Docker Swarm Set-up

Docker Swarm is an orchestration management tool that runs on Docker applications. It helps end-users in creating and deploying a cluster of Docker nodes. Each node of a Docker Swarm is a Docker daemon, and all Docker daemons interact using the Docker API.

Features of Docker Swarm

1. Creating multipul containers of the same image and distributing it to   multiple docker host that are part of docker swarm cluster

2. Scale-up and scale-down the replicas/containers

3. Swarm allows you to roll back environments to previous safe environments

4. Any communication between the manager and client nodes within the Swarm is highly secure

Prerequisites:

1. Requires minimum two hosts, which can either be virtual machine or cloud machine

2. An ubuntu account with Sudo privileges

3. Docker install on both nodes (master and worker)

How does Docker Swarm work-

In Swarm, containers are launched using service. A service is a group of containers of the same image that enables the scaling of applications. Before you can deploy a service in Docker Swarm, you must have at least one node deployed.

There are two type of nodes in Docker Swarm:

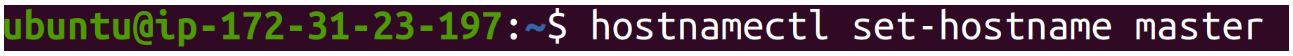
1. **Master node.** Maintains cluster management tasks
2. **Worker node.** Receives and executes tasks from the manager node

Step 1 Change hostname of both machine master and worker both

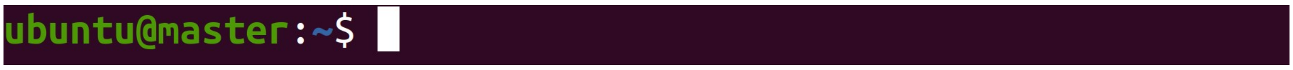
run the following command on both machine

hostnamectl set-hostname master

hostnamectl set-hostname worker



```
ubuntu@ip-172-31-23-197:~$ hostnamectl set-hostname master
```



```
ubuntu@master:~$
```

The above image shows you have changed host name successfully.

Step 2 Update software

run the following command on both machine

sudo apt update

Step 3 Install docker

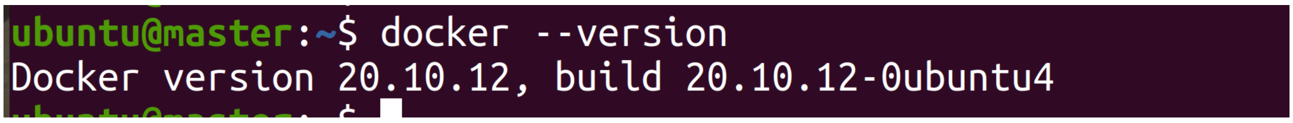
To install docker on ubuntu run the following command on both nodes

