RPi Cluster - Swarm

Materials

- 1 USB power hub with 4 ports 2amp per port minimun.
- 4 Raspberry Pi 2B (1GB, no wifi)
- 4 SD cards minimun 8GB, I use 32GB here.
- 4 wifi dongles (for rpi2)
- 1 router (to be used later)

Research and reference

- Raspberry Pi 3 Super Computing Cluster Part 2 - Software Config <<--- follow this for networking 20170522
- Setup Kubernetes on a Raspberry Pi Cluster easily the official way! <<--- follow this 20170522
- How I setup a Raspberry Pi 3 Cluster Using The New Docker Swarm Mode In 29 Minutes
- Cluster computing on the Raspberry Pi with Kubernetes
- Docker Swarmmode Test Scenarios
- Get Started with Docker on Raspberry Pi has additional notes about create docker image and access GPIO.

Install Docker

Option 1 - Install Docker as Addon to Jessie

- create a SD with Jessie (2017-04-10-raspbian-jessie.img) on it.
- install docker

`$ curl -sSL https://get.docker.com | sh`

- add current user, pi, to docker group to avoid type `sudo` each time.
  `$ sudo usermod -aG docker pi` .
- auto start
  `$ sudo systemctl enable docker`

```
pi@rpi-01:~ $ sudo systemctl enable docker
Synchronizing state for docker.service with sysvinit using update-rc.d...
Executing /usr/sbin/update-rc.d docker defaults
Executing /usr/sbin/update-rc.d docker enable
pi@rpi-01:~ $
```

- this gives use rpi with
  - image 2017-04-10-raspbian-jessie.img
  - dynamic wifi (consider to move to static IP later)
  - docker

This requires installation of docker-machine and docker-compose separately. See procedures at the links below:

- docker end to end app, this covers building OS images for ARM and GPIO applications.

Option 2 - Install Hypriot (used here)

Hypriot provides a distribution, Hypriot, which has docker pre-installed.

Reference

- Docker on ARM Raspberry Pi
- [Guide] Installing Docker Swarm on HypriotOS

hostname.local does not work out of box.

```
HypriotOS/armv7: pirate@rpi-m1 in ~
$ ping rpi-s2.local -c5
ping: unknown host rpi-s2.local
HypriotOS/armv7: pirate@rpi-m1 in ~
$ hostnamectl
   Static hostname: rpi-m1
         Icon name: computer
           Chassis: n/a
        Machine ID: 9989a26f06984d6dbadc01770f018e3b
           Boot ID: f1c4d9d1856343c3a15fcd68b3ad59bd
  Operating System: Raspbian GNU/Linux 8 (jessie)
            Kernel: Linux 4.4.50-hypriotos-v7+
      Architecture: arm
HypriotOS/armv7: pirate@rpi-m1 in ~
```

We need to install mDNS service for example, `avahi-daemon` . There are other ways.

Use the utility program `flash` to download image and copy to SD

There is an utility flash to simple the process of copying image and setup configuration.

- get image from here
- default user “pirate” (password “hypriot”).
- create a config file: device-init.yaml as below;

```
hostname: rpi-m1    <<--- modify this for each node
wifi:
  interfaces:
    wlan0:
      ssid: "<ACCESS-POINT-ID>"
      password: "<PASSWORD>"
```

- download and create a SD with device config file; modify hostname for each node.

`flash -c device-init.yaml https://github.com/hypriot/image-builder-rpi/releases/download/v1.4.0/hypriotos-rpi-v1.4.0.img.zip`

- or copy it by following instruction here to clone image, then change node name or just re-download with different node name in.

```
➜  rpi2cluster git:(master) ✗ nmap -sP 192.168.1.0/24 | grep rpi-
Nmap scan report for rpi-s3 (192.168.1.81)
Nmap scan report for rpi-s2 (192.168.1.84)
Nmap scan report for rpi-s1 (192.168.1.86)
Nmap scan report for rpi-m1 (192.168.1.88)
```

