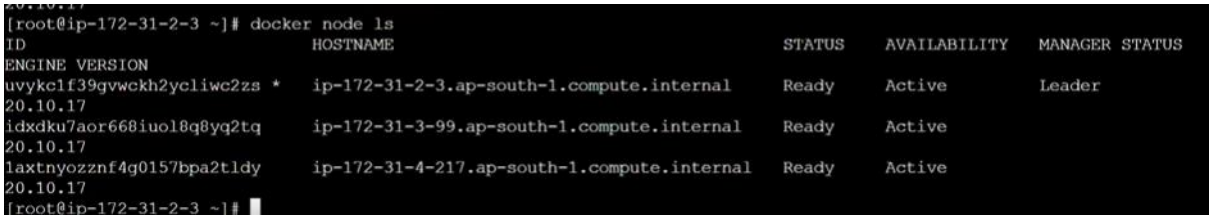
Summary

Session No – 16

- When we have multi-tier architecture or 3-tier architecture & one single container will not work i.e. if one container goes down the entire architecture goes down example MySQL database & word press
- In multi-tier architecture, if one container goes down an entire architecture or website goes down
- In Swarm, we have to give a task. It can be a single container or multiple containers
- If we use docker in it we have an image from it we launch the container so docker can see & manage the container directly
- The minimum thing Swarm can manage is the task
- If due to any reason, the base system goes down & container is deleted swarm will automatically launch the container in another node it is called fault tolerance
- Swarm cluster gives the capability to make the exact copy of the task in another node this concept is called replication
- Let's say we are running the same website in different nodes we can place a program in between the client and the server which has the capability to take the request from the client and connect to the website or node. This program is called a load balancer
- Swarm gives us a pre-created load balancer known as a service
- Whenever we launch a task swarm will always launch one service which is known as a load balancer

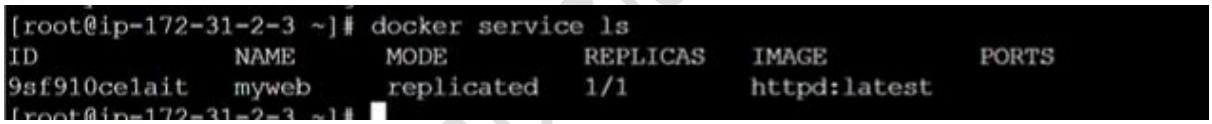
Linux World Informatics Pvt Ltd

- **docker node ls** command is used to list the nodes in the cluster



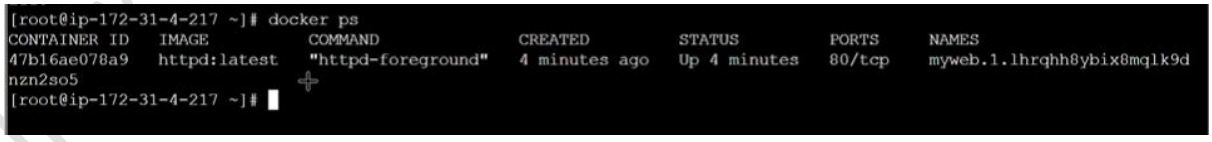
- Creating service
  - Command:- **docker service create --name (Service name) (image name)**

- **docker service ls** command is used to list all services

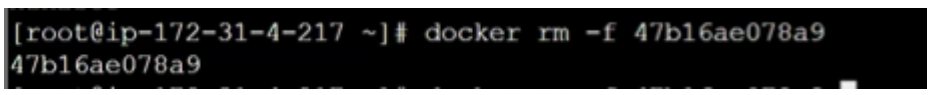


- **docker service ps** command is used to know more detail about the service

- On one of the nodes, we can see the container is created



- Now the duty of the manager node is to monitor the container & if it does down or removed recreate it
- Deleting the container manually