sudo apt install docker.io -y

Step 4 Check version

run the following command on both machine

docker --version



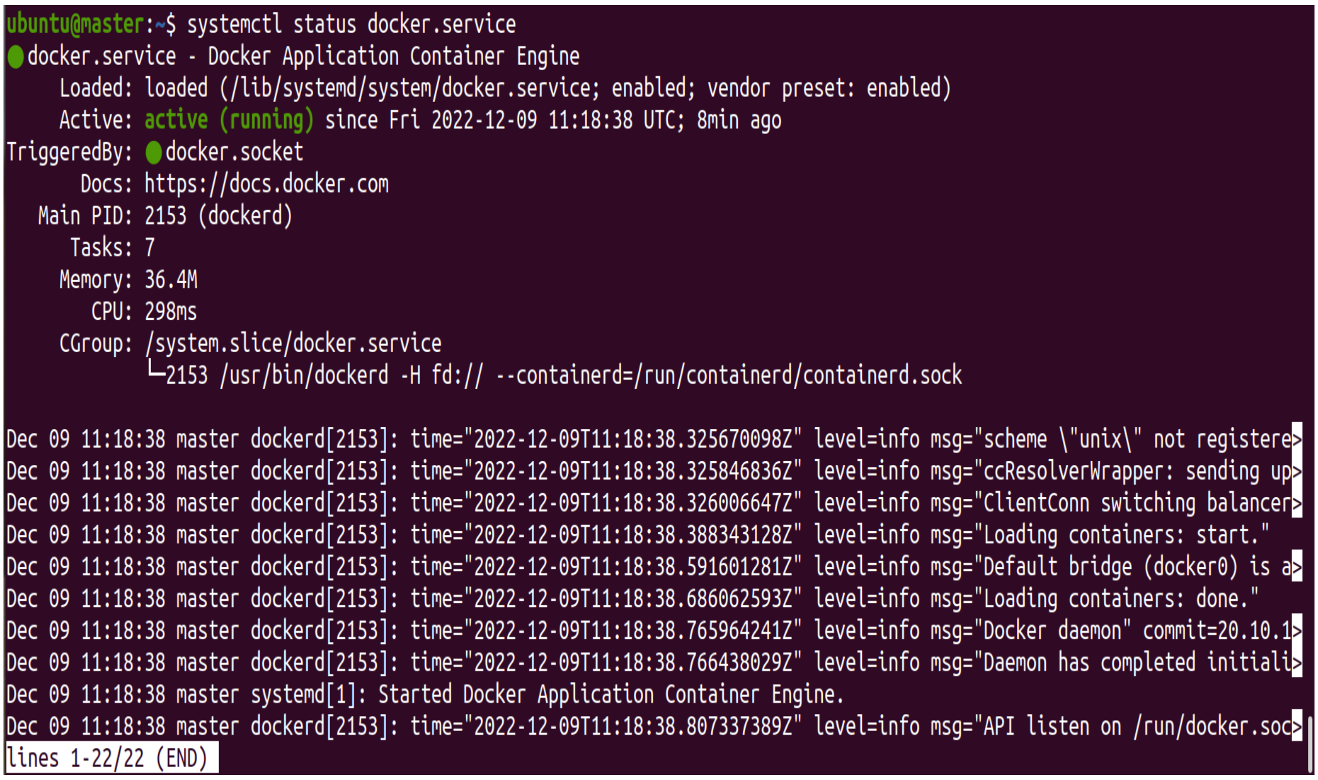
```
ubuntu@master:~$ docker --version
Docker version 20.10.12, build 20.10.12-0ubuntu4
```