- check docker installation.

```
➜  rpi2cluster git:(master) ✗ ssh pirate@192.168.1.88
The authenticity of host '192.168.1.88 (192.168.1.88)' can't be established.
ECDSA key fingerprint is SHA256:CMQc15FF7V/t7N4hfcTmS+/xIuQ00xFlWYezymsUE18.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.88' (ECDSA) to the list of known hosts.
pirate@192.168.1.88's password:

HypriotOS (Debian GNU/Linux 8)

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
HypriotOS/armv7: pirate@rpi-m1 in ~
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
HypriotOS/armv7: pirate@rpi-m1 in ~
$ docker -v
Docker version 17.03.0-ce, build 60ccb22
HypriotOS/armv7: pirate@rpi-m1 in ~
```

check `docker-compose` and `docker-machine`

```
HypriotOS/armv7: pirate@rpi-m1 in ~
$ docker-compose -v
docker-compose version 1.11.2, build dfed245
HypriotOS/armv7: pirate@rpi-m1 in ~
$ docker-machine -v
docker-machine version 0.9.0, build 15fd4c7
HypriotOS/armv7: pirate@rpi-m1 in ~
```

swarm needs to be installed separately,

```
$ swarm  <<--- this is wrong command, use `docker swarm init`
-bash: swarm: command not found
HypriotOS/armv7: pirate@rpi-m1 in ~
```

There is bug for ping,
`$ ping rpi-s1 ping: icmp open socket: Operation not permitted HypriotOS/armv7: pirate@rpi-m1 in ~ $ ping 192.168.1.86 ping: icmp open socket`

Create SSH Keys

There is some error about cloning, see here for solution.

- delete the ssh entry in `~/.ssh/known_hosts` before `ssh` login the node.

`ssh-keyscan -t rsa github.com >> ~/.ssh/known_hosts`

```
HypriotOS/armv7: pirate@rpi-m1 in ~
$ ssh-keygen -t rsa -C "pirate@rpi-m1"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/pirate/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/pirate/.ssh/id_rsa.
Your public key has been saved in /home/pirate/.ssh/id_rsa.pub.
The key fingerprint is:
c7:61:7a:96:68:2e:10:4e:08:3a:7f:f6:a6:63:4a:37 pirate@rpi-m1
The key's randomart image is:
+---[RSA 2048]----+
|.                |
|.. .             |
|o . o     o      |
| o o .   = o     |
|  . =   S *      |
|   o o o +       |
|  . E + .        |
| . .o+ .         |
|  .o..           |
+-----------------+
```

Copy SSH Key of Master Node (rpi-m1) to Workers (rpi-s1, rpi-s2, rpi-s3)

```
HypriotOS/armv7: pirate@rpi-m1 in ~
$ ssh-copy-id pirate@192.168.1.86
The authenticity of host '192.168.1.86 (192.168.1.86)' can't be established.
ECDSA key fingerprint is 3a:da:ef:3e:57:dd:dc:6a:44:8c:21:87:36:a6:5c:2b.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
pirate@192.168.1.86's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'pirate@192.168.1.86'"
and check to make sure that only the key(s) you wanted were added.
$
```

repeat it for rpi-s2, rpi-s3

Test SSH Key

```
HypriotOS/armv7: pirate@rpi-m1 in ~
$ ssh pirate@192.168.1.86

HypriotOS (Debian GNU/Linux 8)

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue May 16 20:36:41 2017 from waterlily
HypriotOS/armv7: pirate@rpi-s1 in ~
```

Works.

