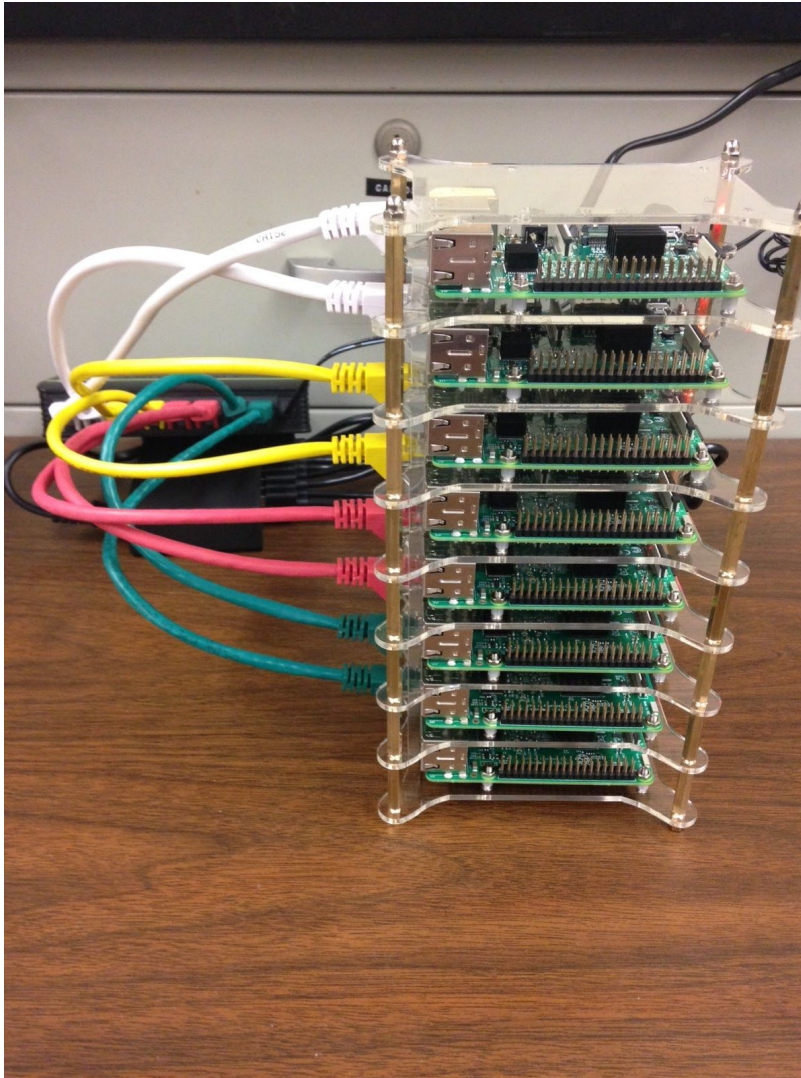


How To Make a Raspberry Pi Cluster

FOR
DUMMIES®



8-node Raspberry Pi 3 Cluster

By: Mike Garcia

© 2017

Contents

| | |
|--|-----------|
| Contents | 2 |
| Parts List | 4 |
| How to Setup the 1st Pi (aka "The Master Node") | 8 |
| How to Install the OS (Operating System) | 8 |
| How to Flash Image to MicroSD | 9 |
| How to Update Your Pi's Settings | 10 |
| How to Setup Internet Connectivity | 11 |
| How to Update Master Node | 12 |
| How to Install Software for Parallelization | 13 |
| How to Install OpenMP | 13 |
| How to Install OpenMPI | 13 |
| How to Install mpi4py | 13 |
| How to Clone a Raspberry Pi | 14 |
| How to Clone a Raspberry Pi using Windows | 14 |
| How to Clone a Raspberry Pi using Mac | 15 |
| How to Clone a Raspberry Pi using Linux | 18 |
| How to Setup DHCP Server | 21 |
| How to Install DHCP Server | 21 |
| How to Adjust DHCP Server Configuration | 23 |
| How to Adjust Interface File | 23 |
| How to Configure NAT | 24 |
| How to Confirm Settings on Master Node | 26 |
| How to Verify DHCP Lease | 26 |
| How to Assign static IP's for slave nodes | 27 |
| How to Test SSH on Slave Node | 27 |
| How to Setup Passwordless SSH | 28 |
| How to Verify Hostnames | 28 |
| How to Generate Master Node SSH Key | 29 |
| How to Send Files to Nodes | 31 |
| How to Setup Parallel-SSH & Parallel-SCP | 31 |
| Appendix | 33 |
| How to Install MPICH (alternative to OpenMPI) | 33 |
| How to Install mpi4py (for MPICH installation) | 33 |
| How to Use a Host File with MPI | 34 |
| How to Compile with OpenMP / MPI | 35 |

Parts List

For this demo, we will be setting up a Raspberry Pi 3 cluster with 8 Raspberry Pi's (nodes). The cluster will consist of one "master" node and seven "slave" nodes. The master node will serve as a DHCP server to the slave nodes (this means the master node will assign IP addresses to the slave nodes and handle all network coordination for the cluster). Once our cluster is complete, we will communicate with the slave nodes through the master node.

In order to make our final product more presentable and less of a wire mess, we opted for a 10-port USB wall charger that all of our Pi's could plug into, along with an optional USB power cord for our ethernet switch too. This reduced the number of electrical outlets required to power the cluster, and makes for easier transport.

All the parts used for this project can be found on this Amazon shopping list:

<http://a.co/53qmkLV>

NOTE: NOTE: You do not need the switch or switch USB power cable if you plan on connecting all of your cluster Pi's into an existing router/switch ****and**** have enough ports to do so.

Also, this shopping list assumes you already have a USB keyboard and mouse available to use.

