WILLIAM PRETTY SECURITY INC

Project Gryphon

Initiated August 2016

Version 2.0

This document will outline the design of a RaspberryPi based Super Cluster

Revision Information

1.0 ……………………………………………………...…………. Draft Release

2.0 ……………………………………………………………..…Docker Release

Last Saved: Monday, July 16, 2018

Table of Contents:

Table of Figures:

Introduction:

The following document outlines the hardware and software required to create a Docker Swarm.

Hardware:

The hardware consists of four worker nodes and one manager. Each worker is a RaspberryPi Model 2B with 512MB of RAM. Each worker has a root file system installed on a 16GB uSD card. The manager system consists of a RaspberryPi Model 2 which has been configured to boot from a 128GB solid state drive. This drive contains the root file system for the RaspberryPi along with Rainbow tables used by the cracking software. It also stores the docker registry, from which the workers pull their image files.

The Pi's are networked together via an 8 port 10/100 Ethernet switch. Each RaspberryPi has a hard coded IP Address. The workers are assigned 192.168.10.121 thru 124 corresponding to nodes 1 thru 4, respectively. The Manager is assigned address 192.168.10.120 The switch also provides access to the internet via an in house LAN.

The manager is capable of being run headless via SSH. However it also is capable of being connected to an HDMI monitor, mouse and keyboard.