The above image shows the version of docker

Step 5 Check docker status

run the following command on both nodes

systemctl status docker.service



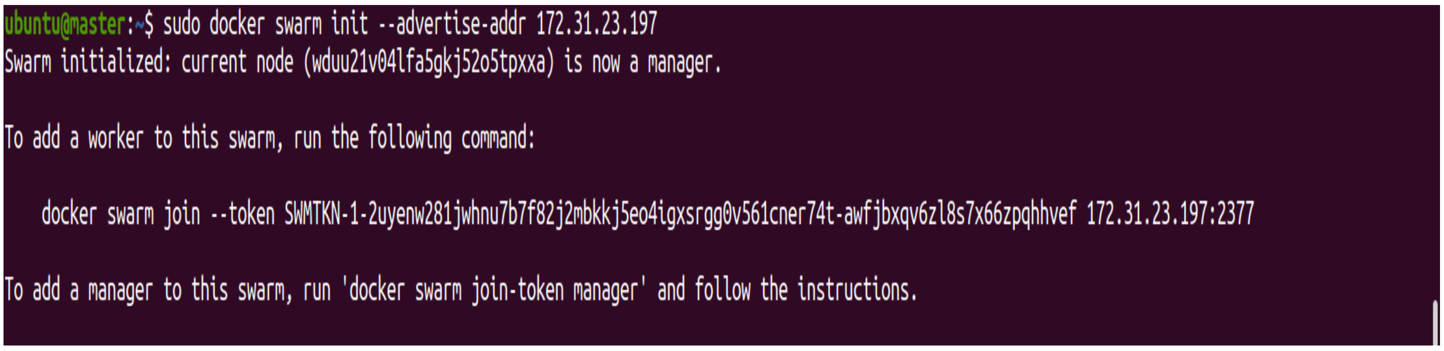
```
ubuntu@master:~$ systemctl status docker.service
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2022-12-09 11:18:38 UTC; 8min ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 2153 (dockerd)
      Tasks: 7
     Memory: 36.4M
        CPU: 298ms
     CGroup: /system.slice/docker.service
             └─2153 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Dec 09 11:18:38 master dockerd[2153]: time="2022-12-09T11:18:38.325670098Z" level=info msg="scheme \"unix\" not registere>
Dec 09 11:18:38 master dockerd[2153]: time="2022-12-09T11:18:38.325846836Z" level=info msg="ccResolverWrapper: sending up>
Dec 09 11:18:38 master dockerd[2153]: time="2022-12-09T11:18:38.326006647Z" level=info msg="ClientConn switching balancer>
Dec 09 11:18:38 master dockerd[2153]: time="2022-12-09T11:18:38.388343128Z" level=info msg="Loading containers: start."
Dec 09 11:18:38 master dockerd[2153]: time="2022-12-09T11:18:38.591601281Z" level=info msg="Default bridge (docker0) is a>
Dec 09 11:18:38 master dockerd[2153]: time="2022-12-09T11:18:38.686062593Z" level=info msg="Loading containers: done."
Dec 09 11:18:38 master dockerd[2153]: time="2022-12-09T11:18:38.765964241Z" level=info msg="Docker daemon" commit=20.10.1>
Dec 09 11:18:38 master dockerd[2153]: time="2022-12-09T11:18:38.766438029Z" level=info msg="Daemon has completed initiali>
Dec 09 11:18:38 master systemd[1]: Started Docker Application Container Engine.
Dec 09 11:18:38 master dockerd[2153]: time="2022-12-09T11:18:38.807337389Z" level=info msg="API listen on /run/docker.soc>
lines 1-22/22 (END)
```

The above image shows status of docker

Step 6: Create docker Swarm

Here, create a cluster with the pvt IP address of the manager node.

sudo Docker Swarm init --advertise-addr 172.31.23.197



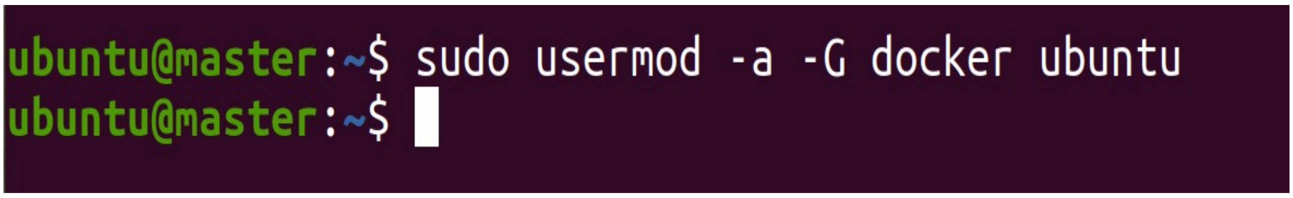
```
ubuntu@master:~$ sudo docker swarm init --advertise-addr 172.31.23.197
Swarm initialized: current node (wduu21v04lfa5gkj52o5tpxxa) is now a manager.

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

    docker swarm join --token SWMTKN-1-2uyenw281jwhnu7b7f82j2mbkkj5eo4igxsrgg0v561cner74t-awfjbxqv6zl8s7x66zpqhhvef 172.31.23.197:2377

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

The above image shows status of manager node is created successfully.

Step 7 add user of nodes into a docker group so that they not ask sudo previlage while ruuning dokcer command.

run the following command on both nodes

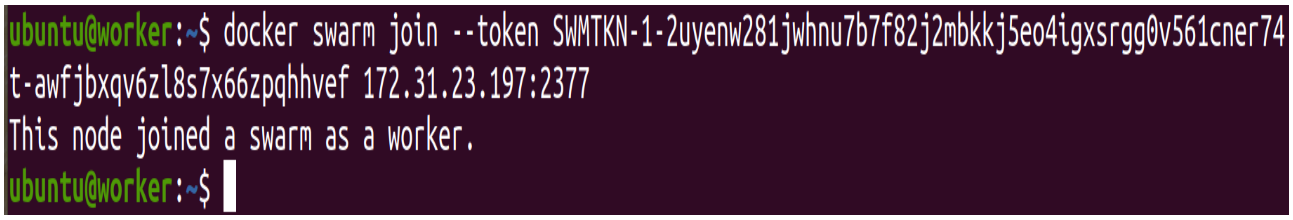
sudo usermod -a -G docker ubuntu



