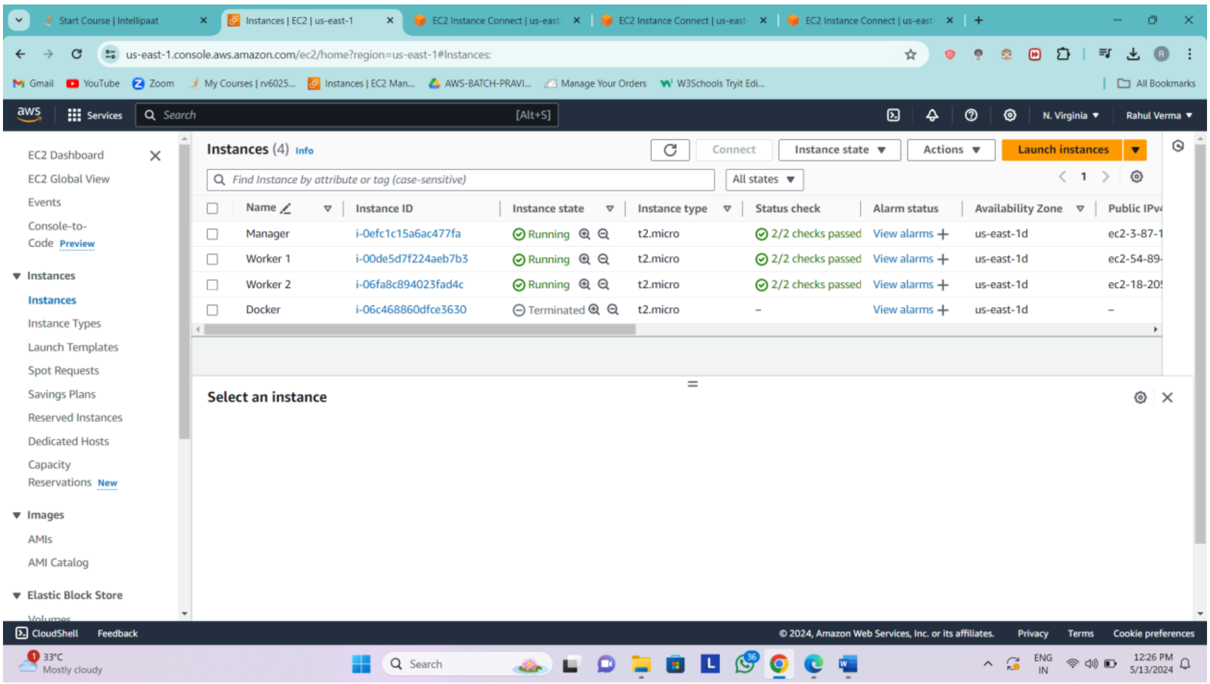
Docker swarm Services

Prerequisite:

Create 3 instance named manager, worker 1 and worker 2

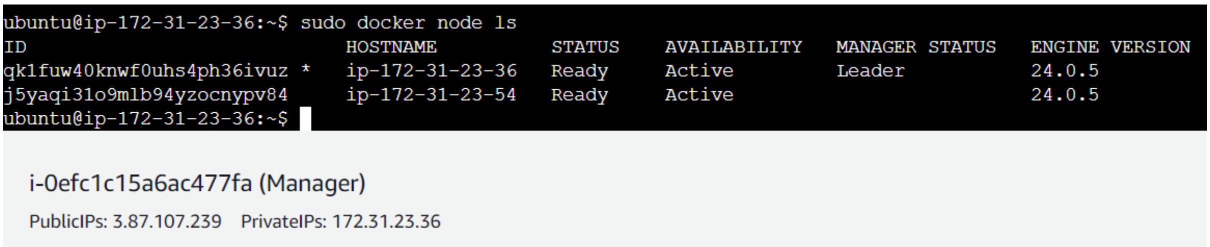


Instances (4)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
Manager	i-0efc1c15a6ac477fa	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1d	ec2-3-87-1
Worker 1	i-00de5d7f224aeb7b3	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1d	ec2-54-89-
Worker 2	i-06fa8c894023fad4c	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1d	ec2-18-20!
Docker	i-06c468860dfce3630	Terminated	t2.micro	–	View alarms +	us-east-1d	–

1. We already have two nodes

sudo docker node ls



```
ubuntu@ip-172-31-23-36:~$ sudo docker node ls
ID                            HOSTNAME          STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
qk1fuw40knwf0uhs4ph36ivuz *   ip-172-31-23-36   Ready     Active         Leader           24.0.5
j5yaqi31o9mlb94yzocnypv84     ip-172-31-23-54   Ready     Active                          24.0.5
ubuntu@ip-172-31-23-36:~$ 
```