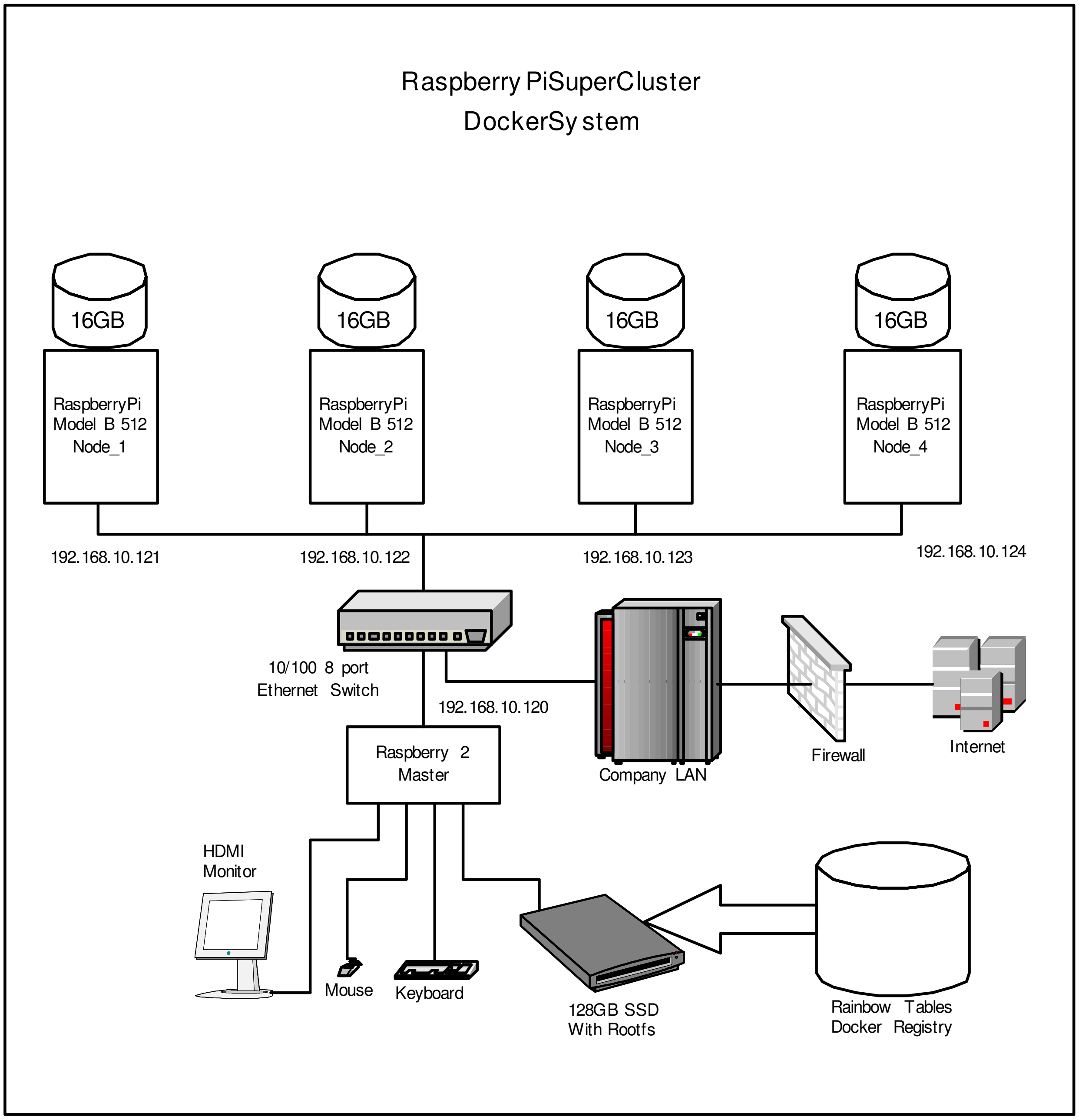


Figure 1 Block Diagram

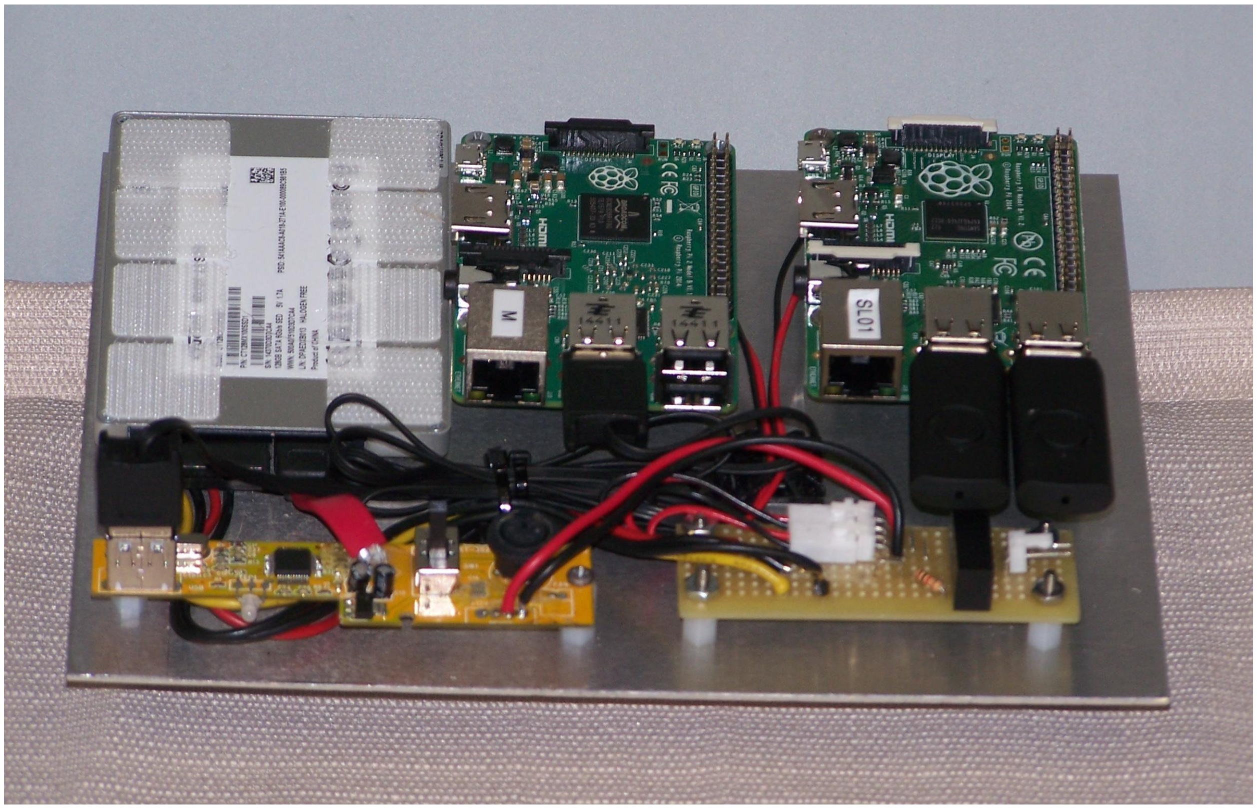


Figure 2 Manager and Node_1

Figure 2 above shows the sub-chassis which contains the Manager and the first worker node. The RaspberryPi in the center is the manager. To the left of it is the solid state drive. The drive is connected to the Pi using a SATA to USB adapter board. This board is located on the bottom left of figure 2.

This figure also shows two optional USB keys attached to node 1. These keys were configured as a type 1 RAID drive for a previous project.

