[day-11] {Docker_swarm, service, stack.} (1)

Kubernetes Into & Installation

↳ Docker Compo → single point failure. (System crash sing point failure)

↳ System crash - No app. available.

{→ high availability
→ workload manage}

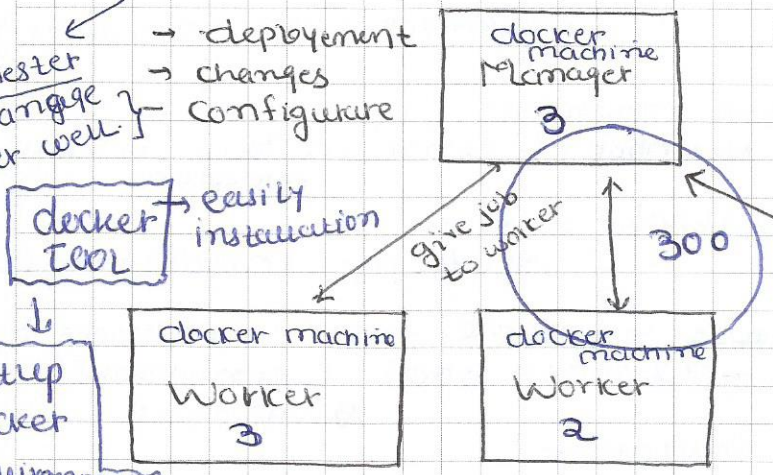
Docker Swarm - a cluster (group of machine)
↳ which work together

{orchester to manage docker well.} → deployement → changes → configurare

(multiple machine)

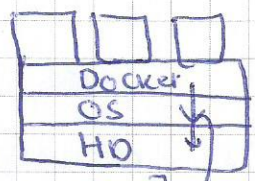
(cluster of Nodes)

(orchester) (Container orchestration)



12

- works job



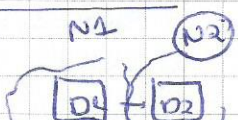
[System call] [OS kernel]

(multiple machine working)

→ manger knows (to be informed first) new information - give info to manger
↳ comm. and worker nodes

→ workers run the job or container.

→ Manger Node also takes the work loads.
↳ {Runs the container}



(tech. manager who works)

{Multi-manager confi.
↳ high availability}

→ We never run container work load on manager.
↓
[{Not possible on K8S}]

→ Multi Manager Concept also available.

18.223.136.47 - M2

18.216.129.200 - W

18.216.22.76 - W

Shell pre-configured
↓
Docker Command Line Environment

{Swarm was not a default feature}

(worker)

(worker)

```
> docker version (master)
    ↳ Swarm: Inactive
> docker info (Swarm availability)
    Swarm inactive
    (Feature was is, default By)
>
> docker swarm init
    Swarm initialized: current node as manager.
```

```
> docker version
> docker swarm join --token
    {docker swarm init} master
    docker
```

```
> docker version
```

past

past

> docker node ls (Run on manager)

id	Host Name	Status	Availability	Status Engine
Node-id1	IP1	Ready	Active	Worker
Node-id2	IP2	Ready	Active	Worker
Master	IP3	Ready	Active	Leader

> docker swarm join-token worker (For worker)

> docker " " manager (For manager)

> docker run, docker compose → one machine (limited) Number of container but on same machine

> Service → multiple container on multiple machine (Run)

> docker service create --name ravi --replaces 5 -p 3000:3000 learndevops/simpleapp

↙ run multiple container.

> docker service ls (List of Service)

id	Name	mode	Replicas	Image	Poder port: No
Service-id	ravi	replicated	3	name	

> docker PS (master) (Raft algo. is used to distribute)

{ three running on master node

> docker PS (slave) → docker distributes the job. (Raft-algo.)

[{ three running }]

> docker service PS ravi

> Id Name image Node desired state

> docker rm -f ga038 (slave machine)

> docker service ps ravi (@ high availability)

Id - Name → docker must run listed

> { auto healing / (non-zero exit (137)) → {Scaling container}

high availability }

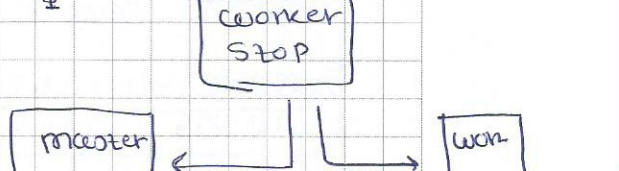
> ~~docker service s~~ (Service docker stop) (slave)

> docker PS (docker manages to run 5 containers always.

