A Guide to Creating a Raspberry Pi Docker Cluster using Clusterhat.

This guide will allow you to utilize several open source technologies in Docker, Raspbian, and others to learn and implement container technologies. The low cost and ease of learning that Raspberry Pi SoC (system on a chip) boards represent, along with their widespread use in I.o.T., robotics, Data Center, and D.I.Y. maker spaces create an additional means to acquire enterprise grade knowledge without the high cost that traditional server and switches represent.

This General Overview will give the user the basic setup as I created the cluster in my office and home labs. Let's Get started!

Steps needed:

What is needed to run a clusterhat (at a minimum)?

1. 1x Controller Raspberry Pi (A+/B+/B2/B3)
2. 1-4x Raspberry Pi Zero (1.2/1.3/W)
3. 2-5x Micro SD memory card (8GB min)
4. Pi Power Supply (2-2.5A capable)
5. Network Cable / WiFi dongle / etc.

The ClusterHAT will work with most models of Raspberry Pi but please be aware the Raspberry Pi 3 may reduce the CPU speed when hot which may happen with high CPU usage with or without a HAT. A 20cm (or longer) USB cable (Type A plug to Micro-B plug) is required to connect the ClusterHAT to the Controller Pi USB port. The Controller and each Zero requires a Micro SD card (8GB or bigger recommended). A power supply capable of supplying 2 Amps is recommended (2.5A when using a Pi 3B or 3B+).

6. Purchase the hardware. I used a 3B+.

   Qty 1 3B+

7. https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/
8. Qty 1 clusterhat
   a. https://clusterhat.com/buy
      i. Each clusterhat package contains the following:

         1. 1x ClusterHAT PCB*
         2. 4x 12mm M2.5 standoff
         3. 8x 6mm M2.5 screw
         4. 4x stick-on plastic feet
         5. USB Cable*
9. Qty 4 Raspberry Pi Zeros

i. https://www.raspberrypi.org/products/raspberry-pi-zero/

10. Clusterhat case (optional)
    i. https://www.pishop.us/product/cluster-hat-case/
11. Assemble the hardware together.
    a. A general warning: Disconnect the Pi from all sources of power (USB Power/HDMI/USB Peripherals/etc.) before connecting the ClusterHAT.
        i. Qty 1 3b+ (or any approved controller)
        ii. Qty 1 clusterhat
        iii. Qty 4 Raspberry Pi Zeros
        iv. Case (if needed)
12. Download and install the Raspbian Operating Systems provided by the Clusterhat creator and developer.
    a. https://clusterctrl.com/setup-software
    b. After you've downloaded follow these next steps:
        i. Format SD Cards to Fat32 using whatever your comfortable with
        ii. use Etcher (https://etcher.io) and flash the controller to the Pi 3B or 4B or whatever you're using as a controller)
        iii. Official from Clusterhat says this:

            1. Follow the standard instructions to unzip and write each of the above images to the corresponding SD Card.
            2. Username: **pi**
               Password: **clusterctrl**

               We strongly advise changing the pi users password and resizing the filesystem using the raspi-config tool on your first login.
            3. SSH is no longer enabled by default on any of the images, to enable SSH you will need to create a file named "ssh" in the boot partition - see the Raspberry Pi blog entry for more details, this needs to be done on both Controller and Pi Zero images (from the Controller Pi the Pi Zeros can be accessed via the serial console when SSH is disabled)

13. Load Docker
    a. Connect via SSH to the controller or via command line of the controller to each of zero nodes and controller.
        i. Details to do this if you don't know how can be found here:
    b. Issue the following commands on *all Pi's* in this order:

```
a. curl -sSL https://get.docker.com | sh
b. sudo usermod -a -G docker pi
c. curl -s https://packagecloud.io/install/repositories/Hypriot/rpi/script.deb.sh | sudo bash
d. sudo apt-get install containerd.io=1.2.6-1
e. iptables -A FORWARD -i br0 -o br0 -j ACCEPT
```

i. This command will need to be entered after every boot or reboot
ii. I'm attempting to automate this in a script
    1. Edit: Add the following lines to crontab -e via CLI:
        a. @reboot sudo iptables -A FORWARD -i br0 -o br0 -j ACCEPT
        b. @reboot /sbin/clusterhat on
        c. @reboot /sbin/clusterhat fan on 20
        d. @reboot /sbin/clusterhat fan on 21