8x - Raspberry Pi 3's



8x - MicroSD Cards



8x - Heatsinks for Pi's



8x - Pi USB Power Cables



8x - Stackable Cases for Pi's



1x - 10-Port USB Power Plug



1x - 8-Port Ethernet Switch



1x - USB Power for Ethernet Switch



8x - Ethernet Cables

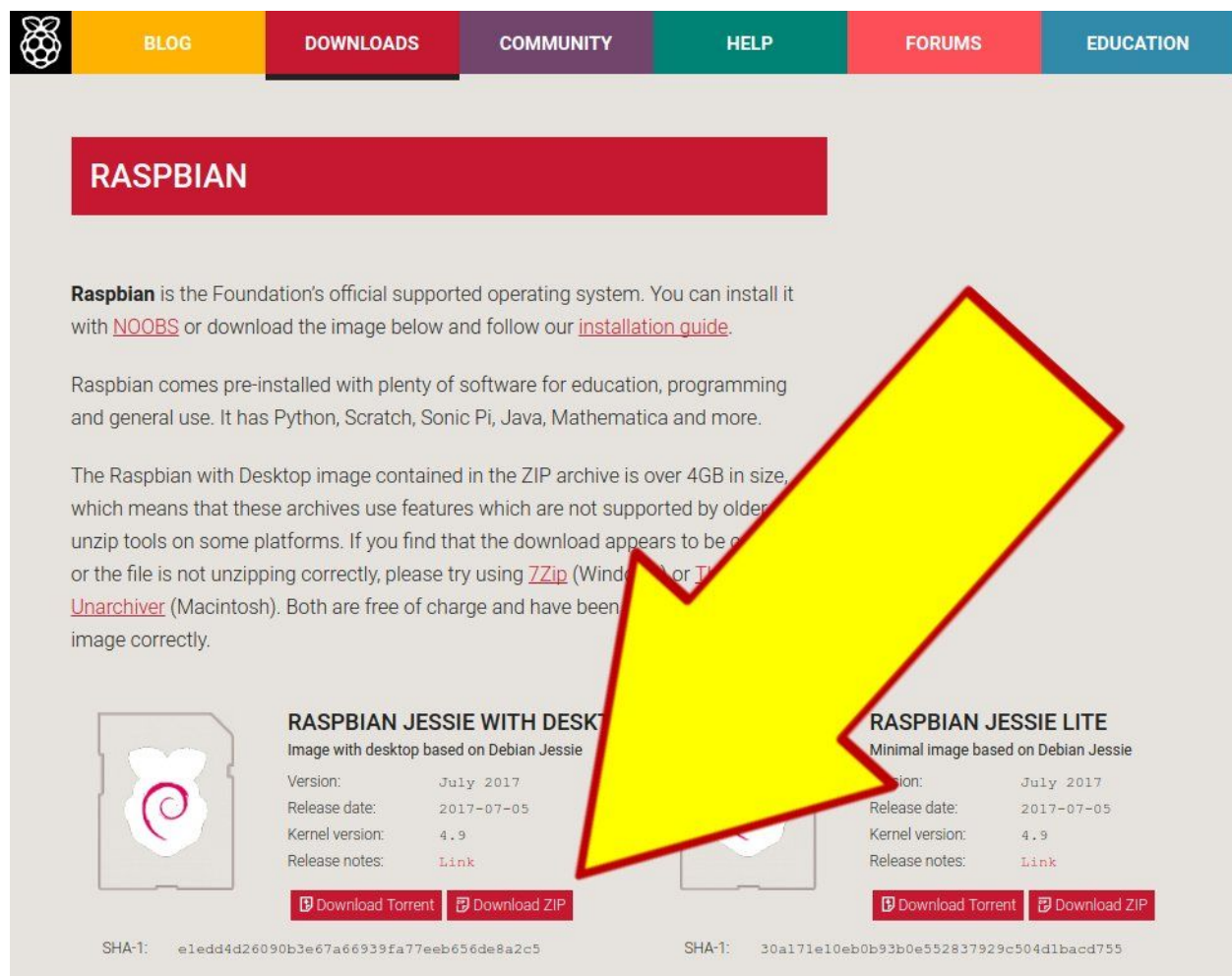


How to Setup the 1st Pi (aka "The Master Node")

How to Install the OS (Operating System)

First things first, we need to pick and install an operating system for all of our nodes. For the best support and documentation we used the official 32 bit Operating System "Raspbian Jessie with Desktop". Go to the Raspberry Pi's official site, navigate to the "Downloads" section and click "Download ZIP" for the aforementioned operating system.

<https://www.raspberrypi.org/downloads/raspbian/>



The screenshot shows the Raspbian Downloads page. A large yellow arrow with a red outline points from the right towards the download links for 'Raspbian Jessie with Desktop' and 'Raspbian Jessie Lite'. The page includes a navigation bar with links to Blog, Downloads, Community, Help, Forums, and Education. The main content area describes Raspbian as the official supported operating system and provides details for two versions: 'Raspbian Jessie with Desktop' and 'Raspbian Jessie Lite'. Both versions are based on Debian Jessie and were released in July 2017. The 'Desktop' version is over 4GB in size, while the 'Lite' version is minimal. Both versions include links to download the image as a torrent or ZIP file, and provide the SHA-1 hash for verification.

| Version | Release date | Kernel version | Release notes |
|------------------------------|--------------|----------------|----------------------|
| Raspbian Jessie with Desktop | July 2017 | 4.9 | Link |
| Raspbian Jessie Lite | July 2017 | 4.9 | Link |

SHA-1: e1edd4d26090b3e67a66939fa77eeb656de8a2c5

SHA-1: 30a171e10eb0b93b0e552837929c504d1bacd755

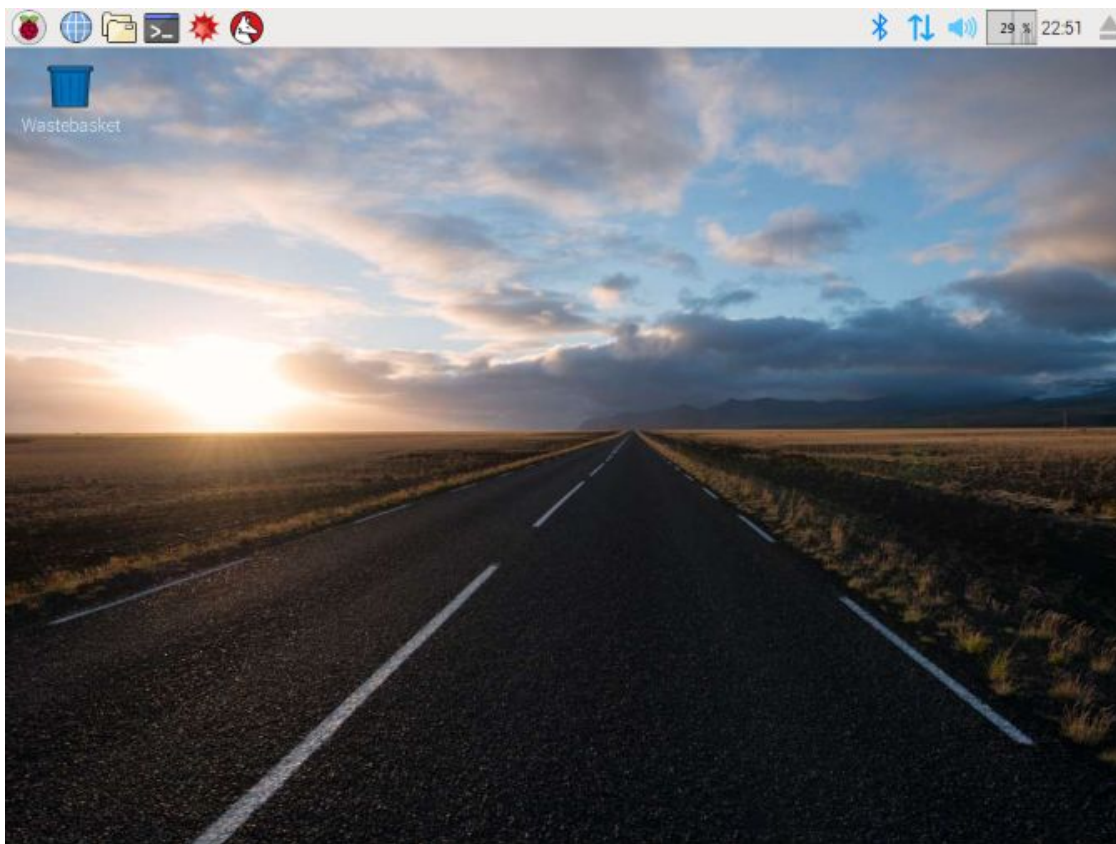
How to Flash Image to MicroSD

Use Win32DiskImager (for Windows) or Etcher (for Mac/Linux) to install your preferred image to the microSD card.

