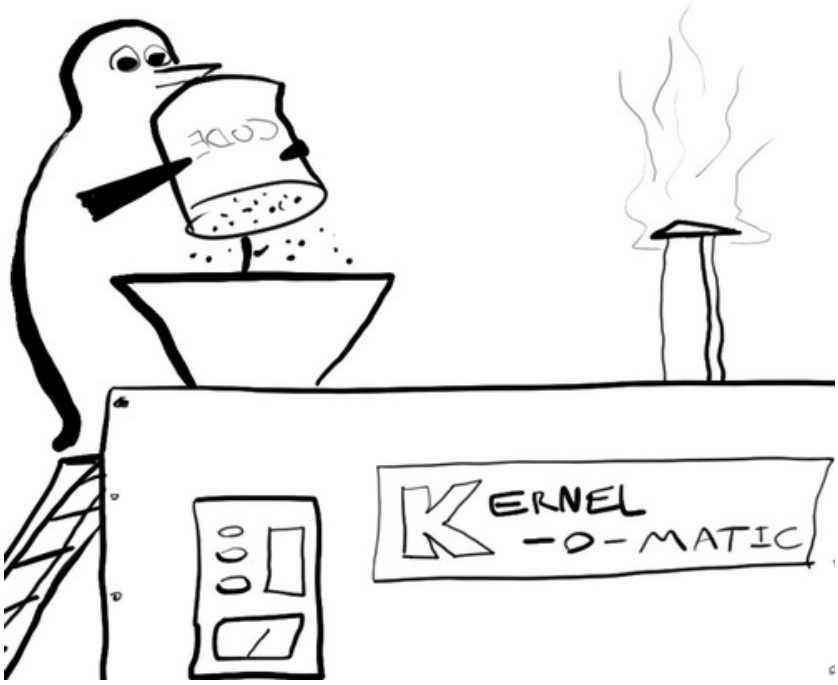
adafruit learning system

Raspberry Pi Kernel-o-Matic

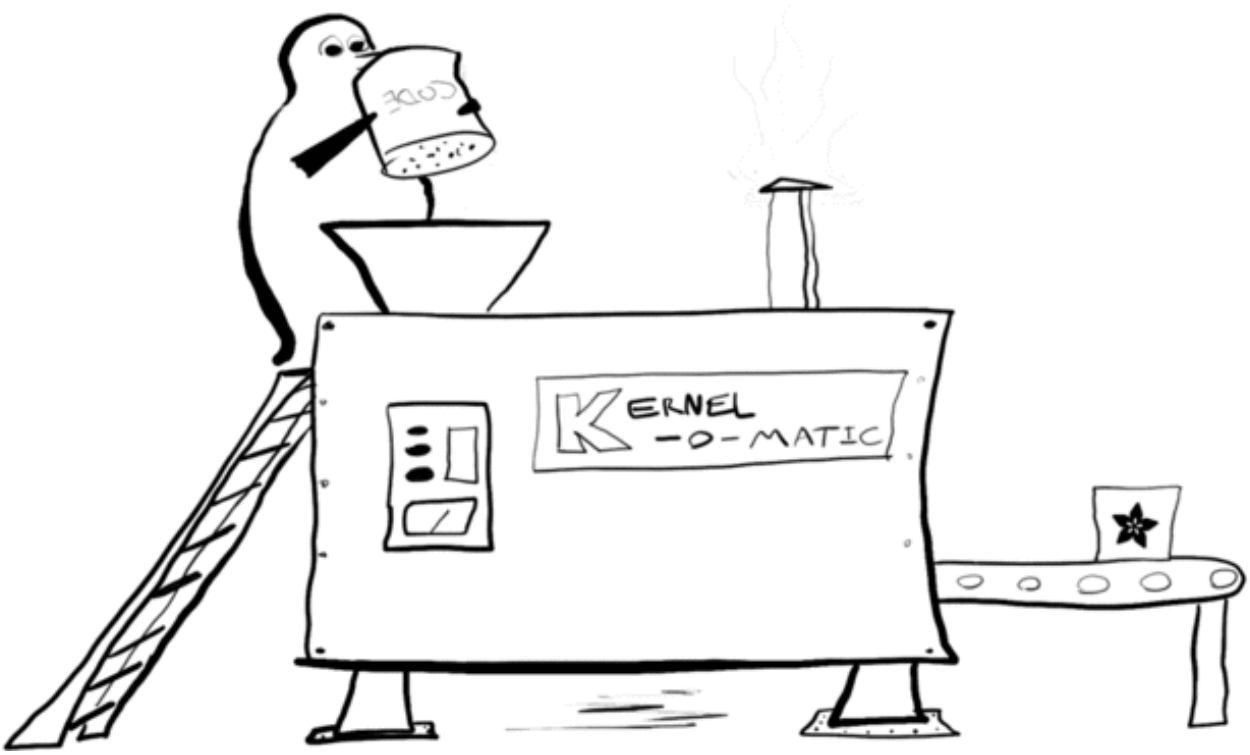
Created by Todd Treece



Last updated on 2015-05-08 08:00:14 AM EDT

Guide Contents

Overview



If you're ever needed to compile the Linux Kernel on a Raspberry Pi, you've probably noticed that it takes a long time. We sure have.

If you have a desktop computer or a laptop with decent hardware specs, it seems like there ought to be an easy way to use all that processing power to generate a new kernel for your Pi, but it can be tricky to figure out the specifics. Enter the Adafruit Pi Kernel-o-Matic (http://adafru.it/epp), which uses **Vagrant** to run a virtual machine pre-configured for compiling kernels and produces a package suitable for installation on a Raspbian machine.

Vagrant is for "creating and configuring virtual development environments". What does that mean? Well, really, it's a simple way to set up a virtual machine (VM) running an operating system of your choice. It uses **VirtualBox** to run VMs, and works on OS X, Windows, and Linux.