```
ubuntu@master:~$ sudo usermod -a -G docker ubuntu
ubuntu@master:~$
```

step 8 Now, add worker node by copying the command of the "docker swarm init" and paste the output onto the worker node:

Docker Swarm join --token SWMTKN-1- xxxxx

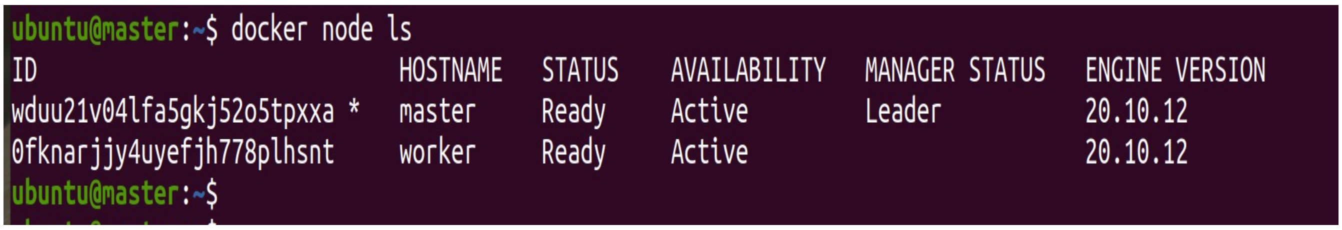


```
ubuntu@worker:~$ docker swarm join --token SWMTKN-1-2uyenw281jwhnu7b7f82j2mbkkj5eo4igxsrgg0v561cner74
t-awfjbxqv6zl8s7x66zpqhhvef 172.31.23.197:2377
This node joined a swarm as a worker.
ubuntu@worker:~$
```

The above image shows status of worker node is joined a swarm successfully.

Step 9 Now, go back to the manager node and execute the following command to list the node.

docker node ls



