RASPBERRY PI SETUP GUIDE

BEFORE YOU START

Download and install *Etcher* from etcher.io. *Etcher* is among the easiest options for writing a Pi OS image to a microSD.

WINDOWS USERS

Download and install *Bonjour Print Services for Windows* (current version 2.0.2) from Apple. Microsoft is only beginning to embrace standard ***Zeroconf*** protocols, such as ***multicast DNS (mDNS)***. Bonjour services fill in the gaps and streamlines the process of connecting to your Raspberry Pi.

Install an SSH client. You'll already have SSH if you have installed Git for Windows with the *Git BASH* environment.

Install a text editor that supports Unix-style line endings, e.g., *Atom* or *Notepad++*.

LINUX USERS

Using your default package manager, install *Avahai* MDNS services.

INSTALL RASPBIAN

The Raspberry Pi is built with Linux distributions in mind. The official distribution, which we'll use in our labs is known as *Raspbian* and is based on *Debian*. If you're familiar at all with *Ubuntu*, you should be mostly at home working in *Raspbian*. Don't worry if Linux is not your jam. We'll provide plenty of guidance so that you can focus your energy on the network concepts.

INSTALL RASPBIAN

Download a current image of *Raspbian* from http://www.raspberrypi.org/downloads. If access to a graphical Linux desktop is not important to you, I'd recommend *Raspbian Jessie Lite*. Otherwise, download the larger image including the *PIXEL desktop environment*.

Use *Etcher* to write the image you've downloaded to a microSD. Depending on how you downloaded *Raspbian*, you may need to unzip the image file. The correct file extension for the disk image is `.img`.

Be aware that this process will overwrite any data you currently have stored on the card.

UPDATE CONFIGURATION

Etcher will eject the microSD when the image is completely rewritten. Since we want to edit some files on the SD, you will need to briefly remove the card before inserting it again.

On macOS or Windows, you'll be limited to accessing the boot partition of the card. Use Explorer or Finder to locate and open the partition.

ENABLE SSH

Due to security considerations, the newest versions of *Raspian* disable SSH by default, but it's easy to turn the feature on so that we can use it for initial setup.

To enable SSH on the first boot, add an empty file named `ssh` to `/boot`. An easy way to create this file from the terminal (Git BASH on Windows) is to navigate to the boot volume (may show up as a drive letter on Windows) and to run `touch ssh`. On the Mac, this looks like:

```
cd /Volumes/boot
touch ssh
```

Raspbian will check for this file during the first startup and proceed to configure the ***SSH daemon*** start automatically. The term ***daemon***, by the way, is the name Unix operating systems use to describe a network service.

CONFIGURE USB ETHERNET FOR PI ZERO

The Pi Zero does not have a physical Ethernet port, but it can be configured to operate in Ethernet Device mode over USB. Since we're building our networking skillset, we'll rely on this feature along with a variety of remote console/desktop protocols to access the Pi.

Open `/boot/config.txt` in a text editor that supports Unix-style line endings. Navigate to the end of the file and add the following statement on its own line:

```
dtoverlay=dwc2
```

Now open `/boot/cmdline.txt` and look for the keyword `rootwait`. Immediately after `rootwait`, insert the following statement:

```
modules-load=dwc2,g_ether
```

The boot process on the Pi is sensitive to formatting. Confirm that you entered the preceding statements exactly as shown without any additional whitespace surrounding the equal sign or comma. The modules-load statement should be separated from surrounding commands with a single leading and trailing space.

INITIAL BOOT

It's time to boot the Pi for the first time. Close your editor and any windows that are open to the microSD so that you can eject the card gracefully from your OS. Remove the card insert into the card slot on your Raspberry Pi.

Connect power to the designated micro-USB port on the Pi. Connect Ethernet (Pi 3) or USB (Pi Zero) and get ready to launch an SSH connection.

From terminal (*Git BASH* on Windows), connect to the Pi for the first time by running the following command:

```
ssh pi@raspberrypi.local
```

SSH will ask you to accept the connection of an unknown device before presenting you with a password prompt. The default password for the `pi` user is `raspberry`.

Before proceeding further with setup, you should change the default password and configure a unique hostname. Both options are available in *Raspbian*'s menu-based configuration utility.

To access this utility, run `sudo raspi-config` from SSH. Review basic and advanced options and make changes as needed.

CONNECT TO WIFI

To complete this guide, you will need to establish Internet connectivity for your Pi. Since the Pi 3 and Pi Zero W, we can solve the problem by connecting to local wifi. This approach is not the only solution.

We could modify the settings for the wired connection to obtain a local address from DHCP and attach the Pi directly to an open switch port on our LAN. We'll still be able to SSH using the hostname with the `.local` suffix. The same physical connection will also provide access to Internet routable addresses.

Another option is to configure your workstation to share its Internet connection with the link we've established to the Pi. This process can be a bit confusing, as it requires us to enable Internet connection sharing and to set up static IP addresses on each end of the physical link.

Our wifi solution blends the advantages of the first option with the freedom of wireless. Likewise, it will pave the way for future projects.

CONFIGURE DHCP FOR WLAN0

When we connect to our wireless network, we want to ensure that it obtains an IP address automatically through ***DHCP***. We can enable DHCP for the wireless interface by modifying the default settings for `wlan0` stored in `/etc/network/interfaces`. If you're new to working in the Linux terminal, you can edit this file using *nano*.

CONFIGURE WPA_SUPPLICANT

Wireless settings for the Pi are controlled by a service called *wpa_supplicant*, which stores network connection settings inside `/etc/wpa_supplicant.conf`.