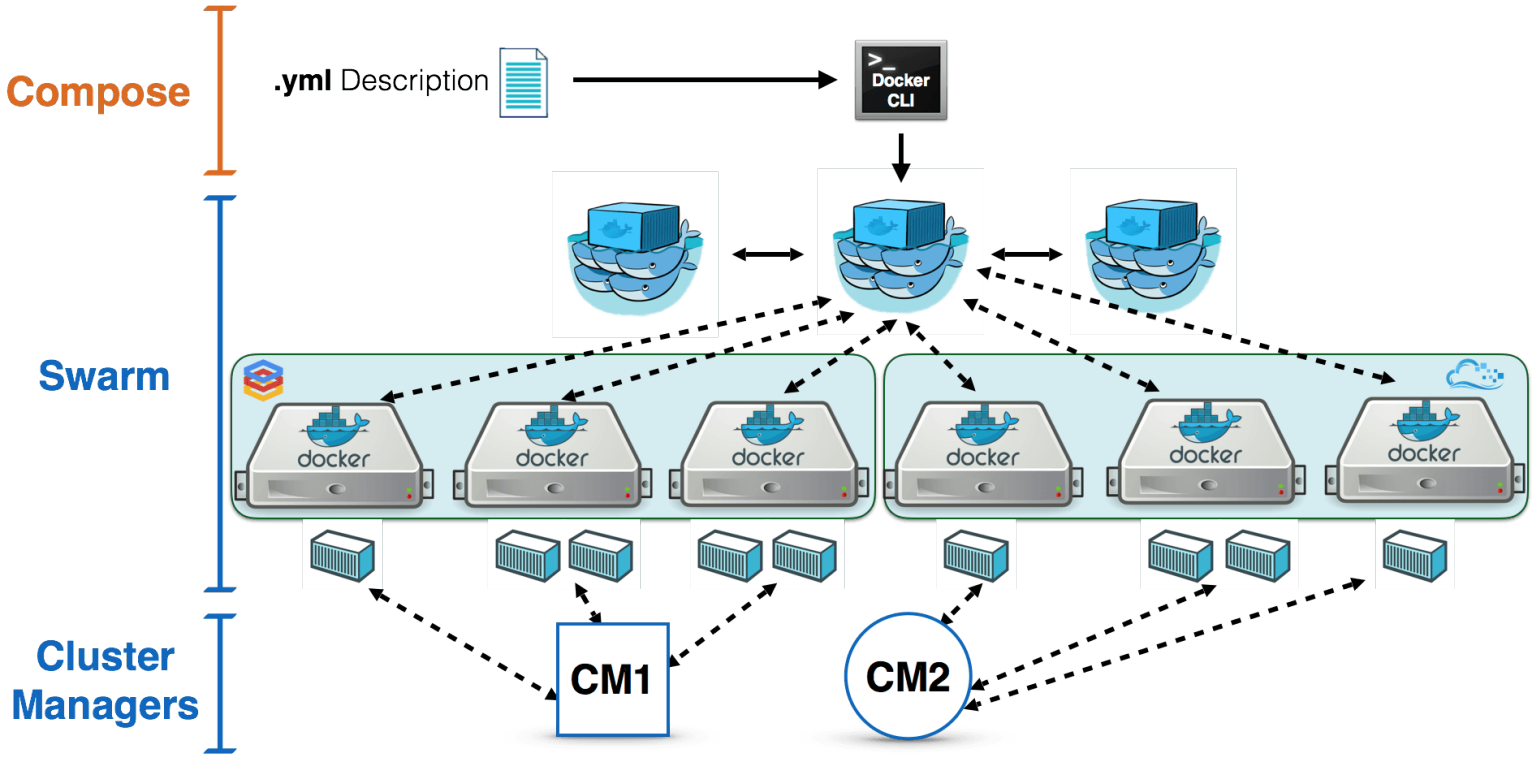
Create Docker Swarm Manager

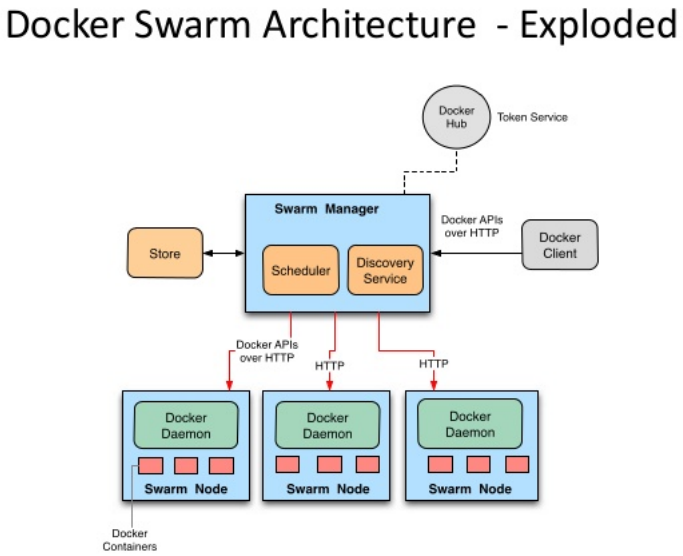
Reference

Docker Swarm Cheatsheet-1
Docker Swarm Cheatsheet-2

docker-machine can be cloud or local VM. Getting started with swarm mode
How to Create a Cluster of Docker Containers with Docker Swarm and DigitalOcean on Ubuntu 16.04