14. Configure Docker
    a. These commands are next:

```
a. docker swarm init
```

2. It should come back with a message similar to:

```
a. docker swarm join \ --token <random token docker generated
   > \ <controller ip>:<port>
```

```
```

Installing portainer for web based cluster management:

You may need to run this as a super user (IE sudo):

```
1. docker pull portainer/portainer
2. docker run -d -p 9000:9000 -v /var/run/docker.sock:/var/run/docker.sock --
   restart always --name portainer portainer/portainer -H unix:///var/run/doc
   ker.sock
```

Playing with Docker

I followed a guide online from a Dutch developer. His instructions worked well except for the "counter" commands. His guide can be found here. I've recreated it here as well.

Deploying containers

I'm going to deploy the last example on this page on my setup. It shows a 'real' application with a database (Redis) that is shared by two simple web applications. In that example we first need to create a network overlay that is used between the web app and the Redis database. We can then create the Redis and web app **services**.

These will take quite a long time to start so don't get worried if they seem 'stuck' for a while. Especially the first time you deploy something it will take quite a bit of time to download and unpack the Docker image. So we need to issue 3 commands:

```
$ sudo docker network create --driver overlay --subnet 20.0.14.0/24 armnet

$ sudo docker service create --name redis   --replicas=1 --network=armnet alexellis2/redis-arm
:v6

$ sudo docker service create --name counter --replicas=2 --network=armnet --publish 3000:3000
alexellis2/arm_redis_counter
```

In the counter service we use two replicas for now.

WARNING You need ARM specific images to deploy on your Raspberry. Deploying official non-ARM Docker images on your raspberry won't work!

After a while both services should be up and running:

```
$ docker service ls

ID            NAME     MODE        REPLICAS  IMAGE

3k3rcebdxpxp  counter  replicated  2/2       alexellis2/arm_redis_counter:latest

melg4lzwro15  redis    replicated  1/1       alexellis2/redis-arm:v6
```

Docker has, as we instructed, created two instances of the 'counter' service (a Node.js web application) and one Redis instance. We can check which nodes a service is running on:

```
$ docker service ps counter

ID            NAME       IMAGE                                NODE   DESIRED STATE  CURRENT ST
ATE           ERROR  PORTS

dmnw6re5h31v  counter.1  alexellis2/arm_redis_counter:latest  mikey  Running        Running 15
seconds ago
```

```
s7b7h12uruzy  counter.2  alexellis2/arm_redis_counter:latest  donny  Running        Running 12
seconds ago
```

We can now curl the web application and they both should increment and report the counter:

```
$ curl -4 localhost:3000/incr {"count":13}

$ curl -4 localhost:3000/incr {"count":14}
```

We can also scale the service to more workers if we want:

```
$ docker service scale counter=4

counter scaled to 4


$ docker service ls

ID            NAME     MODE        REPLICAS  IMAGE

3k3rcebdxpxp  counter  replicated  4/4       alexellis2/arm_redis_counter:latest

melg4lzwro15  redis    replicated  1/1       alexellis2/redis-arm:v6
```

We can also check on which nodes the counter service runs:

```
$ docker service ps counter

ID            NAME       IMAGE                                NODE   DESIRED STATE  CURRENT ST
ATE           ERROR  PORTS

dmnw6re5h31v  counter.1  alexellis2/arm_redis_counter:latest  mikey  Running        Running 2
minutes ago

s7b7h12uruzy  counter.2  alexellis2/arm_redis_counter:latest  donny  Running        Running 2
minutes ago

pap6i8ufk5dm  counter.3  alexellis2/arm_redis_counter:latest  leo    Running        Running 17
seconds ago

8x89pmd5epty  counter.4  alexellis2/arm_redis_counter:latest  raph   Running        Running 18
seconds ago
```

Nice! That they all happen to run on the four zero's (leo, donny, mickey and raph) is simply because of the order they were started in; docker swarm will by default pick the service with the least number of processes on it. If you want to control on which node a service gets started you can do so using for example labels.