- Connect your microSD card to your computer
- Select the image we downloaded and flash it to the microSD card
- Once finished, plug into the Pi and plug in the power!

You can find more details on Raspberry Pi site:

<https://www.raspberrypi.org/documentation/installation/installing-images/README.md>



Raspbian Jessie Desktop

How to Update Your Pi's Settings

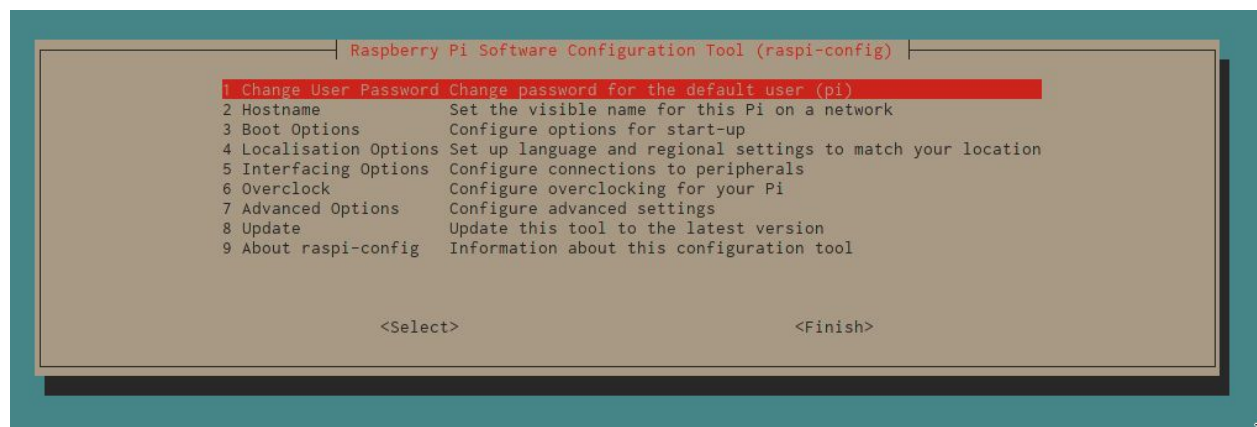
The raspbian jessie operating system comes with default settings that we'll need to change to fit our needs. By default it has United Kingdom localization settings, ssh disabled, etc. Depending on where you're located and what your plans are for the cluster, you will most likely have to edit some of these default settings.

```
pi@raspberrypi:~ $ sudo raspi-config
```

Once your Pi has successfully booted, open a terminal window (press Ctrl+Alt+T) and type the following command:

```
sudo raspi-config
```

You should be presented with a screen that looks like this:



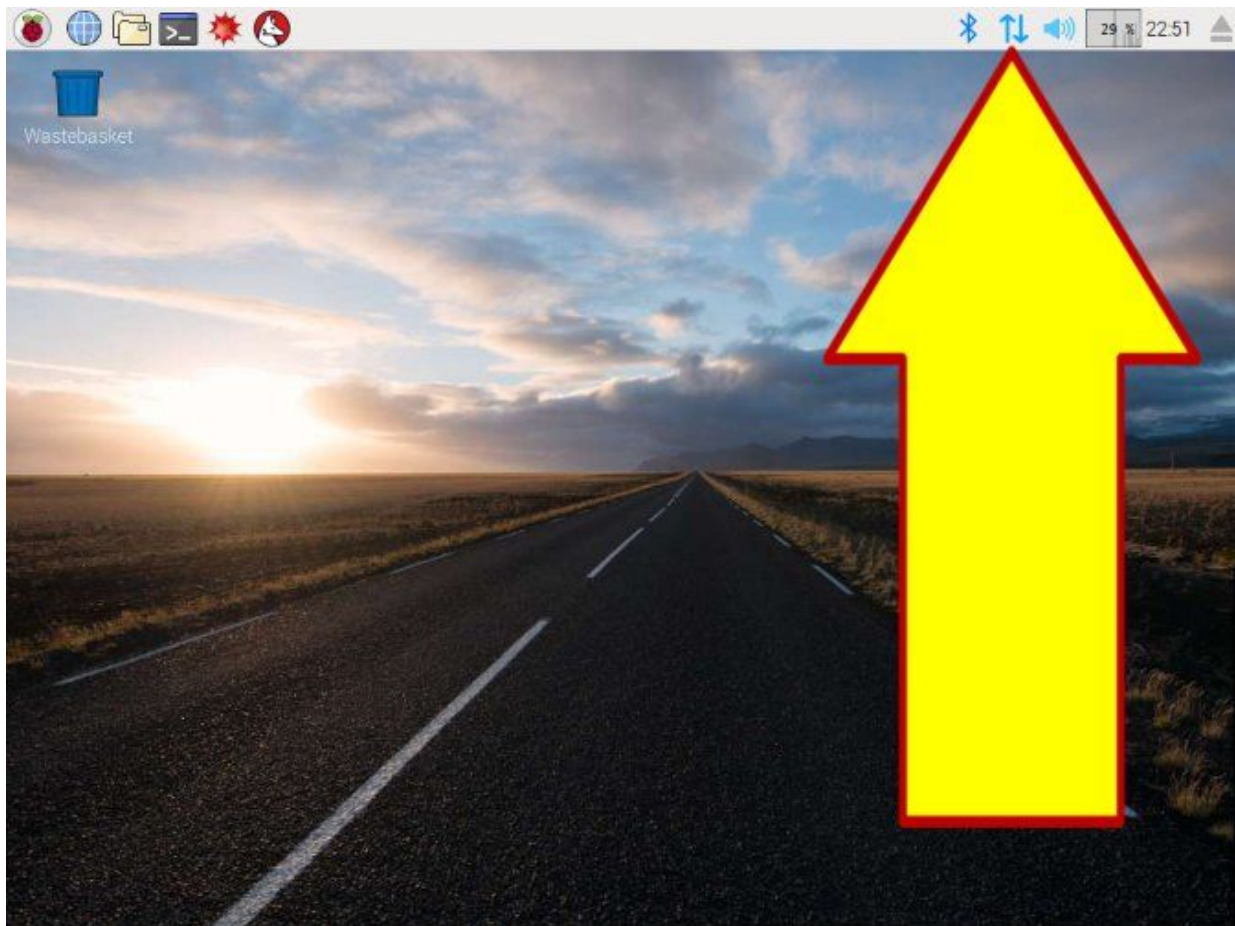
If you're not in Great Britain, you'll probably want to change these settings:

- Localisation Options → Change Locale → **Unselect** "en_GB.UTF-8 UTF-8" and **Select** your locale, in our case we want US settings, which is "en_US.UTF-8 UTF-8".
- Localisation Options → Change Time zone → **Select your timezone**.
- Localisation Options → Change Wifi Country → **Select your country**.
- Interfacing Options → SSH → **Enable**.
- Advanced Options → Memory Split → **Change value to 16** (since we're not using the GUI much, we'd prefer more of the shared memory for computing)

How to Setup Internet Connectivity

The easiest way to get internet connectivity on the master node is to directly plug into an existing modem or router. An ethernet connection will be much faster than the onboard WiFi during the setup, especially for the next section (updating/upgrading Raspbian).

