory was created.

This message will also show you an ID. We will need this ID for our next few steps so write it down somewhere safe.

## Uploading Speed test data to Google Drive

**6.** Now that we have used the gdrive command line interface to create a folder on our Google Drive and have the ID handy we can now try uploading something to it.

For this, we will be uploading our speedtest.csv file, just run the following command and make sure you replace **YOUR_FOLDER_ID** with the id you retrieved in the previous step.

```
./gdrive sync upload ~/speedtest YOUR_FOLDER_ID
```

On the initial sync, you should see a message like below appear on the command line. This message tells you that the file has been successfully synced to your Google Drive account.

```
Starting sync...
Collecting local and remote file information...
Found 1 local files and 0 remote files

1 remote files are missing
[0001/0001] Uploading speedtest.csv -> speedtest/speedtest.csv
Sync finished in 3.119979351s
```

**7.** With Google Drive synching now setup on our Raspberry Pi we can move onto automating the whole process so that we upload the **speedtest.csv** every time we edit it.

Learn how to setup the automation process over the page.

## Automating the speed test process.

**1.** Now that we have setup both the gdrive application and written up and our python script we need to now move onto automating the whole process.

To do this, we will be writing a simple bash script that we will use crontab to run routinely.

Begin writing this simple bash script by running the following command on your Raspberry Pi.

```
nano /home/pi/speedtest.sh
```

**2.** Within this file, we want to enter the following lines.

Make sure that you replace YOUR_FOLDER_ID with the ID that you received when you created the folder on google drive.

```
#!/bin/bash
python /home/pi/speedtest.py
/home/pi/gdrive sync upload  /home/pi/speedtest YOUR_FOLDER_ID
```

**3.** Now save the file by pressing **Ctrl + X** then **Y** and then finally **Enter**.

**4.** Before we set up the crontab for our bash script, we must first give it execution privileges.

We can achieve this by running the following command within the terminal.

```
sudo chmod +x /home/pi/speedtest.sh
```

**5.** Now with everything done, we can finally move onto setting up the crontab.

Begin editing the crontab by running the following command on your Raspberry Pi, if you are asked what editor you want to use we recommend you select Nano (Typically number 2).

```
crontab -e
```

**6.** To this file add the following line at the bottom of the file. This line will tell crontab that it should run our bash script once every hour.

If you want to change the behavior of the crontab, you can look at the Crontab Guru to work out your desired values: https://crontab.guru/

```
0 * * * * /home/pi/speedtest.sh
```

**7.** You should now have your speed test monitor now up and running with it continually uploading its results to your Google Drive account.

# Installing OctoPrint

## Project Description

In this tutorial, we will be walking you through all the steps that you need to follow to be able to set up OctoPrint on your Raspberry Pi and connect to it from Cura.

To simplify things, we will be using the OctoPi distribution. This distribution is a pre-setup version of Raspbian with everything that you need to get OctoPrint up and running already installed. It also helps simplify the process of setting up a WiFi connection among other things.

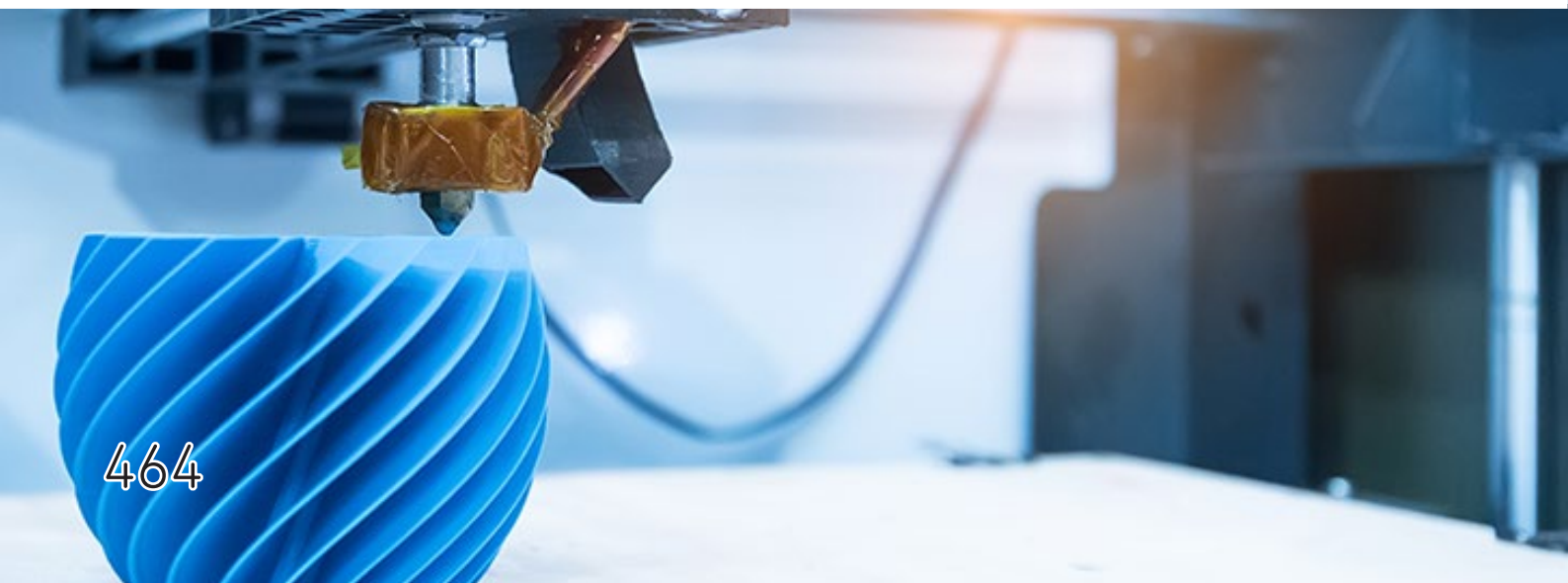
To be able to use OctoPrint make sure that your 3D Printer has an available serial port. Many 3D printers allow this ability over a USB cable connection. However, some might require more complicated methods that we won't be touching on in this tutorial.

Our tutorial will assume you already have your Raspberry Pi connected to your 3D printer via its serial connection.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection
- 3D Printer

**Optional**
- Raspberry Pi Case



# Step 1 - Formatting your SD Card

First off you should do when installing a new image to an SD card is format it correctly.

The following steps will show you the best way to format your SD Card for the Raspberry Pi:

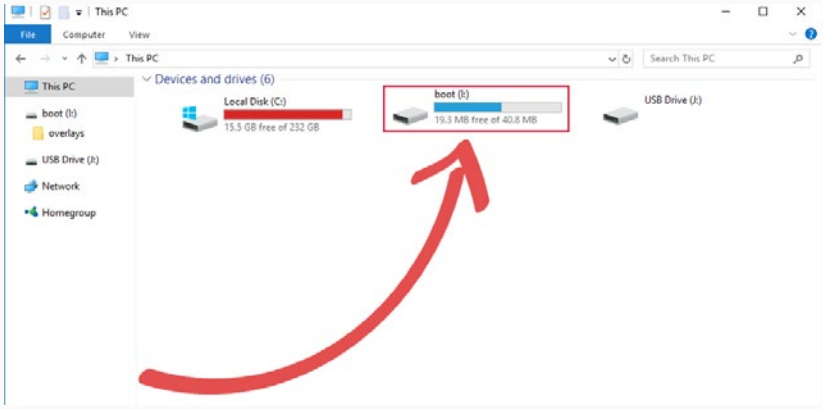
**1.** To begin you need a tool to format your SD Card. You can obtain one of the most robust ones from here: **https://www.sdcard.org/downloads/formatter_4/**

Once you are on the website will find a download link for the SD Formatter.

**2.** Download and install the SD Card Formatter from the SD Card association.

**3.** With the SD Formatter installed to your computer, insert your SD card and check the drive letter that is allocated to it, e.g., **G:/**.

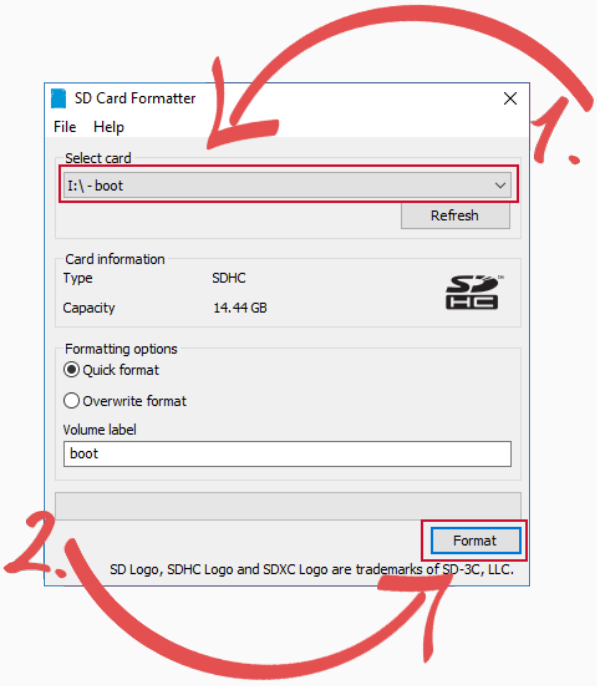
Make sure that you have the correct drive letter otherwise you could accidentally format the wrong device.




**4.** With your SD Card now plugged into your computer, launch the SD Formatter software that we installed earlier.

First set the drive letter to the correct one **(e.g., G:/)**.

Once you have everything set correctly, press the "**FORMAT**" button.



## Step 2 - Installing OctoPrint

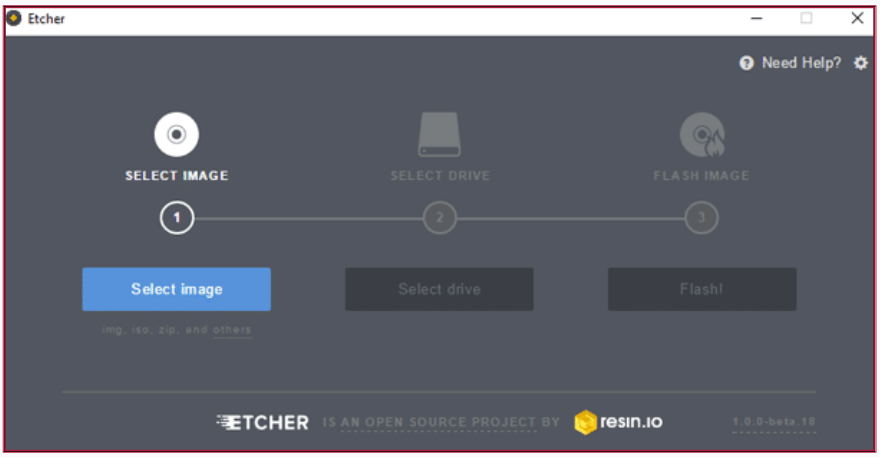
**1.** First head over to the OctoPrint website at https://octoprint.org/download/ and download the latest zip file.

Once downloaded, unzip the file so, you are left with a **.img** file.

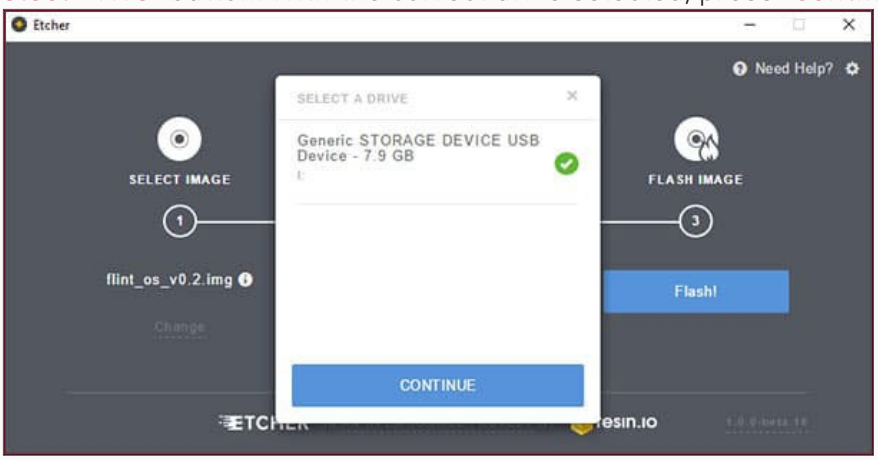
**2.** Next, you will need some tools to be able to write the image to the SD card.

For this we will use a tool called Etcher, you can download at https://etcher.io/.

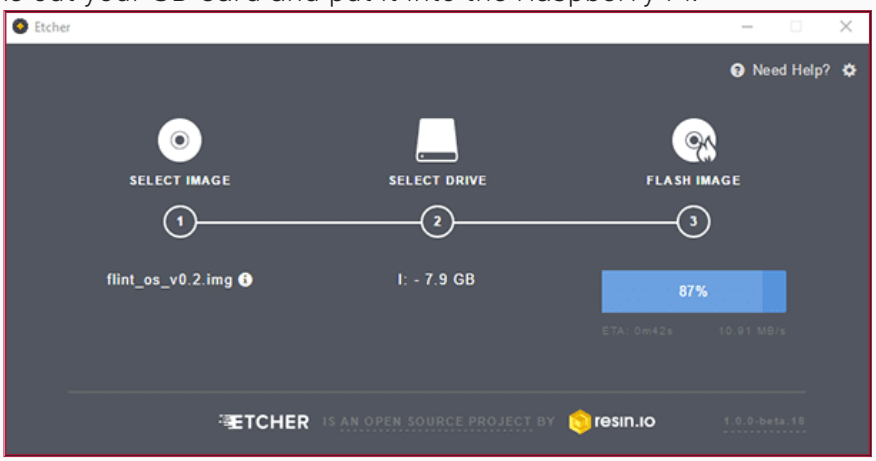
**3.** Open Etcher and press the "**Select Image**" button. Browse to the image you downloaded in **Step 1** and select it.