Getting a new kernel can be (almost!) as simple as:

```
cd Adafruit-Pi-Kernel-o-Matic
vagrant up
vagrant ssh
sudo adabuild
```

Interested? You'll just need to install a bit of software. Read on for specifics.

Installing Dependencies

You'll need a couple of things installed on your workstation to get started. The Kernel-o-Matic runs inside of a Vagrant configured virtual machine running Ubuntu. You will need to have the **latest versions of Vagrant and VirtualBox** installed to use the Kernel-o-Matic.

Once installed, Vagrant will handle downloading and configuring the Ubuntu VM for compiling the Raspberry Pi Kernel. Use the links below to download and install the appropriate versions of VirtualBox and Vagrant for your operating system.

VirtualBox

http://adafru.it/cBK

Vagrant

http://adafru.it/epq

After you have installed Vagrant and VirtualBox, you can grab a copy of the latest version of Adafruit Kernel-o-Matic, and unzip the archive into a directory of your choosing.

Kernel-o-Matic

http://adafru.it/epr

If you are comfortable with *git*, you can also clone a copy of the Kernel-o-Matic repo.

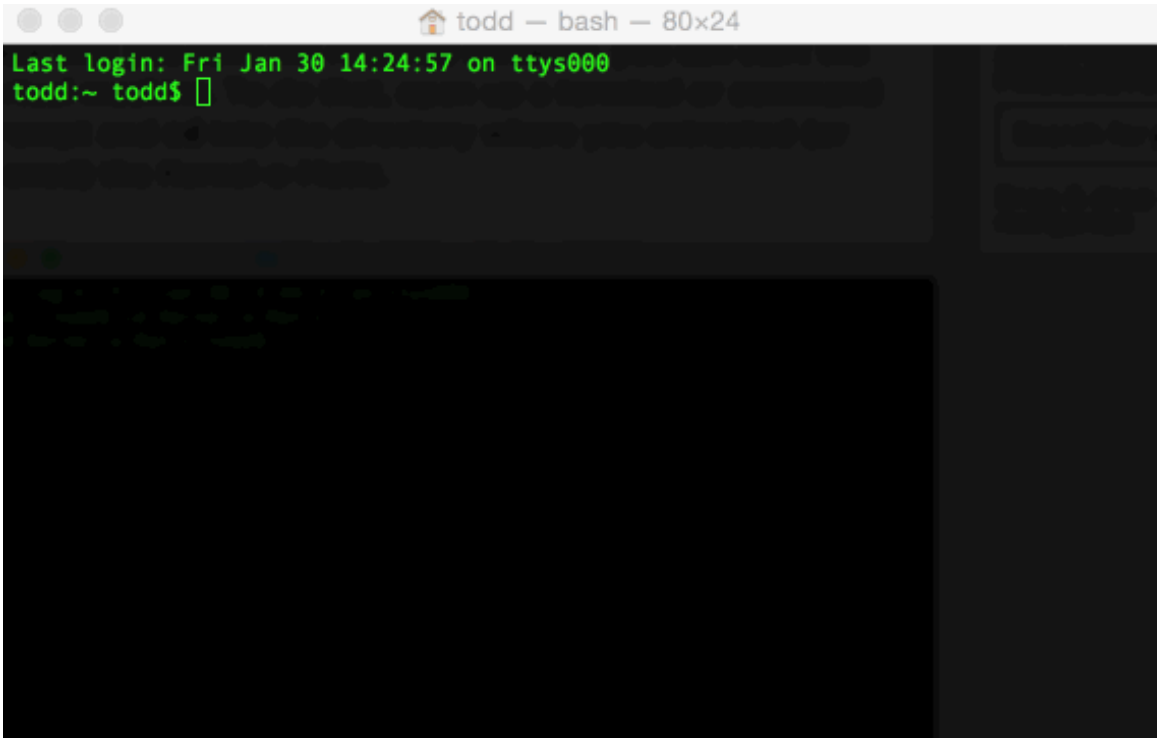
```
git clone https://github.com/adafruit/Adafruit-Pi-Kernel-o-Matic
```

Next, we'll look at how to launch and connect to the virtual machine.