- If we go to the manager node & see the details of the service we can see it is automatically detected & relaunched in another node

```
[root@ip-172-31-2-3 ~]# docker service ps  myweb
ID             NAME          IMAGE          NODE                                          DESIRED STATE   CURRENT STAT
E            ERROR                         PORTS
a54rz2ho5uyq   myweb.1       httpd:latest   ip-172-31-2-3.ap-south-1.compute.internal     Running         Running 37 s
econds ago
lhrqhh8ybix8    \_ myweb.1   httpd:latest   ip-172-31-4-217.ap-south-1.compute.internal   Shutdown        Failed 43 se
conds ago    "task: non-zero exit (137)"
[root@ip-172-31-2-3 ~]#
```

- By default nature of the service is that it is isolated or private which means anyone from the internet who tries to connect to the service will not connect to the container
- **docker service inspect (service name)** command is used to know more detail about the service

```
[root@ip-172-31-2-3 ~]# docker service inspect  myweb
[
    {
        "ID": "9sf910ce1ait4ctqbflha3imb",
        "Version": {
            "Index": 210964
        },
        "CreatedAt": "2022-11-22T16:17:50.613750116Z",
        "UpdatedAt": "2022-11-22T16:17:50.613750116Z",
        "Spec": {
            "Name": "myweb",
            "Labels": {},
            "TaskTemplate": {
                "ContainerSpec": {
                    "Image": "httpd:latest@sha256:f2e89def4c032b02c83e162c1819ccfcbd4ea6bdbc
```

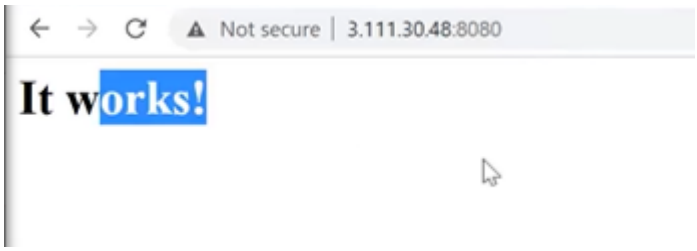
- **--publish** keyword in docker service command is used for exposing it to the outside world
    - Command:- **docker service create --name (service name) --publish 8080:80 (image name)**

```
[root@ip-172-31-2-3 ~]# docker service create --name myweb   --publish  8080:80   httpd
jh461y5219315r2wupeapgvi7
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Waiting 3 seconds to verify that tasks are stable...
```

- Connecting to the website with the curl command

```
[root@ip-172-31-2-3 ~]# curl  http://172.31.2.3:8080
<html><body><h1>It works!</h1></body></html>
[root@ip-172-31-2-3 ~]#
```

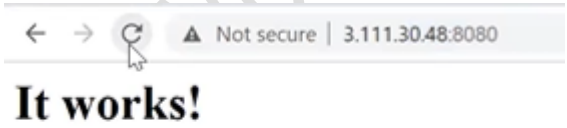
- Connecting to the website from the browser



- Between the client and the task, we have a load balancer running which is taking the request from the client and connecting to the task. The task gives the reply to the load balancer & load balancer gives to the client
- If we delete the container manually client won't see the dysconnectivity

```
[root@ip-172-31-4-217 ~]# docker rm -f 582f32f8f41c
582f32f8f41c
[root@ip-172-31-4-217 ~]# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[root@ip-172-31-4-217 ~]#
```

- Accessing the webpage from the browser



- As soon as the container is deleted swarm master automatically launches another container with the same service
- As soon as we remove the service client won't be able to see the website

```
[root@ip-172-31-2-3 ~]# docker service rm myweb
myweb
[root@ip-172-31-2-3 ~]#
```

- Creating service from **vimal13/apache-webserver-php** image
    - Command**:- docker service create --name(service name) --publish 8080:80 vimal13/apache-webserver-php**