**4.** Next click the "**Select Drive**" button. With the correct drive selected, press "**Continue**".




**5.** Finally, click the "**Flash!**" button to begin the flashing process. This can take some time. Once completed, take out your SD Card and put it into the Raspberry Pi.



## Step 3 - Configuring OctoPrint for WiFi

**1.** With your SD Card still plugged into your device open up its folder on your computer.

**2.** On the root of the SD Card, you should find a file called "**octopi-network.txt**". This file holds all the information needed to connect to a WiFi device.

To add in your WiFi details open the file using a text editor, make sure you use a program such as No**tepad++**, **Atom** or **VSCode** as they don't tend to mangle the file without first warning you.

Avoid editing the file with other programs such as WordPad (Windows) or TextEdit (MacOS X) as they will enforce their encodings on the file making it fail to work.

**3.** Now within this file find the following text, this is where you will set your WiFi networks SSID name and its password.

You will need to uncomment this text and add your networks SSID and password.

**Find**

```
## WPA/WPA2 secured
#iface wlan0-octopi inet manual
#    wpa-ssid "put SSID here"
#    wpa-psk "put password here"
```

**Replace With**

```
## WPA/WPA2 secured
iface wlan0-octopi inet manual
   wpa-ssid "put SSID here"
   wpa-psk "put password here"
```

**4.** Once you have made the required changes to the file, then save it. Then just take your SD Card out and put it on the Raspberry Pi.

These are the last changes we should need to make while the SD card is still plugged into the computer. With these changes made we can now proceed to set up OctoPrint through its interface.

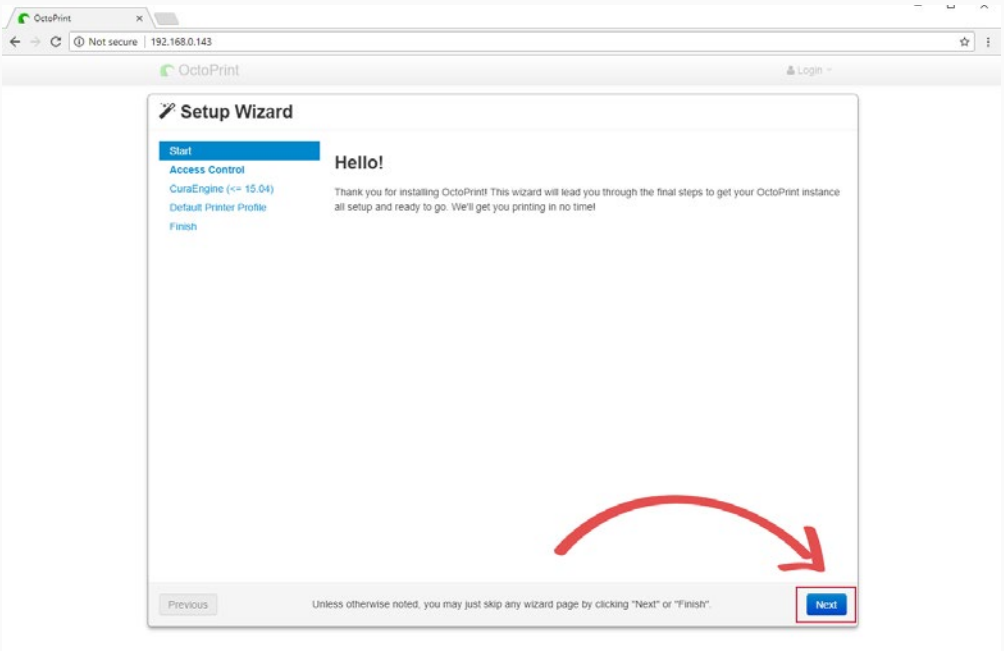
# Step 4 - Getting Started with Octoprint

**1.** Begin by going to your Raspberry Pi's IP Address, if you don't have this you should be able to find it on your router.

In some cases. you can just go to http://octopi.local to be able to load up the interface, but this does not work in all cases.

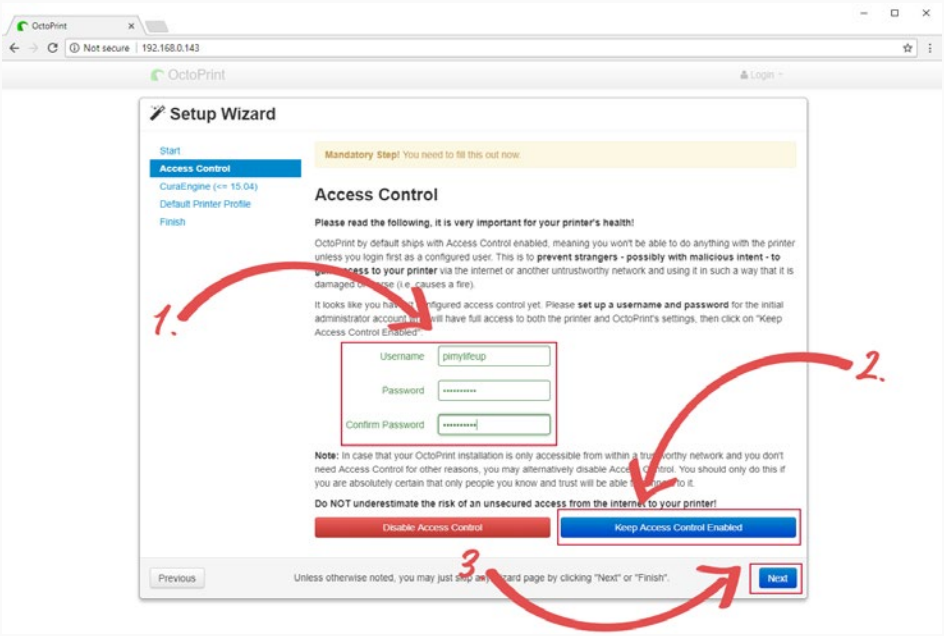
**2.** On your first visit to OctoPrints web interface, you should be greeted with the following message. Over the next few steps, we will walk you through everything that needs doing.

To begin with, just click the "**Next**" button.



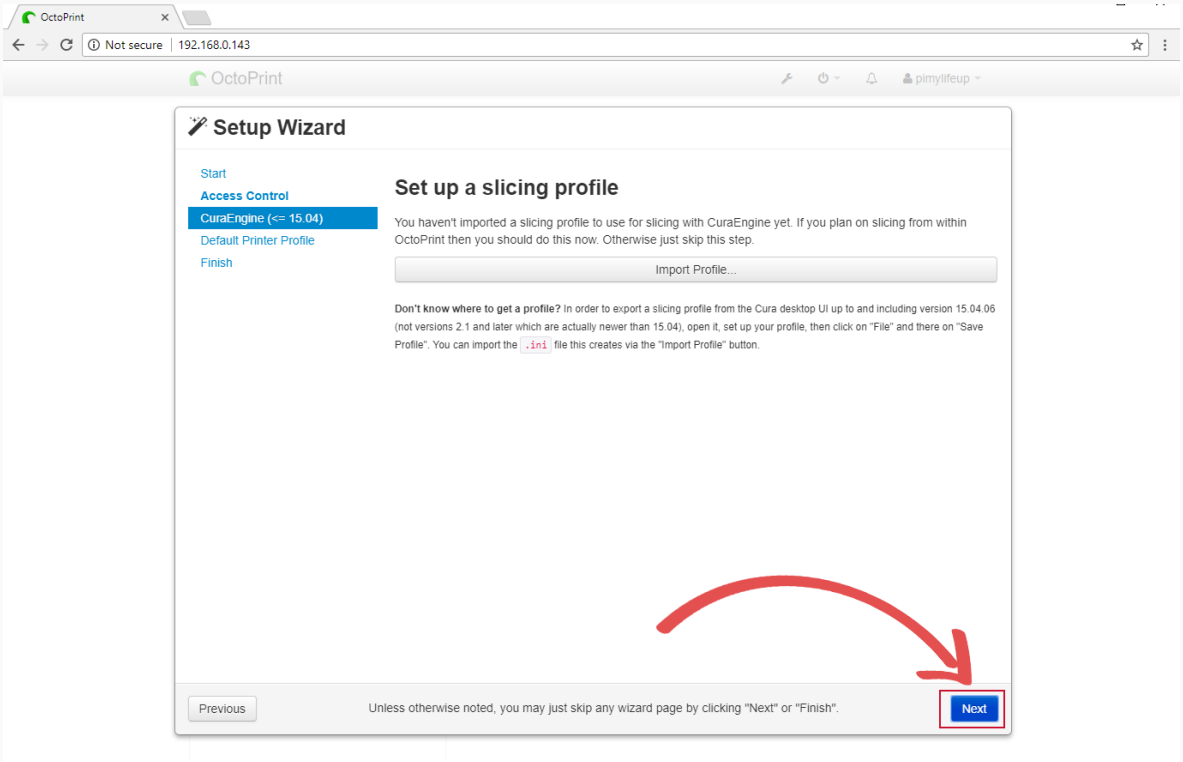

**3.** Now on this screen, you need to enter a username and a password (**1.**) to utilize to access the web interface. Not using this is a huge security risk as someone could take control of your 3D Printer.

Click the "**Keep Access Control Enabled**" button (**2.**) and then the "**Next**" button (**3.**)



# Step 4 - Getting Started with Octoprint

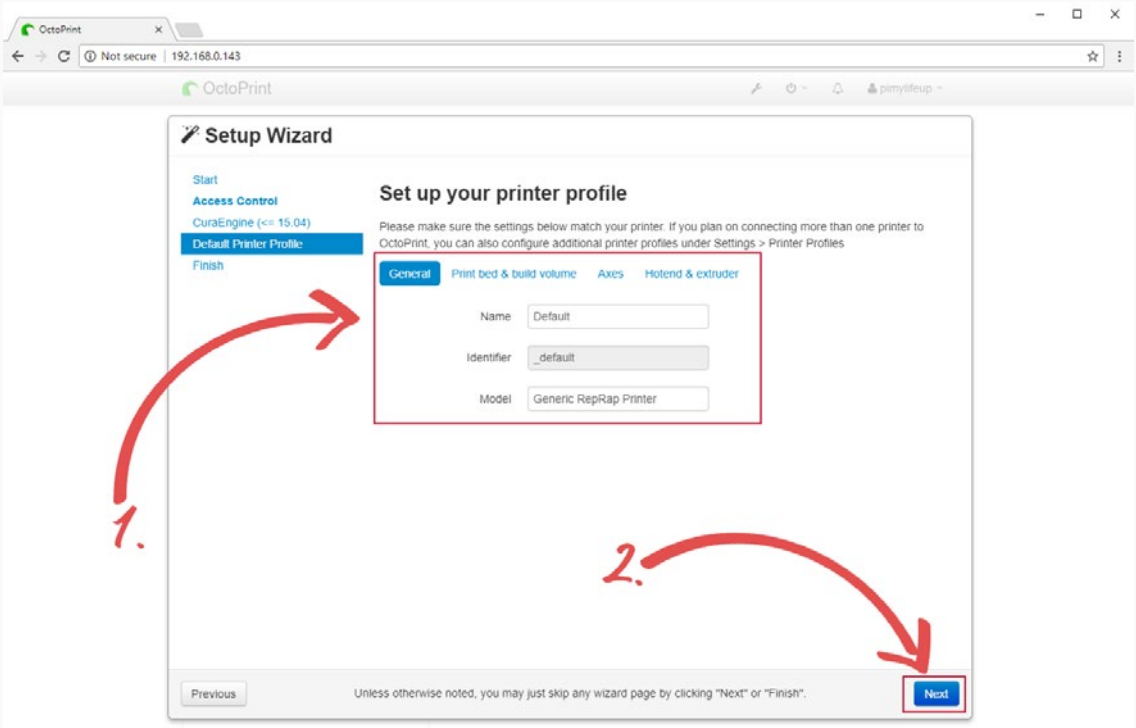
**4.** Unless you have a slicing profile that you want to utilize, just press the "**Next**" button on this screen.