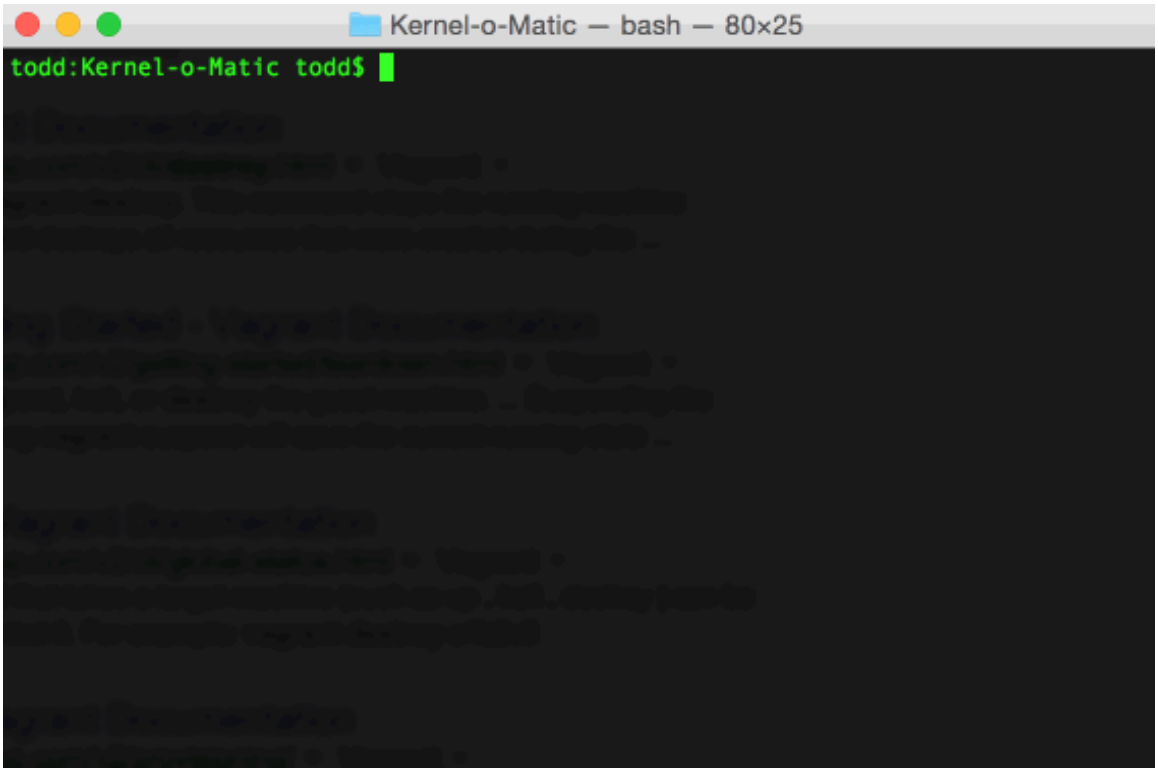
Starting the VM

Now that you have the required software, you can start the virtual machine (VM). To do that, open up a terminal or command prompt and **cd** into the directory where you extracted (or cloned) the Kernel-o-Matic.

> If you previously had Vagrant & VirtualBox installed on your machine, please make sure you are running the latest versions before continuing.

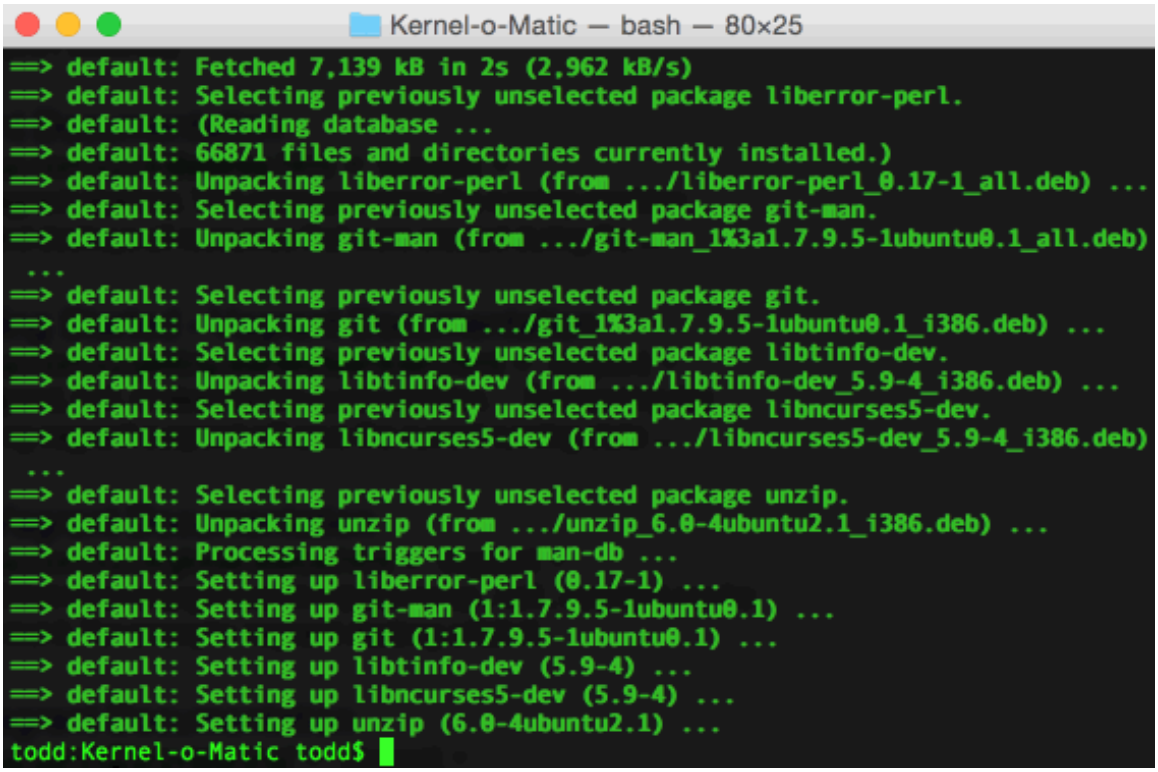


Next we need to download, start, and provision the virtual machine. Luckily, Vagrant makes this simple, and we can do this all in one step by running **vagrant up.**



It will take a few minutes for Vagrant to download and start your VM, and you'll see a long stream of text that describes in detail what Vagrant is currently working on. You may see something like "*Box 'ubuntu/trusty32' could not be found. Attempting to find and install...".* This is normal if you are using a fresh vagrant install.

You will be returned to the command prompt after Vagrant has finished setting up the virtual machine. You can connect to the VM by running **vagrant ssh.**



Now that you are connected to the Ubuntu VM, we are ready to look at the kernel build helper we wrote to make the process easier. The next section will show you the basics of using **adabuild**.

Build the Kernel

Now that you have connected to the virtual machine, we can start the build process. First let's look at some of the build options included with the **adabuild** kernel build helper.

