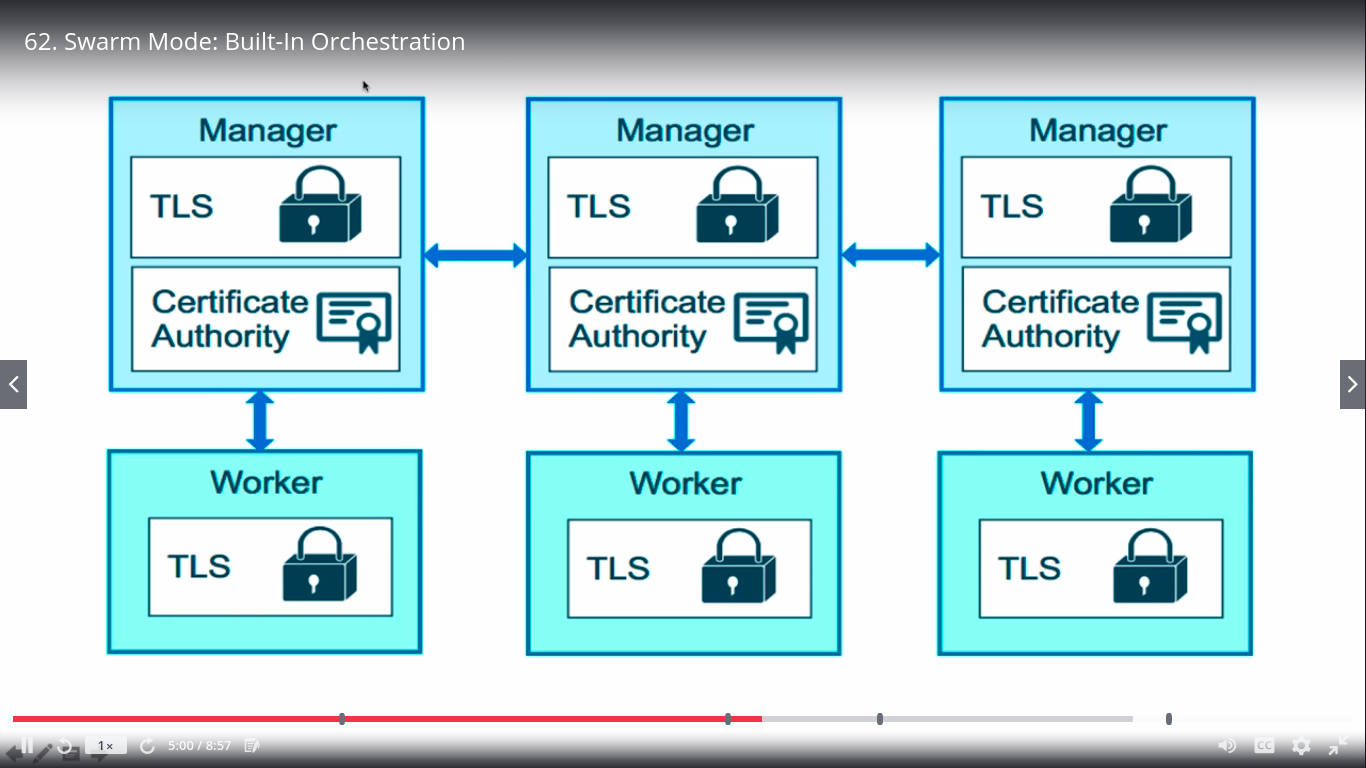
# Docker Swarm

What is docker Swarm .?

A Docker Swarm is a group of either physical or virtual machines that are running the Docker application and that have been configured to join together in a cluster. Once a group of machines have been clustered together, you can still run the Docker commands that you're used to, but they will now be carried out by the machines in your cluster. The activities of the cluster are controlled by a swarm manager, and machines that have joined the cluster are referred to as nodes.

Docker swarm is a container orchestration tool, meaning that it allows the user to manage multiple containers deployed across multiple host machines.

Docker swarm is not enabled by default it has to be enabled.