**5.** This screen is highly important, on this you will need to fill out all the information of your 3D printer.

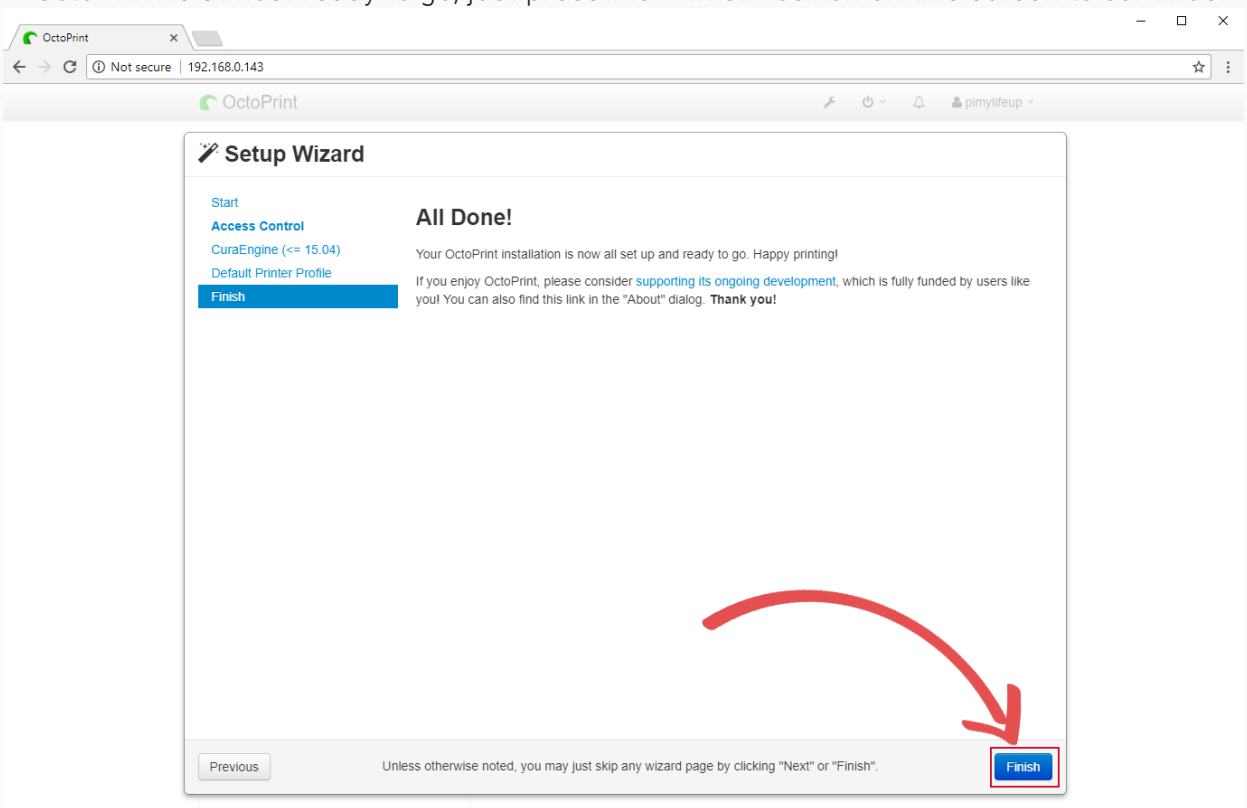
Make sure you go through each tab and ensure that the values there match the specifications of your 3D printer. You may have to google your printer to get the correct information.

Failing to enter the correct information can cause printing issues.

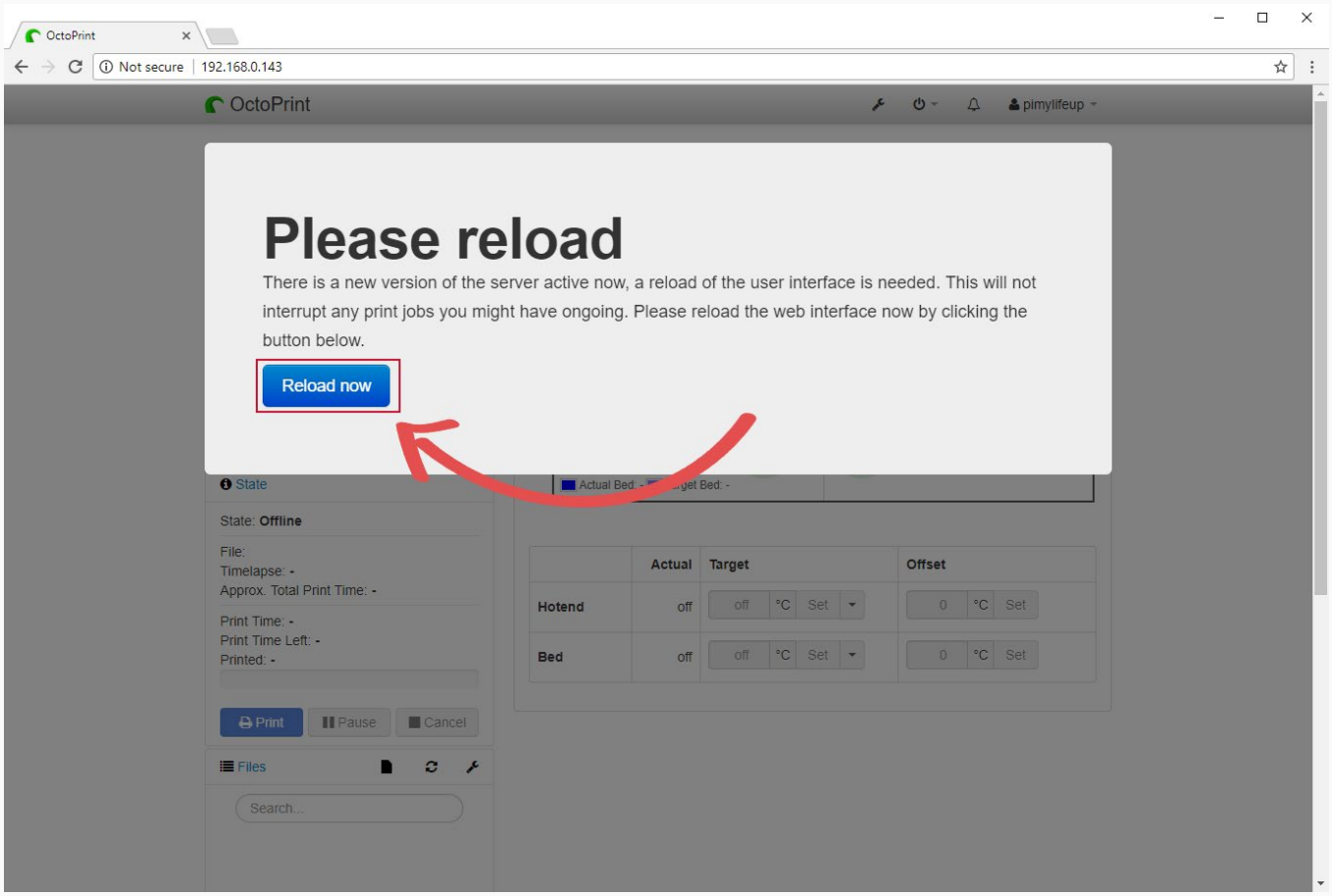


# Step 4 - Getting Started with Octoprint

**6.** Now OctoPrint is almost ready to go, just press the "**Finish**" button on this screen to continue.



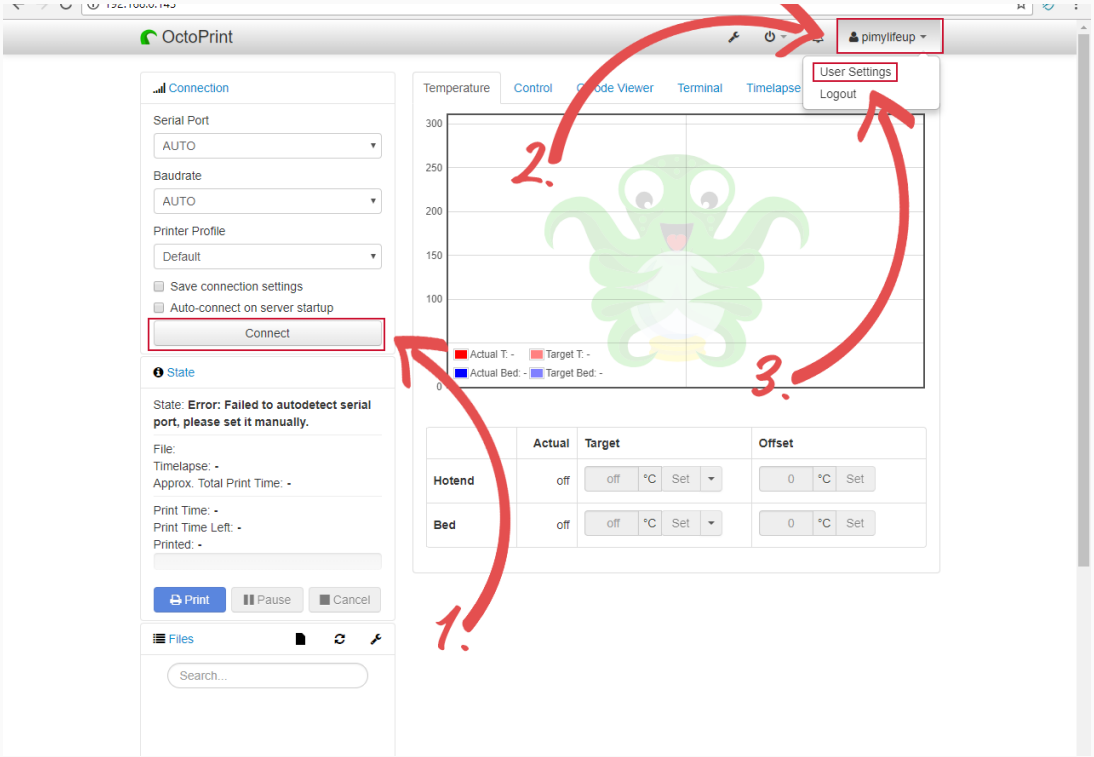

**7.** Now on here press the "**Reload now**" button. After pressing this button, we will be taken to the main OctoPrint interface.



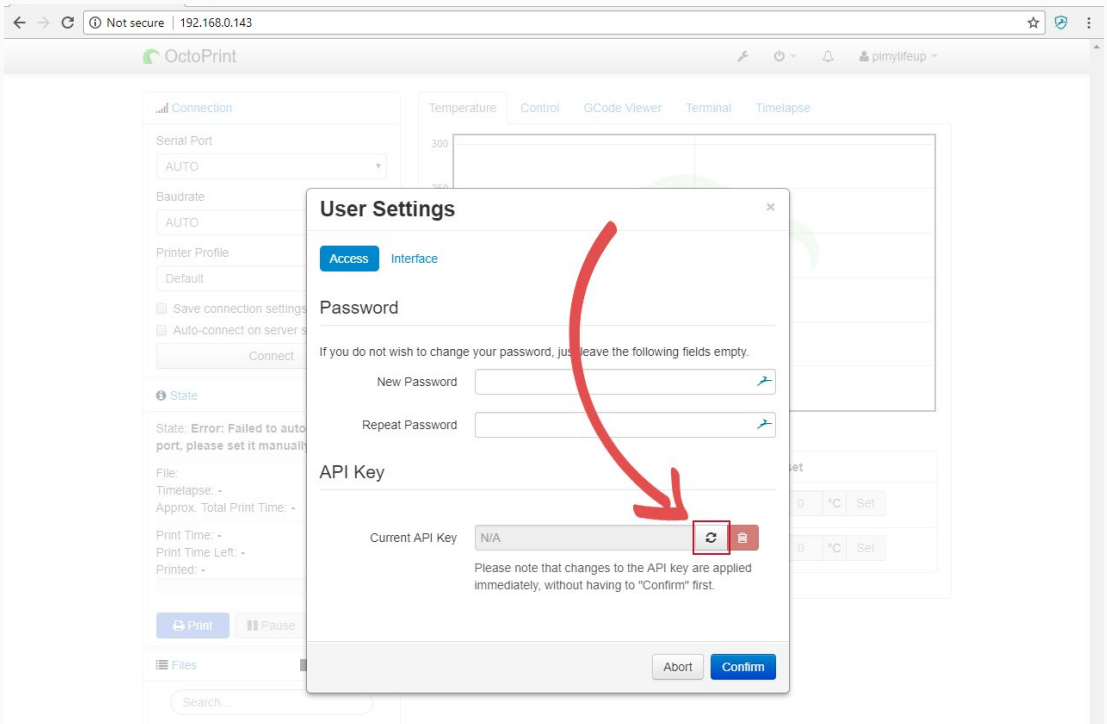
# Step 4 - Getting Started with Octoprint

**8.** Now you should be taken to the main OctoPrint screen, take time to familiarize yourself with this interface. Once you have your Raspberry Pi connected up to your 3D printer, you can then press the "**Connect**" button (**1.**)

To connect up a slicer like CURA we will need to do a couple more steps, begin by clicking your user-name in the top right-hand corner (**2.**) and then clicking the "**User Settings**" button underneath it (**3.**)