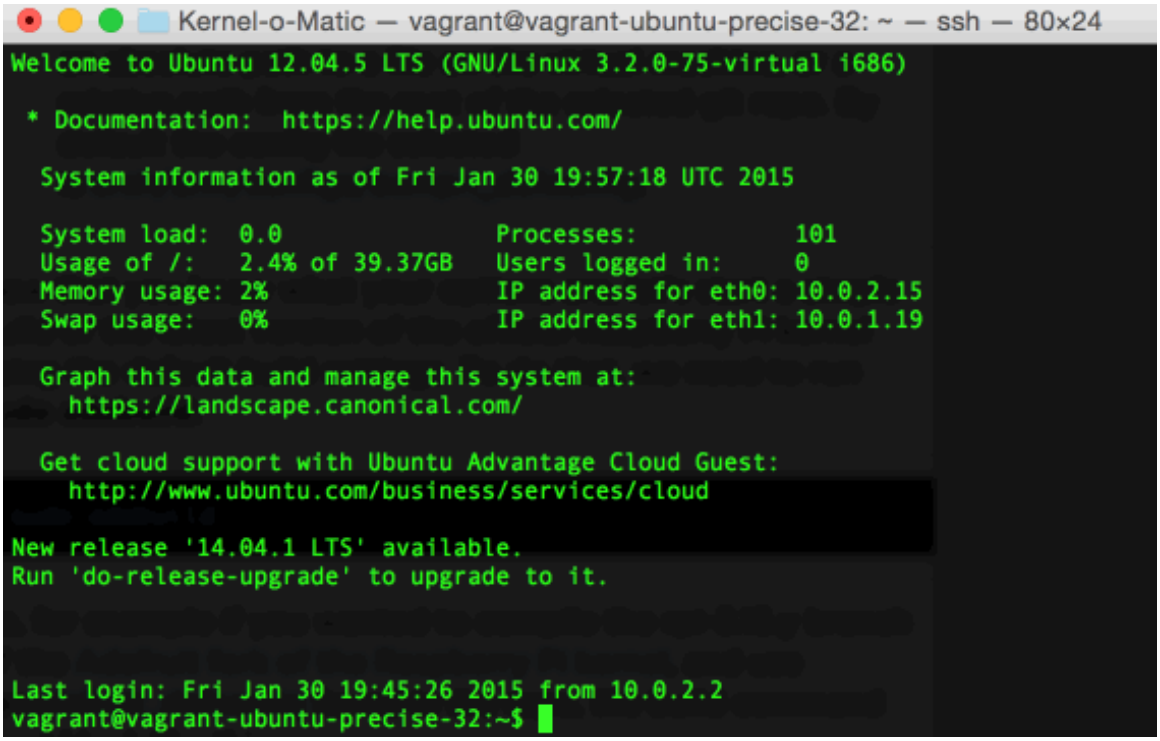
You can view the build options by entering **sudo adabuild -h** and pressing return.

```
$ sudo adabuild -h
usage: adabuild [options]
 This will build the Raspberry Pi Kernel.
 OPTIONS:
   -h      Show this message
   -r      The remote github repo to clone
           Default: raspberrypi/linux
   -b      The git branch to use
           Default: Default git branch of repo
   -1      The config file to use when compiling for Raspi v1
           Default: arch/arm/configs/bcmrpi_defconfig
   -2      The config file to use when compiling for Raspi v2
           Default: arch/arm/configs/bcm2709_defconfig
```

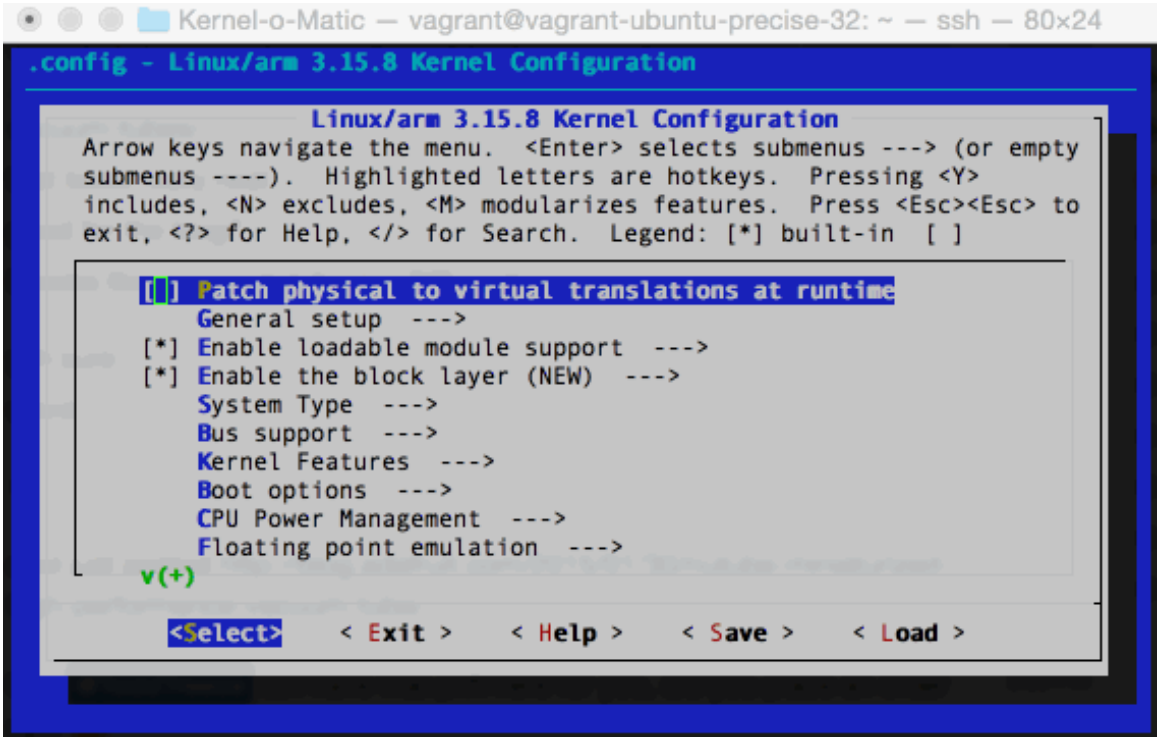
Let's look at the config options in more detail:

- **-h** shows you the usage instructions shown above.
- **-r** allows you to pass a GitHub repo in username/repo format that will be used instead of the official Raspberry Pi linux repo found here: https://github.com/raspberrypi/linux (http://adafru.it/eps)
- **-b** specifies the name of the git branch on the repo to use when compiling. If you do not supply a branch, the default branch of the selected git repo will be used.
- **-1** allows you to point to a custom config file for the Raspberry Pi v1 build. This can be an absolute path to a file on the virtual machine, or a relative path from the root of the selected git repo. By default the config file selected is arch/arm/configs/bcmpri_defconfig (http://adafru.it/ept).
- **-2** allows you to point to a custom config file for the Raspberry Pi v2 build. This can be an absolute path to a file on the virtual machine, or a relative path from the root of the selected git repo. By default the config file selected is arch/arm/configs/bcm2709_defconfig (http://adafru.it/eui).