TIP An easy way to restart all instances of a service is by scaling them to 0 and then back to the desired number of instances again.

Of course we can also completely delete a service:

```
$ docker service rm counter
counter

$ docker service ls
ID            NAME   MODE        REPLICAS  IMAGE
tw57a8c0ixk4  redis  replicated  1/1       alexellis2/redis-arm:v6
```

So this is all you need to be able to deploy a database backed web application on your Raspberry Pi! If you want to deploy a Java application you can use this image as a base and you can use this one for Python applications.

Bonus: Failover

So what happens when a node crashes? With a ClusterHat this is really easy and fun to demonstrate. Instead of yanking out a network cable I can just power down a zero on the command line. So to demo this I created the counter service on three nodes:

```
$ docker service ps counter
ID            NAME       IMAGE                                NODE   DESIRED STATE  CURRENT ST
ATE          ERROR  PORTS
0aaom01ttl0l  counter.1  alexellis2/arm_redis_counter:latest  raph   Running        Running 4
seconds ago
gjgnm04mcewg  counter.2  alexellis2/arm_redis_counter:latest  leo    Running        Running 1
second ago
q5dbgcc6x0c7  counter.3  alexellis2/arm_redis_counter:latest  mikey  Running        Running 6
seconds ago
```

It’s active on three of the four zero’s; Leo, Raph and Mikey. What happens if I kill Leo?

```
$ clusterhat off p1
```

```
Turning off P1


$ $ docker node ls
ID                            HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
08iueg6qv6h4964wf7ahvc4ov     donny     Ready   Active
4ktue0dbx9quqjpb7sucskg1f     raph      Ready   Active
6xb470pw6slr6maxzcusbsx7d *   splinter  Ready   Active        Leader
f4n5p4b5omih0wvlm5qiuswrs     leo       Down    Active
pvwt9odee16ctl2ui4fhiribo     mikey     Ready   Active


$ docker service ps counter
ID            NAME           IMAGE                                NODE   DESIRED STATE  CURREN
T STATE               ERROR  PORTS
0aaom01ttl0l  counter.1      alexellis2/arm_redis_counter:latest  raph   Running        Runnin
g about a minute ago
5e52pnlmlygg  counter.2      alexellis2/arm_redis_counter:latest  donny  Running        Runnin
g 15 seconds ago
gjgnm04mcewg   \_ counter.2  alexellis2/arm_redis_counter:latest  leo    Shutdown       Runnin
g about a minute ago
q5dbgcc6x0c7  counter.3      alexellis2/arm_redis_counter:latest  mikey  Running        Runnin
g about a minute ago
```

So Leo goes down (as shown in the node list) and the counter.2 process gets moved from Leo to Donny automatically! Now let's turn it back on:

```
$ clusterhat on p1
Turning on P1


docker node ls
ID                            HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
08iueg6qv6h4964wf7ahvc4ov     donny     Ready   Active
4ktue0dbx9quqjpb7sucskg1f     raph      Ready   Active
6xb470pw6slr6maxzcusbsx7d *   splinter  Ready   Active        Leader
f4n5p4b5omih0wvlm5qiuswrs     leo       Ready   Active
pvwt9odee16ctl2ui4fhiribo     mikey     Ready   Active
```

After a while Leo is back to Ready / Active again; fully automatically. The Docker daemon starts back up on boot, remembers the Swarm it was part of and reports itself as active (assuming it can find a master).