Alternatively, we can connect the network via WiFi through the WiFi tray icon (see below).



How to Update Master Node

We want to make sure that we're using all the updated packages since when the distribution image file was created, so in a terminal (Ctrl+Alt+T) we'll run:

```
sudo apt-get update
```

This will download information for any packages with newer versions and their dependencies. Next we'll run the command to actually update to the newer versions:

```
sudo apt-get upgrade
```

After this completes, let's go ahead and reboot the system

```
reboot
```

We're about to clone this node to the remaining ones, so this would be a good time to install any applications you'd like to have available on all of the nodes.

```
sudo apt-install p7zip vim ranger git tmux pssh openjdk-8-jdk etc...
```

How to Install Software for Parallelization

For this project, and probably most cluster projects, you will use software designed for multiprocessing and parallel computing. Installing OpenMP and MPI is what we'll cover in this section. It is important to do this step prior to cloning this node because this software will be required by all nodes in the cluster in order to use it.

OpenMP (Open Multi-Processing) is used for parallelizing a program within a multicore CPU. OpenMP creates multiple threads, via forking, to utilize all the cores of a CPU.

MPI (Message Passing Interface) is used for parallelizing a program among multiple node CPUs, as in the case of a cluster. There are two implementations of MPI available for our raspberry pi's. There is MPICH and OpenMPI (not to be confused with OpenMP). For most use cases, especially in our case, they are interchangeable. If you run into issues with your current MPI, you can easily replace it with the other.

We also show how to install mpi4py, which is a python interface to whatever MPI implementation you have installed. This allows for MPI calls within python scripts.

How to Install OpenMP

OpenMP is installed by default if using gcc/g++ as our compiler.

Compatible Compiler List: <http://www.openmp.org/resources/openmp-compilers/>

How to Install OpenMPI

```
sudo apt-get install openmpi-bin openmpi-common openmpi-doc libopenmpi-dev
```

How to Install mpi4py

Installation for Python 2

```
sudo apt-get install python-mpi4py
```

Installation for Python 3

```
sudo apt-get install python3-mpi4py
```

How to Clone a Raspberry Pi

By “clone”, we want to make an exact copy of the raspberry pi’s hard drive that we’ve customized to this point in the tutorial. The cloning process takes place in two steps. Step one is to read the master node hard drive (microSD card) to an image file. Step two is to write that image file to a blank microSD card that will be a slave node’s hard drive.

Since all of the raspberry pi’s will need the software and settings we’ve installed thus far, it is much faster to clone this master node onto the hard drives of the other nodes. Once finished cloning, we can continue to make final changes on our master node, and make even fewer changes to the slave nodes.

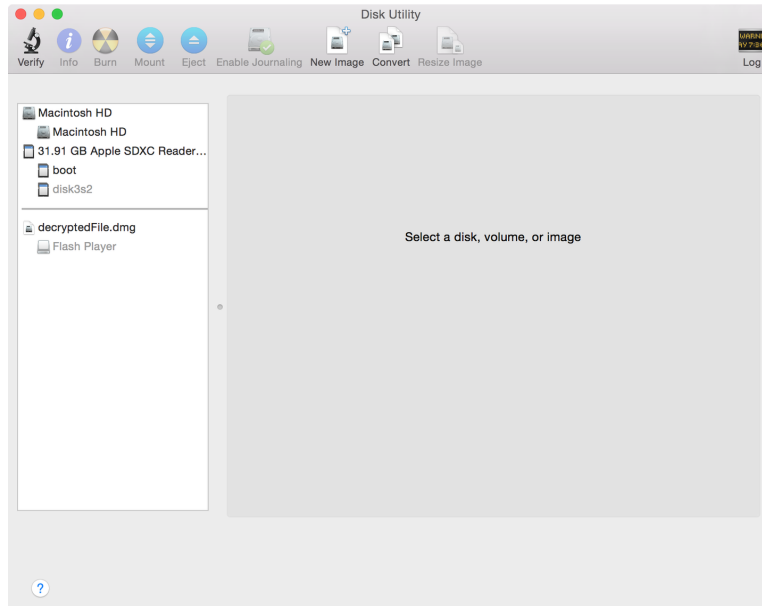
How to Clone a Raspberry Pi using Windows

NOTE: The easiest way to clone is on a Windows machine

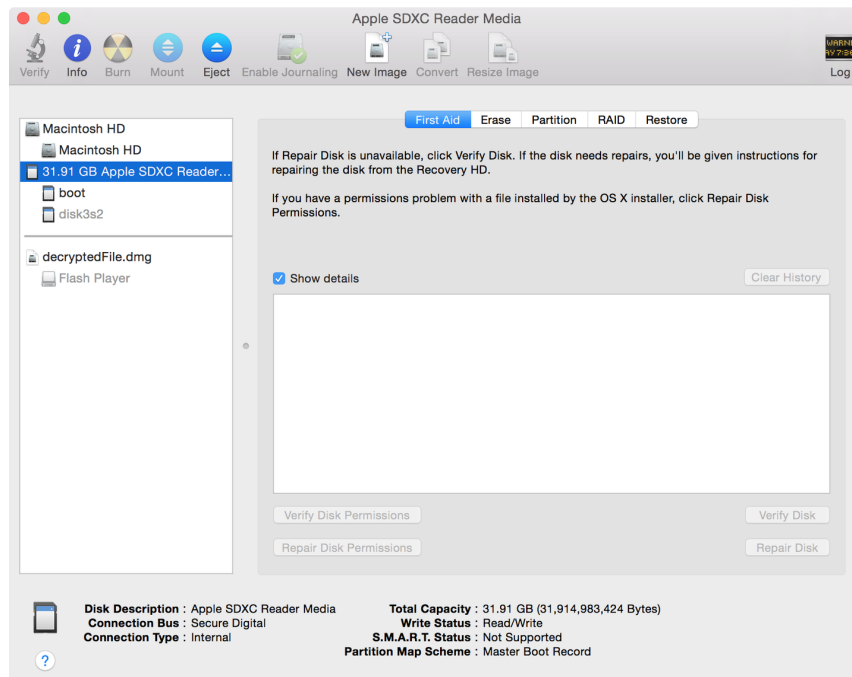
- Install **Win32 Disk Imager** : <https://sourceforge.net/projects/win32diskimager/>
- Insert the updated master node’s MicroSD card into your Windows machine.
- Make up an image name (e.g. “node.img”) and save to your desktop.
- Click "Read" (this will create an image of the MicroSD).
- Once it’s finished reading to an image file ...
- Eject the MicroSD and insert a new one to clone the image to.
- In Win32 Disk Imager, Select the image file you just created, choose the MicroSD drive from the dropdown and click "Write".
- Continue writing your image file to the remaining node’s microSD cards.