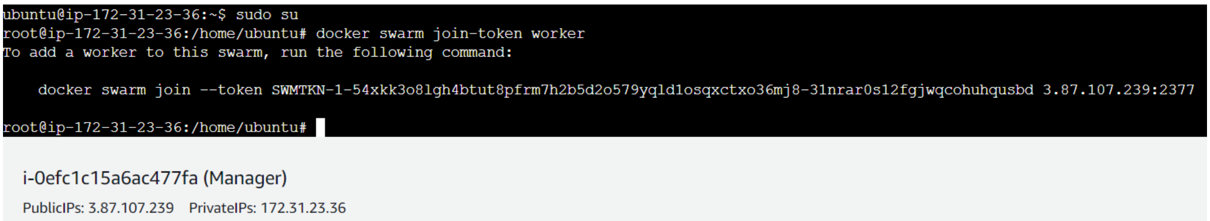
i-0efc1c15a6ac477fa (Manager)

PublicIPs: 3.87.107.239   PrivateIPs: 172.31.23.36

2. Now let's add one more node (worker 2)

To get the token type the below command-

sudo docker swarm join-token worker

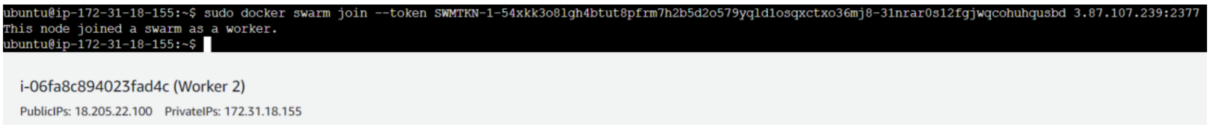


```
ubuntu@ip-172-31-23-36:~$ sudo su
root@ip-172-31-23-36:/home/ubuntu# docker swarm join-token worker
To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-54xkk3o81gh4btut8pfrm7h2b5d2o579yqld1osqxctxo36mj8-31nrar0s12fgjwqcohuhqusbd 3.87.107.239:2377

root@ip-172-31-23-36:/home/ubuntu#
```

i-0efc1c15a6ac477fa (Manager)

PublicIPs: 3.87.107.239   PrivateIPs: 172.31.23.36

Copy the token and paste it in your new worker 2 instance



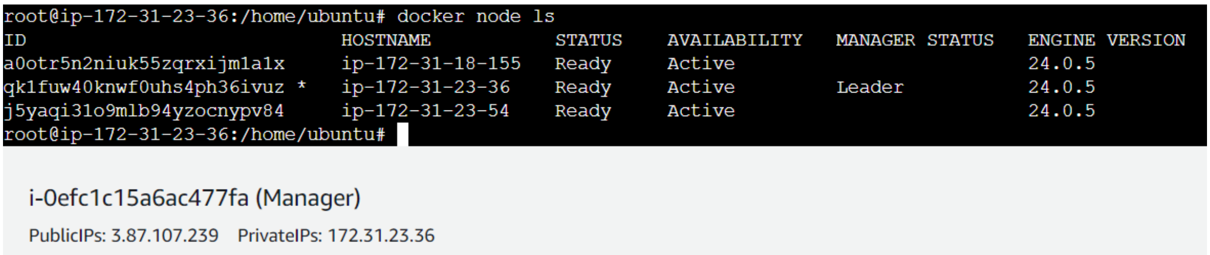
```
ubuntu@ip-172-31-18-155:~$ sudo docker swarm join --token SWMTKN-1-54xkk3o81gh4btut8pfrm7h2b5d2o579yqld1osqxctxo36mj8-31nrar0s12fgjwqcohuhqusbd 3.87.107.239:2377
This node joined a swarm as a worker.
ubuntu@ip-172-31-18-155:~$
```

i-06fa8c894023fad4c (Worker 2)

PublicIPs: 18.205.22.100   PrivateIPs: 172.31.18.155

3. check if it's done or not

docker node ls



```
root@ip-172-31-23-36:/home/ubuntu# docker node ls
ID                            HOSTNAME          STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
a0otr5n2niuk55zqrxijm1a1x     ip-172-31-18-155  Ready     Active                          24.0.5
qk1fuw40knwf0uhs4ph36ivuz *   ip-172-31-23-36   Ready     Active         Leader           24.0.5
j5yaqi31o9mlb94yzocnypv84     ip-172-31-23-54   Ready     Active                          24.0.5
root@ip-172-31-23-36:/home/ubuntu#
```