Architecture





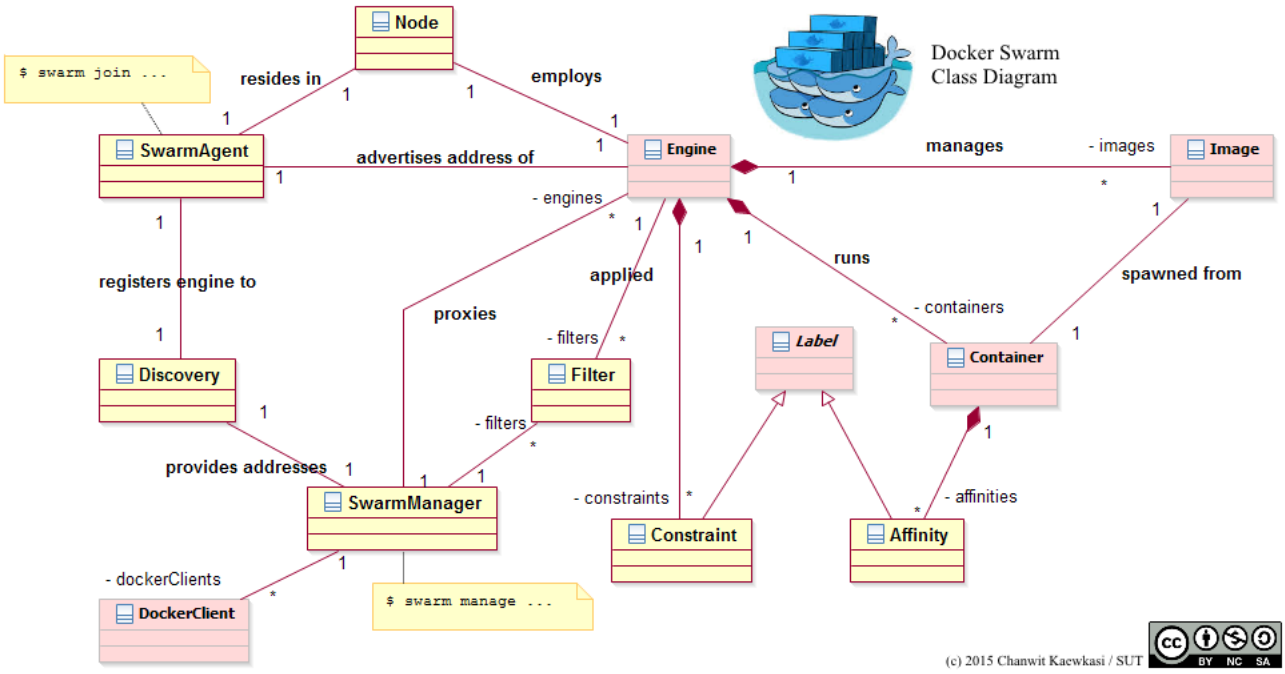
Features

highlights

- Cluster management
- Declarative service model: Docker Engine uses a declarative approach to let you define the desired state of the various services in your application stack.
- Scaling; task can be consider like POTs in Kubernetes.