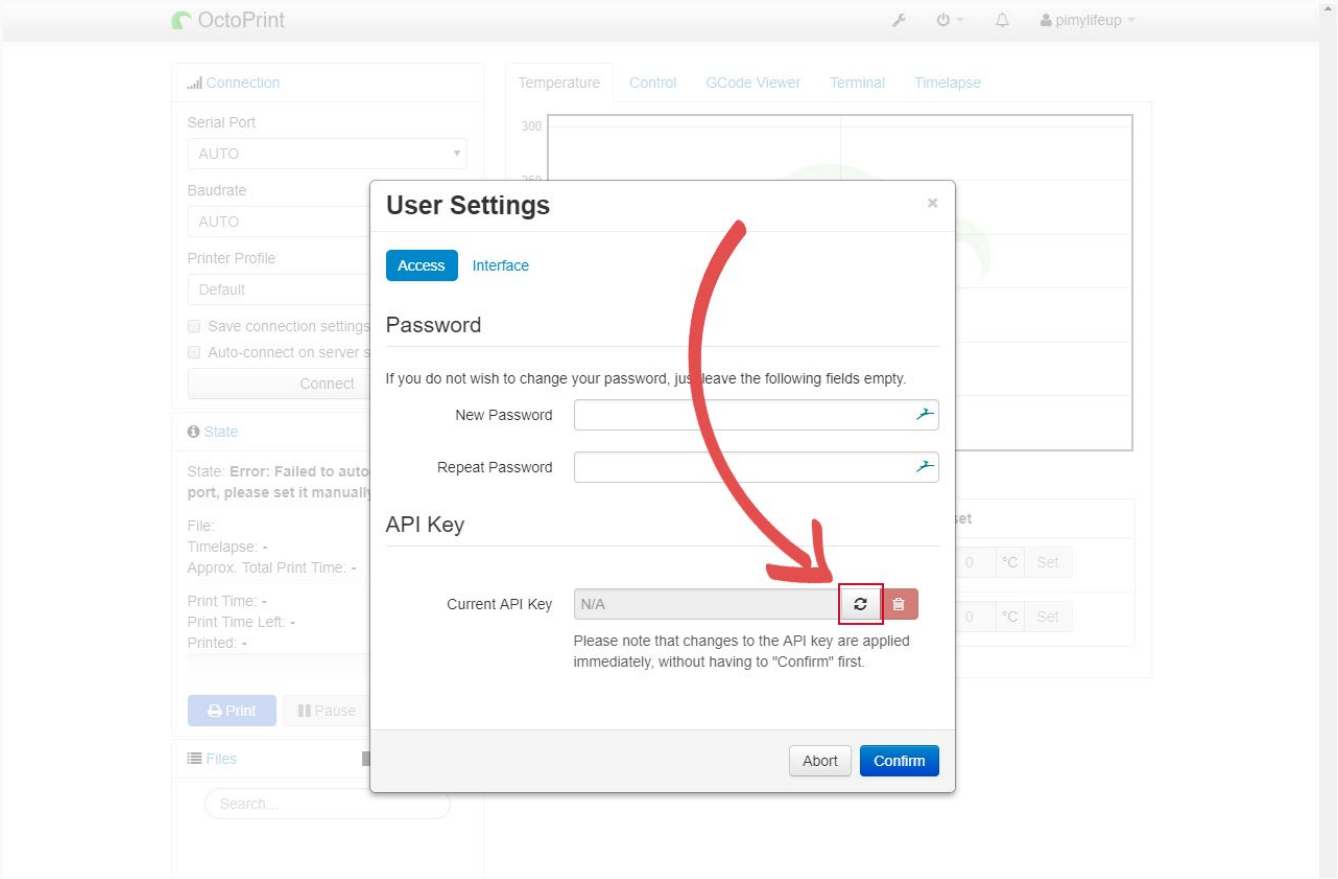

**9.** On this screen in the API Key section click the refresh button as shown below. This button will generate a new API key for us to utilize in CURA.



# Step 4 - Getting Started with Octoprint

**10.** Now copy the current API key that is now shown. We will need this for getting CURA to connect with OctoPrint so keep it safe.

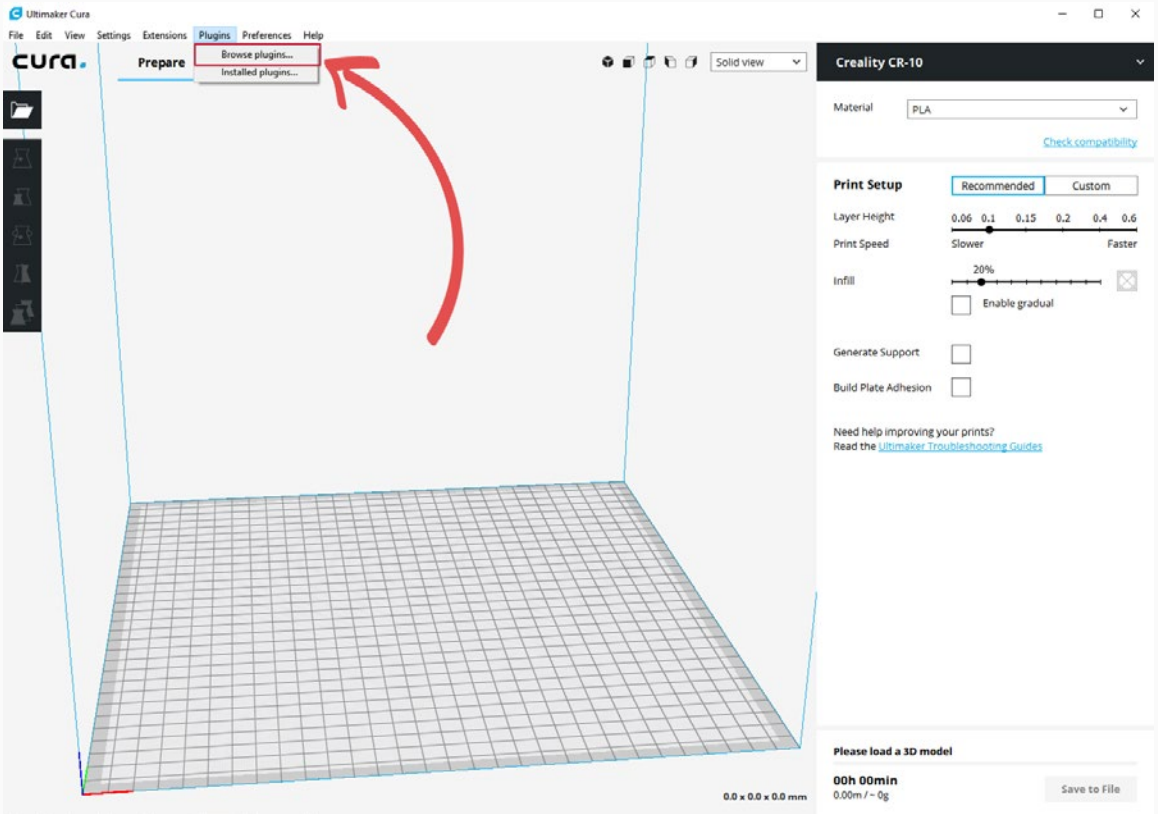
With the API Key copied down, that is everything you need to do within the web interface for now.



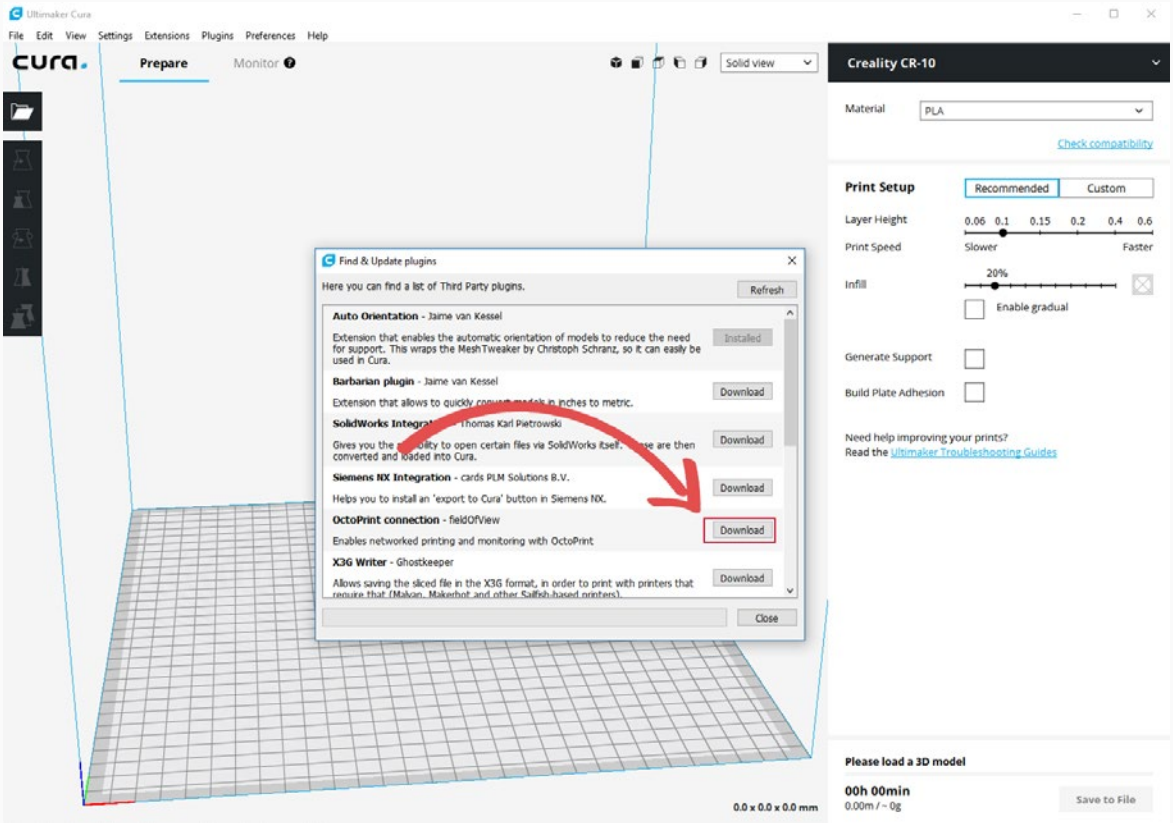
# Step 5 - Connecting Cura to OctoPrint

**1.** In the newer versions of Cura, you will have to actually enable OctoPrint support by installing a plugin. Our next few steps will walk you through the process of doing this.

To begin to launch up Cura, click "**Plugins**" in the toolbar and then select "**Browse Plugins**".



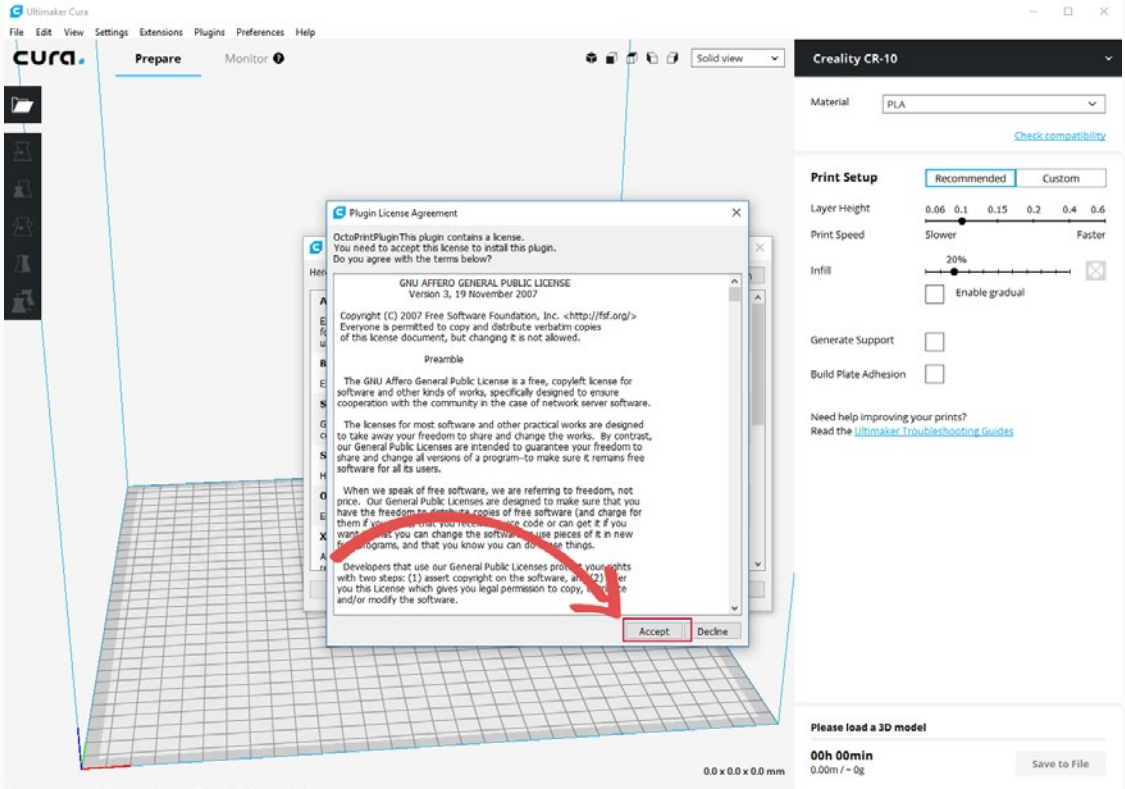

**2.** Within this screen look for "**OctoPrint Connection**" and click the "**Download**" button to install the plugin.



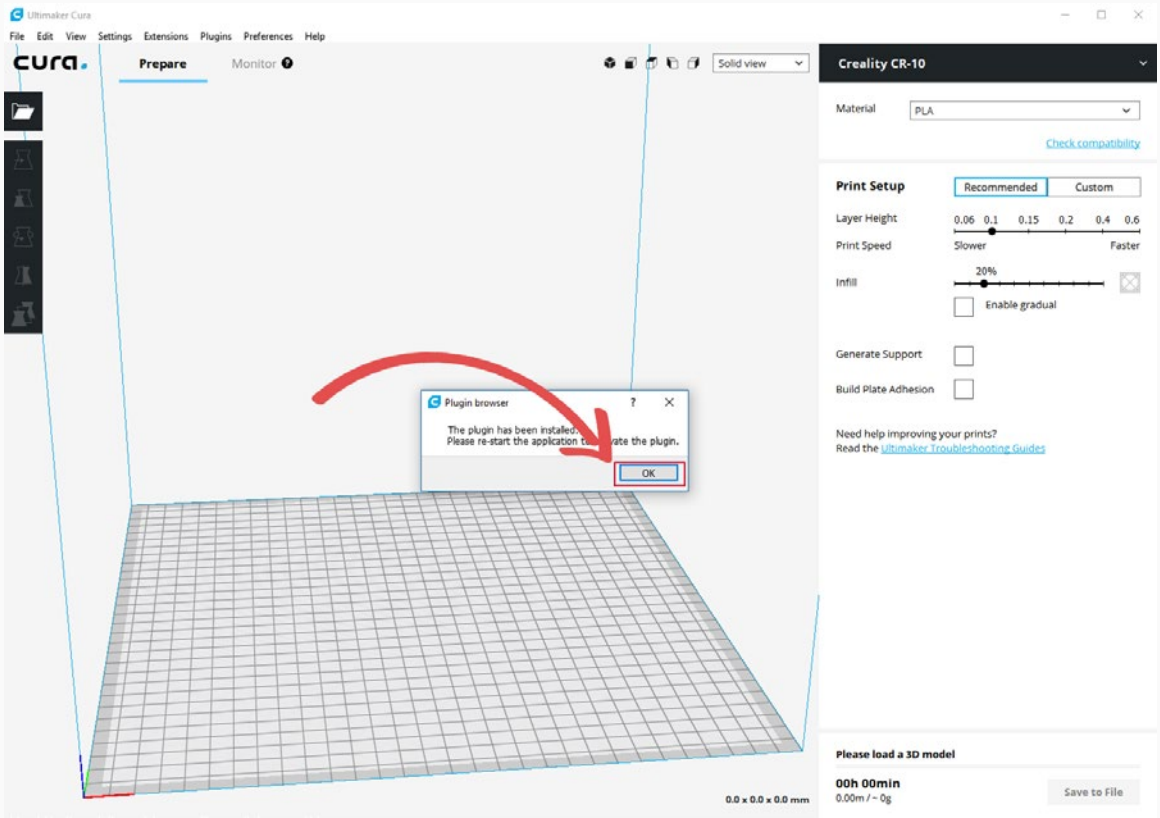
# Step 5 - Connecting CURA to OctoPrint

**3.** You will now be asked whether you accept the license agreement provided by the OctoPrint connection plugin.

If you are happy with agreeing to the license, click the "**Accept**" button.