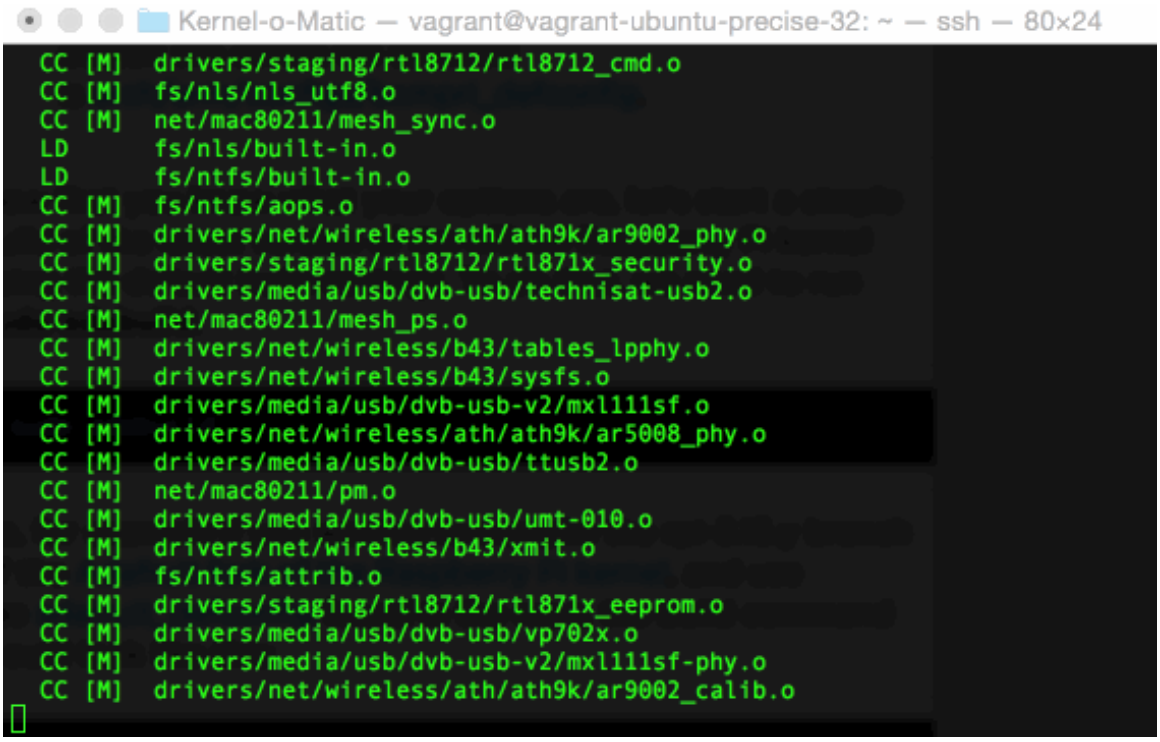
Now that you know what your options are, let's start a simple build of the latest version of the official Raspberry Pi kernel using the default build settings. To do that, we need to run **sudo adabuild**.



After **adabuild** finishes cloning a the required repos for building the kernel, you will be presented with a menu. This is the menu configuration for the kernel, where you can add or remove modules and capabilities. Tweak any options you want here; if you would like to use the defaults in the config file, it is safe to exit and save. Hit TAB and RETURN to exit, and RETURN again to confirm that you want to save.

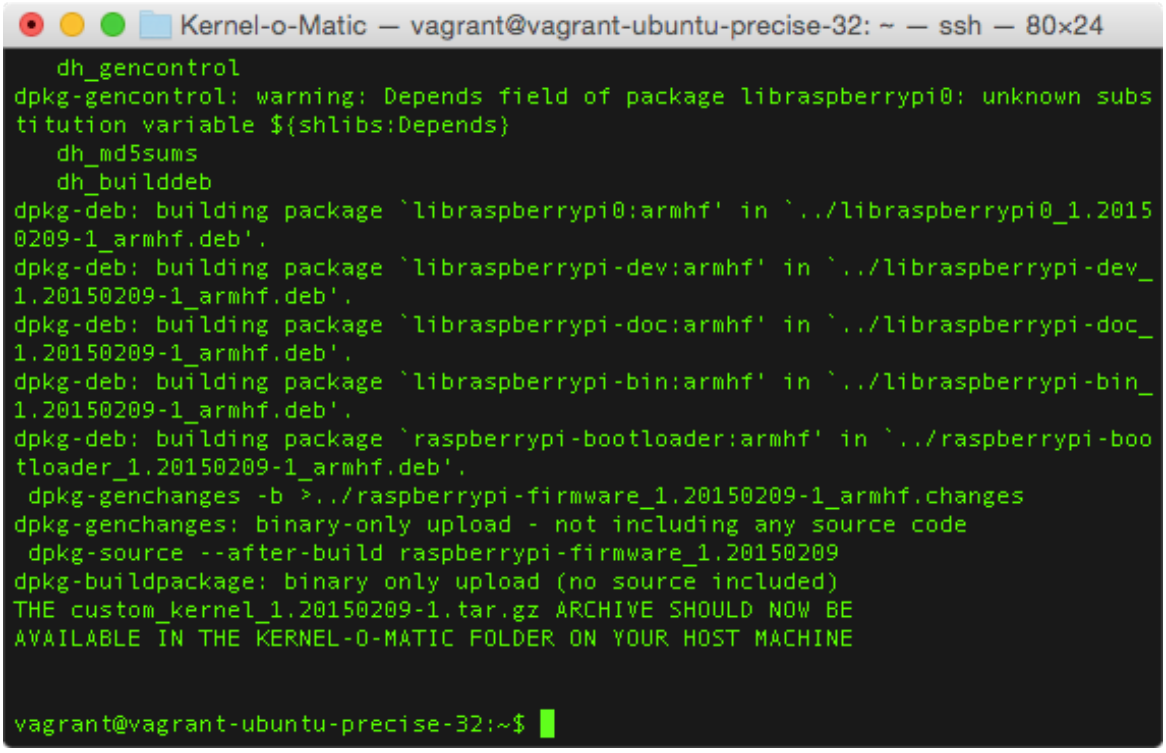


Now comes the part where you sit and watch a kernel compile. It takes about 15 minutes on a 2Ghz Intel i7 with 8 cores working to finish the kernel for the Raspberry Pi v1, but the compile time will vary depending on your machine.