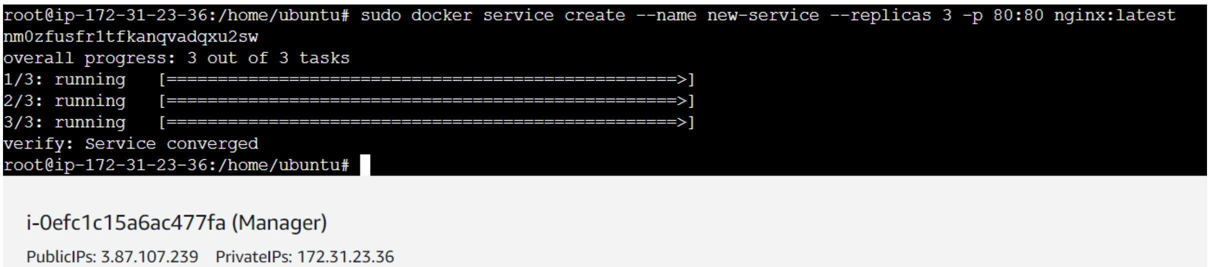
i-0efc1c15a6ac477fa (Manager)

PublicIPs: 3.87.107.239   PrivateIPs: 172.31.23.36

And we can see in the above image we have 3 nodes which means we have successfully added our 3rd node which is worker 2 instance.

4. now let’s create one service

sudo docker service create --name new-service --replicas 3 -p 80:80 nginx:latest



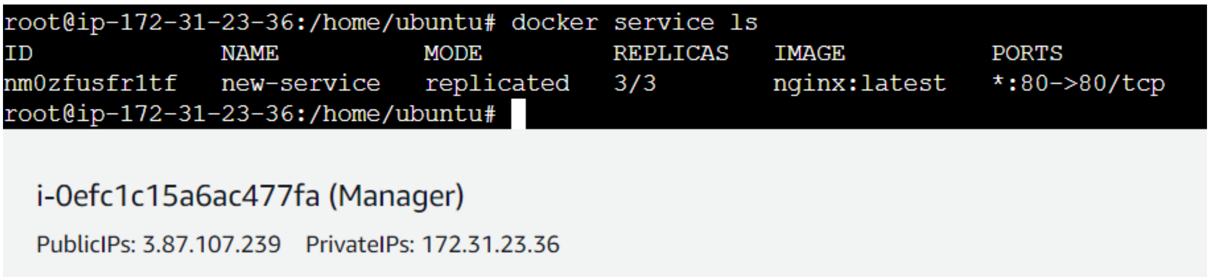
```
root@ip-172-31-23-36:/home/ubuntu# sudo docker service create --name new-service --replicas 3 -p 80:80 nginx:latest
nm0zfusfr1tfkanqvadqxu2sw
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
3/3: running   [==================================================>]
verify: Service converged
root@ip-172-31-23-36:/home/ubuntu#
```

i-0efc1c15a6ac477fa (Manager)

PublicIPs: 3.87.107.239   PrivateIPs: 172.31.23.36

To check service

docker service ls



```
root@ip-172-31-23-36:/home/ubuntu# docker service ls
ID             NAME          MODE         REPLICAS   IMAGE          PORTS
nm0zfusfr1tf   new-service   replicated   3/3        nginx:latest   *:80->80/tcp
root@ip-172-31-23-36:/home/ubuntu#
```

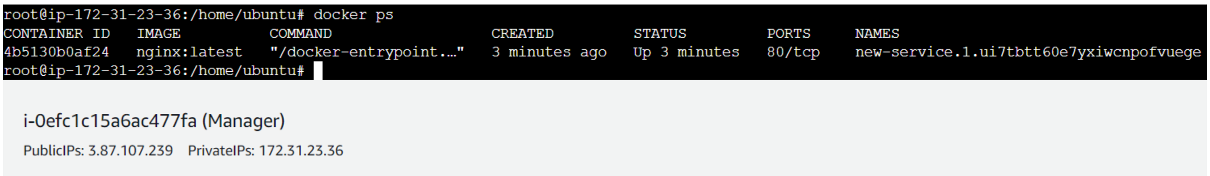
i-0efc1c15a6ac477fa (Manager)