How to Clone a Raspberry Pi using Mac

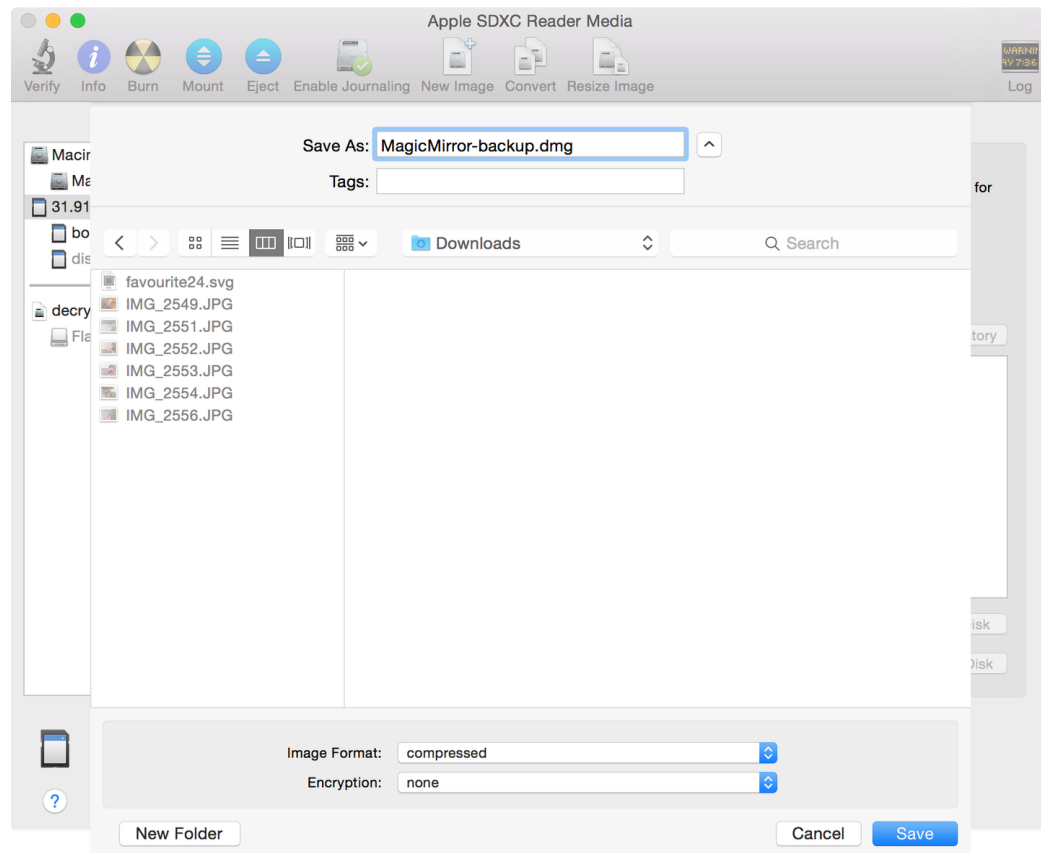
- Plug in the master node microSD card into your Mac.
- Open Disk Utility (Applications → Utilities → Disk Utility).



- Create an Image from your microSD.
 - Select your microSD card from the sidebar, and then click “New Image” in the top bar.



- Name your image file and select a backup destination and click “Save” (this could take 15+ minutes).



- Once it finishes, eject your microSD and prepare to write the new file to a slave node's microSD card.
- Insert a slave node's blank microSD card into your mac.
- Open a terminal window (click the Spotlight, and type “Terminal”)
- Enter the following command:

```
diskutil list
```

Try to identify the device ID of your microSD card (for example, ours shows up as /dev/disk3).

```
Akshays-MacBook-Air:~ akshaygangwar$ diskutil list
/dev/disk0 (internal, physical):
#:           TYPE NAME           SIZE          IDENTIFIER
0:         GUID_partition_scheme   *121.3 GB     disk0
1:             EFI EFI             209.7 MB     disk0s1
2:       Apple_CoreStorage Macintosh HD   120.5 GB     disk0s2
3:       Apple_Boot Recovery HD           650.0 MB     disk0s3

/dev/disk1 (internal, virtual):
#:           TYPE NAME           SIZE          IDENTIFIER
0:             Macintosh HD         +120.1 GB     disk1
               Logical Volume on disk0s2
               BEF9CD9D-BA46-4CC9-B856-AD93A912208F
               Unlocked Encrypted

/dev/disk2 (disk image):
#:           TYPE NAME           SIZE          IDENTIFIER
0:         GUID_partition_scheme   +226.3 MB     disk2
1:             Apple_HFS GIMP 2.8.18    226.3 MB     disk2s1

/dev/disk3 (internal, physical):
#:           TYPE NAME           SIZE          IDENTIFIER
0:         FDisk_partition_scheme   *15.9 GB     disk3
1:             Windows_FAT_32 boot      66.1 MB     disk3s1
2:             Linux                 15.9 GB     disk3s2
```

Once you've identified your slave node's microSD, enter the following command and **replace "disk#" with whatever yours is**:

```
diskutil unmountDisk /dev/disk#
```

Use the following command to write the image file to the microSD card:

```
sudo dd if=~/.path/to/node_image.dmg of=/dev/disk#
```

Once the write is complete, you will see a confirmation from dd. You can then remove the card from your Mac, and insert it back in the Raspberry Pi.

Repeat steps to write the image file to another slave node microSD that hasn't been setup yet.

How to Clone a Raspberry Pi using Linux

- Insert the microSD card into your Linux computer.
- Open a terminal window, and enter the command:

```
sudo fdisk -l
```

Look for the device name of your master node microSD card. We have a 16GB SD card, so we see it in the image below as the device `/dev/sdb` (which has a size of 14.9GB). The size difference is typical since the actual storage is always a bit smaller than the advertised size.