```
ubuntu@master:~$ docker node ls
ID                            HOSTNAME   STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
wduu21v04lfa5gkj52o5tpxxa *   master     Ready     Active         Leader           20.10.12
0fknarjjy4uyefjh778plhsnt     worker     Ready     Active                          20.10.12
ubuntu@master:~$
```

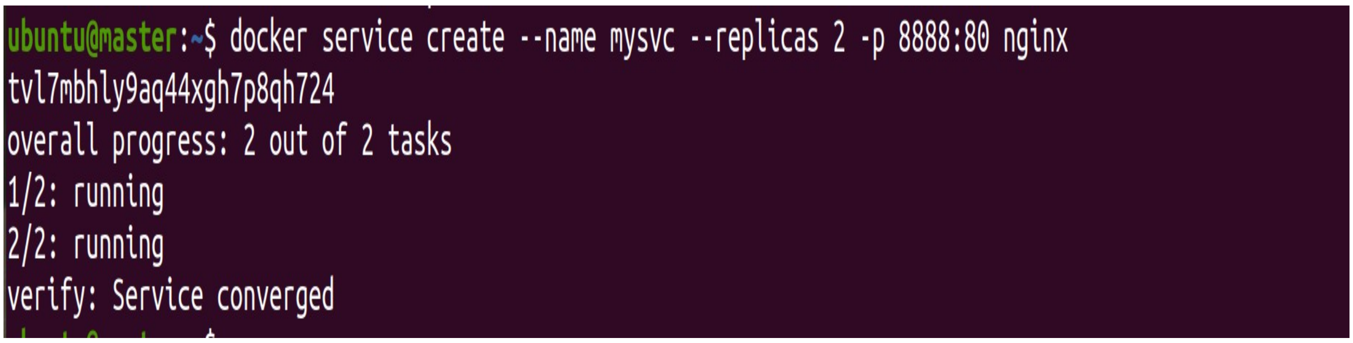
The above image shows Swarm Cluster created successfully.

Now, launch the service in Swarm Mode.

Step 10 - Go to your the manager node and execute the command below to deploy a service of nginx.

Here we will create a service of nginx images with 2 replicas and perform port mapping

docker service create –name mysvc –replicas 2 -p 8888:80 nginx

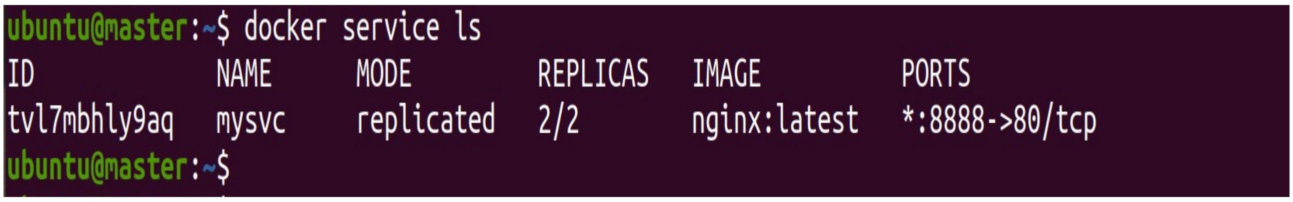


```
ubuntu@master:~$ docker service create --name mysvc --replicas 2 -p 8888:80 nginx
tvl7mbhly9aq44xgh7p8qh724
overall progress: 2 out of 2 tasks
1/2: running
2/2: running
verify: Service converged
```

Step 11 List all service

run the following command

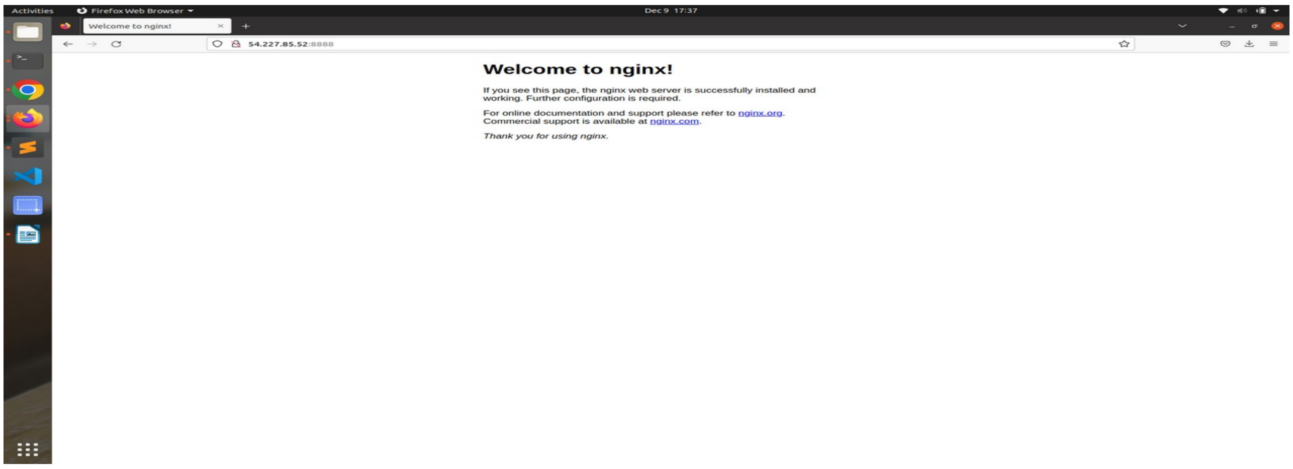
docker service ls