- Maintain desired state via reconciliation.
- Multi-host networking: You can specify an overlay network for your services.
- Service discovery: Swarm manager nodes assign each service in the swarm a unique DNS name and load balances running containers.
- Load balancing: You can expose the ports for services to an external load balancer.
- Secure by default.
- Rolling updates the service.

UML



Initialize the Swarm Manager at Master Node

```
$ clear
HypriotOS/armv7: pirate@rpi-m1 in ~
$ docker swarm init
Swarm initialized: current node (g4vy8ewdl9ci8ct0lheqpxjbg) is now a manager.

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

    docker swarm join \
    --token SWMTKN-1-0obaaqul2si12iu0mxlxwmoa0ig1frdsnjtyalpxnk7clp434e-9l9sx9idajcloasfdw4a01g2y \
    192.168.1.88:2377

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

HypriotOS/armv7: pirate@rpi-m1 in ~
```

Add 3 Worker Nodes to Join Swarm

```
HypriotOS/armv7: pirate@rpi-s1 in ~
$ docker swarm join \
>     --token SWMTKN-1-0obaaqul2si12iu0mxlxwmoa0ig1frdsnjtyalpxnk7clp434e-9l9sx9idajcloasfdw4a01g2y \
>     192.168.1.88:2377
This node joined a swarm as a worker.
HypriotOS/armv7: pirate@rpi-s1 in ~
$
```

Review Swarm Info

```
HypriotOS/armv7: pirate@rpi-m1 in ~
$ docker info
Containers: 4
 Running: 0
 Paused: 0
 Stopped: 4
Images: 17
Server Version: 17.03.0-ce
Storage Driver: overlay2
 Backing Filesystem: extfs
 Supports d_type: true
 Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host macvlan null overlay
Swarm: active  <<-------------
 NodeID: g4vy8ewdl9ci8ct0lheqpxjbg
 Is Manager: true
 ClusterID: pyrvg5qzok221w7pfzetgcoxl
 Managers: 1
 Nodes: 4
 Orchestration:
  Task History Retention Limit: 5
 Raft:
  Snapshot Interval: 10000
  Number of Old Snapshots to Retain: 0
  Heartbeat Tick: 1
  Election Tick: 3
 Dispatcher:
  Heartbeat Period: 5 seconds
 CA Configuration:
  Expiry Duration: 3 months
 Node Address: 192.168.1.88
 Manager Addresses:
  192.168.1.88:2377
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 977c511eda0925a723debdc94d09459af49d082a
runc version: a01dafd48bc1c7cc12bdb01206f9fea7dd6feb70
init version: 949e6fa
Kernel Version: 4.4.50-hypriotos-v7+
Operating System: Raspbian GNU/Linux 8 (jessie)
OSType: linux
Architecture: armv7l
CPUs: 4
Total Memory: 861.9 MiB
Name: rpi-m1
ID: ZGHJ:GLKM:QIHZ:QKIZ:ZQAG:T7VW:STPB:3JX7:ESHQ:Y2ZQ:UDPA:NCRD
Docker Root Dir: /var/lib/docker
Debug Mode (client): false
Debug Mode (server): false
Registry: https://index.docker.io/v1/
Experimental: false
Insecure Registries:
 127.0.0.0/8
Live Restore Enabled: false
HypriotOS/armv7: pirate@rpi-m1 in ~
$
```

List of Nodes

```
HypriotOS/armv7: pirate@rpi-m1 in ~
$ docker node ls
ID                           HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
g4vy8ewdl9ci8ct0lheqpxjbg *  rpi-m1    Ready   Active        Leader
u5oktv61epxq0jui2oq1iaytj    rpi-s2    Ready   Active
y7umdundbo2zm5cljz1wd708i    rpi-s3    Ready   Active
ywf90mm103qd28r6a0upldtu1    rpi-s1    Ready   Active
HypriotOS/armv7: pirate@rpi-m1 in ~
```