- Note down this device name.

```
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disklabel type: dos
Disk identifier: 0xa11cac6d

Device      Boot    Start        End    Sectors    Size Id Type
/dev/sda1   *         2048     206847     204800    100M  7 HPFS/NTFS/exFAT
/dev/sda2             206848  184322047  184115200    87.8G  7 HPFS/NTFS/exFAT
/dev/sda3       184322048  434464767  250142720   119.3G 83 Linux
/dev/sda4       434466814  976771071  542304258   258.6G  f W95 Ext'd (LBA)
/dev/sda5       440324096  696324095  256000000    122.1G  7 HPFS/NTFS/exFAT
/dev/sda6       696326144  976771071  280444928    133.7G  7 HPFS/NTFS/exFAT
/dev/sda7       434466816  440322047    5855232     2.8G 82 Linux swap / Solaris

Partition 4 does not start on physical sector boundary.
Partition table entries are not in disk order.

Disk /dev/sdb: 14.9 GiB, 15931539456 bytes, 31116288 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x04ac9d6c

Device      Boot    Start        End    Sectors    Size Id Type
/dev/sdb1             8192    137215     129024     63M  c W95 FAT32 (LBA)
/dev/sdb2       137216  31116287  30979072   14.8G 83 Linux
shivam@beebom-HP-15-Notebook-PC ~ $
```

Now we'll use the `dd` command to write the image to your hard disk by entering the command:

CAUTION: Double check the parameters in the command above. Entering the wrong parameters will destroy data on specified drive.

```
sudo dd if=/dev/sdb of=~/.master_node_backup.img
```

Above, the “**if**” parameter (input file) specifies the file to clone. In our case, it is **/dev/sdb**, which is our microSD card's device name. Make sure to change “**sdb**” to whatever your device is. The “**of**” parameter (output file) specifies the filename to write to. We chose “**master_node_backup.img**” in our home directory.

Once this process is finished, the output will print a number of records in and out and the total bytes copied. It is now safe to remove the master node microSD card and plug back into the master node.

Insert slave node's microSD into Linux machine. Before continuing, we need to make sure that the microSD is **unmounted**. To check this, open a terminal and run the following command:

```
sudo mount | grep sdb
```

(replace “**sdb**” with whatever you found your slave microSD to be)

If the output of this command is blank, the drive is unmounted, so we can continue writing the image to the microSD card.

If there is some output showing the mounted partitions, then we need to unmount each of them:

```
sudo umount /dev/sdb1 /dev/sdb2 /dev/sdb3
```

Now we'll use **dd** to write the image we made to the slave node's microSD card:

CAUTION: Double check the parameters in the command above. Entering the wrong parameters will destroy data on specified drive.

```
sudo dd if=~/.master_node_backup.img of=/dev/sdb
```

This is the command we used to make a clone, but reversed. The input file is the backup image, while the output file is the microSD card.

Once the copy is completed, **dd** will output a confirmation. At this point we can eject the microSD and place back into the slave node.

Repeat these steps for the remaining slave node's microSD cards.

How to Setup DHCP Server

How to Install DHCP Server

NOTE: If you plan on plugging every Pi into an existing router that already controls the DHCP than you do not need to setup DHCP on the master node.

In our project we are using an unmanaged switch to network the pi cluster. Our master node will connect to the outside internet via it's built-in wifi connecting to some existing router. With our master node as the DHCP server, it will assign static IP addresses to all the other raspberry pi's in the cluster and handle internet on the rest of the cluster as well.

We'll start by changing the **host name** of the master node to something short and recognizable:

```
sudo raspi-config
```

Follow the on-screen options:

Advanced Options → Host Name → **rpi3-node0**

Install DHCP server software so master node can assign IP addresses and coordinate network traffic & jobs:

```
sudo apt-get install isc-dhcp-server
```

Now we need to configure our DHCP server:

```
sudo nano /etc/dhcp/dhcpd.conf
```

Look about 12 lines down from the top of the file and make the following edits:

- **ADD** a comment tag (#) in front of `option domain-name "example.org";`
- **ADD** a comment tag (#) in front of `option domain-name-servers ns1.example.org, ns2.example.org;`
- **REMOVE** the comment tag (#) from the line `#authoritative;`

The result should look like this:

```
GNU nano 2.2.6 F
#
# Sample configuration file for ISC dhcpd for Debian
#
#
# The ddns-updates-style parameter controls whether or not the serv
# attempt to do a DNS update when a lease is confirmed. We default
# behavior of the version 2 packages ('none', since DHCP v2 didn't
# have support for DDNS.)
ddns-update-style none;

# option definitions common to all supported networks...
# option domain-name "example.org";
# option domain-name-servers ns1.example.org, ns2.example.org;

default-lease-time 600;
max-lease-time 7200;

# If this DHCP server is the official DHCP server for the local
# network, the authoritative directive should be uncommented.
authoritative;

# Use this to send dhcp log messages to a different log file (you a
```

In the same file, at the very bottom ...

- **ADD** the following entire subnet block:

```
subnet 192.168.8.0 netmask 255.255.255.0 {
    range 192.168.8.100 192.168.8.200;
    option broadcast-address 192.168.8.255;
    option routers 192.168.8.1;
    max-lease-time 7200;
    option domain-name "rpi3";
    option domain-name-servers 8.8.8.8, 8.8.4.4;
}
```

NOTE: We're using the subnet *192.168.8.0* but you can use a different one

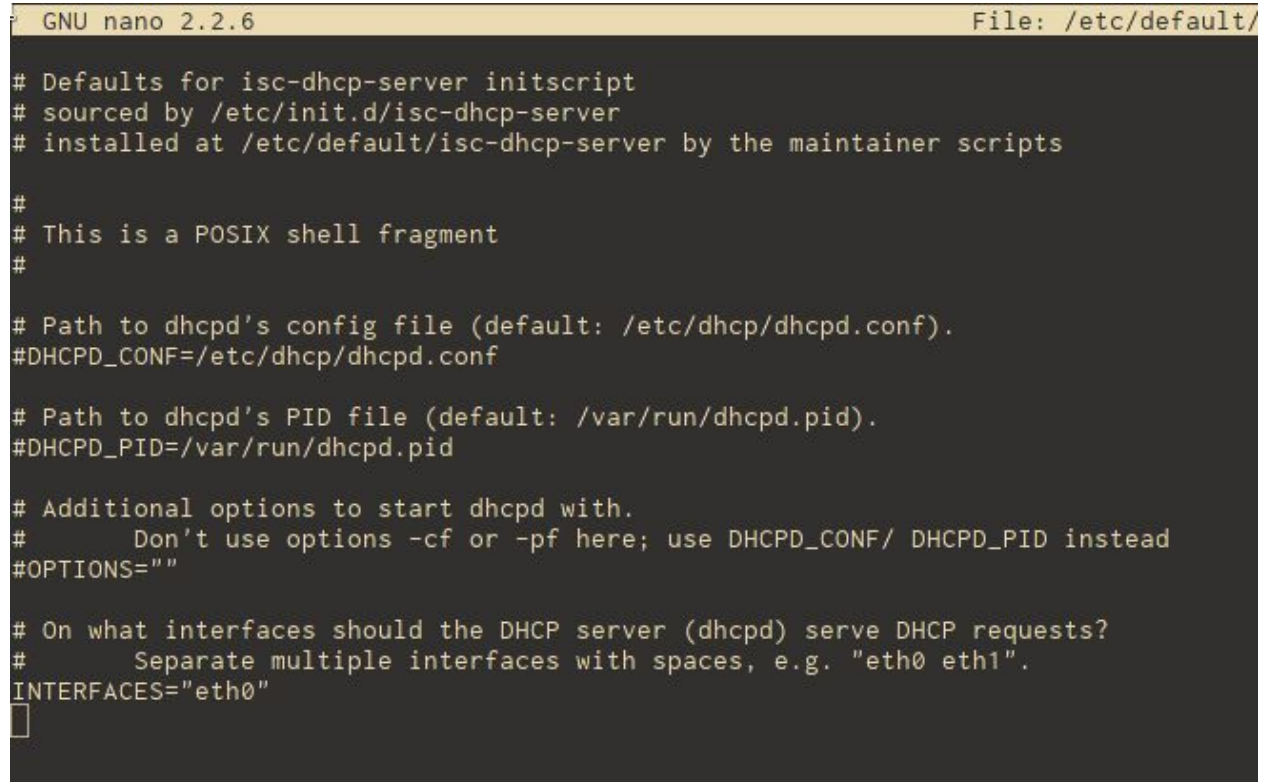
- **Save & Exit** (press **Ctrl+X** followed by **Y**)