> Service docker start,

> docker service scale ravi=5,

> docker service ps ravi



? # docker service scale ravi=7 (desire state) [should desired state up & running] [Run in any case (3) Containers.]

> How to access the application?

* -p 3000:3000 (← this is Service port Number)

(small p) (Slower case) Host docker port → Container port

(Request: 3000) {service available on all Layer}

Service port Number (Virtual Layer access your Container)

↳ Request hit Service then it Navigate to Container

(Load Balancing) [

↳ distribution List (group List)

Service → DL (distribution List) [group List distribution]

> 18.223.136.47:3000 (request Served Container Id)
(master) IP

> Multiple request route to multiple Container)

(It can go to any Container)

any-port No. allowed, NO Rule

? > X.Y.Z.Y42:3000 (Slave_IP) (Forwared Request to all Node & Container)

↳ request Serered from Container

> Service Belongs to all node :3000 (Service port)

(any-ip) → forward request to Swarm Container.

Load Balancing

> Service → auto healing, auto Scaling.

> docker service rm ravi
> docker ps. (Removed Service)

> docker service create --name test --replicas 6 -P 1234:80 nginx

1234: Service port. 80: Nginx default port (exp.)

> docker inspect nginx
> docker pull nginx
> docker service rm test.

> docker service create --name test --mode global -P 1234:80 nginx

> 3 out of 3 tasks.

> docker service ps test

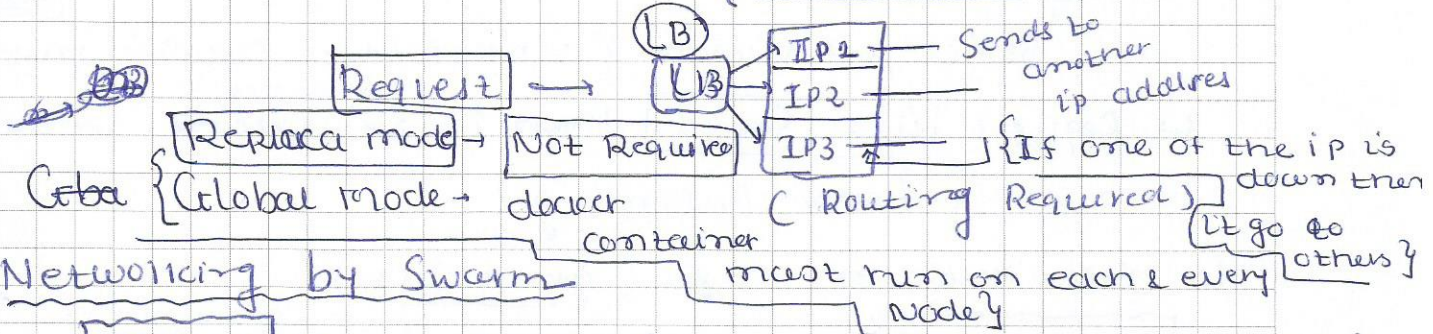
1. > one container per node. (4:26) {must Run on each node} (4)

> Log [Aggregator], Node [Log Collectors], [Monitoring Containers] (monitor the infrast.)

[global mode] → [one for each node]

> Monitoring or global modes. (update, Rollback Service)

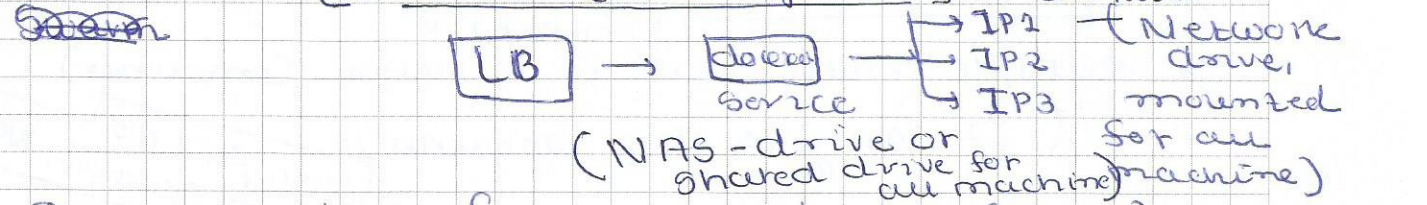
* If Master Node is down, {No Changes possible}



* Networking by Swarm

[global] Runs Single Container one each machine)

[docker Service logs ravi] → displays Logs from all nodes.