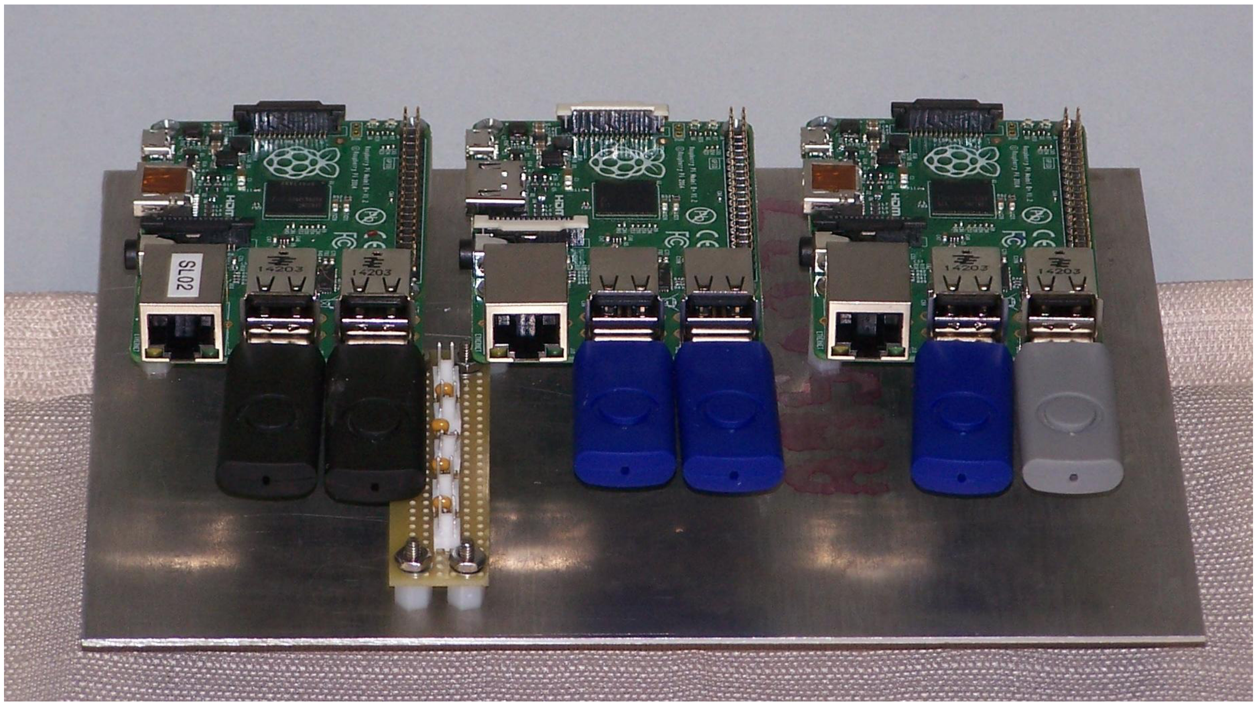


Figure 3 Nodes 2, 3 and 4

Figure 3 above shows nodes 2, 3 and 4 mounted on their own chassis. The small PCB is used to distribute power to the RaspberryPi's in the cluster. Future revisions of hardware may include a relay board capable of cycling power on the worker nodes.

Once again optional USB drives are shown connected to each node.

