Docker Swarm

To use Docker in swarm mode, install Docker. See installation instructions for all operating systems and platforms.

Current versions of Docker include *swarm mode* for natively managing a cluster of Docker Engines called a *swarm*. Use the Docker CLI to create a swarm, deploy application services to a swarm, and manage swarm behavior.

If you are using a Docker version prior to `1.12.0`, you can use standalone swarm, but we recommend updating

Why we need swarm :

- Cluster of docker machines/hosts
- Cluster is a collection of 2 or more nodes/machines/servers
- High availability
- Reliability
- Fail over
- 99.999% high availability
- < 5 min downtime in a year

Help Link : https://docs.docker.com/engine/reference/commandline/swarm_init/#extended-description

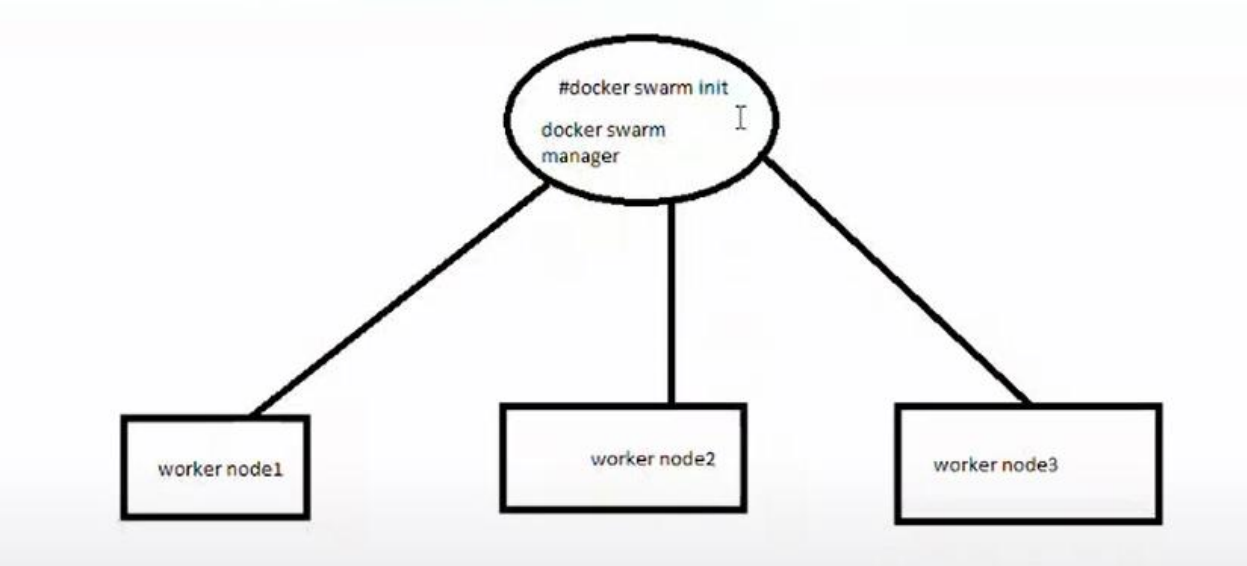
Suppose we have 3 different operating systems and on top of those docker engines has been installed.

So for creating a container what we will do means we will do login one of any docker engine and we will try to create a container. When ever our systems are running out of memory we will not get notify till the moment we login.

SO that is why we are going to maintain central server to manage all our docker engines.

So we just integrated all 3 docker engines with another docker engine called docker swarm.

Now instead of creating containers individually we are going to create containers from the central server.



First we need to have more than docker engines installed systems.

Then we are going to make one system of docker engine as a docker swarm to manage all our remaining engines.

First once logged into our first docker engine we have to check wether swarm is enabled or not by using below command

docker info

```
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 773c489c9c1b21a6d78b5c538cd395416ec50f88
runc version: 4fc53a81fb7c994640722ac585fa9ca548971871
init version: 949e6fa
```

Now we can intiate our first docker engine as a docker swarm using below command.

docker swarm init

```
root@ip-172-31-10-11:~# docker swarm init
Swarm initialized: current node (nf6qnegpohks3funedgrp0ii8) is now a manager.

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

    docker swarm join --token SWMTKN-1-0wtj02mw6ypkhrtfcutfozplaughbjo1c372odmekd3m908cgs-acku9bcs6pe9i0tmygtlix
72n 172.31.10.11:2377

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

root@ip-172-31-10-11:~#
```

Now check for swarm info in our docker engine

docker info

```
Swarm: active
 NodeID: nf6qnegpohks3funedgrp0ii8
 Is Manager: true
 ClusterID: m95jyx7lncxzz6f8up5st83pr
 Managers: 1
 Nodes: 1
 Orchestration:
  Task History Retention Limit: 5
 Raft:
  Snapshot Interval: 10000
  Number of Old Snapshots to Retain: 0
  Heartbeat Tick: 1
  Election Tick: 10
 Dispatcher:
```