How to Adjust DHCP Server Configuration

Enter the following command in the terminal:

```
sudo nano /etc/default/isc-dhcp-server
```

- **ADD** "eth0" to the empty interfaces line at bottom, so it reads `INTERFACES="eth0"`



```
GNU nano 2.2.6 File: /etc/default/isc-dhcp-server

# Defaults for isc-dhcp-server initscript
# sourced by /etc/init.d/isc-dhcp-server
# installed at /etc/default/isc-dhcp-server by the maintainer scripts
#
# This is a POSIX shell fragment
#
# Path to dhcpd's config file (default: /etc/dhcp/dhcpd.conf).
#DHCPD_CONF=/etc/dhcp/dhcpd.conf
#
# Path to dhcpd's PID file (default: /var/run/dhcpd.pid).
#DHCPD_PID=/var/run/dhcpd.pid
#
# Additional options to start dhcpd with.
# Don't use options -cf or -pf here; use DHCPD_CONF/ DHCPD_PID instead
#OPTIONS=""
#
# On what interfaces should the DHCP server (dhcpd) serve DHCP requests?
# Separate multiple interfaces with spaces, e.g. "eth0 eth1".
INTERFACES="eth0"
```

- **Save & Exit** (press **Ctrl+X** followed by **Y**)

How to Adjust Interface File

Now we need to get our master node to serve as DHCP & NAT server for cluster.

Enter the following command in a terminal:

```
sudo nano /etc/network/interfaces
```

Find the line that reads: `iface eth0 inet manual` and ...

- **ADD** the line `auto eth0` right above that
- **CHANGE** the line `iface eth0 inet manual` to
`iface eth0 inet static`

Now we'll assign a static IP address for the master node

- **ADD** the line `address 192.168.8.1` directly below the line we just modified

And underneath that, create a new line ...

- **ADD** the line `netmask 255.255.255.0`

Our final file should look something like this:

```
...
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
    address 192.168.8.1
    netmask 255.255.255.0

allow-hotplug wlan0
auto wlan0
iface wlan0 inet dhcp
...
```

- **Save & Exit** (press **Ctrl+X** followed by **Y**)
- **RESTART** the Pi

All DHCP settings are now completed!

How to Configure NAT

Now we'll configure IP tables to provide network address translation services to our master node.

Enter the following command into the terminal:

```
sudo nano /etc/sysctl.conf
```

- **REMOVE** comment tag (#) on the line `#net.ipv4.ip_forward=1` and make sure in this line it is set equal to 1, which it should be by default.
- **Save & Exit** (press **Ctrl+X** followed by **Y**)

```
sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

```
sudo iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
```

```
sudo iptables -A FORWARD -i wlan0 -o eth0 -m state --state RELATED
```

```
sudo iptables -A FORWARD -i eth0 -o wlan0 -j ACCEPT
```

Now we'll check everything is correct by running:

```
sudo iptables -t nat -S
```

Which should output:

```
-P PREROUTING ACCEPT
-P INPUT ACCEPT
-P OUTPUT ACCEPT
-P POSTROUTING ACCEPT
-A POSTROUTING -o wlan0 -j MASQUERADE
```

One more command to verify everything is correct:

```
sudo iptables -S
```

Which should output:

```
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A FORWARD -i wlan0 -o eth0 -m state --state RELATED
-A FORWARD -i eth0 -o wlan0 -j ACCEPT
```

To save these settings

```
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

To make sure this is used every time interface comes up...

```
sudo nano /etc/network/interfaces
```

- **ADD** `post-up iptables-restore < /etc/iptables.ipv4.nat` under our previously added line `netmask 255.255.255.0` in the **eth0** section.
- **Save & Exit** (press **Ctrl+X** followed by **Y**)

Now, every time eth0 successfully comes up, it will run the iptables restore command to reset the routes and NAT we need.

How to Confirm Settings on Master Node

Enter the following command in the terminal:

```
ifconfig
```

It should output the IP address we assigned it `inet addr:192.168.8.1` under the **eth0** section.

How to Verify DHCP Lease

Power on one of the cloned slave node raspberry pi's we created earlier and make sure it's connected to the same switch as the master node.

This slave node should have automatically received an IP address from the master node's DHCP server we just setup.

To verify this happened, run this command on the master node terminal:

```
cat /var/lib/dhcp/dhcpd.leases
```

You should see one or more entry blocks for a lease on a new IP address.

Example Output:

```
lease 192.168.8.103 {
  starts 3 2017/06/10 05:11:28;
  ...
  hardware ethernet bc:4f:f5:8c:14:d9
  ...
  client-hostname "rpi3-node1";
}
```

COPY the MAC address (hardware ethernet address) listed in your output

How to Assign static IP's for slave nodes

On the master node still, run:

```
sudo nano /etc/dhcp/dhcpd.conf
```

At the very bottom of the file add :

```
host rpi3-node1 {
  hardware ethernet bc:4f:f5:8c:14:d9;
  fixed-address 192.168.8.101;
}
```

- **Save & Exit** (press **Ctrl+X** followed by **Y**)
- **REBOOT** the slave node we just powered on.

We should now be able to SSH into the slave node at the IP address we just assigned it.

How to Test SSH on Slave Node

On the master node, run:

```
ssh pi@ip-address-we-just-assigned
```


If we are able to login, then we've succeeded so far. Now we should make sure our Slave node's host name is updated.

In the SSH session, run:

```
sudo raspi-config
```

Advanced Options → Hostname → **rpi3-node1**

- **Save & Exit** (press **Ctrl+X** followed by **Y**)
- **EXIT** SSH Session by simply typing `exit` in the SSH terminal.

Repeat the Last 3 “How to” Sections for Each Slave Node before Continuing

How to Setup Passwordless SSH

Secure Shell (SSH) is the network protocol we will utilize to send commands to the slave nodes from the master node. This enables us to do everything from the master node and avoid changing keyboard, mouse, and monitor plugs over and over! This is not ideal because we'll still need to enter n (however many nodes you have) SSH commands into the master node terminal to make a change on all n nodes. We will further simplify this process later on by using *parallel-ssh* which will enable us to enter one command on the master node and have it executed on all slave nodes. But before we get there, we need to set up regular SSH first.

How to Verify Hostnames

The hostnames used for our cluster are:

- rpi3-node0
- rpi3-node1
- ...
- rpi3-node7

You can verify that your current Pi has an appropriate hostname by running in a terminal:

```
cat /etc/hostname
```

and see if you get the appropriate node hostname that we specified earlier.