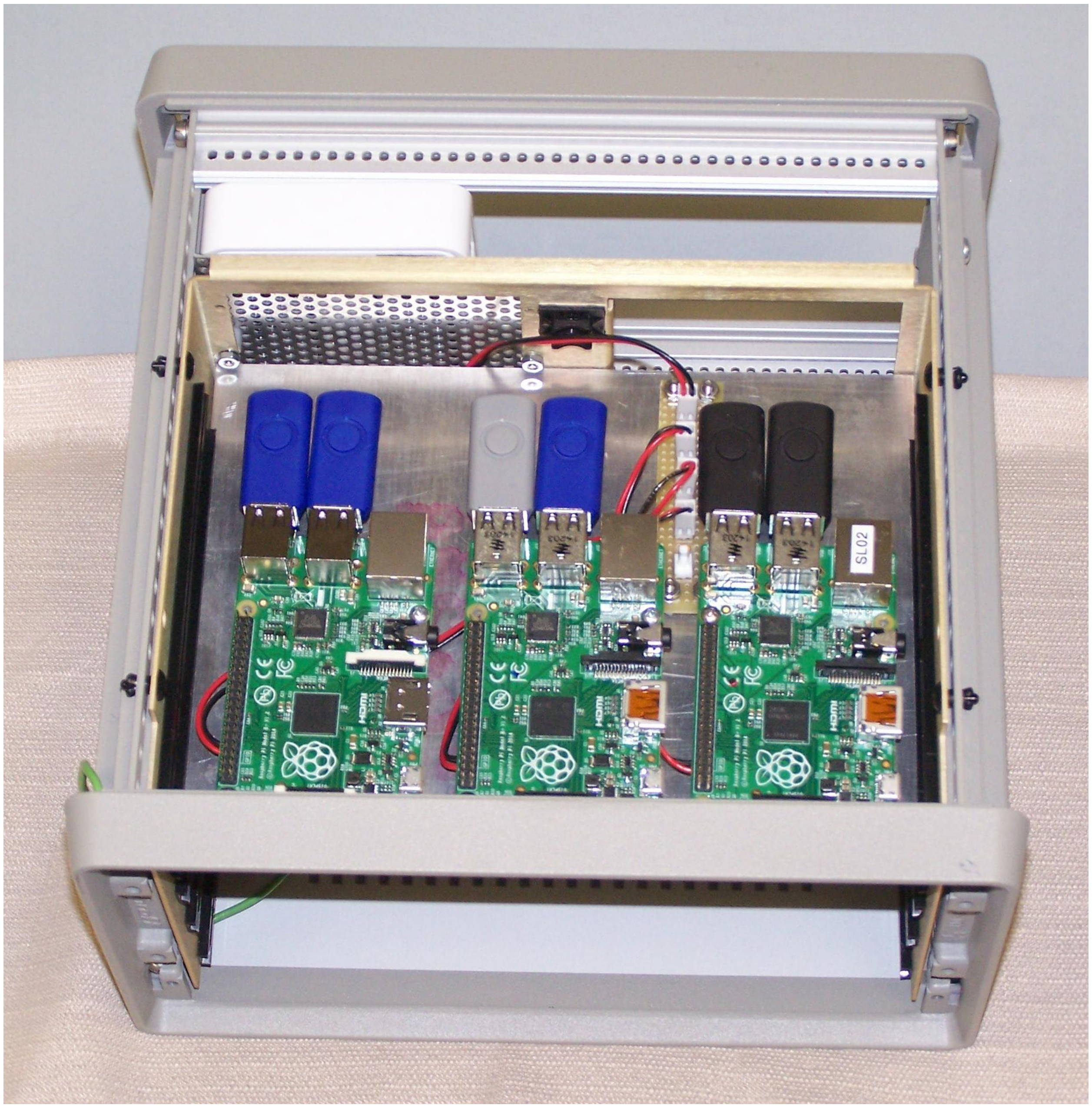


Figure 4 Bottom Deck Installed in Chassis

Figure 4 above shows how the lower sub-chassis is installed in the case. Note the power wiring is now in place and is connected to the distribution board. This 10/100 Ethernet switch was chosen because it is also powered by five volts. The switch is the white box at the left rear of the chassis.