If you want to make one more engine as a manager or if we want to attach any workers like nodes to this running manager we have to execute these following commands at their terminals.

```
root@ip-172-31-10-11:~# docker swarm join-token manager
To add a manager to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-0wtj02mw6ypkhrtfcutfozplaughbjo1c372odmekd3m908cgs-1mv4van9xzkqzx5r7x8wwn
1d0 172.31.10.11:2377

root@ip-172-31-10-11:~# docker swarm join-token worker
To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-0wtj02mw6ypkhrtfcutfozplaughbjo1c372odmekd3m908cgs-acku9bcs6pe9i0tmygtlix
72n 172.31.10.11:2377

root@ip-172-31-10-11:~#
```

Why we need one more manager means we are high availability reg managers of docker swarm.

Use case:

Launch 3 ubuntu 16.04 servers on ec2 instances

3 servers : Just create physical servers

1. Manager
2. Worker1
3. Worker2

- ❖ Cluster of 3 machines
- ❖ Create NGINX web server --- http://machine:80 --→ Access the website
- ❖ Scale up ---- how many containers are created
- ❖ Shut down of one node

Install docker on 3 ubuntu machines using the below link

https://docs.docker.com/install/linux/docker-ce/ubuntu/#set-up-the-repository

Then name it is one server as manager and remaining 2 as worker1 and worker2

Then go to Manager node and type below command to create swarm on manager and as a manager. And Manager node can act as manager as well as node.

docker swarm init

```
root@ip-172-31-2-217:~# docker swarm init
Swarm initialized: current node (4hu4k42io67hdpg2umv5tqjbo) is now a manager.

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

    docker swarm join --token SWMTKN-1-5h3fv2hp4xr1351x0fuse3dknmlw0y6ync8426w3w
2md88tt1z-77osibckohxnd1lodfe84r3qq 172.31.2.217:2377

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

root@ip-172-31-2-217:~# docker info
```

Then login remaining 2 nodes with root user and type docker swarm join line in above output . That means we are going to make remaining 2 nodes are workers to manager node.

And then verify on manager node using below command

docker info

```
Swarm: active
 NodeID: 4hu4k42io67hdpg2umv5tqjbo
 Is Manager: true
 ClusterID: oo57m9e5gb9zadpecria53ziz
 Managers: 1
 Nodes: 2
```

Now pulling one image from docker hub to create nginx web server

docker pull sixeyed/docker-swarm-walkthrough

docker service create --name website --publish 80:80 sixeyed/docker-swarm-walkthrough

Now we can verify on manager node process is running or not using

docker ps

It has to show one process is running

But when you go to worker nodes and type same ps command you are unable to see running process

But when we go to browser and type all 3 servers ip addresses same nginx server can be available that's called clustering containers.

make it more scalable:

scale up to 10 instances:

docker service inspect website --pretty

docker service update --replicas 20 website

Then just check on all termnals below command to check for running containers

docker ps | grep six | wc –l

3 servers can be shared all running 20 conatiners like below

```
root@ip-172-31-9-164:~# docker ps | grep six | wc -l
6
root@ip-172-31-9-164:~# docker ps | grep six | wc -l
6
root@ip-172-31-9-164:~#
```

```
root@ip-172-31-14-172:~# docker ps | grep six | wc -l
7
root@ip-172-31-14-172:~# docker ps | grep six | wc -l
7
root@ip-172-31-14-172:~#
```

```
root@ip-172-31-2-217:~# docker ps | grep six | wc -l
7
root@ip-172-31-2-217:~# docker ps | grep six | wc -l
7
root@ip-172-31-2-217:~#
```

containers can run on manager or workers

Now if you want to move containers from one server to another server we can remove worker from cluster so that those conatiners can be moved and distributed to remaining 2 servers.

```
root@ip-172-31-14-172:~# docker ps | grep six | wc -l
7
root@ip-172-31-14-172:~# docker swarm leave
Node left the swarm.
root@ip-172-31-14-172:~#
```

```
root@ip-172-31-9-164:~# docker ps | grep six | wc -l
6
root@ip-172-31-9-164:~# docker ps | grep six | wc -l
10
root@ip-172-31-9-164:~#
root@ip-172-31-2-217:~# docker ps | grep six | wc -l
7
root@ip-172-31-2-217:~# docker ps | grep six | wc -l
10
root@ip-172-31-2-217:~#
```

Important command in docker swarm :

Swarm mode CLI commands

Explore swarm mode CLI commands

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

That’s all about Docker Swarm and same like Kuberentes we will discuss tomorrow