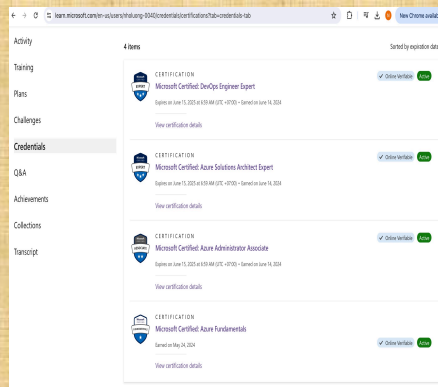
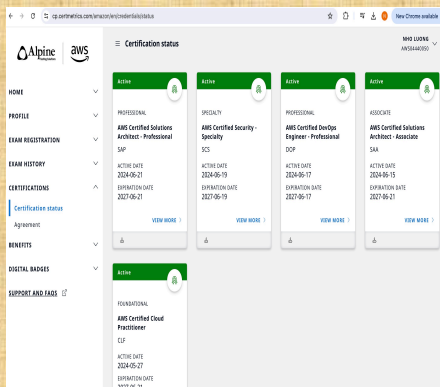


Docker Swarm

Author: Nho Luong
Skill: DevOps Engineer Lead



Overview

Docker Swarm is a clustering and scheduling tool for Docker containers. With it, you can establish and manage a cluster of Docker nodes as a single virtual system.

Swarm, IT administrators and developers

Docker Swarm Manager

A swarm is a group of machines that are running Docker and joined into a cluster. After that has happened, you continue to run the Docker commands you're used to, but now they are executed on a cluster by a swarm manager.

Using a Raft implementation, the managers maintain a consistent internal state of the entire swarm and all the services running on it. For testing purposes it is OK to run a swarm with a single manager. If the manager in a single-manager swarm fails, your services continue to run, but you need to create a new cluster to recover.

To take advantage of swarm mode's fault-tolerance features, Docker recommends you implement an odd number of nodes according to your organization's high-availability requirements.

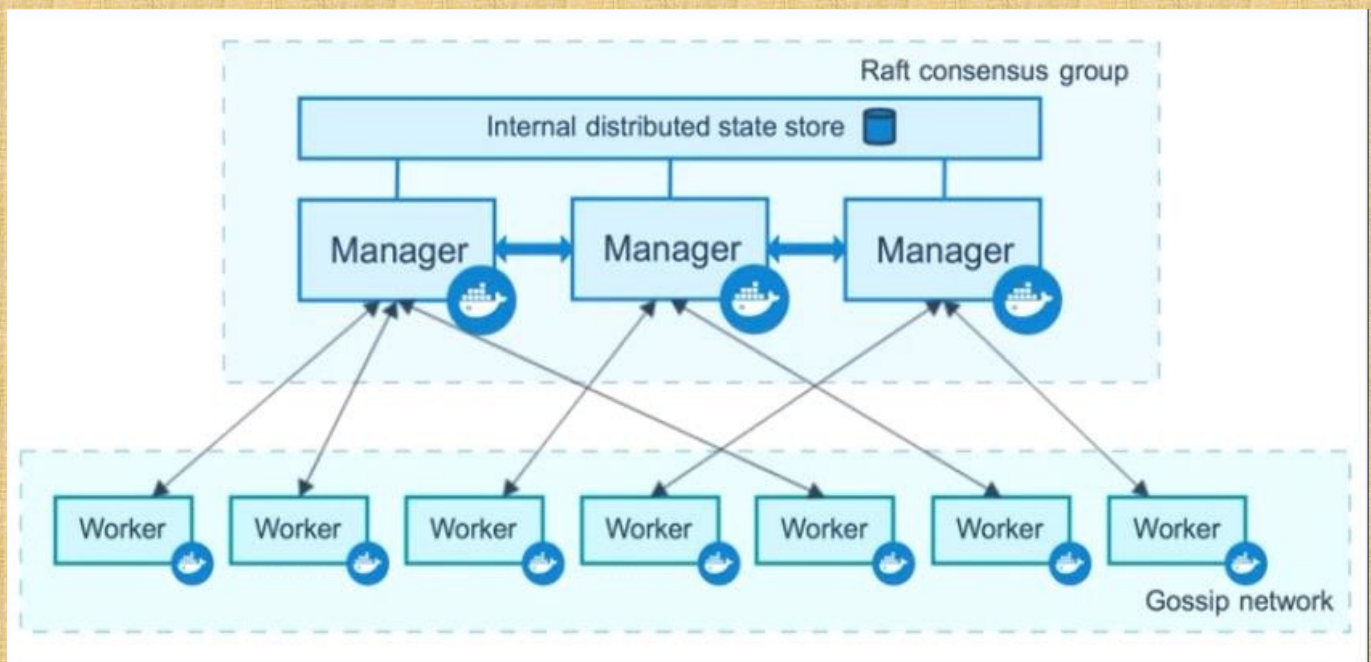
When you have multiple managers you can recover from the failure of a manager node without downtime.