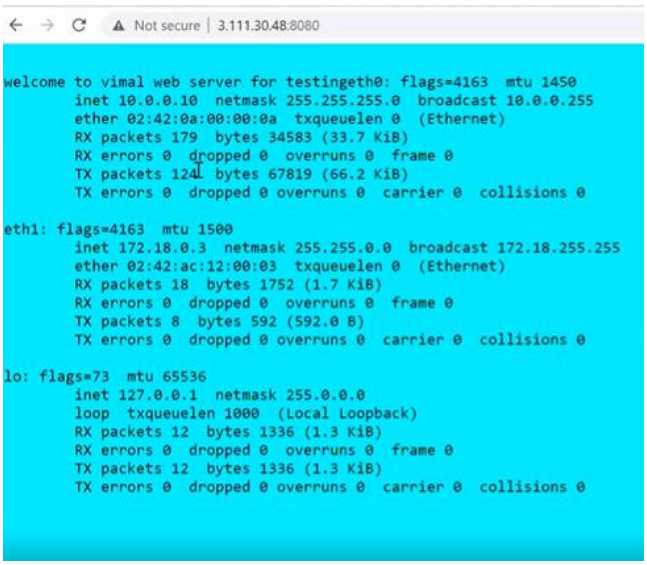
```
[root@ip-172-31-2-3 ~]# docker service create --name myweb --publish 8080:80  vimal13/apache-webserver-php
td9yevstpebnt4rnihh6fyzdu
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
[root@ip-172-31-2-3 ~]# docker service ls
ID             NAME      MODE         REPLICAS   IMAGE                                  PORTS
td9yevstpebn   myweb     replicated   1/1        vimal13/apache-webserver-php:latest   *:8080->80/tcp
[root@ip-172-31-2-3 ~]# docker service ps myweb
ID             NAME      IMAGE                                  NODE                                                DESIRED
TATE   CURRENT STATE           ERROR     PORTS
0klv16iigy0i   myweb.1   vimal13/apache-webserver-php:latest   ip-172-31-4-217.ap-south-1.compute.internal   Running
       Running 21 seconds ago
[root@ip-172-31-2-3 ~]#
```

- Connecting to the website from the browser

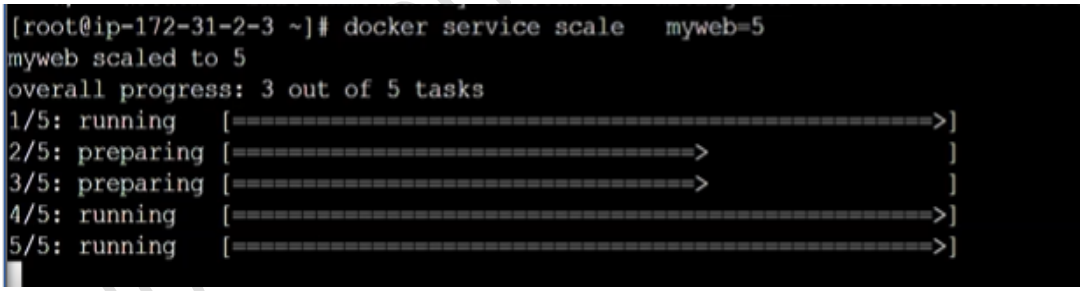
- Now if we delete the container manually service will launch the container automatically

```
[root@ip-172-31-4-217 ~]# docker ps
CONTAINER ID   IMAGE                                  COMMAND                  CREATED              STATUS
 PORTS     NAMES
81a64f235a69   vimal13/apache-webserver-php:latest   "/usr/sbin/httpd -DF…"   About a minute ago   Up About a minute
 80/tcp    myweb.1.0klv16iigy0ib2ma0t7bjg6j6
[root@ip-172-31-4-217 ~]# docker rm -f 81a6
81a6
[root@ip-172-31-4-217 ~]#
```

- Now this time we connect to our website we can see the IP address has changed

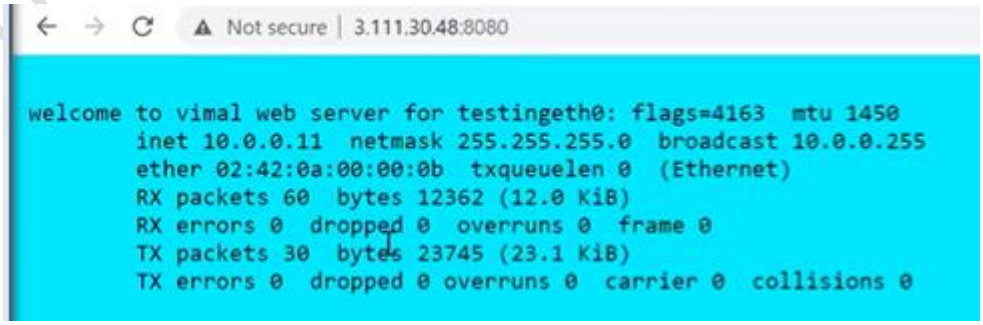


- **scale** keyword in the docker service is used for horizontally scaling the service
    - Command:- **docker service scale (Service name)=5**