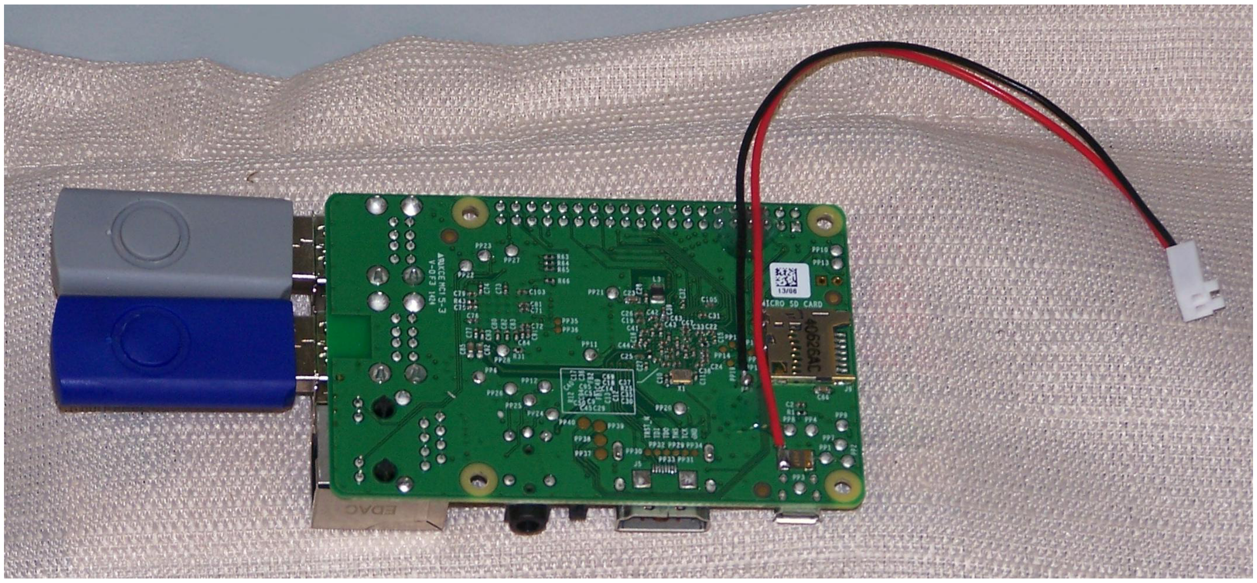


Figure 5 RaspberryPi Power Connection

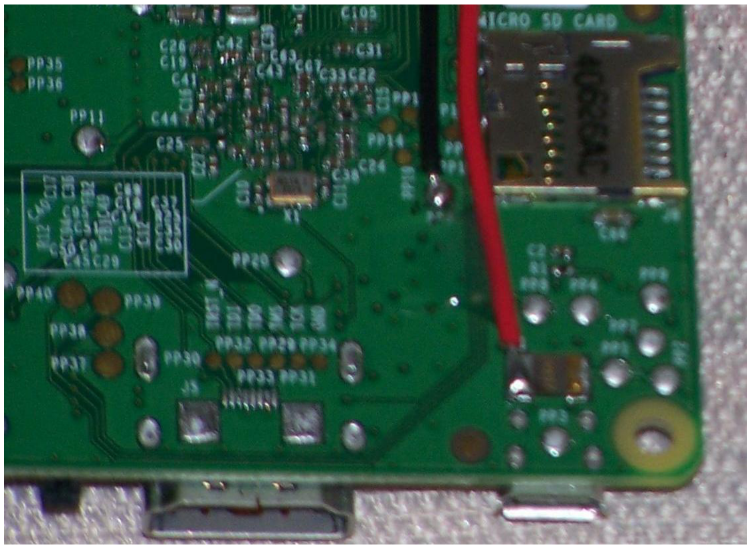


Figure 6 Power Connection Detail

Rather than use the mini USB connector for power, I installed my own power connections to the bottom of each board. The details of this are shown above.

Booting from an external USB HDD

Booting the Raspberry Pi up to use a USB HDD will give you extra storage capacity and also provide you with a device that is faster and more robust when it comes to accessing data repeatedly. The USB HDD should use external power, if possible as the max recommended current on the USB ports is 100 mA.

Follow the listed steps to set up your USB HDD and configure your SD card:

1. Plug in the USB device to one of your Raspberry Pi's USB ports.
2. Run fdisk

```
fdisk /dev/sda
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-250069679, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-250069679, default
250069679):

Created a new partition 1 of type 'Linux' and of size 119.2 GiB.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.


Command (m for help): p
Disk /dev/sda: 119.2 GiB, 128035676160 bytes, 250069680 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: 0x8d6be700

Device     Boot Start       End   Sectors   Size Id Type
/dev/sda1        2048 250069679 250067632 119.2G 83 Linux
```

Format sda1:
root@raspberrypi: mkfs.ext4 /dev/sda1

```
mke2fs 1.42.12 (29-Aug-2014)
Creating filesystem with 31258454 4k blocks and 7815168 inodes
Filesystem UUID: 7e9da99d-c4e7-48d8-9477-1479df428f45
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632,
2654208,
        4096000, 7962624, 11239424, 20480000, 23887872

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

After formatting the USB device mount it to a directory locally, for example:
mkdir /dev/usb
mount /dev/sda1 /dev/usb

You can use ls /dev/ | grep "sd" to list connected devices and then to mount the device use ***sudo mount /dev/sda1 ~/usb***, where sda1 is your USB device.
This will confirm the USB device is working.

Copy the image of your SD card root file system to the USB drive using the dd command, for example:
sudo dd bs=1M if=/dev/mmcblk0p2 of=/dev/sda1

```
7575+0 records in
7575+0 records out
7942963200 bytes (7.9 GB) copied, 430.608 s, 18.4 MB/s
root@Master:~#
```

We can now edit/create the current SD card's cmdline.txt file as follows:
nano /boot/cmdline.txt

Change the file to the following line:
dwc_otg.lpm_enable=0 console=ttyAMA0,115200
kgdboc=ttyAMA0,115200 console=tty1 root=/dev/sda1
rootfstype=ext4 elevator=deadline rootwait

Reboot the system.

Run the partition resize tool on sda1 using the following command:

resize2fs /dev/sda1