A three-manager swarm tolerates a maximum loss of one manager.

- A five-manager swarm tolerates a maximum loss of at most $(N-1)/2$ manager nodes. A seven-manager swarm tolerates a maximum simultaneous loss of two manager nodes.
- Docker recommends a maximum of seven manager nodes for a swarm.

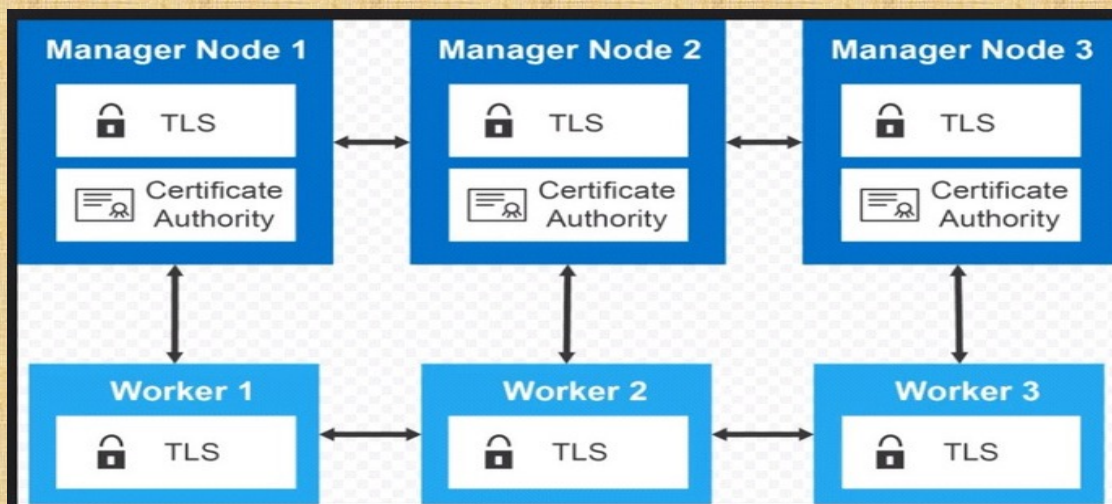
Docker Swarm Worker Node Worker nodes are also instances of Docker Engine whose sole purpose is to execute containers. Worker nodes don't participate in the Raft distributed state, make scheduling decisions, or serve the swarm mode HTTP API.

You can create a swarm of one manager node, but you cannot have a worker node without at least one manager node. By default, all managers are also workers. In a single manager node cluster, you can run commands like `docker service create` and the scheduler places all tasks on the local Engine.



Manage swarm security with public key infrastructure (PKI)

- The swarm mode public key infrastructure (PKI) system built into Docker makes it simple to securely deploy a container orchestration system. The nodes in a swarm use mutual Transport Layer Security (TLS) to authenticate, authorize, and encrypt the communications with other nodes in the swarm.
- When you create a [swarm](#) by running `docker swarm init`, Docker designates itself as a manager node.
- By default, the manager node generates a new root Certificate Authority (CA) along with a key pair, which are used to secure communications with other nodes that join the swarm. If you prefer, you can specify your own externally-generated root CA, using the `--external-caflag` of the `docker swarm init` command.
- The manager node also generates two tokens to use when you join additional nodes to the swarm: one **worker token** and one **manager token**. Each token includes the digest of the root CA's certificate and a randomly generated secret. When a node joins the swarm, the joining node uses the digest to validate the root CA certificate from the remote manager. The remote manager uses the secret to ensure the joining node is an approved node. Each time a new node joins the swarm, the manager issues a certificate to the node. The certificate contains a randomly generated node ID to identify the node under the certificate common name (CN) and the role under the organizational unit (OU). The node ID serves as the cryptographically secure node identity for the lifetime of the node in the current swarm.