PublicIPs: 3.87.107.239   PrivateIPs: 172.31.23.36

5. if we check all the nodes, we should have containers running on it

Manager node:

docker ps

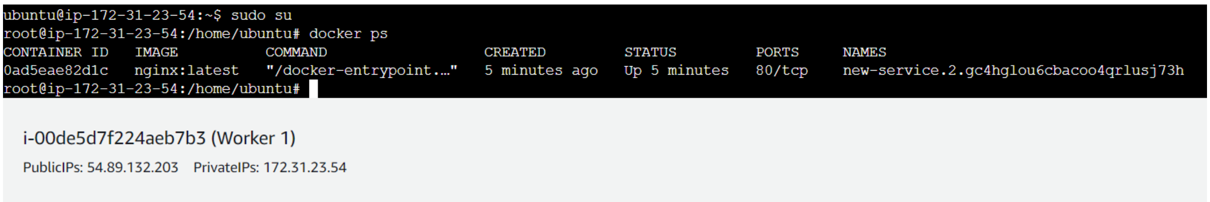


```
root@ip-172-31-23-36:/home/ubuntu# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS     NAMES
4b5130b0af24   nginx:latest   "/docker-entrypoint.…"   3 minutes ago   Up 3 minutes   80/tcp    new-service.1.ui7tbtt60e7yxiwcnpofvuege
root@ip-172-31-23-36:/home/ubuntu#
```

i-0efc1c15a6ac477fa (Manager)

PublicIPs: 3.87.107.239   PrivateIPs: 172.31.23.36

Worker 1 node:

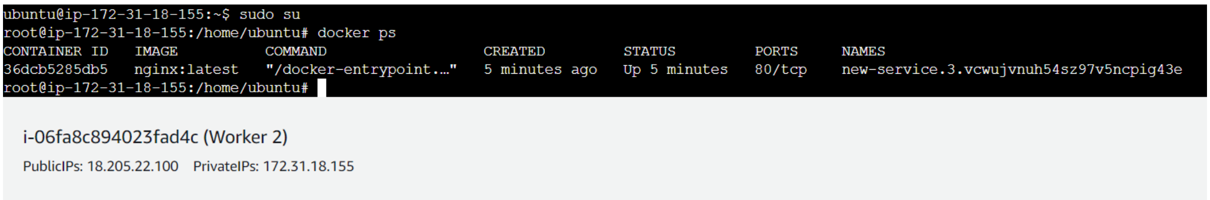


```
ubuntu@ip-172-31-23-54:~$ sudo su
root@ip-172-31-23-54:/home/ubuntu# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS     NAMES
0ad5eae82d1c   nginx:latest   "/docker-entrypoint.…"   5 minutes ago   Up 5 minutes   80/tcp    new-service.2.gc4hglou6cbacoo4qrlusj73h
root@ip-172-31-23-54:/home/ubuntu#
```

i-00de5d7f224aeb7b3 (Worker 1)

PublicIPs: 54.89.132.203   PrivateIPs: 172.31.23.54

Worker 2 node:

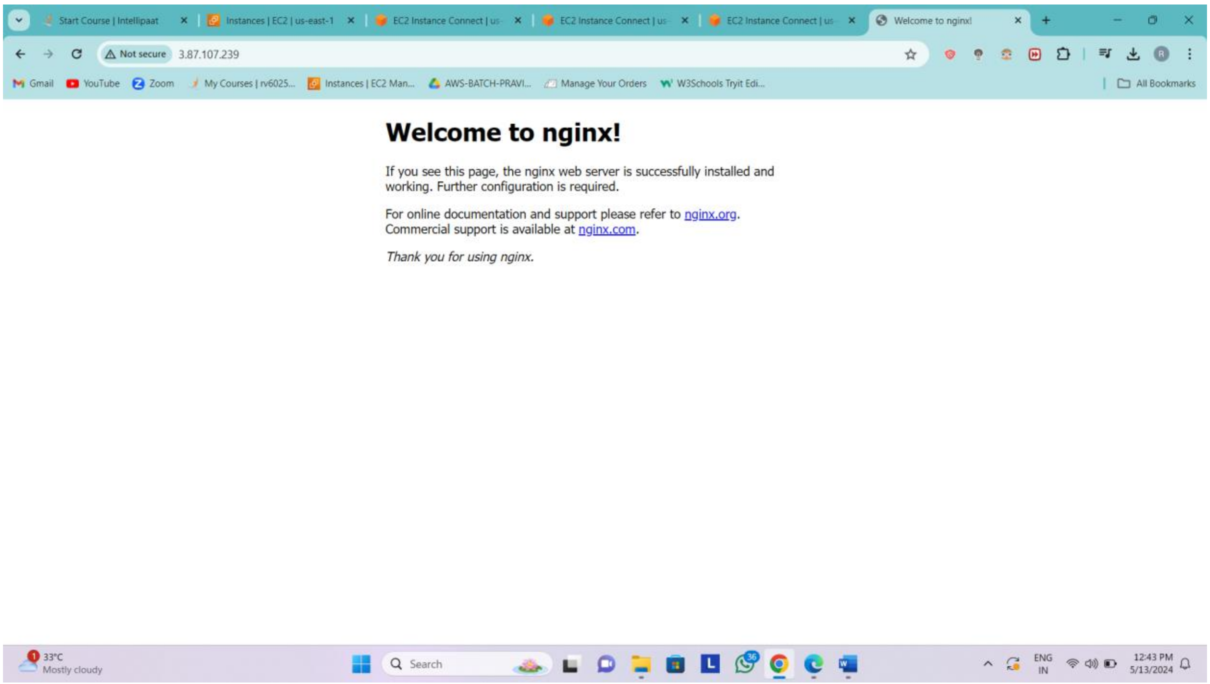


```
ubuntu@ip-172-31-18-155:~$ sudo su
root@ip-172-31-18-155:/home/ubuntu# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS     NAMES
36dcb5285db5   nginx:latest   "/docker-entrypoint.…"   5 minutes ago   Up 5 minutes   80/tcp    new-service.3.vcwujvnuh54sz97v5ncpig43e
root@ip-172-31-18-155:/home/ubuntu#
```