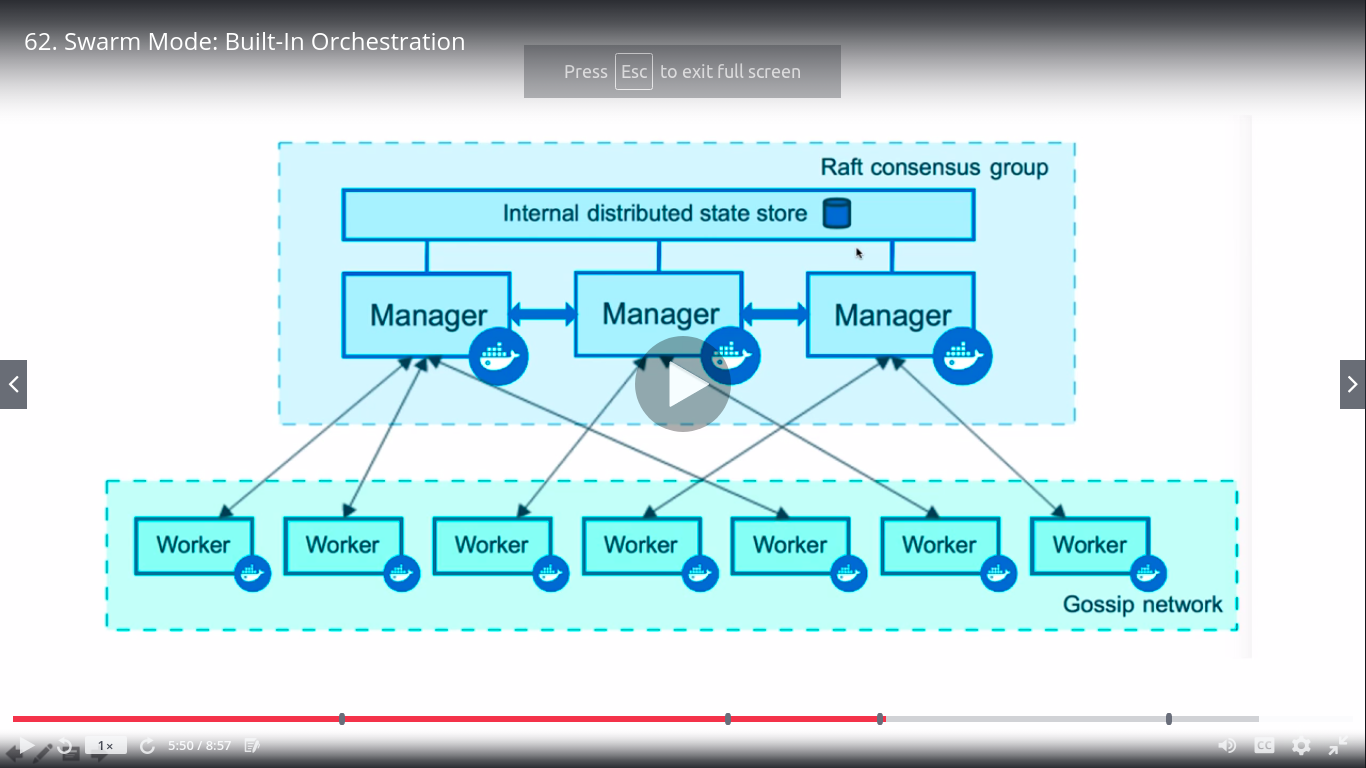
Blue boxes on top are called manager nodes, and they they actually have a db locally on them called a raft Database. It stores the config to have the authority to get inside a swarm

These are 3 different managers added to the swarm they all keep a copy of the DB.

Below in green we have worker nodes.

Each one of them is A PHYSICAL MACHINE or a virtual machine.

Consider a below image of more complicated scenario:



Where all managers share a raft database and manage worker nodes in the swarm cluster

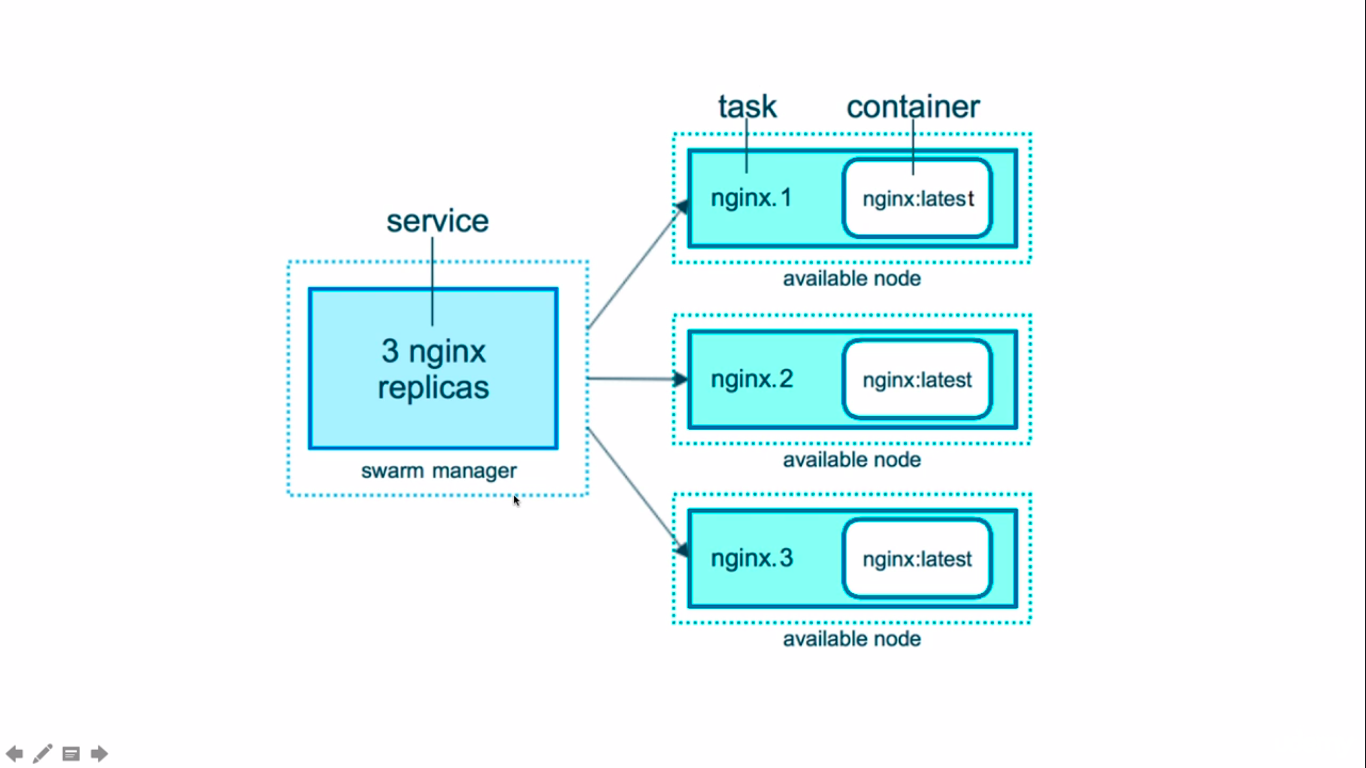
Managers themselves are also workers and also workers can be promoted or demoted.