Lab Session:

- Create 3 Linux machines and install docker on it.
- Name it as master ,worker01,worker02
- On master machine run `docker swarm init` to make it as manager node 4. Above step will create master node and it will return a token for worker nodes

```
root@master:/home/ubuntu# docker swarm init
Swarm initialized: current node (klui6jpr0flt679me9ewvrkg) is now a manager.

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

    docker swarm join --token SWMTKN-1-0iabdwepndt8wipfhnppwvcgcjydwcnvp9ylqzhzxmmgs0vx6f-cvfj79c4rgil26nmxvoc7ug1s 172.31.84.230:2377

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

```

root@master:/home/ubuntu# docker swarm init
Swarm initialized: current node (klui6jpvr0flt679me9ewvrkg) is now a manager.

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

    docker swarm join --token SWMTKN-1-0iabdwepndt8wipfhnppwvcgcydwcnp9ylqzhzxmms0vx6f-cvfj79c4rgil26nmxvoc7ugls 172.31.84.230:2377

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

```

Copy **docker swarm join** command and run it on **worker01** and **worker02** On master node run **docker node ls** command , it will show all the nodes in cluster

```

root@master:/home/ubuntu# docker node ls

```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE VERSION
klui6jpvr0flt679me9ewvrkg *	master	Ready	Active	Leader	18.09.7
w4tkg4vfpgwtep3eanotj6916	worker01	Ready	Active		18.09.7
8xp67hq694jbmegg0154jae8cz	worker02	Ready	Active		18.09.7

Command **docker swarm join-token worker** will give token value for worker node

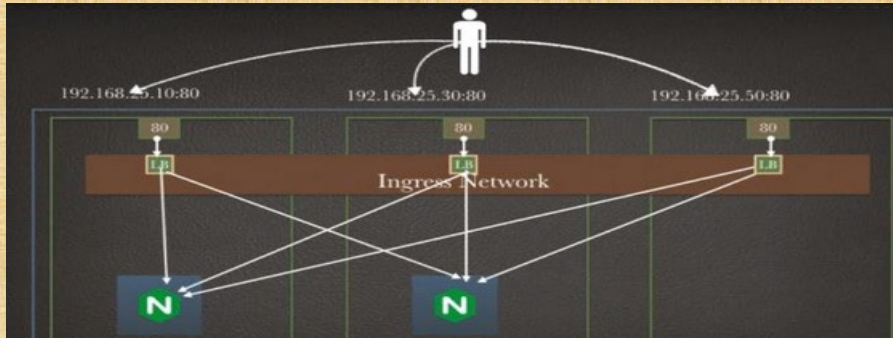
Command **docker swarm join-token manager** will give token value for the manage node

- Command **docker info** will give you swarm information as well like it is active or not
- Command **docker swarm leave** will leave the node(worker node) from the cluster and docker info will return Swarm as inactive
- Command **docker node rm worker01** will remove the worker from the list.
- Command **docker node rm -f worker02** will forcefully remove the worker from the list .
- Command **docker node inspect worker01** will inspect worker01 node.
- Command **docker node promote worker01 worker02** will promote worker01 and worker02 as manager node
- Command **docker node demote worker01 worker02** will promote worker01 and worker02 as worker node
- Command **watch docker container ls** will watch the running containers, run this command on each node (master as well as on worker nodes)
- Command **docker service --help** will show the options related to service which helps to create the service.
- Command **docker service create -d nginx** on master node and will create container in the cluster
- Command **docker service rm <<service name >>** will remove the service docker service create --name nginx --replicas 3 -p 80:80 nginx
- Command **docker service scale m2fapj0o5s3t=2** to scale up or scale down
- Command **docker service create --mode=global nginx** will create a service on each node Command **docker service create --replicas=3 --constraint="node.role==manager"** nginx will run the containers on the master node only
- Command **docker service create --replicas=3 --constraint="node.role==worker"** nginx will run the containers on the master node only
- Docker swarm runs on overlay network

Docker swarm runs on overlay network



Load Balancer in Overlay network



Create an overlay network

```
docker network create -d overlay my-overlay
docker service create --name website --replicas 3 --network my-overlay --publish 84:80
hshar/webapp
create db service
```

```
mysql -u root -p
intelli Create database docker; Use docker; mysql> create
table emp(name varchar(30), phone varchar(30)) -> ; Query
OK, 0 rows affected (0.01 sec) mysql> select * from emp;
Empty set (0.00 sec) mysql> select * from emp; +-----+-----+
+ | name | phone | +-----+-----+ | xyz | 5555555 | +-----+--
-----+ 1 row in set (0.00 sec)
```

```
docker service create --name db --replicas 1 --network my-overlay hshar/mysql:5.6
```

go inside db



Thank You