i-06fa8c894023fad4c (Worker 2)

PublicIPs: 18.205.22.100   PrivateIPs: 172.31.18.155

6. and if we visit one the ip address it should show us nginx welcome page



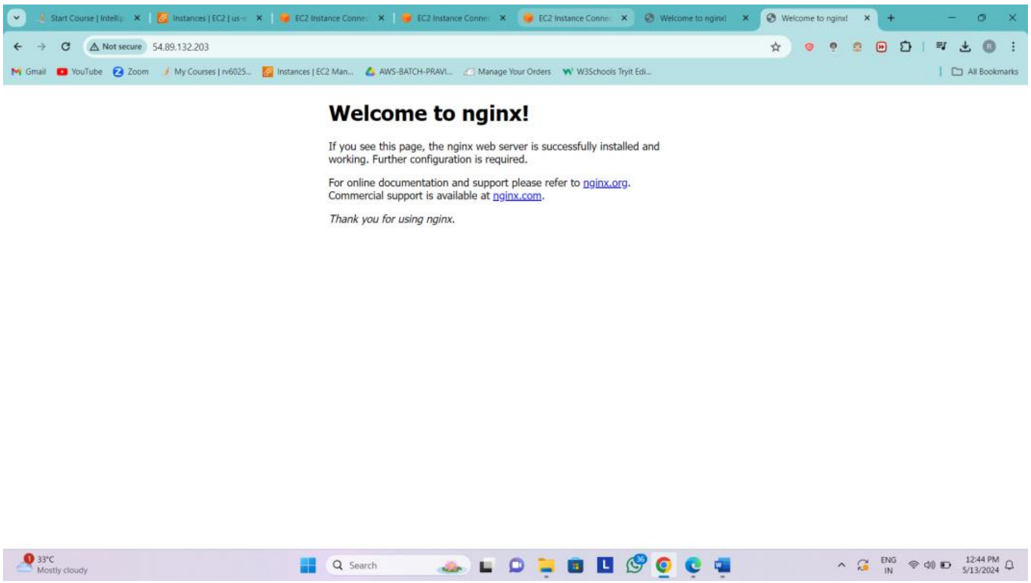
Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Worker 1:



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

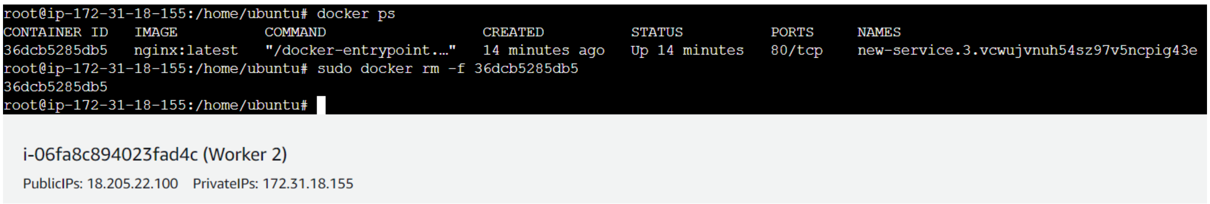
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