How to Generate Master Node SSH Key

To make SSH connection/authorization faster and more secure, we will generate an SSH key for the master node and then register it as an authorized SSH key on each of the slave nodes. Then we remove the need for passing passwords when issuing SSH commands and increase the speed of our system.

You generate an SSH key on the master node by running the following in its terminal:

```
ssh-keygen -t rsa -C "pi@master-node0"
```

The `-t` in the above command is specifying the type of encryption protocol ("rsa" in this case).

The `-C` in the above command is just to add our own comment to the SSH key.

- You will be prompted for a file to save the key, just **leave it blank** by **pressing enter**.
- You will be prompted for a passphrase, again, just **leave it blank** by **pressing enter twice**.

We're leaving the passphrase blank to avoid having to type passwords when running parallel programs in our cluster.

Once the master node SSH key is generated, we need add it to the other slave nodes as an authorized key.

```
cat ~/.ssh/id_rsa.pub | ssh pi@ip_of_slave "mkdir .ssh;cat >>
.ssh/authorized_keys"
```

Repeat for all the slave nodes by replacing the IP address

Advanced: Alternatively, if you have many nodes to copy this to, you can automate this process with a bash script and create a txt file listing all the IP addresses of the nodes you wish to copy this to.

Bash Script:

```
while read ip; do
  ssh-copy-id -i ~/.ssh/id_rsa.pub pi@$ip
done < IPlistfile.txt
```

Once you've added the master node's SSH key as an authorized key to all slave nodes in the cluster, we need to create an SSH key for each slave node and add it to the master node's list of authorized SSH keys.

SSH into each of the slave nodes:

```
ssh pi@ip_of_slave
```

and create the ssh key using the same command we used previously with the master node:

```
ssh-keygen -t rsa -C "pi@slave-node#"
```

- You will be prompted for a file to save the key, just **leave it blank** by **pressing enter**.
- You will be prompted for a passphrase, again, just **leave it blank** by **pressing enter twice**.

Once ssh key is generated, the public key of each slave node needs to be added to the authorized key of the master node.

Using a similar command as previously done with the master node:

```
cat ~/.ssh/id_rsa.pub | ssh pi@ip_of_master "cat >> .ssh/authorized_keys"
```

Once the command finishes, exit the SSH session and SSH into the next slave node and

Repeat for all remaining slave nodes.

How to Send Files to Nodes

Once the passwordless SSH is setup and working, we can easily copy files to all nodes with Secure Copy (**SCP**). SCP is used to send files between networked nodes with a simple terminal command. A common use would be something like:

```
scp /src/path/fileToBeSent pi@ip-of-destination:/dest/file/path/
```

We will further simplify this process later on by using *parallel-scp* which will enable us to enter one command on the master node and have it copy a file to all slave nodes. Parallel-scp is packaged with Parallel-SSH, so when you install parallel-ssh, you'll automatically have parallel-scp.

How to Setup Parallel-SSH & Parallel-SCP

As we mentioned above, Parallel-SSH is an application that will allow us to issue commands to all slave nodes with one command from the master node. Parallel-SCP is installed by default when installing parallel-ssh. Parallel-SCP is the same idea, except you copy files to all slave nodes instead of issuing commands.

First we need to install parallel-ssh/scp on the master node.

Enter the following command in a terminal on the master node:

```
sudo apt-get install pssh
```

Let it install. Once it finishes, we just need to create a *host_file* which will include user names and IP addresses of all the nodes in the cluster. This will make issuing parallel-ssh/scp much simpler. Normally you have to pass each node's username and password in the command, but with a *host_file* we can just reference that instead.

Create a blank file in the home directory of the master node by entering in its terminal:

```
nano ~/slave_hosts
```

Type all of your slave nodes SSH login info in the pattern **[user@]host[:port]**

It should look very similar to this when done:

```
pi@192.168.8.101:22
pi@192.168.8.102:22
pi@192.168.8.103:22
pi@192.168.8.104:22
pi@192.168.8.105:22
pi@192.168.8.106:22
pi@192.168.8.107:22
```

- **Save & Exit** (press **Ctrl+X** followed by **Y**)

Now when we want to run a command (parallel-ssh) on all slave nodes, we can enter a simple command on the master node terminal:

```
parallel-ssh -i -h ~/slave_hosts echo "hello world"
```

When we want to copy a file (parallel-scp) to all slave nodes, we can enter a simple command on the master node terminal:

```
parallel-scp -v -h ~/slave_hosts /src/path/fileToBeSent /dest/file/path/
```

Appendix

How to Install MPICH (alternative to OpenMPI)

NOTE: Only Install OpenMPI **OR** MPICH, **NOT** both. We recommend OpenMPI, but if you experience any errors or bugs, try MPICH.

NOTE: If you install MPICH, you will need to uninstall mpi4py and install/compile from source with the instructions below

```
sudo apt-get install mpich
```

How to Install mpi4py (for MPICH installation)

Go to home directory.

```
cd ~
```

Download mpi4py tarball.

```
wget https://bitbucket.org/mpi4py/mpi4py/downloads/mpi4py-2.0.0.tar.gz
```

Update

```
sudo apt-get update --fix-missing
```

Unzip the file

```
tar xzf mpi4py-2.0.0.tar.gz
```

Navigate to the directory

```
cd mpi4py-2.0.0
```


Installation for Python 3

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

```
sudo python3 setup.py build --mpicc=/usr/bin/mpicc
```

```
sudo python3 setup.py install
```

Installation for Python 2

```
sudo apt-get install python-dev
```

```
sudo python setup.py build --mpicc=/usr/bin/mpicc
```

```
sudo python setup.py install
```

How to Use a Host File with MPI

If you plan on using MPI, you'll want to make a `host_file` that lists all of the node's IP addresses. This is very similar to what we did for `parallel-ssh`, but instead used for **mpirun** and will specify all nodes, including the master node.

Create a blank file in the home directory of the master node by entering in its terminal:

```
nano ~/mpi_hostfile
```

Type all of your node's IP addresses (including the master node's), one per line.

It should look very similar to this when done:

```
192.168.8.1
192.168.8.101
192.168.8.102
192.168.8.103
192.168.8.104
192.168.8.105
192.168.8.106
192.168.8.107
```

- **Save & Exit** (press **Ctrl+X** followed by **Y**)

Now when we want to run an mpi program (mpirun) on all nodes, we can enter a simple command on the master node terminal:

```
mpirun -n 8 -hostfile ~/mpi_hostfile ~/path/to/mpi_executable
```

How to Compile with OpenMP / MPI

If using without MPI, compile it with g++/gcc :

```
gcc -fopenmp -o my_executable my_source.c
```

If using with MPI, compile it with mpicc:

```
mpicc -fopenmp -o mpi_executable mpi_source.c
```

and then run it with the same mpirun commands as the others:

```
mpirun -n 8 -hostfile ~/mpi_hostfile ~/path/to/mpi_executable
```