> One container for each node in Swarm)
(global) → Replicas it's different.

* Round-Robin selection. (auto-scaling possible in K8S not in Swarm)

* Service → how many types of Container, you are running Single type.

↳ Limited to one image

↳ tomcate & Nginix (Service at Single-Shot)

- three Container on master, 2- Container on Slave.

* Multiple Service type on Single-Shot (docker-Stack)

↳ (deploying different Service in Single-Shot)

> [Service imparative] & Stack Can be used {.yml}

↳ declarative

↳ {example.yml}

> deploy Container

↳ dlb Service

> Vi test.yml (Set of Service)

Redis

image:

db:
  image: —
  deploy:
    placement:
      constraints: [node = "manager"]

{ Constraints with Condition }

Result:

> docker Servie ~~ls~~ LS.
> docker Stack Services capp (set of Servie in Singe shot)
  ↳ deploying everything (
> docker Stack rm 'Stack_name' (Remove in Single-shot) n

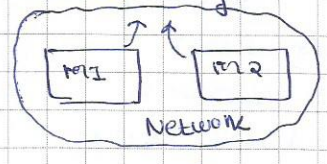
[docker stack deploy -c Stack.yml capp]

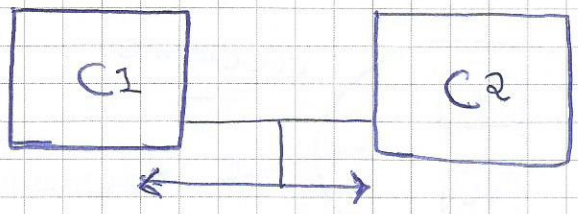
Stack:

(all service will be deployed in single-shot on swarm machine.)

```
Services:
  redis:
    image: redis:alpine
    deploy:
      replicas: 1
      update_config:
        parallelism: 2
        delay: 10s
      restart_policy:
        condition: on-failure
```

*
docker Networking.





Network (one connect with others) (how connected and the comm.)

CNM [Container Network manager] (container Network model)

→ docker gets default Networking Capability. (Networking Capability)

> ip a → docker 0 (Bridge Network, docker create) (Host machine)

Network (From host machine)

inet 172.17.0.1/16 ← container ip address from host
172.17.0.2/16

[Network from host machine it takes)

> docker inspect image / container / volume / Network
  ↳ Network: {
    Bridge: {
      gatway
      ip - 172.17.0.2

Contain Can be Accessed from Host machine

Container has their own ip address.

docker0 → ip address from {docker0}

R5 →

V4 →

A.3 →

(Ping nginx Container_id)

Ping Contair_ip addres from one

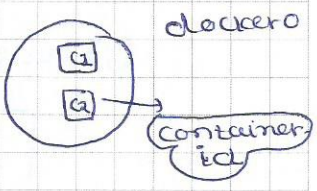
→ docker PS

→ docker run -d -P nginx

docker exec -it httpd ~~bash~~ bash

> apt-get install iputil-ping

> Ping Container_id

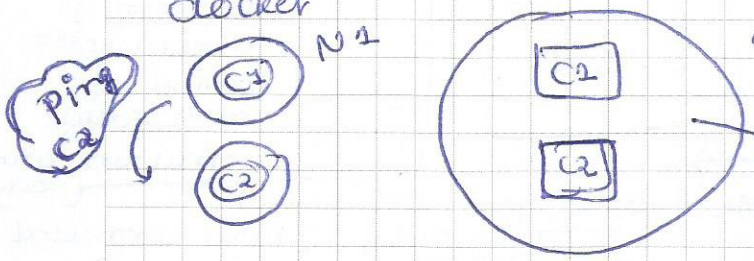


> docker PS

> docker exec 254167 ping 4abcs

{Bridge0 - provide comm. across Docker containers and external world including host}

docker



our own Network then default Net.

? → Bridge - [By default, Container to Container Comm.] → one container com. with other container)

→ host (host network) - [host IP + default] [Device]

→ none - NO Networking (waste container)

Swarm mode → overlay

→ macvlan

docker 0 → Network

→ docker Network Create -d

docker network create Network1
["Driver": "bridge"]

two comm. machine to machine Container

→ docker network ls

Name driver

→ Nobody can access this container (useless)

> docker run -d -P --network none nginx

> docker PS

host (Not showing any port)

> docker