7. now what happen if one of the container goes down

To remove container

sudo docker rm -f <container_id>

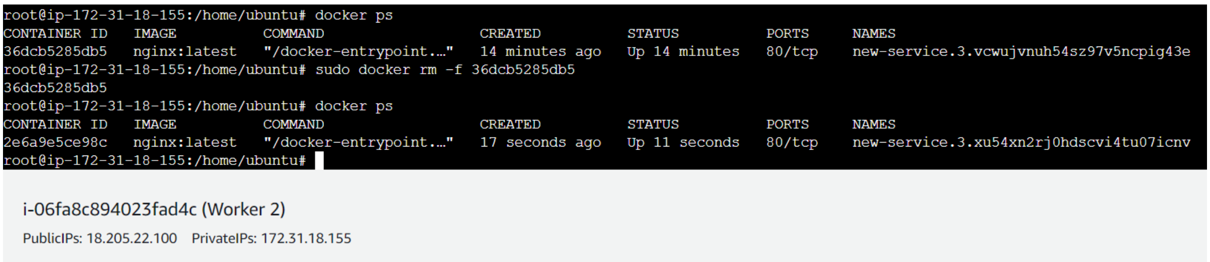


```
root@ip-172-31-18-155:/home/ubuntu# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS     NAMES
36dcb5285db5   nginx:latest   "/docker-entrypoint.…"   14 minutes ago   Up 14 minutes   80/tcp    new-service.3.vcwujvnuh54sz97v5ncpig43e
root@ip-172-31-18-155:/home/ubuntu# sudo docker rm -f 36dcb5285db5
36dcb5285db5
root@ip-172-31-18-155:/home/ubuntu#
```

i-06fa8c894023fad4c (Worker 2)

PublicIPs: 18.205.22.100   PrivateIPs: 172.31.18.155

And we see it has automatically created container



```
root@ip-172-31-18-155:/home/ubuntu# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS     NAMES
36dcb5285db5   nginx:latest   "/docker-entrypoint.…"   14 minutes ago   Up 14 minutes   80/tcp    new-service.3.vcwujvnuh54sz97v5ncpig43e
root@ip-172-31-18-155:/home/ubuntu# sudo docker rm -f 36dcb5285db5
36dcb5285db5
root@ip-172-31-18-155:/home/ubuntu# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS     NAMES
2e6a9e5ce98c   nginx:latest   "/docker-entrypoint.…"   17 seconds ago   Up 11 seconds   80/tcp    new-service.3.xu54xn2rj0hdscvi4tu07icnv
root@ip-172-31-18-155:/home/ubuntu#
```

i-06fa8c894023fad4c (Worker 2)

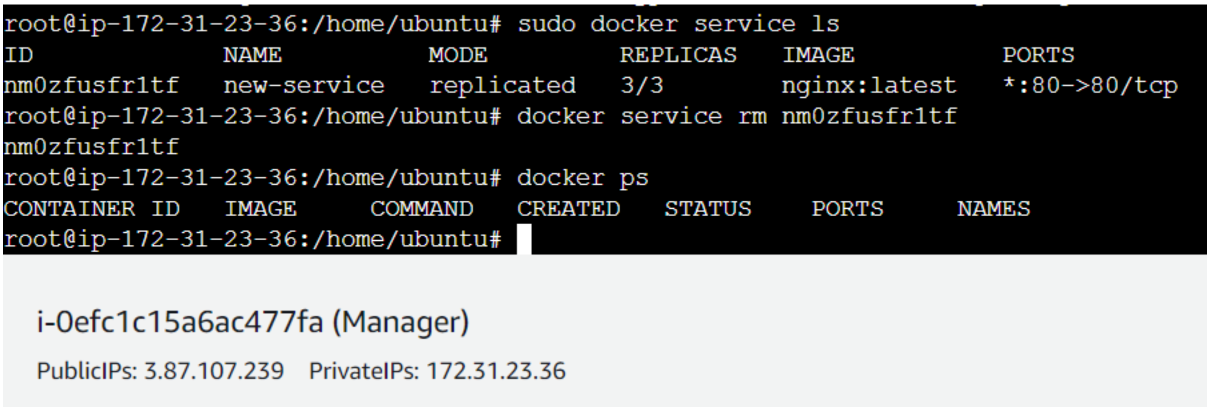
PublicIPs: 18.205.22.100   PrivateIPs: 172.31.18.155

So what docker swarm do is, it will detect if any of the container is down in a service then it will restart them or launch a new one. So this will help us to reduce any downtime.

Note: if we remove our service everything will be gone

8. to remove a service

docker service rm <id>



```
root@ip-172-31-23-36:/home/ubuntu# sudo docker service ls
ID             NAME          MODE         REPLICAS   IMAGE          PORTS
nm0zfusfr1tf   new-service   replicated   3/3        nginx:latest   *:80->80/tcp
root@ip-172-31-23-36:/home/ubuntu# docker service rm nm0zfusfr1tf
nm0zfusfr1tf
root@ip-172-31-23-36:/home/ubuntu# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@ip-172-31-23-36:/home/ubuntu#
```