Running Services in the Docker Swarm

No service yet.

```
HypriotOS/armv7: pirate@rpi-m1 in ~
$ docker node ps
ID  NAME  IMAGE  NODE  DESIRED STATE  CURRENT STATE  ERROR  PORTS
HypriotOS/armv7: pirate@rpi-m1 in ~
$ docker service ls
ID  NAME  MODE  REPLICAS  IMAGE
HypriotOS/armv7: pirate@rpi-m1 in ~
$
```

Deploy it.

```
HypriotOS/armv7: pirate@rpi-m1 in ~
$ docker service create -p 80:80 --name webserver nginx
0eyzq92p6oduhcu56arphi23o
HypriotOS/armv7: pirate@rpi-m1 in ~
$ docker node ps
ID            NAME         IMAGE         NODE    DESIRED STATE  CURRENT STATE             ERROR  PORTS
vy1gymsrar89  webserver.1  nginx:latest  rpi-m1  Running        Preparing 26 seconds ago
HypriotOS/armv7: pirate@rpi-m1 in ~
$ docker service ls
ID            NAME       MODE        REPLICAS  IMAGE
0eyzq92p6odu  webserver  replicated  0/1       nginx:latest
HypriotOS/armv7: pirate@rpi-m1 in ~
```

```
HypriotOS/armv7: pirate@rpi-m1 in ~
$ docker service ls
ID  NAME  MODE  REPLICAS  IMAGE
HypriotOS/armv7: pirate@rpi-m1 in ~
$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
647353628c83        bridge              bridge              local
698f31954f63        docker_gwbridge     bridge              local
ce2a385643c4        host                host                local
ifah9kmkvht2        ingress             overlay             swarm
a2800c300829        none                null                local
HypriotOS/armv7: pirate@rpi-m1 in ~
$ docker network create \
> --driver overlay \
> --subnet 10.10.1.0/24 \
> --opt encrypted \
> services
e3npyw8gejcyr5dkijqoyqwkp
HypriotOS/armv7: pirate@rpi-m1 in ~
$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
647353628c83        bridge              bridge              local
698f31954f63        docker_gwbridge     bridge              local
ce2a385643c4        host                host                local
ifah9kmkvht2        ingress             overlay             swarm
a2800c300829        none                null                local
e3npyw8gejcy        services            overlay             swarm
HypriotOS/armv7: pirate@rpi-m1 in ~
```

why the service is not running?? ``` HypriotOS/armv7: pirate@rpi-m1 in ~ $ docker service create \

> --replicas 2 \ --name nginx \ --network services \ --publish 80:80 \ nginx y50id9yg99z3ehada3mhgga1b HypriotOS/armv7: pirate@rpi-m1 in ~ $ docker service ls ID NAME MODE REPLICAS IMAGE y50id9yg99z3 nginx replicated 0/2 nginx:latest HypriotOS/armv7: pirate@rpi-m1 in ~ ```

do it again.

```
HypriotOS/armv7: pirate@rpi-m1 in ~ $ docker node ls ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS g4vy8ewdl9ci8ct0lheqpxjbg * rpi-m1 Ready
```

Service is running at rpi-s2.

```
HypriotOS/armv7: pirate@rpi-m1 in ~
$ docker service ls
ID            NAME  MODE        REPLICAS  IMAGE
r25uzlw259zs  ping  replicated  1/1       hypriot/rpi-alpine-scratch:latest
ttsu5t4sml64  web   replicated  0/2       nginx:latest
HypriotOS/armv7: pirate@rpi-m1 in ~
$
```

```
HypriotOS/armv7: pirate@rpi-s2 in ~
$ docker ps
CONTAINER ID        IMAGE                                                                                                  COMMAND             CREATED
4b22f39ebc1f        hypriot/rpi-alpine-scratch@sha256:708171e6a1bd7c60a0ec9a5657900ada854bf14623be053f5865f918e0e2691c   "ping 8.8.8.8"      12 minutes ago
HypriotOS/armv7: pirate@rpi-s2 in ~
```