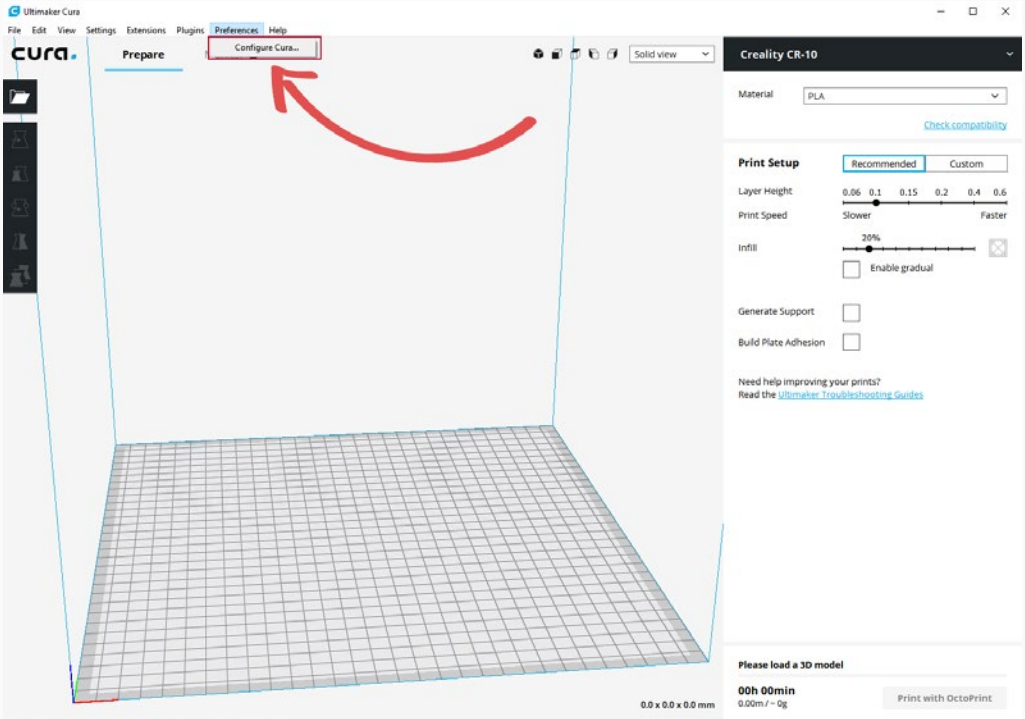



**4.** Now you will need to click the "**OK**" button. Once pressed you will need to manually close out of Cura and re-open it for the plugin to be loaded in.



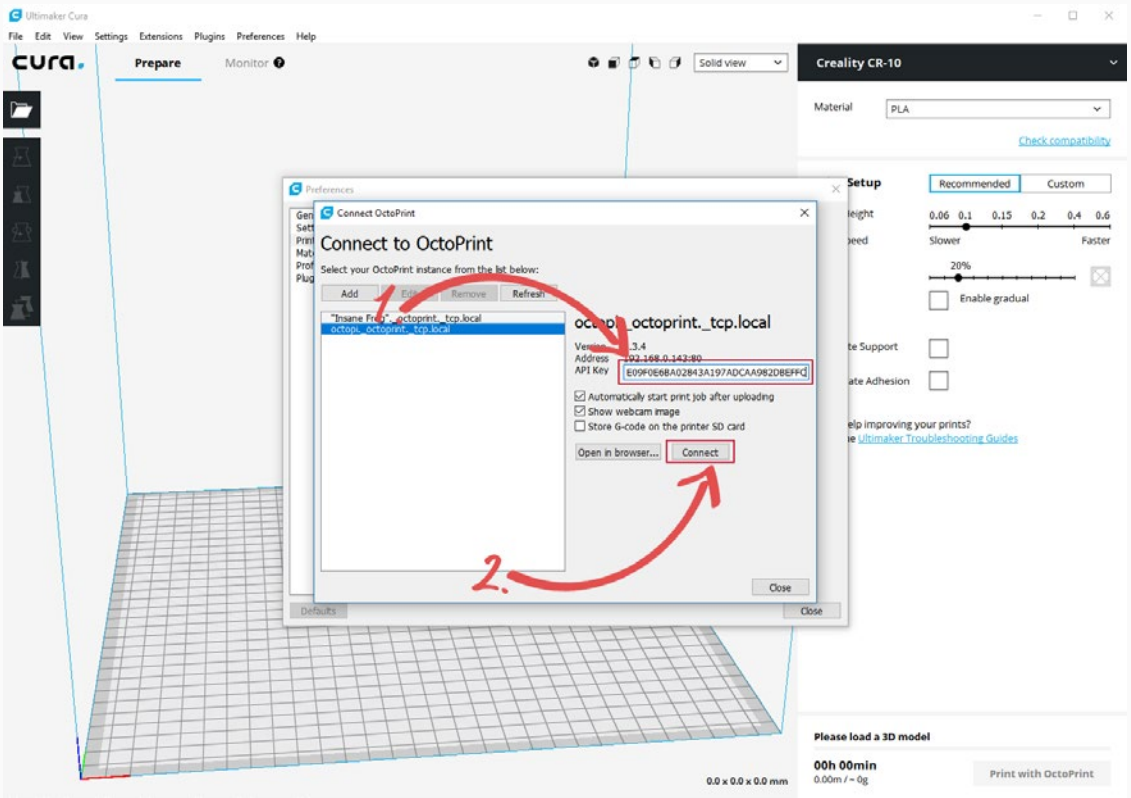
# Step 5 - Connecting CURA to OctoPrint

**5.** Now we need to set Cura up so it will use OctoPrint, to do this we need to click "**Preferences**" in the toolbar then click "**Configure Cura**".




**6.** On this screen click you OctoPrint server you want to connect to in the left-hand side then enter the API Key that you got earlier from OctoPrints web interface (**1.**)

With the correct API key entered the "**Connect**" button (**2.**) should become clickable. Once the button is clicked, you have successfully set up Cura to be able to print using your OctoPrint web server.



# File Management
## Projects

# Torrent Box

## Project Description

A Raspberry Pi TorrentBox is a great way to have a cheap affordable always on torrent machine. If you're a heavy or light torrent user, then this still works well for both!

With this project be prepared to have to deal with a bit of self-troubleshooting when running into file permission errors.

In this tutorial, we will be making use of the transmission package. I have spent a fair bit of time re-searching deluge vs.transmission and decided on transmission.

The reason for choosing Transmission is that we have used transmission before and it can handle magnet links through the web interface.  It has also been relatively stable, and we haven't had any issues once we got past the initial setup stage.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

**Optional**
- Raspberry Pi Case
- External Hard Drive or USB Drive

## Setting up the Raspberry Pi Torrent Box

Setting up the torrent client is incredibly easy and won't take much work before you have this up and running.

The thing you will need to be careful of the most in this tutorial is the folder and file permissions as they can be somewhat tricky.

**1.** First, let's bring the Raspberry Pi up to date to avoid any potential issues installing the torrent client.

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

**2.** Now, let's download and install transmission by using the following command

```
sudo apt-get install transmission-daemon
```

**3.** If you plan to store the files on an external drive, for example, a USB hard drive then we need to make directories for the torrents to stored. (**NASHDD1** is the drive I mounted)

```
sudo mkdir -p /media/NASHDD1/torrent-inprogress
sudo mkdir -p /media/NASHDD1/torrent-complete
```

**4.** Now we will need to make a few changes to the transmission settings file, enter the following command:

```
sudo nano /etc/transmission-daemon/settings.json
```

**5.** We recommended making the following changes. However, there are other settings you may want to update depending on your preference.

If you want information on all the settings, you can change then check out transmissions guide at https://github.com/transmission/transmission/wiki/Editing-Configuration-Files.

```
"incomplete-dir": "/media/NASHDD1/torrent-inprogress",
"incomplete-dir-enabled": true,
"download-dir": "/media/NASHDD1/torrent_complete",
"rpc-password": "Your_Password",
"rpc-username": "Your_Username",
"rpc-whitelist": "192.168.*.*",
```

We need to now save and quit out of the file by pressing **Ctrl+X** then pressing **Y** and then **Enter**.

**6.** Now we need to get transmission to reload so that it loads in our modified configuration file. Please note that if you restart your Raspberry Pi your changes to the config file will be lost.

You can reload the service by entering the following command:

```
sudo service transmission-daemon reload
```

**7.** If you didn't change the download directories as we did in the example configuration above, then you can skip right ahead to **step 17**.

Otherwise, if you did what we did you will need to complete the next several steps to ensure that Transmission will successfully work with these folders.

**8.** We will also need to update a few more places to make sure that transmission runs as the correct user, if we don't we'll run into a lot of errors.

First, let's stop the transmission service by entering the following command:

```
sudo service transmission-daemon stop
```

**9.** Now we need to edit the daemon, so it launches as the user who owns the directory we wish to store in (To check who owns the folder use **ls -l /your_directory/download_dir**)

First open the file by entering the following command.

```
sudo nano /etc/init.d/transmission-daemon
```

**10.** In here edit the user, so it runs as the user who owns the directory we're storing our downloads in. In my case, it is **pimylifeup** instead of **debian-transmission**.

We need do this because the folder we are going to store our torrents is likely to be owned by a different user to the default transmission.

To see who owns the directories use **ls -l** in the folder that contains the download folders.

Make sure you replace **pimylifeup** with the relevent user.

```
USER=pimylifeup
```

**11.** We also need to chown some files since we are no longer using the debian-transmission user. If you don't do this, you will start seeing errors, and transmission will fail to launch.

To take ownership of these files we just need to enter the following command in to the terminal.

Make sure you again replace **pimylifeup** with the user that you are utilizing.

```
sudo chown -R pimylifeup:pimylifeup /etc/transmission-daemon
sudo chown -R pimylifeup:pimylifeup /etc/init.d/transmission-daemon
sudo chown -R pimylifeup:pimylifeup /var/lib/transmission-daemon
```

**12.** We also need to change the user defined in the following file.

Failing to will result in an error when going to relaunch the service.

```
sudo nano /etc/systemd/system/multi-user.target.wants/transmission-daemon.service
```

**13.** In here change the user to the relevant user, in my case it is **pimylifeup**.

```
user=pimylifeup
```

**14.** Now reload the **systemctl** file by entering the following:

```
sudo systemctl daemon-reload
```

# Setting up the Raspberry Pi Torrent Box

**15.** Next, we need to create the directory where the **setting.json** file will be accessed by the transmission-daemon.