A minimal network entry contains the ***ESSID***, i.e., the name of the network, and the passphrase (if configured) as in the following example:

```
network={
  ssid="My Fancy Wifi"
  psk="super secret squirrels"
}
```

If you don't feel comfortable storing the password in plaintext, you can use a utility called `wpa_passphrase` to generate the raw encryption key for the network as follows:

```
wpa_passphrase "My Fancy Wifi" "super secret squirrels"
```

Some networks don't have encryption enabled. When configuring such a network, you will replace the `psk` variable with `key_mgmt=NONE`.

Once you've determined the appropriate setup for the file, add it to the end of the existing `wpa_supplicant.conf` using *nano*. The network connection should be established automatically once you save the file. You can check by running `ifconfig wlan0` and checking that a proper IP address has been configured. It is sometimes necessary to reset the underlying process by calling `sudo wpa_cli reconfigure`.

UPDATE SOFTWARE PACKAGES

Let's finalize the initial setup by checking for updates to *Raspbian* and its default packages (this can take awhile).

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

ENABLE REMOTE EDITING

Editing configuration files is something we need to do quite often. We can always work with files using ***nano*** or other CLI-based editors, but these tools do have a steep learning curve. You may not be aware that some of the most popular text editors support network-based editing of remote files through SSH. While this configuration can be mildly confusing, the payoff is typically worth the effort.

Remote editing will rely on a Pi-based ***client*** that connects to a ***server*** running inside your local text editor. The client and server communicate through an application protocol called *rmate* that piggybacks on the SSH connection between two endpoints.

We'll approach setup in three parts. First, we'll configure a local text-editor to run an *rmate* server. Once the server is running, we will establish an SSH-tunnel between workstation and Pi. Finally, we'll install the client script on the Pi and use it to load a file. Pay close attention to this process. While the installation of the server and client is a one-time process, you must launch the server and initiate the SSH tunnel each time you want to edit remote files.

INSTALL AND LAUNCH SERVER

To get started, open a new window in an editor that supports the rmate protocol. Rmate is natively supported in *TextMate*, but can be added through third-party plugins for a variety of popular editors, including: Atom, Sublime Text, and Visual Studio Code. I'll use Atom, a cross-platform editor from Github that can be downloaded from Atom.io.

Plugins in Atom are managed from preferences. Launch Atom, open preferences and look for remote-atom in the plugin repository. Once the plugin is installed, you'll be able to configure further preferences to change the port on which the server listens or to launch automatically with Atom.

You can manually start the server from the Packages Menu, by extending sub-menus for Remote Atom and selecting Start Server. By default, the server will listen to ***port 52698*** on ***the loopback network*** of your workstation.

LAUNCH REVERSE SSH

The loopback network is not accessible from external hosts, including your Raspberry Pi. As mentioned before, we use SSH to build a path from the Pi back to the server. The SSH process will redirect new *rmate* sessions to the listening server on our loopback network. Close your existing SSH session by running the exit command from within the session, and use the following command to launch a ***reverse SSH connection*** that forwards *rmate* packets to the server on the ***loopback***:

```
ssh 52698:127.0.0.1:52698 pi@raspberrypi.local
```

INSTALL CLIENT

Finally, we need to install an *rmate* client on the Pi. The following commands will download the the *rmate* script from Github to `/usr/local/bin` and update its file permissions to allow it to execute:

```
wget -O /usr/local/bin/rmate
  https://raw.githubusercontent.com/aurora/rmate/master/rmate

sudo chmod a+x /usr/local/bin/rmate
```

START EDITING

You can now edit a file by calling `rmate` from within the SSH session you've established to the Pi and passing the name of a file as an argument. If everything works as intended, Atom will open a new tab to edit your file. As long your network connection remains healthy, changes you save from Atom will be pushed directly to storage on the Pi.

As always, you should save changes regularly. You may also consider creating a backup of important configuration files before making changes. Finally, don't forget to use `sudo` if the file you are editing requires elevated privileges.

The following example demonstrates these concepts and launches a remote editing session to modify network interfaces configuration:

```
sudo cp /etc/network/interfaces ~/interfaces.bak
sudo rmate /etc/network/interfaces
```

Be sure to save and close the file from Atom when you're finished working.

GRACEFUL SHUTDOWN

Don't just yank the power from your Pi when it's time to quit. You should always issue a proper shutdown via SSH before to prevent data loss or corruption of the microSD.

From an SSH connection, run `sudo shutdown -h now`. If you ever need to reboot the Pi, you can use the same command in conjunction with the `-r` option.

After halting the Pi, wait about 20 - 30 seconds to allow the Pi to complete the process before disconnecting the power.

TROUBLESHOOTING

Make sure that you've installed an mDNS client as described early in this document. Though mDNS is now a draft standard, Microsoft has not included a full implementation yet as of Windows 10.1. If you are missing mDNS, you are likely to receive an error the first time you connect to the Raspberry Pi.

If you are sure that mDNS is installed correctly, you will need to check the configuration of the Ethernet port or USB device on your local computer. Run ipconfig on Windows or ifconfig on other devices to confirm that the port registers a network connection. The local IPv4 address for this port should begin with 169.254.x.x.

If you are connecting the Raspberry Pi Zero via USB or using a USB Ethernet port for the first time on your computer, it's possible that your first attempt to connect will be unsuccessful. Many of these issues are resolved by disconnecting the USB briefly and trying after you have plugged everything back in. If issues persist, search online to determine whether your computer is missing a driver needed for the Ethernet adapter.

At some point, you are likely to receive an error or warning related to SSH. In most cases, what you are seeing is expected behavior. The first time you connect to a host, SSH will bind a cryptographic signature to the hostname, e.g., `raspberrypi.local`. SSH will let us know if another host provides the same name with a different signature. Most of the time, SSH will tell us how to resolve the problem. Likely, you will need to edit the `known_hosts` file and remove the specified line number.