Docker Session 16 (17-03-20):

Docker Swarm Networking: Docker swarm by default uses the overlay network. This network enables the replicas to be accessible from all the nodes present in the cluster.

Use case:- Create 2 overlay networks intelliq1 and intelliq2

- Start tomcat with 5 replicas in the swarm cluster and at the time of starting specify that these replicas should run on intelliq1 network
- start nginx with 5 replicas on the default ~~network~~ Overlay network (ingress) and later perform a rolling network update to intelliq2 network.

① Create 2 overlay networks

# docker network create --driver overlay intelliq1

# docker network create --driver overlay intelliq2

② To see the list of all networks

# docker network ls

③ Create tomcat with 5 replicas on intelliq1 network

# docker service create --name webserver -p 9090:8080 --replicas 5 --network intelliq1 tomcat

④ To check if tomcat (webserver) service running on intelliq1 network

# docker network inspect webserver

The command generated the service info in JSON file format

To see that output in normal text format

# docker service inspect webserver --pretty

⑤ Create nginx with 5 replicas on the default overlay network (ingress)

# docker service create --name appserver -p 8888:80 --replicas 5 nginx

⑥ Perform a rolling network update from ingress network to intelliq2 network

# docker service update --network-add intelliq2 appserver

⑦ Check if nginx (appserver) is now running on intelliq2 network

# docker service inspect appserver --pretty

Docker Stack: Docker stack is used for creating Multicontainer architecture in the Swarm Cluster. A single stack can define the behaviour of an entire application where multiple services are linked with each other

docker compose + swarm = docker stack

docker compose + kubernetes = kompose

① To create a stack

# docker stack deploy -c stack/compose_file_name stack_name

② To see where the replicas of stack are running

# docker stack ps stack_name

③ To see the list of stacks in swarm cluster

# docker stack ls

④ To delete a stack

# docker stack rm stack_name

Use Case: Create a docker stack file with 3 replicas of wordpress will be linked with one replica of mysql

Vim stack1.yml

```
---
version: '3'
services:
  mydb:
    image: mysql:5
    environment:
      MYSQL_ROOT_PASSWORD: intellipaat

  mywordpress:
    image: wordpress
    ports:
      - 5050:80
    deploy:
      replicas: 3
...
```

To create a stack from the above file

# docker stack deploy -c stack1.yml wordpress

To check of the stack services are running

# docker stack ps wordpress

⇒

Vim stack4.yml

```
---
version: '3'

services:
  devserver:
    image: jenkins
    ports:
      - 6060:8080
    deploy:
      replicas: 2
      placement:
        constraints:
          - node.hostname == Manager

  qaserver:
    image: tomcat
    ports:
      - 7070:8080
    deploy:
      replicas: 3
      placement:
        constraints:
          - node.hostname == worker1

  prodserver:
    image: tomcat
    ports:
      - 8080:8080
    deploy:
      replicas: 4
```

```
    placement:
      constraints:
        - node.hostname == worker2
...
wq!
```

To create a stack from the above file

```
# docker stack deploy -c stack2.yml ci-cd
```

To check if the stack services are running

```
# docker stack ps ci-cd
```

## Docker Session 17 (18-03-20):

⇒ Create a docker stack file for establishing the LAMP architecture and also specify upper limits on the amount of hardware resources.

```
Vim stack3.yml
---
version: '3'
services:
  mydb:
    image: mysql
    environment:
      MYSQL_ROOT_PASSWORD= intellig
    deploy:
      replicas: 2
      resources:
        limits:
          cpus: "0.01"
          memory: 150M

  apache:
    image: httpd
    ports:
      - 8989: 80

    deploy:
      replicas: 3
      resources:
        limits:
          cpus: "0.01"
          memory: 50M
```

```
php:
  image: php:7.2-apache
  deploy:
    replicas: 4
    resources:
      limits:
        cpus: "0.01"
        memory: 50M
...
wq!
```

⇒ When handling failover scenarios of a worker machine crashes, the replicas running on worker will automatically migrates to Manager and other workers. But if the Manager collapses there will be no Leader to maintain docker swarm. To handle this kind of challenges, we maintain Managers This Managers information will be maintained using "quorum". The Minimum no. of Managers that should be maintained is

$$\text{Min\_Managers} = \underbrace{\frac{\text{Total Managers}}{2}}_{\text{Next whole number}}$$

| No. of Managers | Failure Tolerence | Required fo Quoram |
|---|---|---|
| 1 | 0 | 1 |
| 2 | 0 | 2 |
| 3 | 1 | 2 |
| 4 | 1 | 3 |
| 5 | 2 | 3 |
| 6 | 2 | 4 |
| 7 | 3 | 4 |