Once the kernel for v1 has finished, you will be presented with the same menu again to configure the options for the Raspberry Pi v2 kernel build. Configure, exit and save the configuration to start building the kernel for the Raspberry Pi v2.

Once both builds have finished, **adabuild** will create a tar.gz archive that you can transfer to your Raspberry Pi. The archive will be available in the Kernel-o-Matic folder on your host computer.



If you would like to build something other than the current official kernel, you can use the options we went over earlier to change the git repo, git branch, and config files the compiler uses. Let's say you wanted to compile the *rpi-3.15.y* branch of the Adafruit fork of the Raspberry Pi kernel (http://adafru.it/epu), and use the adafruit_defconfig (http://adafru.it/epv) configuration file found in the repo for the Raspberry Pi v1 build. The build command would look like this:

```
$ sudo adabuild -r https://github.com/adafruit/adafruit-raspberrypi-linux -b rpi-3.15.y -1 arch/arm/configs/adafruit_
```

Custom PiTFT Builds

If you would like to build a custom kernel with our PiTFT additions included, you should use the pitft branch of the Kernel-o-Matic repo (http://adafru.it/eLy). First, switch to the **pitft** branch in the cloned Kernel-o-Matic folder on your host machine.

```
git fetch && git checkout pitft
```

SSH back into the Kernel-o-Matic VM.

```
vagrant ssh
```

Then, run **adabuild** with no arguments. The *pitft* branch is pointed to the proper linux repo, branch,

device tree overlays, and custom defconfigs for the PiTFT hardware.

```
sudo adabuild
```

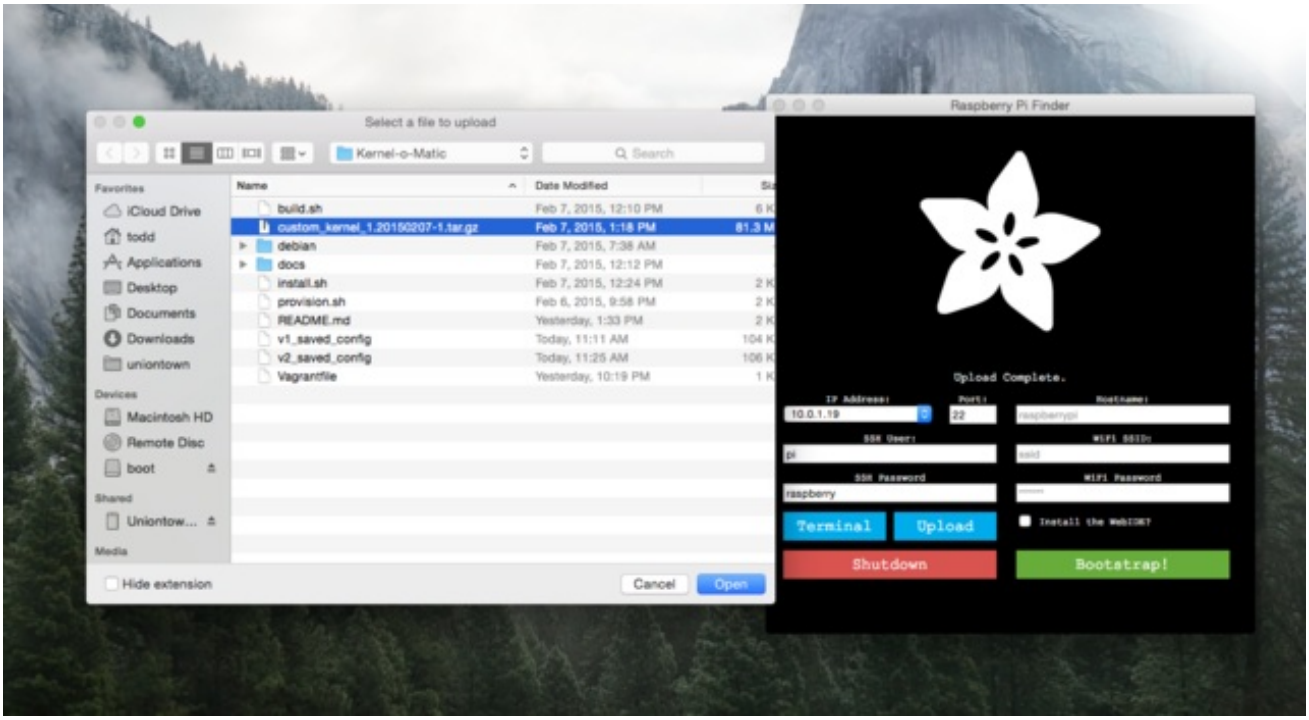
The rest of the build process will look the same as using the default settings described earlier, and you can use both instances of menuconfig to make any changes you would like to the kernel for both versions of the Raspberry Pi. A custom pitft tar.gz archive will be available in the Kernel-o-Matic folder on the host machine when the build has finished. In the next section, we'll show you how to install the custom kernel on your Raspberry Pi.

Install the Kernel

Warning! Installing a broken kernel can render your Raspberry Pi unusable. Please backup your SD card before continuing.