We then need to create a symbolic link (**ln**) back to the settings file we edited earlier, update **pimylifeup** with the relevant username.

```
sudo mkdir -p /home/pimylifeup/.config/transmission-daemon/
sudo ln -s /etc/transmission-daemon/settings.json /home/pimylifeup/.config/transmission-daemon/
sudo chown -R pimylifeup:pimylifeup /home/pimylifeup/.config/transmission-daemon/
```

**16.** Let's start the transmission service back up by entering the following.

```
sudo service transmission-daemon start
```

**17.** Now you can check out your torrent box by opening it up in a web browser.

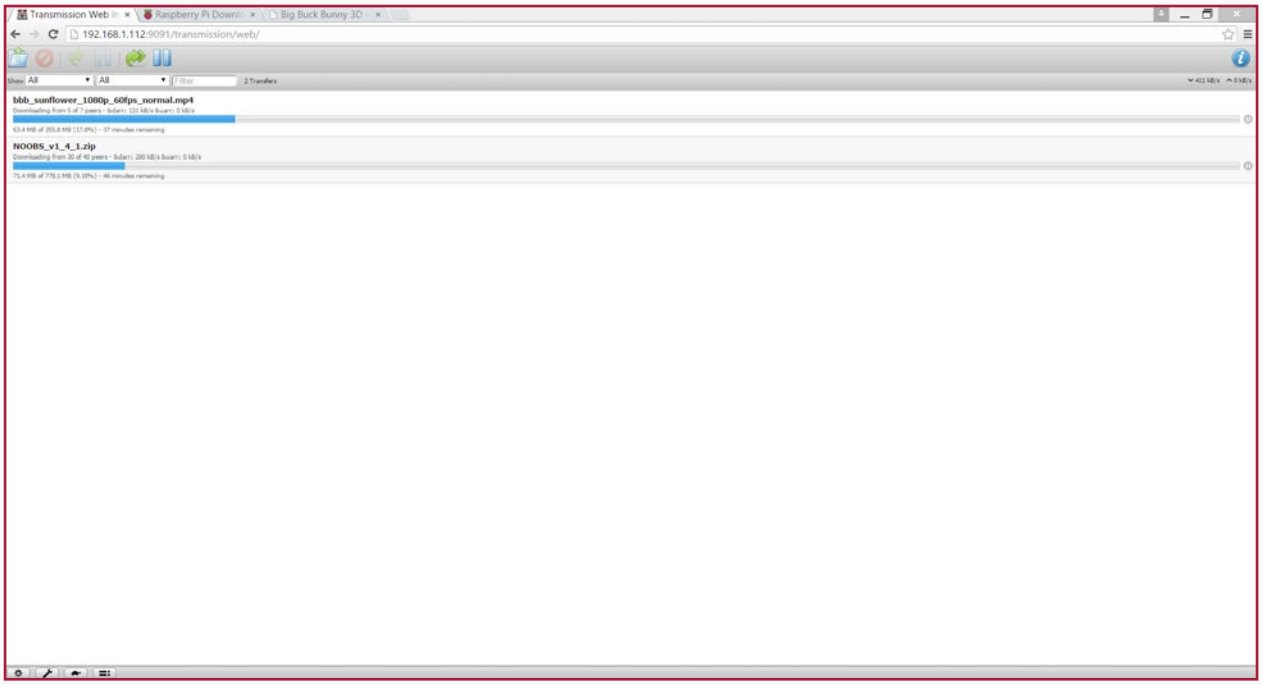
If you don't have your Raspberry Pi's IP just use the command **hostname -I**.

Enter the following URL to get to the interface. (Change the **your_pi_ip** with your Raspberry Pi's IP)

```
http://your_pi_ip:9091
```

**18.** In here you can add & remove torrents, change settings, set schedules and much more.

If you need more information on all these settings be sure to hit up transmissions GitHub website at: https://github.com/transmission/transmission/wiki/Editing-Configuration-Files.



# Setup your own
## eBook Library

## Project Description

In this project, we will be showing you how to set up your very own Raspberry Pi eBook library by utilizing the COPS (**C**alibre **O**PDS (and HTML) **P**HP **S**erver) PHP software.

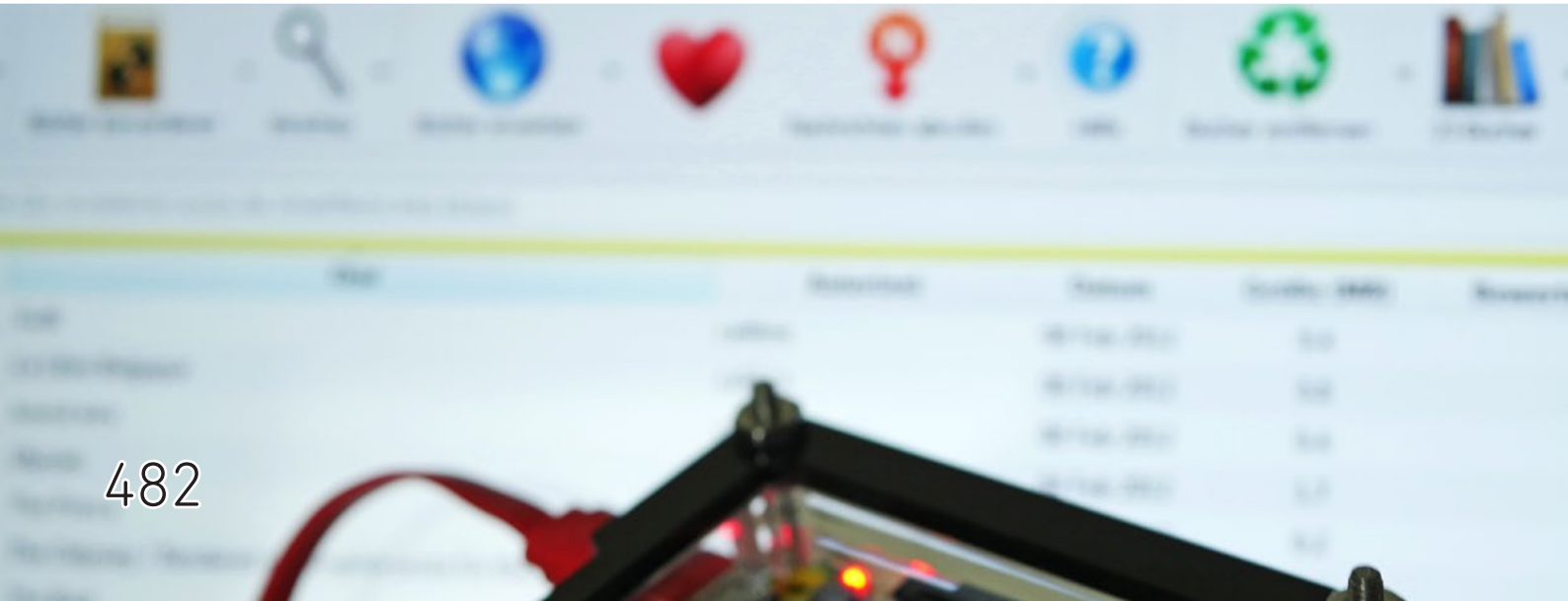
This software acts as a web interface on your Raspberry Pi where you can access all of your eBooks either through its web interface or by downloading them directly from your Raspberry Pi to your devices.

This software works as a fantastic middleman for your eBook's especially if you often share your eBooks with a wide number of devices. It's also a nice interface to sort your eBooks by author and other things.

## Equipment

**Required**
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection (Ethernet Reccommended)

**Optional**
- Raspberry Pi Case
- External Hard drive or USB Drive
- USB Keybaord
- USB Mouse



# Setting up your eBook Library with COPS

Before you begin this tutorial, please make sure that you have followed our "**Setting up NGINX**" guide and installed PHP.

We will be using that as a base for our Raspberry Pi eBook library server, due to its speed and memory efficiency.

**1.** Before we begin, let's bring all our packages on Raspbian up to date by running the following command in terminal.

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

**2.** With our Raspberry Pi up to date, we can now install the extra PHP packages that we need.

We will be assuming you have already installed PHP in this guide.

Run the following command on your Raspberry Pi to install the extra packages.

### Raspbian Stretch and later

```
sudo apt-get install php7.0-gd php7.0-sqlite3 php7.0-json php7.0-intl
sudo apt-get install php7.0-xml php7.0-mbstring php7.0-zip
```

### Raspbian Jessie and earlier

```
sudo apt-get install php5-gd php5-sqlite3 php5-json php5-intl
sudo apt-get install php5-xml php5-mbstring php5-zip
```

**3.** With our new PHP packages installed we will first have to restart NGINX for it to load in the new modules.

Just type the following command into the terminal on your Raspberry Pi to restart NGINX.

```
sudo /etc/init.d/nginx reload
```

**4.** With the PHP modules, we need now installed and NGINX restarted we can move onto setting up COPS (**C**alibre **O**PDS (and HTML) **P**HP **S**erver).

COPS is the PHP software that we will be using as our library maintainer. This software serves your eBooks to any device that connects to it.

Let's clone the software from its Github page to our Raspberry Pi by running the following commands.

The first command makes a folder in our www folder to store our eBook library. Make sure you type in the **.** after the git clone command as this makes it clone to the current directory and not create a new one.

```
sudo mkdir -p /var/www/html/ebooks
cd /var/www/html/ebooks
sudo git clone https://github.com/seblucas/cops.git .
```

## Setting up your eBook Library with COPS

**5.** Now that we have cloned the COPS software into our eBooks directory we will need to make use of composer to grab all its dependencies and set up the software for use.

To do this, we will be running the following commands on our Raspberry Pi.

```
sudo wget https://getcomposer.org/composer.phar
sudo php composer.phar global require "fxp/composer-asset-plugin:~1.1"
sudo php composer.phar install --no-dev --optimize-autoloader
```

**6.** Now before we go configuring our Raspberry Pi eBook Library, we must first create one more folder.

This folder is where you will keep your eBook library itself and its where COPS will read the files from once we configure it.

Let's create the folder by running the following command on our Raspberry Pi:

```
mkdir -p /home/pi/storage/eBooks
```

**7.** With our folder now created let's work on setting up the configuration file of COPS.

We will do this by copying the example file and immediately start to edit it by running a command for nano.

Type the following two commands in terminal to begin editing the file.

```
sudo cp config_local.php.example config_local.php
sudo nano config_local.php
```

**8.** Within this file make the following changes.

**Find**

```
$config['calibre_directory'] = './';
```

**Replace With**

```
$config['calibre_directory'] = '/home/pi/storage/eBooks/';
```

**Find**

```
$config['cops_title_default'] = "COPS";
```

**Replace With**

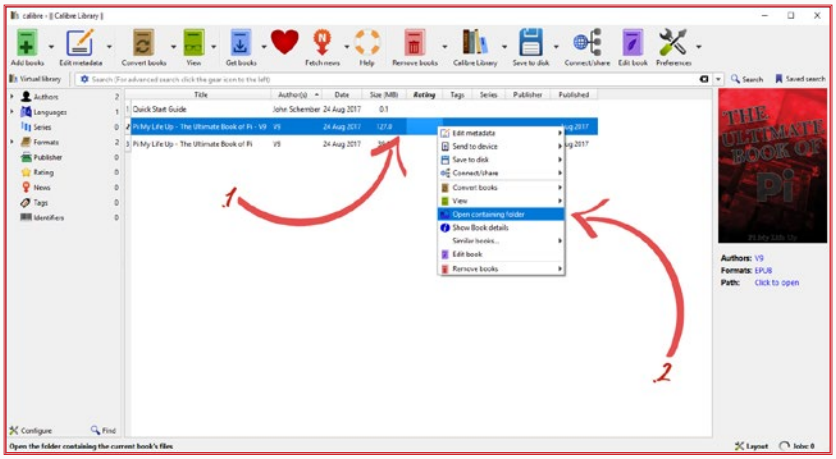
```
$config['cops_title_default'] = "My Raspberry Pi eBook Library";
```

Once you have made the changes, save the file by pressing **Ctrl + X** then **Y** then press **Enter**.

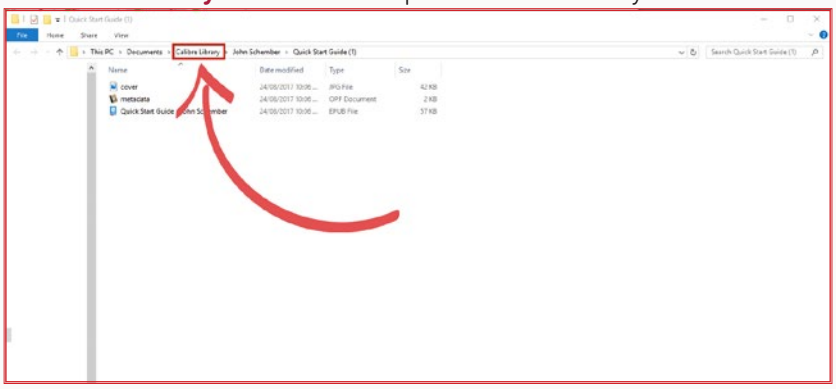
# Copying your library from Calibre to COPS

**1.** In this segment of the tutorial, we will be explaining what you must do to be able to get your Calibre library to show up through COPS. While in this tutorial we will be utilizing SFTP you can use multiple other methods of getting your eBooks to your Raspberry Pi.

Begin by opening **Calibre** on your computer, once it has opened up **right click** on any **book** (**1.**) and **click** "**Open containing folder**" (**2.**).

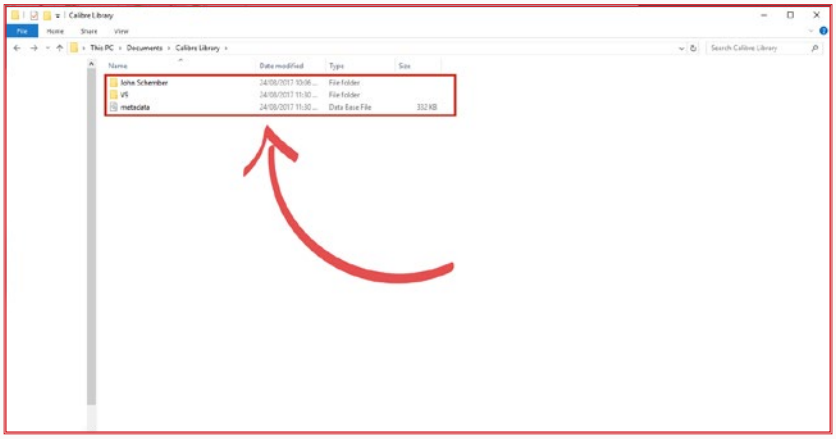



**2.** You should now see the file explorer opened up in the location of where that book is, in the address bar you should see "**Calibre Library**" click it to skip to that directory.



**3.** Now you have found the location of all your Calibre Library files, copy these files and folders to your **/home/pi/storage/eBooks/** folder.

All these files will be automatically found by COPS and displayed on the web interface. If you are unsure of how to get these files to your Raspberry Pi then continue with this tutorial.