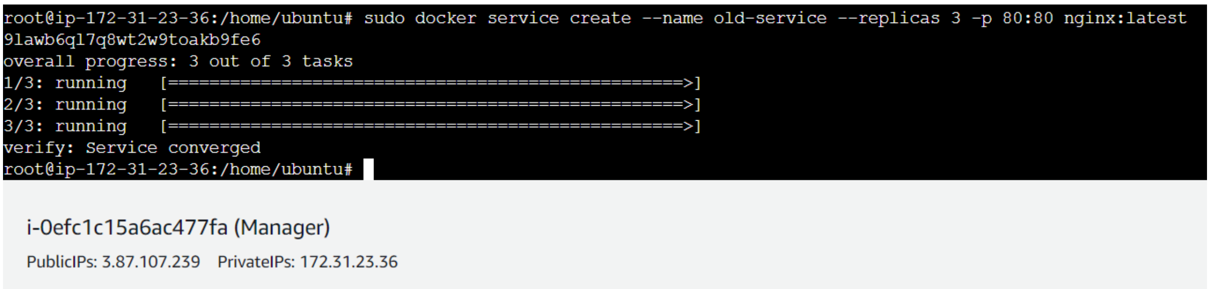
i-0efc1c15a6ac477fa (Manager)

PublicIPs: 3.87.107.239   PrivateIPs: 172.31.23.36

In the above image we can see removing service removes all the container as well.

Will create service one more time and will show inspect it

sudo docker service create --name old-service --replicas 3 -p 80:80 nginx:latest

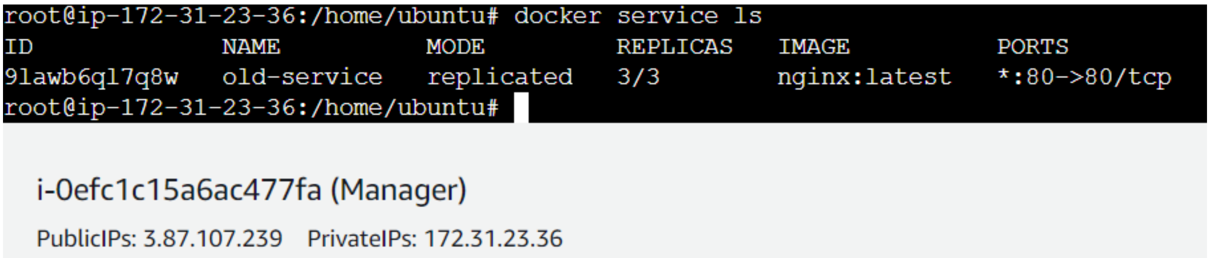


```
root@ip-172-31-23-36:/home/ubuntu# sudo docker service create --name old-service --replicas 3 -p 80:80 nginx:latest
9lawb6ql7q8wt2w9toakb9fe6
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
3/3: running   [==================================================>]
verify: Service converged
root@ip-172-31-23-36:/home/ubuntu#
```

i-0efc1c15a6ac477fa (Manager)

PublicIPs: 3.87.107.239   PrivateIPs: 172.31.23.36

docker service ls



```
root@ip-172-31-23-36:/home/ubuntu# docker service ls
ID             NAME          MODE         REPLICAS   IMAGE          PORTS
9lawb6ql7q8w   old-service   replicated   3/3        nginx:latest   *:80->80/tcp
root@ip-172-31-23-36:/home/ubuntu#
```