```
ubuntu@master:~$ docker service ls
ID             NAME      MODE         REPLICAS   IMAGE          PORTS
tvl7mbhly9aq   mysvc     replicated   2/2        nginx:latest   *:8888->80/tcp
ubuntu@master:~$
```

Step 12 Expose conatiner for public access using public IP of nodes and give ports number **"54.227.85.52:8888"**

Note – make sure your port number should be allow in your security group.



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.

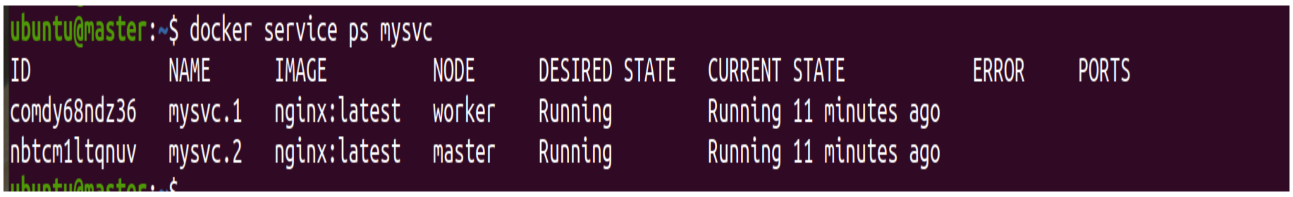
Thank you for using nginx.

The above images show we successfully access the conatiner form outside

Step 13 – To check the service

run the following commands

docker service ls <service_name>



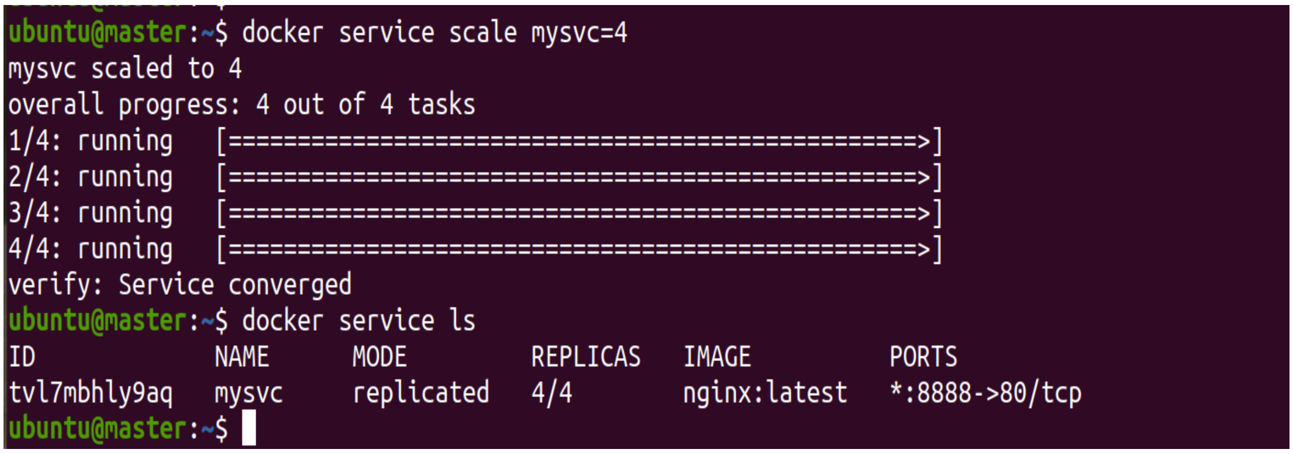
```
ubuntu@master:~$ docker service ps mysvc
ID             NAME      IMAGE          NODE      DESIRED STATE   CURRENT STATE            ERROR     PORTS
comdy68ndz36   mysvc.1   nginx:latest   worker    Running         Running 11 minutes ago
nbtcm1ltqnuv   mysvc.2   nginx:latest   master    Running         Running 11 minutes ago
```

Step 14- Using the concept of scaling we can increase or reduce the replicas

Now scale up the replicas form 2 to 4

run the following command