The easiest way to copy the resulting .tar.gz archive on to your Raspberry Pi is to mount a FAT32 formatted USB flash drive on your computer and copy the archive to that. The build process will output the archive to the Kernel-o-Matic folder with a name that starts with **custom_kernel**.

You can also use the latest version of the Adafruit Pi Finder (http://adafru.it/enj) to copy the archive to your Pi over the local network.



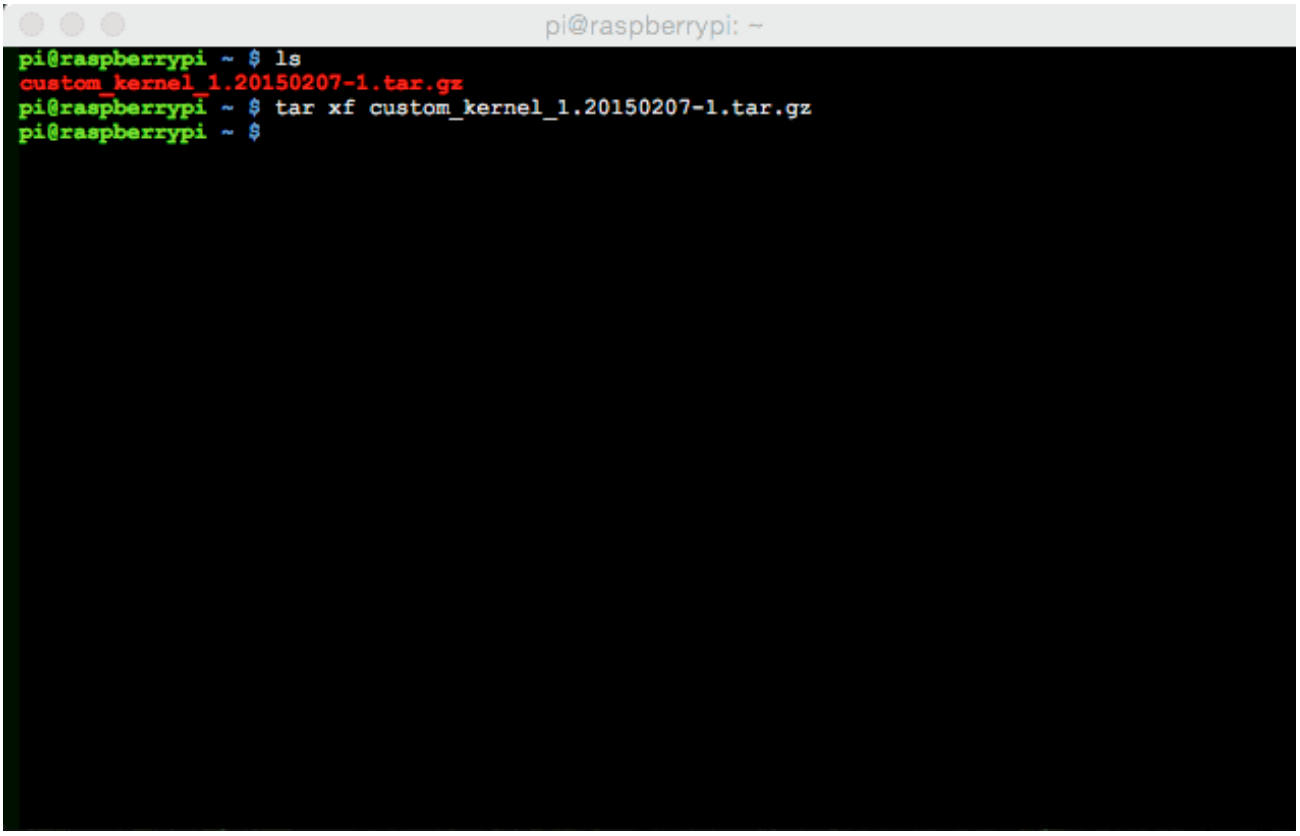
Adafruit Pi Finder

http://adafru.it/euj

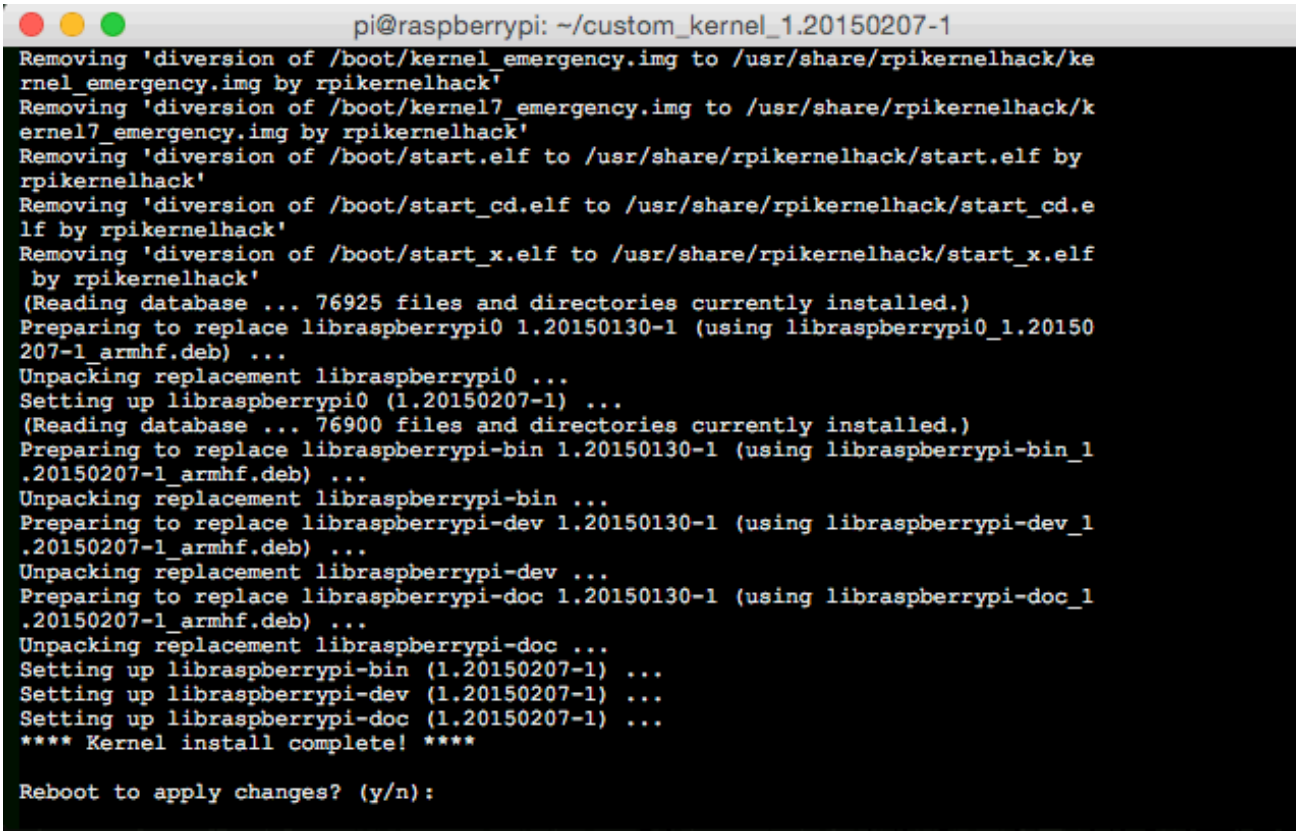
Now that you have the .tar.gz archive copied to your Raspberry Pi, you can extract the archive using **tar**. In this example, the archive we will be using is called *custom_kernel_1.20150207-1.tar.gz*. Replace the name of the example archive with the name of your custom kernel tar.gz archive.

```
$ tar xf custom_kernel_1.20150207-1.tar.gz
```