With this concept of swarm of managers,

With docker run command we can only deploy one container , but that one container dont have concept of scale out and scale in to achieve that we need new commands , this is where service command comes to picture.

Service command allows to run containers with replicas, where all 3 containers run the same services and we can scale out one container to update and redeploy , this will give us 0 down-time deployment.(it replaces docker run command)

In docker services , there are multiple tasks associated to it and each of task run containers.



Docker swarm is inactive by default: to view that enter **docker info**

Below will be output:

Client:

Debug Mode: false

Server:

Containers: 3

Running: 0

Paused: 0

Stopped: 3

Images: 22

Server Version: 19.03.6

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 ipvlan macvlan null overlay

Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog

**Swarm: inactive**

Runtimes: runc

Default Runtime: runc

Init Binary: docker-init

containerd version: 7ad184331fa3e55e52b890ea95e65ba581ae3429

runc version: dc9208a3303feef5b3839f4323d9beb36df0a9dd

init version: fec3683

Security Options:

apparmor

seccomp

Profile: default

Kernel Version: 5.0.0-38-generic

Operating System: Ubuntu 19.04

OSType: linux

Architecture: x86\_64

CPUs: 4

Total Memory: 7.647GiB

Name: ubuntu

ID: MXXE:TVU2:JEI5:MJFI:LO5O:HBXY:YGAE:YK7B:CEEB:OHDG:GEVK:W2WP

Docker Root Dir: /var/lib/docker

Debug Mode: false

Username: skarwa4491

Registry: https://index.docker.io/v1/

Labels:

Experimental: false

Insecure Registries:

127.0.0.0/8

Live Restore Enabled: false

WARNING: No swap limit support

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

To activate the swarm : we can say

“Docker swarm init”

**output:**

Swarm initialized: current node (znhfi409kdsa1jb60zijobob6) is now a manager.

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

docker swarm join --token SWMTKN-1-62tfxycawa7zeam2tpm1eg1q38a940mdw5nkr5jxyf9h76lgpc-esi54zwwsn0xbv9m95pvmtwzh 192.168.43.207:2377

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

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

And then if we say “docker info”

**output:**

Client:

Debug Mode: false

Server:

Containers: 3

Running: 0

Paused: 0

Stopped: 3

Images: 22

Server Version: 19.03.6

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 ipvlan macvlan null overlay

Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog

**Swarm: active**

Runtimes: runc

Default Runtime: runc

Init Binary: docker-init

containerd version: 7ad184331fa3e55e52b890ea95e65ba581ae3429

runc version: dc9208a3303feef5b3839f4323d9beb36df0a9dd

init version: fec3683

Security Options:

apparmor

seccomp

Profile: default

Kernel Version: 5.0.0-38-generic

Operating System: Ubuntu 19.04

OSType: linux

Architecture: x86\_64

CPUs: 4

Total Memory: 7.647GiB

Name: ubuntu

ID: MXXE:TVU2:JEI5:MJFI:LO5O:HBXY:YGAE:YK7B:CEEB:OHDG:GEVK:W2WP

Docker Root Dir: /var/lib/docker

Debug Mode: false

Username: skarwa4491

Registry: https://index.docker.io/v1/

Labels:

Experimental: false

Insecure Registries:

127.0.0.0/8

Live Restore Enabled: false

WARNING: No swap limit support