Steps to Create a New SD Card Image

Written by Nick Masso in May 2021

The source of this file is at https://github.com/engr16x/projects-rpi-setup-6/blob/main/pre_deploy_manual.md and should be edited on github first, then changes propogated to the GDrive and Eagle.

This document explains how to create the OS state that can then be imaged, as well as how to image the OS.

If you are making any changes, they should be made to the github repository, and then copied onto the RPi.

Steps

1. Find the smallest SD card that can fit everything you need on it.

   An imaged hard drive can only be flashed to a storage medium that is equal or larger in capacity. This was actually an issue in spring 2019, as advertised 8GB cards purchased for the class had a 7.8GB capacity, smaller than the 7.9GB SD card that was used as a template. This required following these steps to create a new image on those smaller cards.

   The command `sudo raspi-config --expand-rootfs` (expand root filesystem) is run during deploy, so the full size of the cards is utilized properly.

   8GB cards seems to be the limit. I tried on a 4GB card earlier, and I ran out of space.

   There might be a way to shrink partitions or something, but I havent looked into it. This works well enough.

2. Download a version of Raspberry Pi OS (Raspbian pre-2020) to use as the base.

   v6.0 uses `2019-09-26-raspbian-buster` because of some problems the config files were causing that I didnt want to fix. All official archived images can be found here

3. Flash the official version to an SD card.

   Make sure to copy an empty file called `ssh` into the `/boot` directory of the SD card while its connected to your computer. This will allow you to directly SSH without having to log in with a monitor.

4. Power on the RPi and connect it to a network.

   Setup 1 (My preferred way):

   You have your own router, and connect the RPi and your computer to this same router, the RPi using ethernet. You SSH to `raspberrypi.local`, or if you do not have hostname resolution enabled, you can log into your router and find the IP. If you have nmap installed, use `nmap -sS -R -p ssh -O` and find the IP.

   Setup 2:

Theres some way that if you connect the RPi directly to your computer with ethernet, you can share your computer's network connection with the tethered device. I don't know how to do it, so someone can fill this out later.

Setup 3 (you are in a dorm, campus building, or your router is managed by the apartment complex):

Connect the RPi to your computer with ethernet, SSH in, connect the RPi to the wifi by editing `/etc/wpa_supplicant.conf` or whatever. **Make sure to erase this content when you are done running ./pre_deploy or you will be giving your network credentials to every student who uses the image.**

5. Connect to the RPi and use a tool like WinSCP or some other FTP client to get the folder `projects-rpi-setup-6` onto the home directory of the Pi user.
6. SSH into the RPi. username is `pi` and password is `raspberry`
7. `cd` into the `~/projects-rpi-setup-6` directory.
8. `chmod +x ./pre_deploy.sh` to make it executable.
9. `sudo ./pre_deploy` to start the script.

   Assuming you already made your changes before copying your files over, this will take like half an hour to run and everything will be cool. About 5 minutes in, during the installation of `samba`, it will stop everything and prompt you whether you should use some DHCP thing about managed networks. Hit `No` and everything will continue. If you figure out how to automatically say no to this, please edit it in. When done, it will tell you you are good to image the SD card.
10. After nicely powering off the RPi, remove the SD card and connect it to a linux computer.
11. If your computer automatically mounts the drives, unmmount them. Check this by running

    ```
    $ lsblk
    ```

    Unmount drives with

    ```
    $ umount /dev/sdxX
    ```
12. Now we will copy the SD card into a compressed disk image file

    we are copying the disk, not a partition on the disk, so 'sdx' will not be followed by a number. The `status=progress` flag will give some "is this still working?" kind of information, but you dont get a remaining time or anything.

    ```
    $ sudo dd if=/dev/sdx conv=sync,noerror status=progress bs=64K | gzip -c >
    [path to resulting file].img.gz
    ```

If you have `pv` (pipe viewer) installed and want a fun progress bar, you can pipe it through there (removing the `status=progress` flag). However, you will have to give pv the approximate size of the disk you are copying.

```
$ sudo dd if=/dev/sdx conv=sync,noerror bs=64K | pv -s [Size of disk][units]
| gzip -c > [path to resulting file].img.gz
```

For example, I would use:

```
$ sudo dd if=/dev/sdc conv=sync,noerror bs=64K | pv -s 7400M | gzip -c >
~/disk_image.img.gz
```

You can play with the blocksize if you want but 64K has never given me any problems.

That compressed file now has your image in it. You can use your own image flashing workflow as normal with that file.