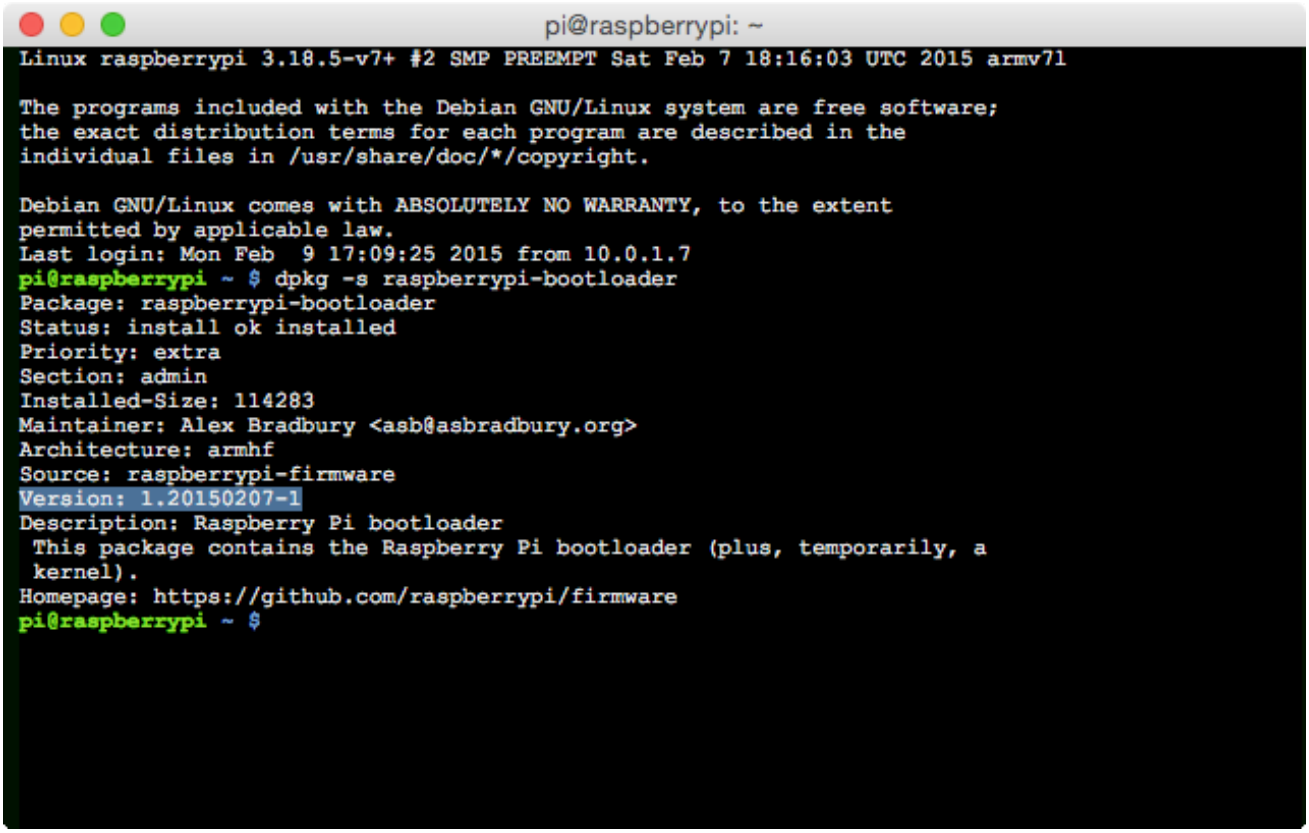
Next, we can use the bundled install script to install the package on your Raspberry Pi. **cd** into the extracted folder and run **sudo ./install.sh** to start the installation.



If you don't see any errors from the install process, you can answer "**y**" to the reboot prompt to apply the changes**.**



Finally, after the reboot, we can check to make sure that the package installed with **dpkg -s raspberrypi-bootloader.**



As you can see above, the custom build is the currently installed version on the Raspberry Pi.

That's all there is to building your own kernel! If you notice any bugs, please use the link below to file an issue on GitHub.

Report a Bug

http://adafru.it/epw