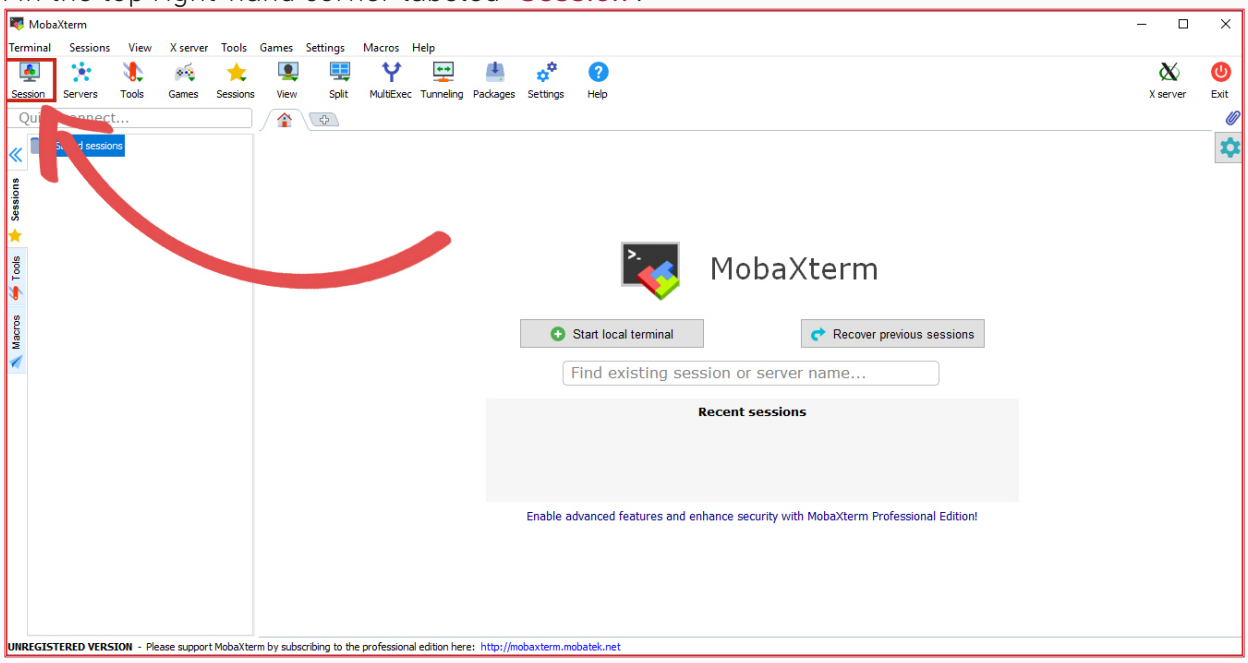
# Copying your library from Calibre to COPS

**4.** Now to get all of our Calibre files and folders to our Raspberry Pi, we will be making use of the piece of software called MobaXterm.

MobaXterm is an excellent piece of software that offers support for a wide variety of protocols, one of these being SFTP. You can grab MobaXterm from their official website: http://mobaxterm.mobatek.net/

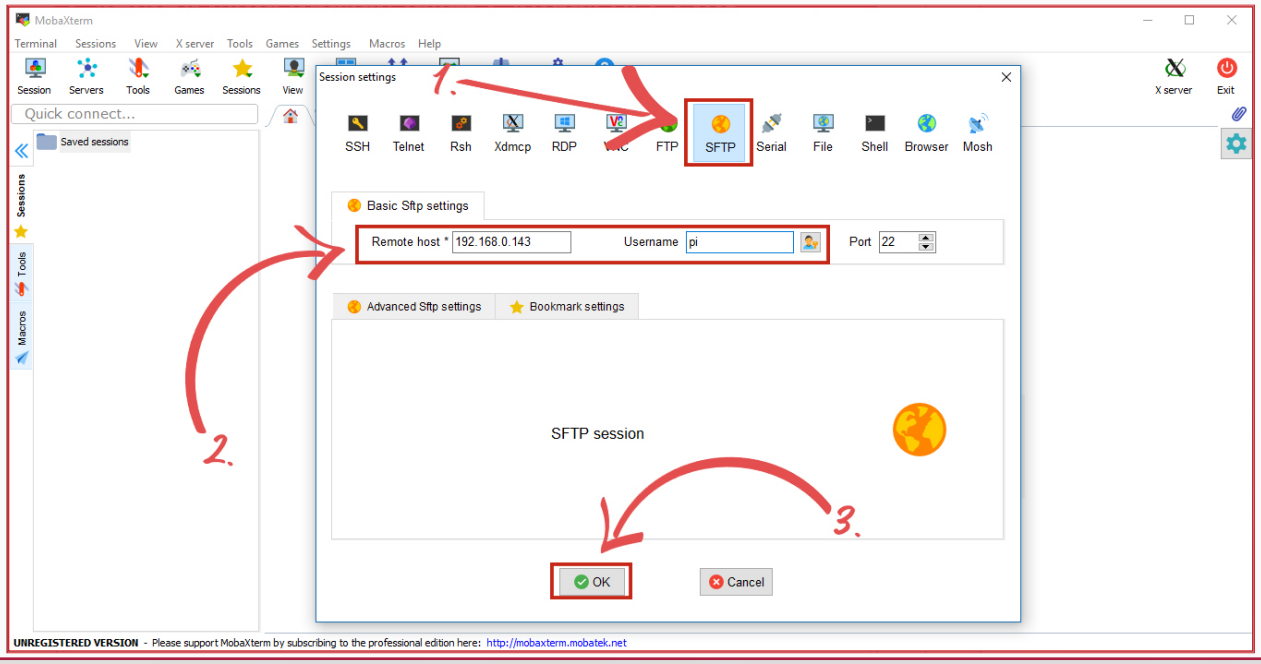
Begin by downloading, installing and running MobaXterm.

**5.** Once you run MobaXterm, you should be greeted with the following screen. Begin by clicking the button in the top right-hand corner labeled "**Session**".




**6.** On this next screen, click the **SFTP** tab (**1.**) then type in both your Raspberry Pi's local IP Address and type in your Pi user's username, **pi** is the username of the default user (**2.**)

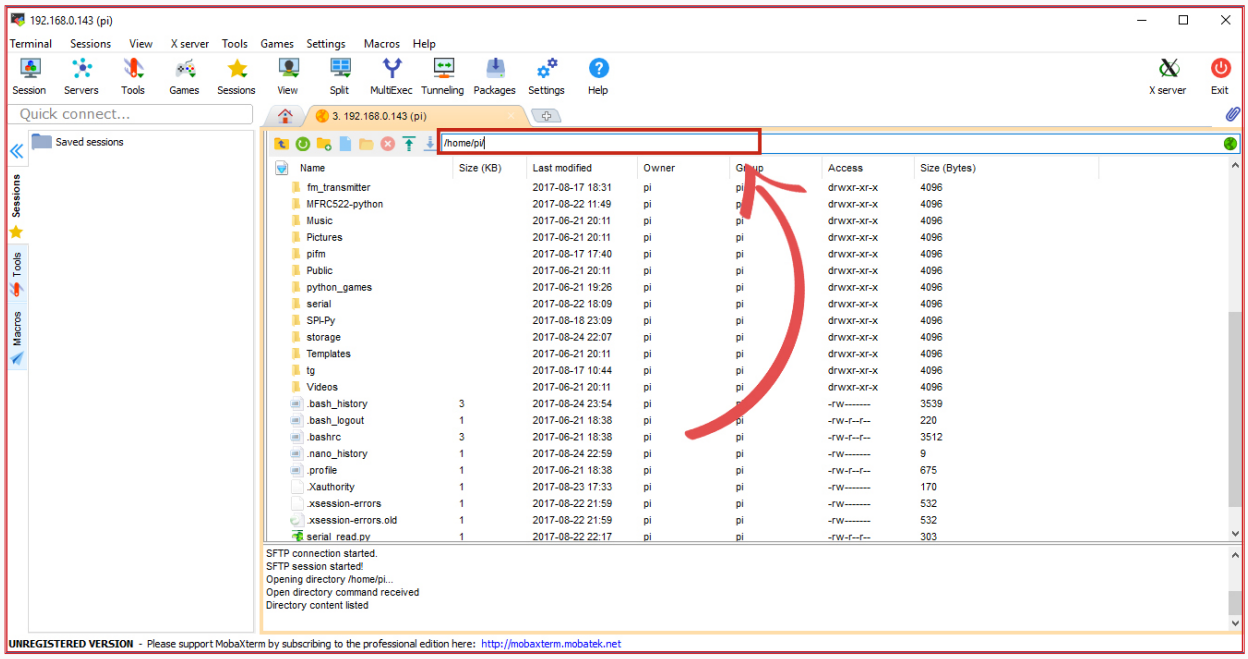
Finally, press the "**Ok**" button to initiate the connection (**3.**).



# Copying your library from Calibre to COPS

**7.** On this next screen click on the address bar as shown below, and type in **/home/pi/storage/eBooks/** and then press enter.

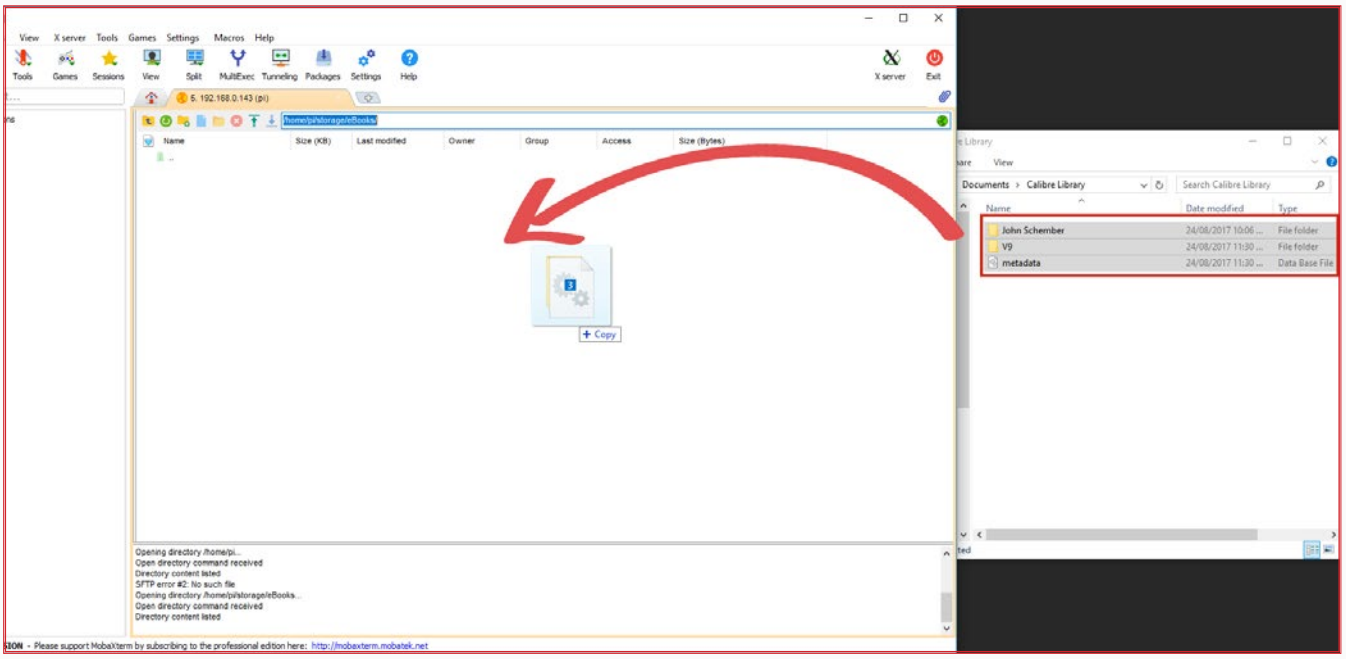
Doing this will allow us to jump directly to the folder we need.




**8.** Finally, we can copy the files from your computer to the Raspberry Pi by dragging them from the folder onto the MobaXterm SFTP window like that we have shown below.

Doing this will immediately begin the transfer process.

When it's complete, you can continue to our last step.



## Checking out COPS

**1.** To check out our Raspberry Pi eBook library, you will need to browse to your Raspberry Pi's IP Address with **/ebooks** at the end of it.