```
resize2fs 1.42.12 (29-Aug-2014)
Filesystem at /dev/sda1 is mounted on /; on-line resizing required
old_desc_blocks = 1, new_desc_blocks = 8
The filesystem on /dev/sda1 is now 31258454 (4k) blocks long.
```

Once it ends, edit the fstab file to include the following code:

nano /etc/fastab

```
proc            /proc           proc    defaults          0       0
/dev/mmcblk0p1  /boot           vfat    defaults          0       2
/dev/sda1       /               ext4    defaults,noatime  0       1 <- **
# a swapfile is not a swap partition, no line here
#   use  dphys-swapfile swap[on|off]  for that

** Change mmcblk0p2 -> sda1
```

Save the changes and reboot your Raspberry Pi.

Your Raspberry Pi is now set up to run from the external USB device. The boot loader will stay on the SD card; however your data is now stored on the HDD.

Setting up Master:

Batch Files:

Shutdown_all.sh *** FIX ME ***

```
#!/bin/bash

echo “Shutting Down Slaves”

#Slave01
ssh root@192.168.10.121 “shutdown now -h”

#Slave02
ssh root@192.168.10.122 “shutdown now -h”

#Slave03
ssh root@192.168.10.123 “shutdown now -h”

#Slave04
ssh root@192.168.10.124 “shutdown now -h”

echo “Slaves Shut Down”
echo “Shutting Down Master”
shutdown now -h
```

Change Desktop Background:

Background (.png) files located here:

/usr/share/raspberrypi-artwork/

Install Nmap

1) ***apt-get install nmap***

nmap –T4 –F <your_local_ip_range>

or

nmap –Pn <your_local_ip_range>

Example:
nmap –Pn 192.168.10.120-125

Install Docker

These instructions are based on the following link by Alex Ellis:
http://blog.alexellis.io/getting-started-with-docker-on-raspberry-pi/

1) Start the docker installer:
   curl -sSL get.docker.com | sh
2) Once the script has finished, set docker to auto-start:
   sudo systemctl enable docker
3) Either reboot the Pi, or enter the following to start the docker daemon:
   sudo systemctl start docker
4) Enable the docker client for user "pi":
   sudo usermod -aG docker pi
5) To test the installation:

```
root@Master:/usr/src# docker info
Containers: 0
 Running: 0
 Paused: 0
 Stopped: 0
Images: 2
Server Version: 1.12.1
Storage Driver: overlay
 Backing Filesystem: extfs
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host null overlay
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Security Options:
Kernel Version: 4.1.19-v7+
Operating System: Raspbian GNU/Linux 8 (jessie)
OSType: linux
Architecture: armv7l
CPUs: 4
Total Memory: 925.8 MiB
Name: Master
ID: FNWQ:4RKW:M6XN:IFH7:PA6J:47BF:2HPF:VPPD:HQ5Z:U6UZ:PHM3:VTGQ
Docker Root Dir: /var/lib/docker
Debug Mode (client): false
Debug Mode (server): false
Registry: https://index.docker.io/v1/
WARNING: No memory limit support
WARNING: No swap limit support
WARNING: No kernel memory limit support
WARNING: No oom kill disable support
WARNING: No cpu cfs quota support
WARNING: No cpu cfs period support
Insecure Registries:
 127.0.0.0/8
```

Building a new Docker Image

1) Create a "Dockerfile" make file: ***nano Dockerfile***

```
FROM resin/rpi-raspbian:jessie-20160831

RUN apt-get -q update && \
    apt-get -qy install \
    john
```

2) A Docker file is similar to a make file in C/C++
build -t crack . <- don't forget period
3) This will create an image file called "crack"

Testing the new Image

Enter: ***docker run –ti crack***

```
root@Master:~# docker run -ti crack

root@23dd83b1bfdc:/# john
Created directory: /root/.john
John the Ripper password cracker, version 1.8.0
Copyright (c) 1996-2013 by Solar Designer
Homepage: http://www.openwall.com/john/
.
.
.
```

Setup a Swarm

1) Initialize the swarm by entering: ***docker swarm init –advertise-addr <IP of master>*** and you will see the following message, although your token will be different. Copy the command to the clipboard.