It is no at rpi-s3 or rpi-s1. scale up to 3;

```
HypriotOS/armv7: pirate@rpi-m1 in ~
$ docker service update --replicas 3 ping
ping
HypriotOS/armv7: pirate@rpi-m1 in ~
$ docker service ls
ID            NAME  MODE        REPLICAS  IMAGE
r25uzlw259zs  ping  replicated  1/3       hypriot/rpi-alpine-scratch:latest
ttsu5t4sml64  web   replicated  0/2       nginx:latest
HypriotOS/armv7: pirate@rpi-m1 in ~
$
```

still

```
HypriotOS/armv7: pirate@rpi-s2 in ~
$ docker ps
CONTAINER ID        IMAGE                                                                                                  COMMAND             CREATED
4b22f39ebc1f        hypriot/rpi-alpine-scratch@sha256:708171e6a1bd7c60a0ec9a5657900ada854bf14623be053f5865f918e0e2691c   "ping 8.8.8.8"      22 minutes ago
HypriotOS/armv7: pirate@rpi-s2 in ~
$
```

Monitoring

- Visualize your Raspberry Pi containers with Portainer or UI for Docker
- Deploying an IoT Swarm with Docker Machine

Physical IO

- Let's get physical with Docker on the Raspberry Pi
- Wiring Pi - GPIO Interface library for the Raspberry Pi

Summary

Cheat Sheet

Docker Cheat Sheet

Initialize swarm mode and listen on a specific interface
```
docker swarm init --advertise-addr 10.1.0.2
```

Join an existing swarm as a manager node
```
docker swarm join --token <manager-token> 10.1.0.2:2377
```

Join an existing swarm as a worker node
```
docker swarm join --token <worker-token> 10.1.0.2:2377
```

List the nodes participating in a swarm
```
docker node ls
```

Create a service from an image exposed on a specific port and deploy 3 instances
```
docker service create --replicas 3 -p 80:80 --name web nginx
```

List the services running in a swarm
```
docker service ls
```

Scale a service
```
docker service scale web=5
```

List the tasks of a service
```
docker service tasks web
```

BUILD

Build an image from the Dockerfile in the current directory and tag the image
```
docker build -t myapp:1.0 .
```

List all images that are locally stored with the Docker engine
```
docker images
```

Delete an image from the local image store
```
docker rmi alpine:3.4
```

SHIP

Pull an image from a registry
```
docker pull alpine:3.4
```

Retag a local image with a new image name and tag
```
docker tag alpine:3.4 myrepo/myalpine:3.4
```

Log in to a registry (the Docker Hub by default)
```
docker login my.registry.com:8000
```

Push an image to a registry
```
docker push myrepo/myalpine:3.4
```

RUN

```
docker run
          --rm remove container automatically after it exits
           -it connect the container to terminal
    --name web name the container
     -p 5000:80 expose port 5000 externally and map to port 80
-v ~/dev:/code create a host mapped volume inside the container
    alpine:3.4 the image from which the container is instantiated
       /bin/sh the command to run inside the container
```

Stop a running container through SIGTERM
```
docker stop web
```

Stop a running container through SIGKILL
```
docker kill web
```

Create an overlay network and specify a subnet
```
docker network create --subnet 10.1.0.0/24
--gateway 10.1.0.1 -d overlay mynet
```

List the networks
```
docker network ls
```

List the running containers
```
docker ps
```

Delete all running and stopped containers
```
docker rm -f $(docker ps -aq)
```

Create a new bash process inside the container and connect it to the terminal
```
docker exec -it web bash
```

Print the last 100 lines of a container's logs
```
docker logs --tail 100 web
```

CLI

swarm init swarm join service create service inspect service ls service rm service scale service ps service update