To grab your Raspberry Pi's local IP address, you can type the following command into the terminal.

```
hostname -I
```

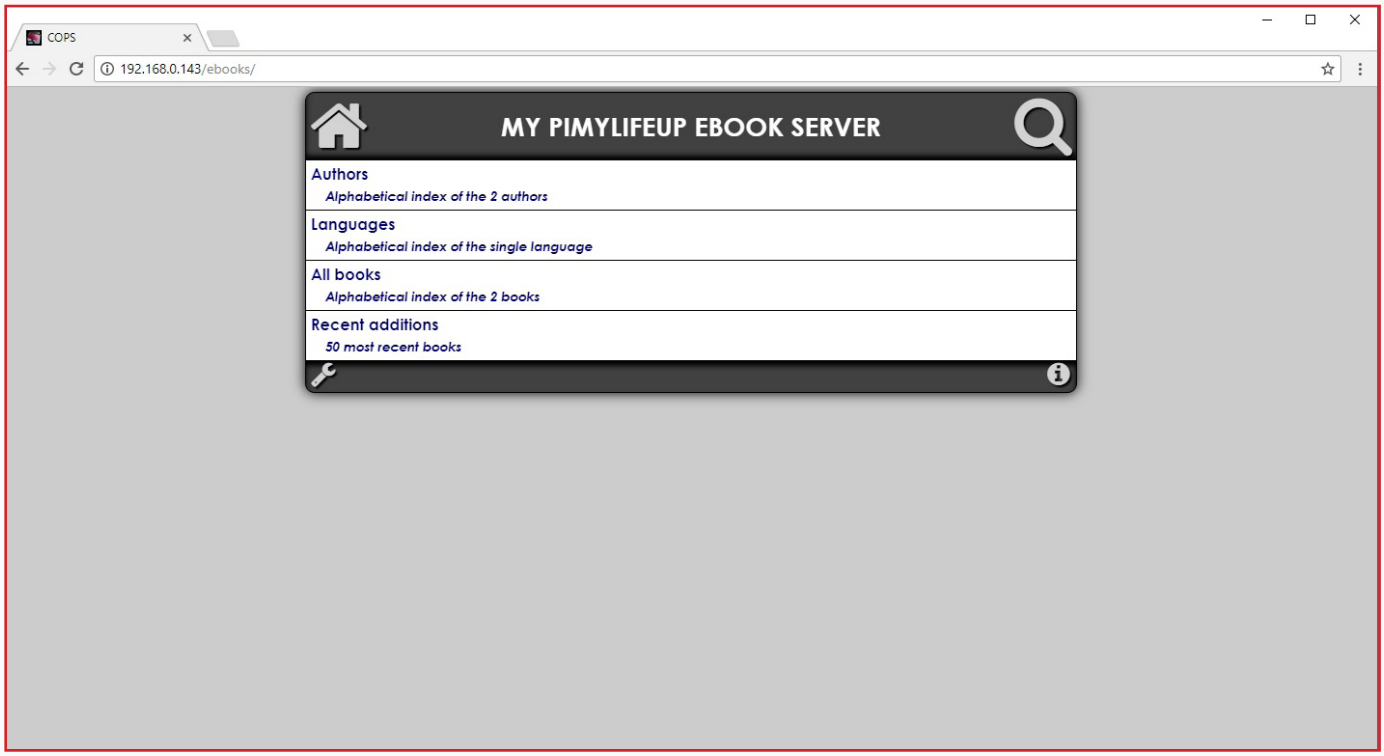
**2.** Once you have your Raspberry Pi's local IP address, then go to it in your favorite web browser with **/ebooks** tagged at the end of it.

You can also use this URL with various eBook readers so you can access all the data through there interface.

The URL you visit should look something like below, of course replacing **192.168.0.143** with your IP Address.

```
http://192.168.0.143/ebooks
```

You should be greeted with a webpage like below when you browse to the URL.



PWR IN
ACT PWR
DISPLAY
CAMERA
HDMI
FCC ID: 2ABCB-RPI32
IC: 20953-RPI32
Made in PRC
GPIO
Raspberry Pi 3 Model B V1.2
© Raspberry Pi 2015

# References

# Linux Cheat Sheet

## File System

| | |
|---|---|
| **ls** | Directory Listing |
| **tree** | Indented file/directory listing |
| **cd** | Change directory |
| **pwd** | Display current directory |
| **mkdir** newDir | Creates a directory |
| **rmdir** oldDir | Removes a directory |
| **cp** file newfilename | Copies a file to a new location |
| **mv** file.txt ./newDir | Moves a file to a new location |
| **touch** file | Sets the last modified timestamp |
| cat | List contents of a file |
| **head** file | Outputs first 10 lines of a file |
| **tail** file | Outputs last 10 lines of a file |
| **chmod 777 file** | CHMOD is used to alter the permissions of a file or files. |
| **chmod u=rw file** | You can use symbols or numbers depending on what you prefer. |
| **chown** pi:root file | CHOWN will change the user and/or the group that owns a file. |
| **unzip** archive.zip | Unzip will extract the files and directories from a compressed zip archive. |
| **tar** -cvzf<br>**tar** -xvzf | Compress and extract the contents of an archive in the tar format.<br>-c to compress & -x to extract |

## Users Management

| | |
|---|---|
| **id** | Get the id of a user or group |
| **who** | List users that are logged in |
| **last** | List of users recently logged in |
| **groupadd** name | Adds a new group |
| **useradd** name | Adds a new user |
| **userdel** name | Deletes a user & all files related to that user |
| **deluser** name | Deletes user with options to remove certain data |
| **usermod** | Modify a user account |
| **passwd** | Change your password or a password of another account |

## Process Management

| | |
|---|---|
| **ps** | Show snapshot of current processes |
| **top** | Show real time processes |
| **kill** pid | Kill process with id pid |
| **pkill** name | Kill process with name |
| **killall** name | Kill processes with name |

## Networking

| | |
|---|---|
| **ping** host | Pings a host |
| **hostname** | Hostname of the system |
| **ifconfig** | Shows network configuration details for the interfaces on current system |
| **ssh** | Connect to another computer using SSH |
| **scp** | Transfer files over SSH |
| **wget** {URL} | Downloads a file directly to the device |
| **netstat -tupl** | Active connections to/from device |

## Shortcuts

| | |
|---|---|
| !! | Repeats last command |
| ctrl+z | Stops command, resume using **fg** for foreground or **bg** for background |
| ctrl+c | Halts the current command |
| ctrl+w | Deletes 1 word on the current line |
| ctrl+u | Deletes entire line |
| ctrl+r | Search previous commands |
| exit | log out of current session |

## General Commands

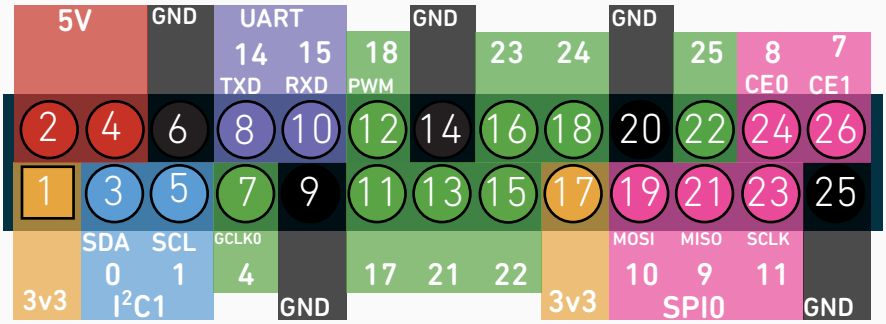
| | |
|---|---|
| **Man** ls | Brings up the manual page for ls |
| ls **\|** head | Pipes \| allows you to direct output from one command into another |
| **date** | Shows system date |
| **whoami** | Shows your username |
| **uptime** | Shows uptime |
| **uname -a** | Show system & kernel |

## Searching

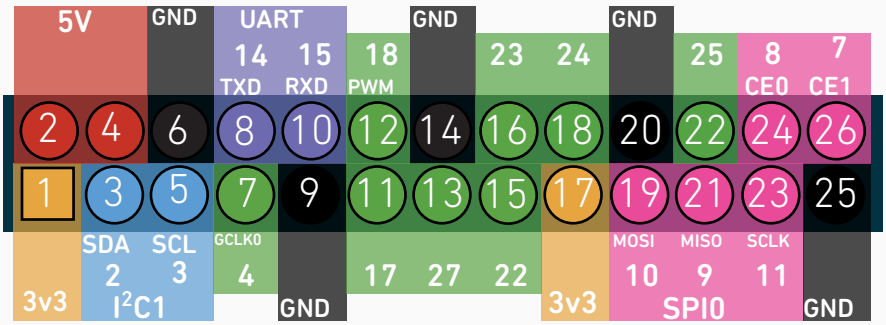
| | |
|---|---|
| **grep** "search" *.txt | Search for a pattern inside files |
| **find** . -name 'help' | Will search for directories & files that match a pattern |
| **whereis** ls | Displays documentation, binaries & source files of a command |

# Raspberry Pi GPIO Pins

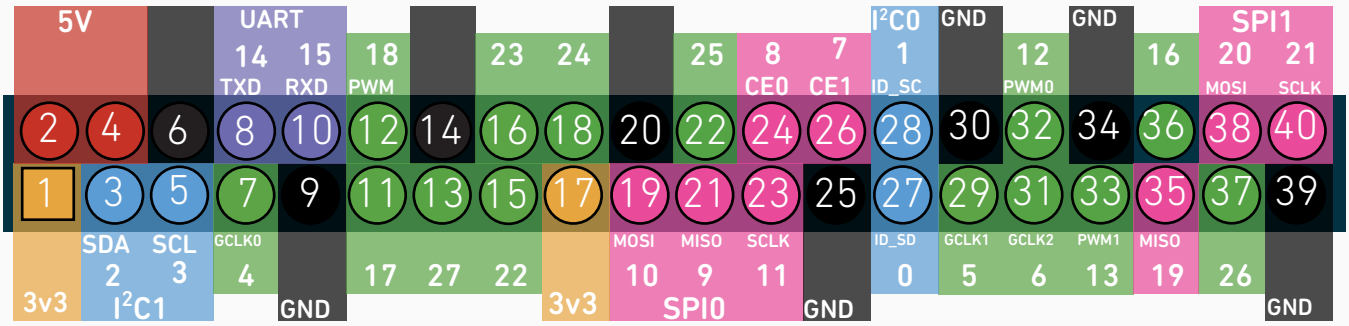
## Raspberry Pi (Revision 1)




## Raspberry Pi (Revision 2)



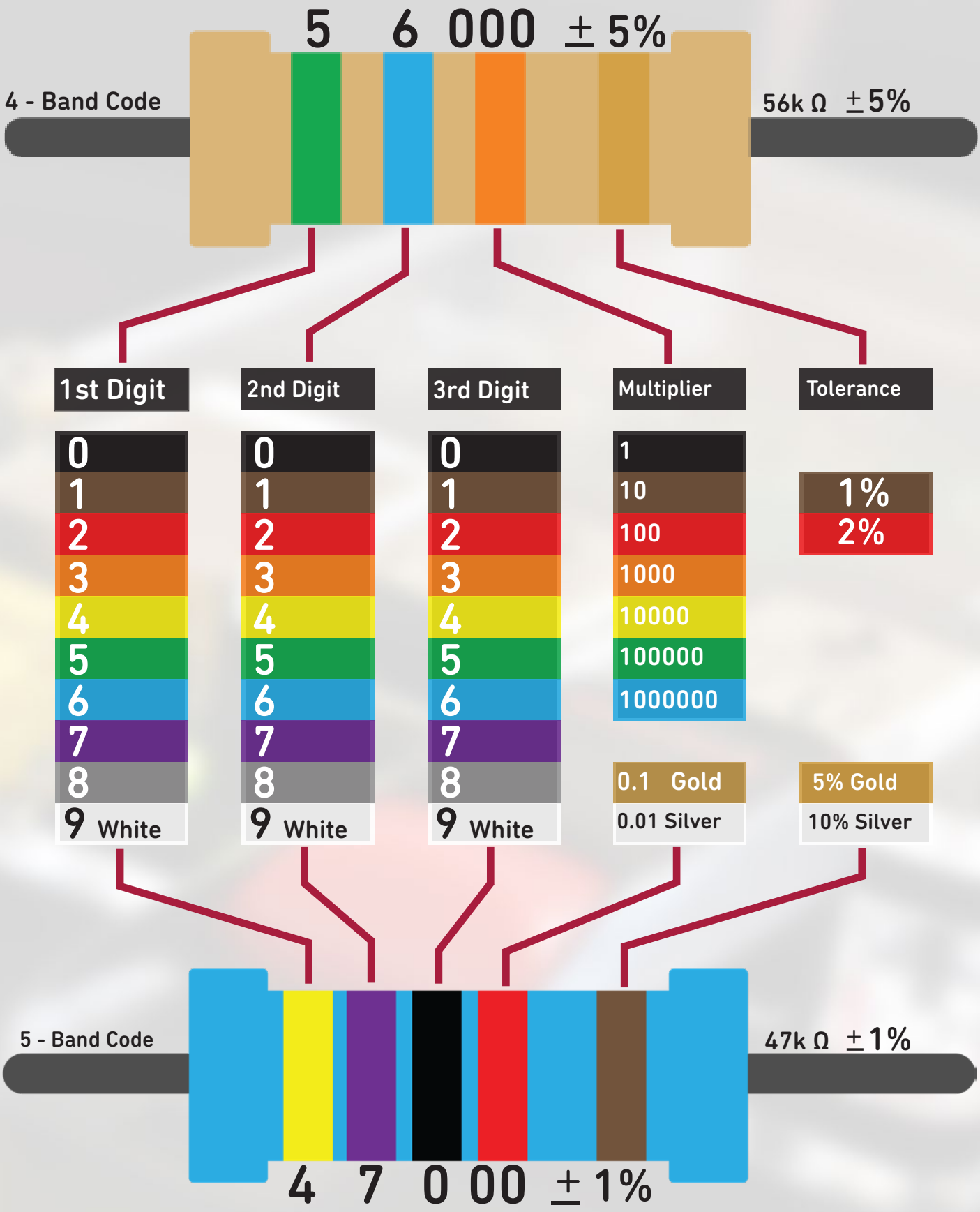

## Raspberry Pi B+, 2, 3 & Zero




## Key

- GPIO Pins
- Ground
- Serial Peripheral Interface (SPI)
- Universal asynchronous receiver/transmitter (UART)
- Inter-Intergrated Circuit (I²C)
- 5v Power Pins
- 3v3 Power Pins

# Resistor Colour Guide



# Voltage Divider Equation

**Calculating the Voltage Out**

$$V_{out} = V_{in} \times \frac{R_2}{R_1 + R_2}$$

**Calculating the R2 Value**

$$R_2 = \frac{V_{out} * R_1}{V_{in} - V_{out}}$$