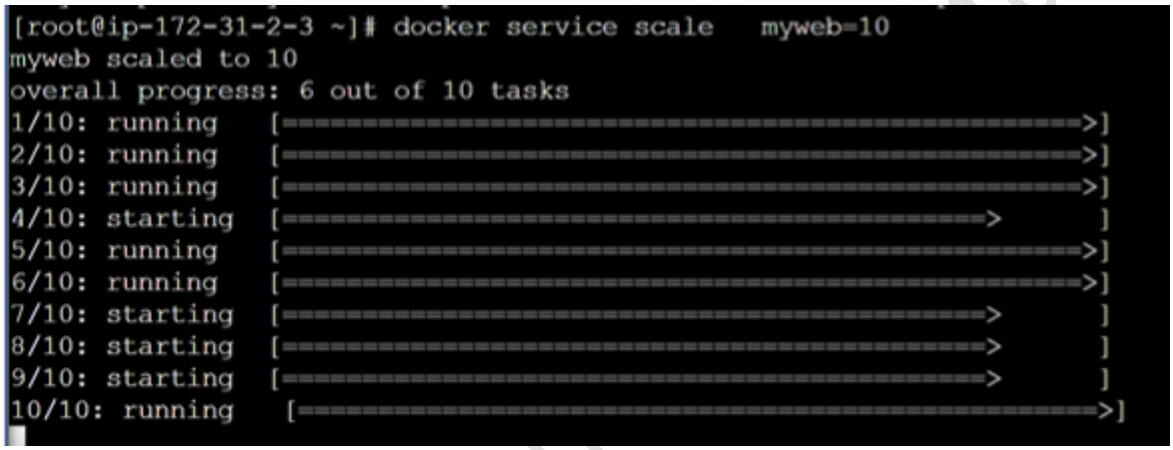


- Listing the service

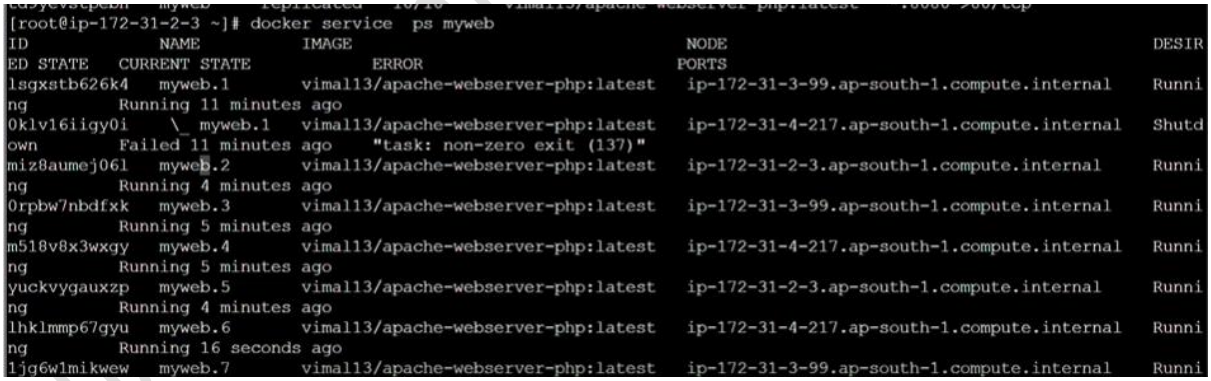
- Now if we refresh the web page we can see the IP address is changing



- The good thing about scaling is in just one click we can scale the task or container



- Now if we see the service we can see 10 containers are running



- Adding more containers or instances are called scale-out & removing the containers or instances is called scale in
- Scaling-in the container