docker service scale mysvc=4



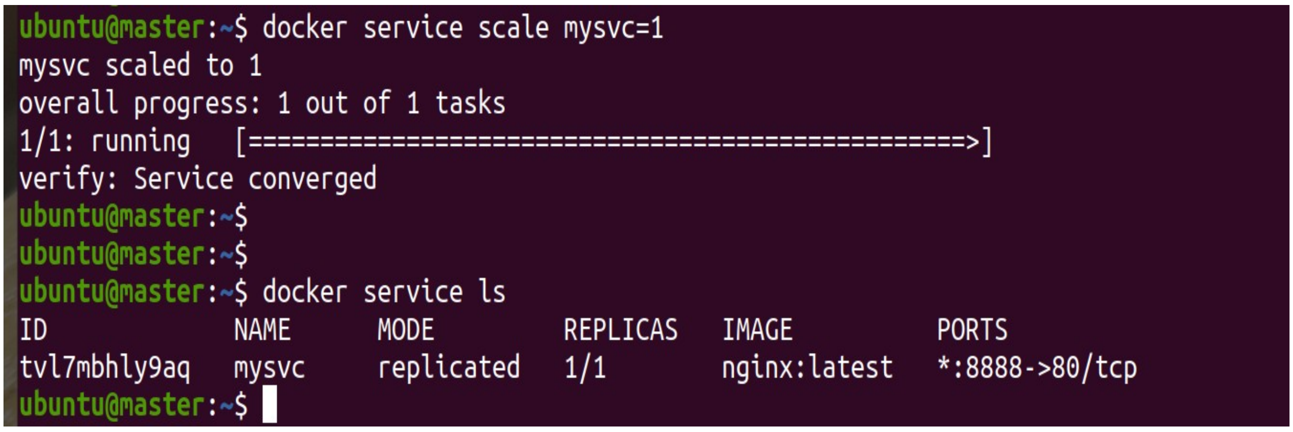
```
ubuntu@master:~$ docker service scale mysvc=4
mysvc scaled to 4
overall progress: 4 out of 4 tasks
1/4: running   [==================================================>]
2/4: running   [==================================================>]
3/4: running   [==================================================>]
4/4: running   [==================================================>]
verify: Service converged
ubuntu@master:~$ docker service ls
ID             NAME      MODE         REPLICAS   IMAGE          PORTS
tvl7mbhly9aq   mysvc     replicated   4/4        nginx:latest   *:8888->80/tcp
ubuntu@master:~$
```

The above image show replicas has increased

Step 15 -Now scale down the replicas form 4 to 1

run the following command

docker service scale mysvc=1



```
ubuntu@master:~$ docker service scale mysvc=1
mysvc scaled to 1
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
ubuntu@master:~$
ubuntu@master:~$
ubuntu@master:~$ docker service ls
ID             NAME      MODE         REPLICAS   IMAGE          PORTS
tvl7mbhly9aq   mysvc     replicated   1/1        nginx:latest   *:8888->80/tcp
ubuntu@master:~$
```

The above image show replica has reduced