```
root@Master:~# docker swarm init --advertise-addr 192.168.10.120
Swarm initialized: current node (d6rsdd3zmodoq76kbmvpxms55) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join \
    --token SWMTKN-1-0mme3h4lh2mkcjdrm6uy7ztwy8l4pguq6q6pshk8z3zv2jsniy-
aknz18cg7hdf2df4pat14sxeb \
    192.168.10.120:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and
follow the instructions.
```

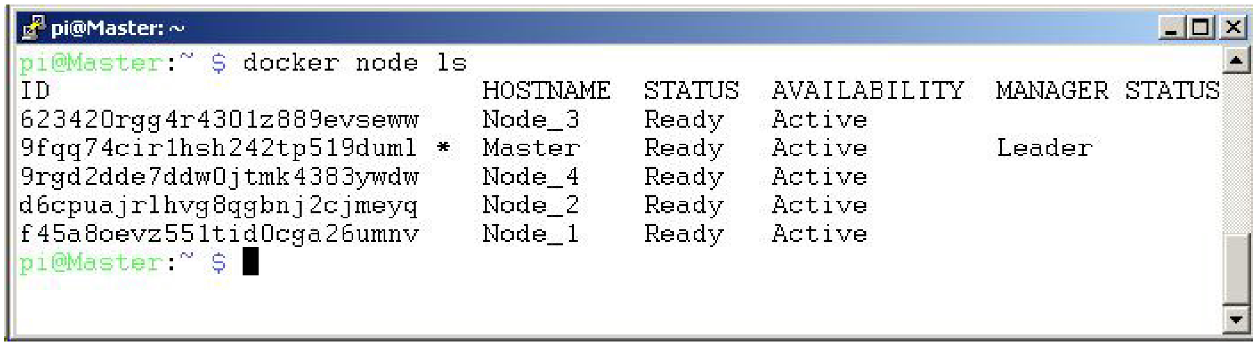
2) Enter: ***docker node ls*** to verify that the swarm is active.

```
root@Master:~# docker node ls
ID                           HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
bxf2nkdwqml13o9m8e5oztifp *  Master    Ready   Active        Leader
```

3) Using the command that you copied to the clipboard, add node_1 and node_2 to the swarm and recheck the swarm as seen below.

```
root@Master:~# docker node ls
ID                           HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
43mb3rcpcj87zok0iglkxdpnl    Node_2    Ready   Active
4dg362b4e6pq15risn8jsfigh    Node_1    Ready   Active
bxf2nkdwqml13o9m8e5oztifp *  Master    Ready   Active        Leader
```

4) Using the command from step 3 , add the final two nodes to the swarm.



```
pi@Master:~ $ docker node ls
ID                           HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
623420rgg4r4301z889evseww    Node_3    Ready   Active
9fqq74cir1hsh242tp519duml *  Master    Ready   Active        Leader
9rgd2dde7ddw0jtmk4383ywdw    Node_4    Ready   Active
d6cpuajrlhvg8qgbnj2cjmeyq    Node_2    Ready   Active
f45a8oevz551tid0cga26umnv    Node_1    Ready   Active
pi@Master:~ $
```

Figure 7 Five node swarm

Setting up a Registry

1) On each Pi edit /etc/systemd/system/docker.service.d/overlay.conf

Change this line:

```
`ExecStart=/usr/bin/dockerd --storage-driver overlay -H fd://`
```

to:

```
`ExecStart=/usr/bin/dockerd --storage-driver overlay -H fd:// --insecure-
registry 192.168.10.120:5000`
```

2) On each Pi edit /lib/systemd/system/docker.service
   Modify this line from: ExecStart=/usr/bin/dockerd -H fd://
   To: ExecStart=/usr/bin/dockerd -H fd:// --insecure-registry 192.168.10.120:5000

3) On the Master get the registry:
   ```
   docker run -p 5000:5000 -d alexellisio/registry-arm
   ```

4) On the Master tag the registry:
   ```
   docker tag num7 192.168.10.120:5000/num7
   ```

5) On the master push the image to the registry:
   ```
   docker push 192.168.10.120:5000/num7
   ```

6) On each slave, pull the image from the master registry:
   ```
   docker pull 192.168.10.120:5000/num7
   ```

Deploying a Service

1) Use the following command on the master:

```
docker service create --replicas=5 --name node7
192.168.10.120:5000/num7 rcracki_mt -l hash.txt -t 15 md5_numeric#1-
7_0_2400x400000_oxid#000.rt
```

Check that the service is running properly.