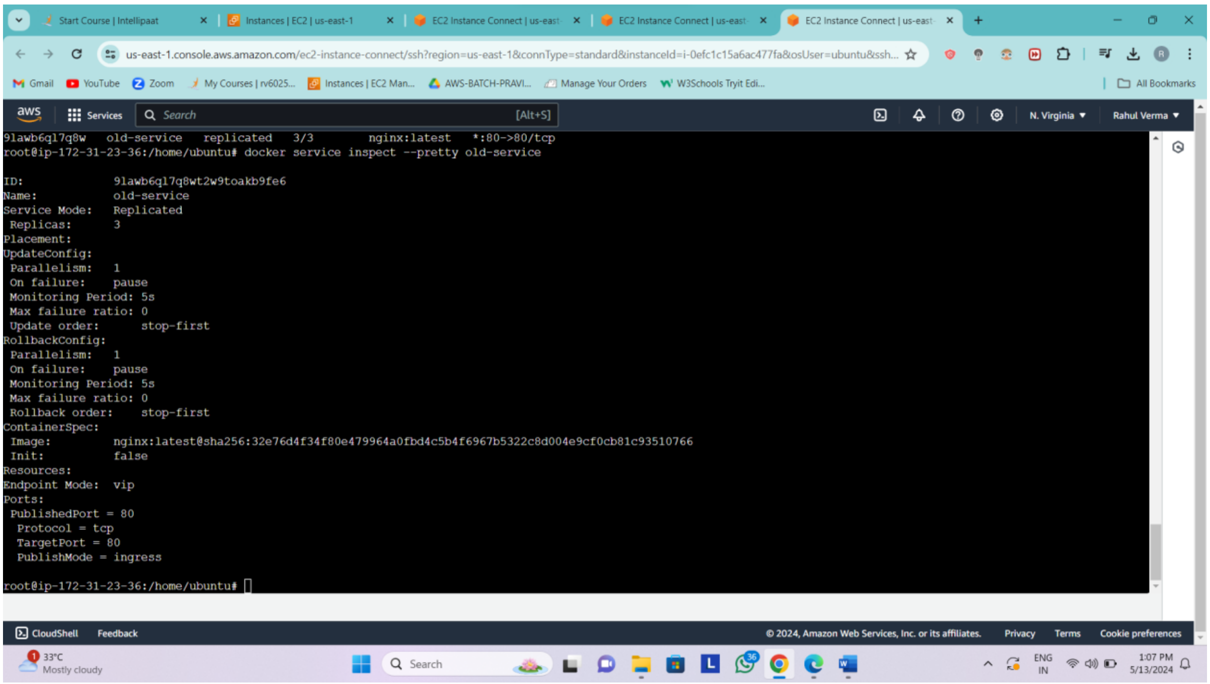
i-0efc1c15a6ac477fa (Manager)

PublicIPs: 3.87.107.239   PrivateIPs: 172.31.23.36

9. To inspect service on (manager node)

docker service inspect --pretty <name_of_service>

this will show all the info about our service

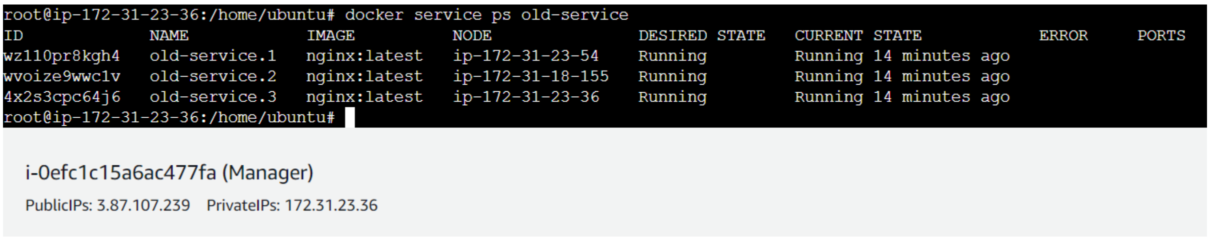


```
9lawb6ql7q8w   old-service   replicated   3/3       nginx:latest   *:80->80/tcp
root@ip-172-31-23-36:/home/ubuntu# docker service inspect --pretty old-service

ID:             9lawb6ql7q8wt2w9toakb9fe6
Name:           old-service
Service Mode:   Replicated
 Replicas:      3
Placement:
UpdateConfig:
 Parallelism:   1
 On failure:    pause
 Monitoring Period: 5s
 Max failure ratio: 0
 Update order:      stop-first
RollbackConfig:
 Parallelism:   1
 On failure:    pause
 Monitoring Period: 5s
 Max failure ratio: 0
 Rollback order:    stop-first
ContainerSpec:
 Image:         nginx:latest@sha256:32e76d4f34f80e479964a0fbd4c5b4f6967b5322c8d004e9cf0cb81c93510766
 Init:          false
Resources:
Endpoint Mode:  vip
Ports:
 PublishedPort = 80
  Protocol = tcp
  TargetPort = 80
  PublishMode = ingress

root@ip-172-31-23-36:/home/ubuntu#
```

10. To check all the containers at once

docker service ps <service_name>

this will help us to check the status of all the containers at once



```
root@ip-172-31-23-36:/home/ubuntu# docker service ps old-service
ID             NAME            IMAGE          NODE               DESIRED STATE   CURRENT STATE            ERROR     PORTS
wz110pr8kgh4   old-service.1   nginx:latest   ip-172-31-23-54    Running         Running 14 minutes ago
wvoize9wwc1v   old-service.2   nginx:latest   ip-172-31-18-155   Running         Running 14 minutes ago
4x2s3cpc64j6   old-service.3   nginx:latest   ip-172-31-23-36    Running         Running 14 minutes ago
root@ip-172-31-23-36:/home/ubuntu#
```

i-0efc1c15a6ac477fa (Manager)

PublicIPs: 3.87.107.239 PrivateIPs: 172.31.23.36

In this we learned:

How to create a service in docker swarm

And how docker swarm manages different replicas