```
pi@Master:~ $ docker service ls
ID            NAME   REPLICAS  IMAGE                     COMMAND
41dmprq27s6m  node7  5/5       192.168.10.120:5000/num7  rcracki_mt -l
hash.txt -t 15 md5_numeric#1-7_0_2400x400000_oxid#000.rt

pi@Master:~ $ docker service inspect --pretty 41dmprq27s6m
ID:             41dmprq27s6mo287io0yc7lhz
Name:           node7
Mode:           Replicated
 Replicas:      5
Placement:
UpdateConfig:
 Parallelism:   1
 On failure:    pause
ContainerSpec:
 Image:         192.168.10.120:5000/num7
 Args:          rcracki_mt -l hash.txt -t 15 md5_numeric#1-
7_0_2400x400000_oxid#000.rt
Resources:
```

Setting up the First Slave:

1) Change the name of the Slave:
   Edit the hostname file: ***nano /etc/hostname***
   Change current name to "Node_1"
   Then: ***nano /etc/hosts*** and do the same.

2) Change the IP Address of the slave:

nano /etc/network/interfaces

```
auto eth0
iface eth0 inet static
address 192.168.10.12n <- the IP address of this slave
netmask 255.255.255.0
gateway 192.168.10.1
```

3) Add the insecure-registry flag to "docker.service":

nano /lib/systemd/system/docker.service

```
ADD: --insecure-registry 192.168.10.120:5000
```

```
ExecStart=/usr/bin/dockerd -H fd:// --insecure-registry 192.168.10.120:5000
```

4) Add insecure-registry flag to "overlay.conf":

nano /etc/systemd/system/docker.service.d/overlay.conf

```
ADD: --insecure-registry 192.168.10.120:5000

ExecStart=/usr/bin/dockerd --storage-driver overlay -H fd:// --
insecure-registry 192.168.10.120:5000
```

5) Restart the daemon and docker

systemctl daemon-reload && systemctl restart docker

6) Make sure the registry is available on the Master by "starting" the container if necessary.

7) Pull the Image from the registry on the Master:

docker pull 192.168.10.120:5000/num7

Cloning Slaves:

1) Use “Win32Disk Imager” to burn three more copies of the “Node_1.img” file.
2) Change the name of the Slave:
   nano /etc/hostname
   Change current name to “node_n” where “n” is the number of the slave you are creating.

3) Change the IP Address of the slave:

nano /etc/network/interfaces

```
auto eth0
iface eth0 inet static
address 192.168.10.12n <- the IP address of this slave
netmask 255.255.255.0
gateway 192.168.10.1
```

Appendix

Installing Rainbow Crack in a Container

1) Create a new container using the following command:
   docker run -ti --entrypoint /bin/bash resin/rpi-raspbian:jessie
2) Find the name of the new container:
   docker ps -a
3) Rename the container:
   docker rename <old_name> rcrack
4) Start the container:
   docker start rcrack
5) Open a bash shell inside the container:
   docker exec -it rcrack bash
6) From inside the container:
   apt-get update
   apt-get install build-essential
   apt-get install libssl-dev
   apt-get install nano
   exit
7) Copy the source directory from the host system to the container:
   docker cp /usr/src/rcracki_mt rcrack:/usr/src/rcracki_mt
8) Open a bash shell inside the container again:
   docker exec -it rcrack bash
9) From inside the container:
   cd /usr/src/rcracki_mt
   make clean
   make
10) Test the application:
   ./rcracki_mt
11) Install the application:
   make install

Creating a Rainbow Crack Image

1) Create the Dockerfile, using nano: ***nano Dockerfile***

```
FROM resin/rpi-raspbian:jessie-20160831
WORKDIR /usr/local/bin
COPY ./rcracki_mt /usr/local/bin/rcracki_mt
COPY ./hash.txt  /usr/local/bin/hash.txt
COPY ./md5_numeric#1-7_0_2400x400000_oxid#000.rt /usr/local/bin/md5_numeric#1-
7_0_2400x400000_oxid#000.rt
CMD ["rcracki_mt", "-l", "hash.txt", "-t", "15", "md5_numeric#1-_0_2400x400000_oxid#000.rt"]
```

2) Build the Image:

docker build -t num7 .

```
Sending build context to Docker daemon 1.294 GB
Step 1 : FROM resin/rpi-raspbian:jessie-20160831
 ---> 0c648c095aee
Step 2 : ADD ./rcracki_mt /usr/local/bin/rcracki_mt
 ---> Using cache
 ---> 57ffe81396bb
Step 3 : ADD ./hash.txt /usr/local/bin/hash.txt
 ---> Using cache
 ---> 15fc27116ea9
Step 4 : ADD ./md5_numeric#1-7_0_2400x400000_oxid#000.rt
/usr/local/bin/md5_numeric#1-7_0_2400x400000_oxid#000.rt
 ---> Using cache
 ---> 2bc37e8520a3
Step 5 : CMD rcracki_mt -l hash.txt -t  15 md5_numeric#1-
7_0_2400x400000_oxid#000.rt
 ---> Using cache
 ---> 5cb699f0a4a1
Successfully built 5cb699f0a4a1
```

3) Run the Dockerfile: