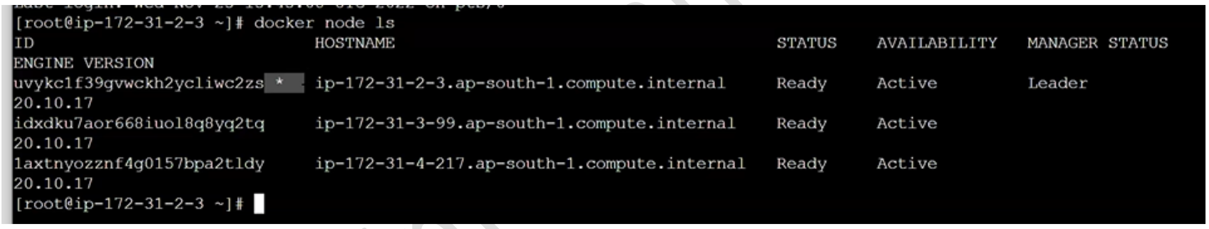
Summary

Session No- 18

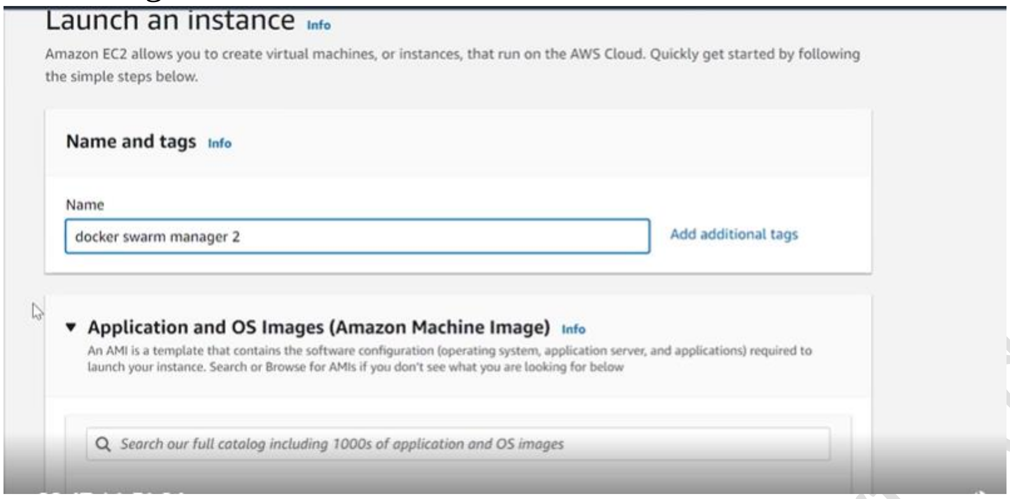
- If we have a bigger workload & to manage the workload inside the containers we need multiple servers or host
- To manage multiple nodes, we need a centralized program called a manager nodes
- **docker node ls command** is used to list the nodes in the cluster
    - ***** Means we are running the command from that server



```
[root@ip-172-31-2-3 ~]# docker node ls
ID                            HOSTNAME                                    STATUS    AVAILABILITY   MANAGER STATUS
ENGINE VERSION
uvykc1f39gvwckh2ycliwc2zs *   ip-172-31-2-3.ap-south-1.compute.internal   Ready     Active         Leader
20.10.17
idxdku7aor668iuol8q8yq2tq     ip-172-31-3-99.ap-south-1.compute.internal  Ready     Active
20.10.17
1axtnyozznf4g0157bpa2tldy     ip-172-31-4-217.ap-south-1.compute.internal Ready     Active
20.10.17
[root@ip-172-31-2-3 ~]#
```

- If we want to do management in a cluster it can be done only through the manager node
- Management means if we want to scale the cluster so the manager has the capability to do it
- If the client wants to connect to the web app it does not need a manager node, a client can directly connect to the worker node
- If the manager nodes go down & there are no manager nodes left, so here are manager is a single point of failure
- If we increase the manager node & any one of the manager nodes goes down we have another manager node with us this concept is called high availability
- If we don’t want a single point of failure we can go for high availability

- Launching 2 new amazon Linux instances



Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

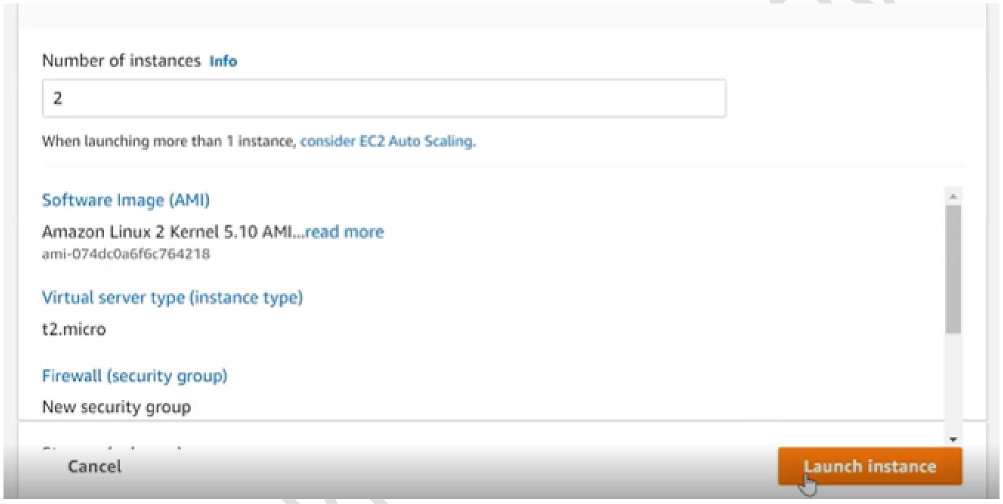
Name: docker swarm manager 2

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

- Creating 2 instances



Number of instances Info: 2

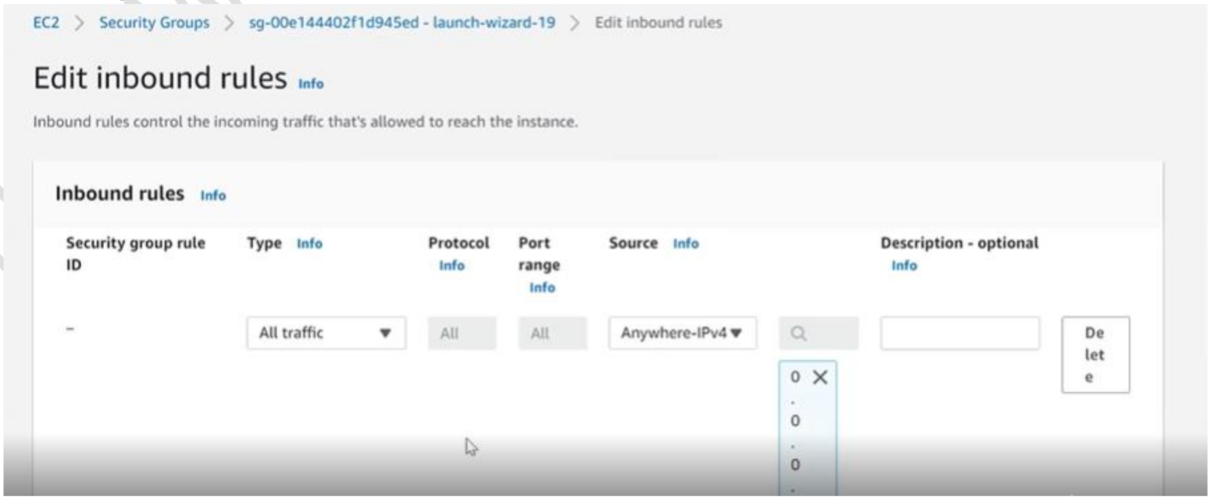
When launching more than 1 instance, consider EC2 Auto Scaling.

Software Image (AMI): Amazon Linux 2 Kernel 5.10 AMI...read more
ami-074dc0a6f6c764218

Virtual server type (instance type): t2.micro

Firewall (security group): New security group

- Now let's say we have 3 manager nodes & anywhere any fault happens our setup can tolerate the fault this is called fault tolerance
- Creating new inbound rules for the instances



EC2 > Security Groups > sg-00e144402f1d945ed - launch-wizard-19 > Edit inbound rules

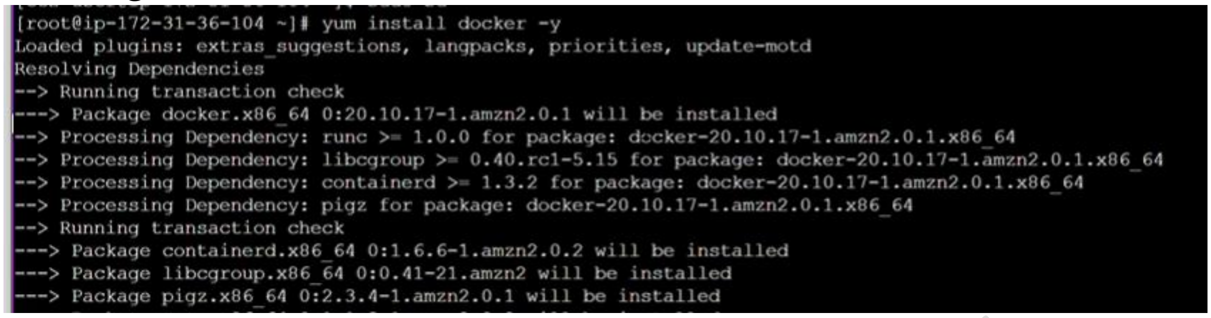
Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules Info

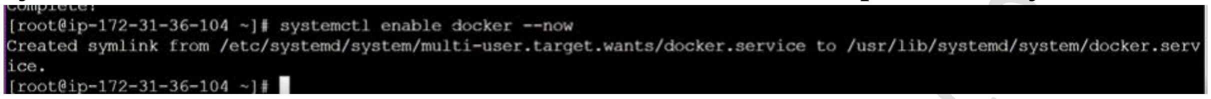
Security group rule ID	Type	Protocol	Port range	Source	Description - optional
–	All traffic	All	All	Anywhere-IPv4 0.0.0.0/0	

- Installing docker on the instance



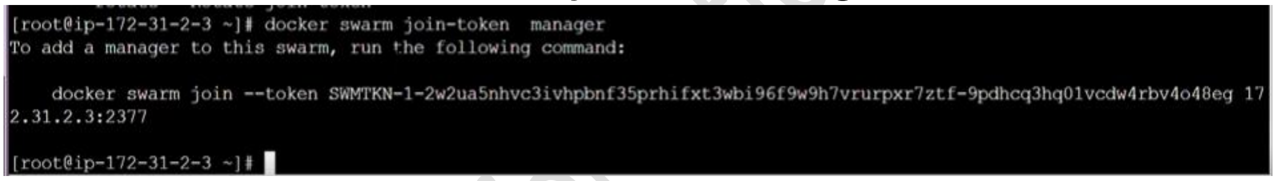
```
[root@ip-172-31-36-104 ~]# yum install docker -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
---> Package docker.x86_64 0:20.10.17-1.amzn2.0.1 will be installed
--> Processing Dependency: runc >= 1.0.0 for package: docker-20.10.17-1.amzn2.0.1.x86_64
--> Processing Dependency: libcgroup >= 0.40.rc1-5.15 for package: docker-20.10.17-1.amzn2.0.1.x86_64
--> Processing Dependency: containerd >= 1.3.2 for package: docker-20.10.17-1.amzn2.0.1.x86_64
--> Processing Dependency: pigz for package: docker-20.10.17-1.amzn2.0.1.x86_64
--> Running transaction check
---> Package containerd.x86_64 0:1.6.6-1.amzn2.0.2 will be installed
---> Package libcgroup.x86_64 0:0.41-21.amzn2 will be installed
---> Package pigz.x86_64 0:2.3.4-1.amzn2.0.1 will be installed
```

- **systemctl enable docker --now** to start & enable docker permanently



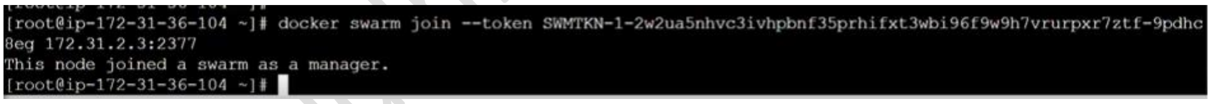
```
[root@ip-172-31-36-104 ~]# systemctl enable docker --now
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.serv
ice.
[root@ip-172-31-36-104 ~]#
```

- If we want to join a node in the existing cluster every node has different tokens, a manager has a different token & a worker node has a different token
- Creating a token for the manager node
    - Command:- **docker swarm join-token manager**



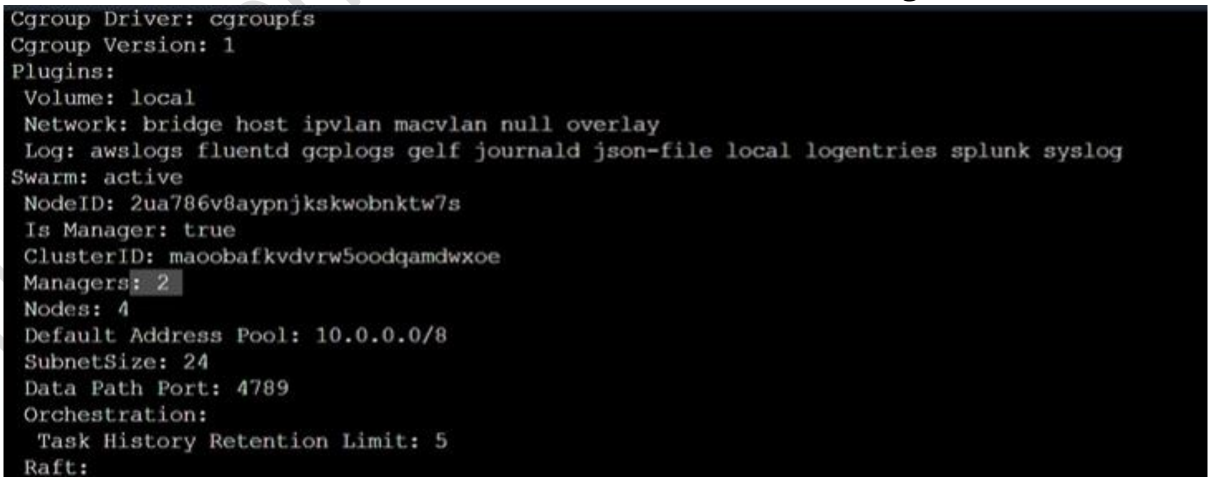
```
[root@ip-172-31-2-3 ~]# docker swarm join-token  manager
To add a manager to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-2w2ua5nhvc3ivhpbnf35prhifxt3wbi96f9w9h7vrurpxr7ztf-9pdhcq3hq01vcdw4rbv4o48eg 17
2.31.2.3:2377

[root@ip-172-31-2-3 ~]#
```

- Running the command given by the docker swarm join-token manager on the new node



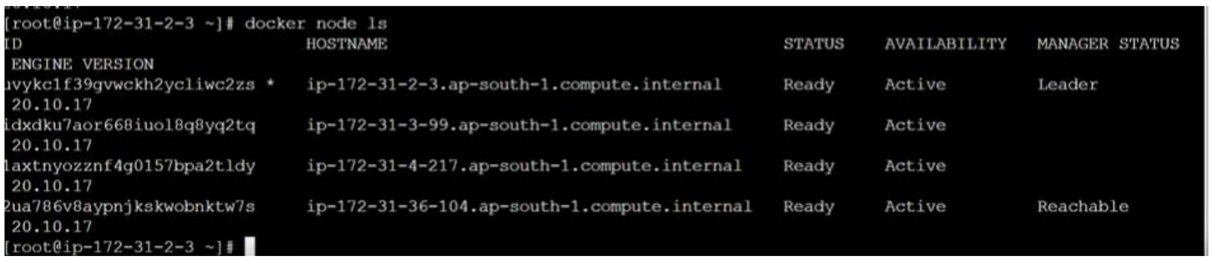
```
[root@ip-172-31-36-104 ~]# docker swarm join --token SWMTKN-1-2w2ua5nhvc3ivhpbnf35prhifxt3wbi96f9w9h7vrurpxr7ztf-9pdhc
8eg 172.31.2.3:2377
This node joined a swarm as a manager.
[root@ip-172-31-36-104 ~]#
```

- Now if we run the docker info command we can see 2 manager nodes



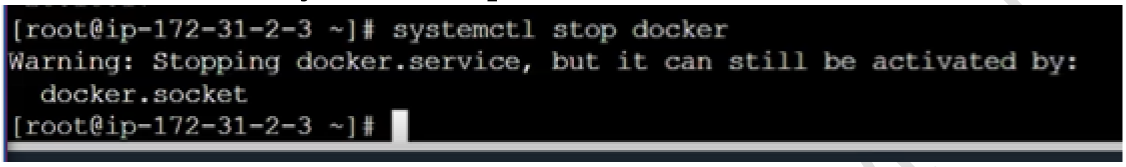
```
Cgroup Driver: cgroupfs
Cgroup Version: 1
Plugins:
 Volume: local
 Network: bridge host ipvlan macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
Swarm: active
 NodeID: 2ua786v8aypnjkskwobnktw7s
 Is Manager: true
 ClusterID: maoobafkvdvrw5oodqamdwxoe
 Managers: 2
 Nodes: 4
 Default Address Pool: 10.0.0.0/8
 SubnetSize: 24
 Data Path Port: 4789
 Orchestration:
  Task History Retention Limit: 5
 Raft:
```

- Now if we list the docker nodes we can see node is added & is reachable



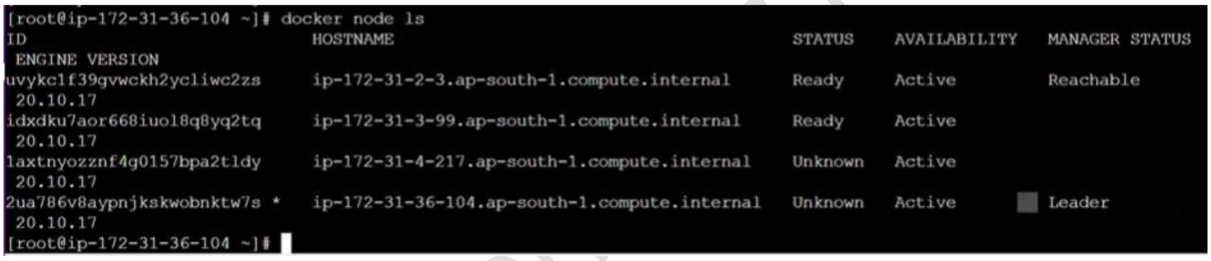
```
[root@ip-172-31-2-3 ~]# docker node ls
ID                          HOSTNAME                                      STATUS    AVAILABILITY   MANAGER STATUS
 ENGINE VERSION
uvykc1f39gvwckh2ycliwc2zs *   ip-172-31-2-3.ap-south-1.compute.internal     Ready     Active         Leader
 20.10.17
idxdku7aor668iuol8q8yq2tq     ip-172-31-3-99.ap-south-1.compute.internal    Ready     Active
 20.10.17
1axtnyozznf4g0157bpa2tldy     ip-172-31-4-217.ap-south-1.compute.internal   Ready     Active
 20.10.17
2ua786v8aypnjkskwobnktw7s     ip-172-31-36-104.ap-south-1.compute.internal  Ready     Active         Reachable
 20.10.17
[root@ip-172-31-2-3 ~]#
```

- Stopping the docker services on the leader node
  - Command:- **systemctl stop docker**



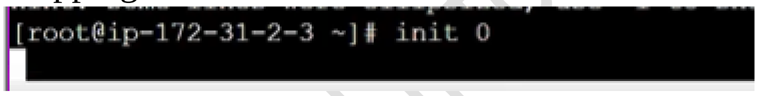
```
[root@ip-172-31-2-3 ~]# systemctl stop docker
Warning: Stopping docker.service, but it can still be activated by:
  docker.socket
[root@ip-172-31-2-3 ~]#
```

- Now if we list the docker nodes we can see the other node became the leader node



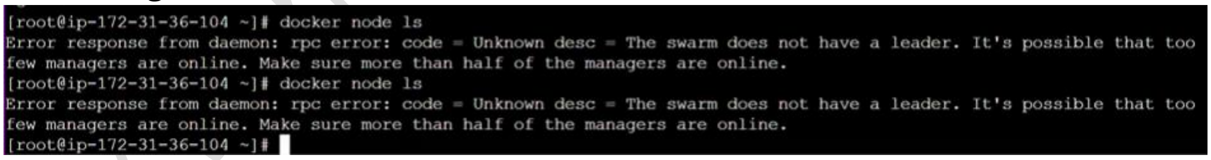
```
[root@ip-172-31-36-104 ~]# docker node ls
ID                          HOSTNAME                                      STATUS    AVAILABILITY   MANAGER STATUS
 ENGINE VERSION
uvykc1f39gvwckh2ycliwc2zs     ip-172-31-2-3.ap-south-1.compute.internal     Ready     Active         Reachable
 20.10.17
idxdku7aor668iuol8q8yq2tq     ip-172-31-3-99.ap-south-1.compute.internal    Ready     Active
 20.10.17
1axtnyozznf4g0157bpa2tldy     ip-172-31-4-217.ap-south-1.compute.internal   Unknown   Active
 20.10.17
2ua786v8aypnjkskwobnktw7s *   ip-172-31-36-104.ap-south-1.compute.internal  Unknown   Active         Leader
 20.10.17
[root@ip-172-31-36-104 ~]#
```

- Stopping the master instance with the **init 0** command



```
[root@ip-172-31-2-3 ~]# init 0
```

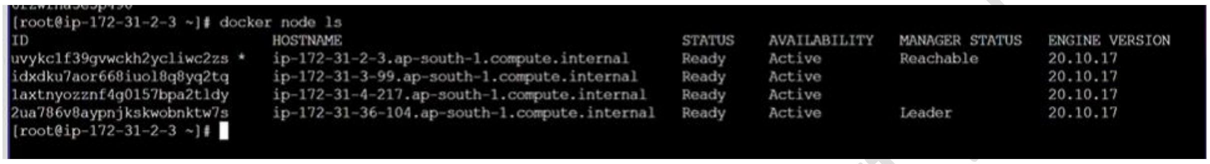
- If the main master goes down and we list the node from another leader it is showing error it is a kind of cluster shutdown



```
[root@ip-172-31-36-104 ~]# docker node ls
Error response from daemon: rpc error: code = Unknown desc = The swarm does not have a leader. It's possible that too
few managers are online. Make sure more than half of the managers are online.
[root@ip-172-31-36-104 ~]# docker node ls
Error response from daemon: rpc error: code = Unknown desc = The swarm does not have a leader. It's possible that too
few managers are online. Make sure more than half of the managers are online.
[root@ip-172-31-36-104 ~]#
```

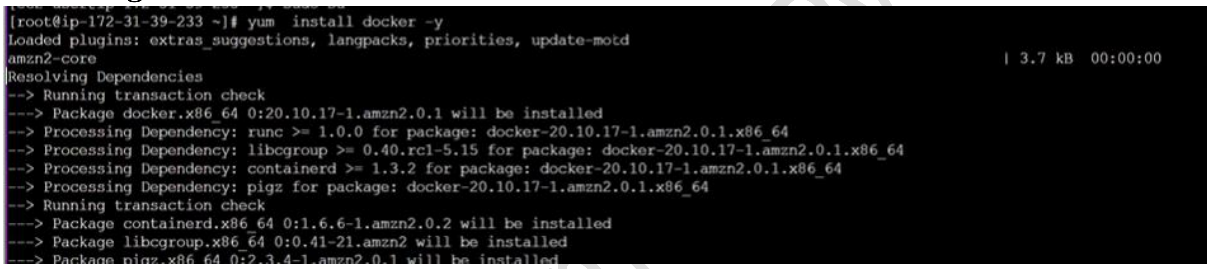
- The main purpose of the master node is to manage the task in the worker node. The master node maintains the current information of the worker node
- Raft is a kind of protocol or algorithm to share information between two different nodes. The main duty of the RAFT algorithm is to share the current status of the cluster with another master node
- If both masters give a different task, then there will be some kind of conflict between the masters this is called a split-brain problem
- That's the reason in the high availability where we have more than one master, we have to make one master node as a leader and others as a manager

- If the main master nodes goes down and we have two more master nodes in that case of high availability election kind of thing happens based on multiple parameters e.g. which master joins first
- **(N-1)/2** this formula from quorum will help us to find the fault tolerance where N is no of nodes
- Now if we start the main master again and list the nodes, we can see it is reachable which means it is a passive node and the leader is an active node



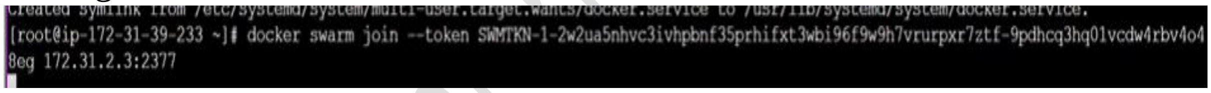
```
[root@ip-172-31-2-3 ~]# docker node ls
ID                            HOSTNAME                                      STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
uvykc1f39gvwckh2ycliwc2zs *   ip-172-31-2-3.ap-south-1.compute.internal     Ready     Active         Reachable        20.10.17
idxdku7aor668iuol8q8yq2tq     ip-172-31-3-99.ap-south-1.compute.internal    Ready     Active                          20.10.17
laxtnyozznf4g0157bpa2tldy     ip-172-31-4-217.ap-south-1.compute.internal   Ready     Active                          20.10.17
2ua786v8aypnjkskwobnktw7s     ip-172-31-36-104.ap-south-1.compute.internal  Ready     Active         Leader           20.10.17
[root@ip-172-31-2-3 ~]#
```

- Installing docker on 3 rd. master node



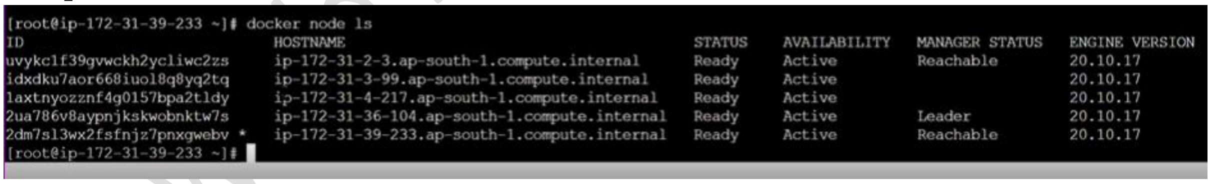
```
[root@ip-172-31-39-233 ~]# yum  install docker -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                                                                          | 3.7 kB  00:00:00
Resolving Dependencies
--> Running transaction check
---> Package docker.x86_64 0:20.10.17-1.amzn2.0.1 will be installed
--> Processing Dependency: runc >= 1.0.0 for package: docker-20.10.17-1.amzn2.0.1.x86_64
--> Processing Dependency: libcgroup >= 0.40.rc1-5.15 for package: docker-20.10.17-1.amzn2.0.1.x86_64
--> Processing Dependency: containerd >= 1.3.2 for package: docker-20.10.17-1.amzn2.0.1.x86_64
--> Processing Dependency: pigz for package: docker-20.10.17-1.amzn2.0.1.x86_64
--> Running transaction check
---> Package containerd.x86_64 0:1.6.6-1.amzn2.0.2 will be installed
---> Package libcgroup.x86_64 0:0.41-21.amzn2 will be installed
---> Package pigz.x86_64 0:2.3.4-1.amzn2.0.1 will be installed
```

- Joining the third master node in the cluster with token



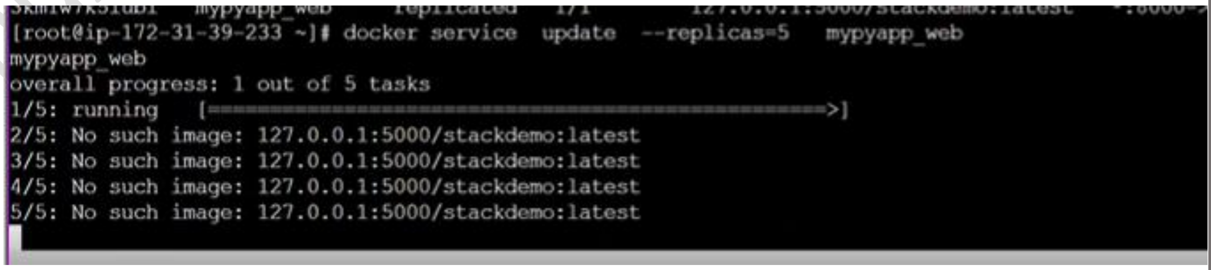
```
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.
[root@ip-172-31-39-233 ~]# docker swarm join --token SWMTKN-1-2w2ua5nhvc3ivhpbnf35prhifxt3wbi96f9w9h7vrurpxr7ztf-9pdhcq3hq01vcdw4rbv4o4
8eg 172.31.2.3:2377
```

- If we list the node in a cluster we can see node is joined in a cluster and is a passive node



```
[root@ip-172-31-39-233 ~]# docker node ls
ID                            HOSTNAME                                       STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
uvykc1f39gvwckh2ycliwc2zs     ip-172-31-2-3.ap-south-1.compute.internal      Ready     Active         Reachable        20.10.17
idxdku7aor668iuol8q8yq2tq     ip-172-31-3-99.ap-south-1.compute.internal     Ready     Active                          20.10.17
laxtnyozznf4g0157bpa2tldy     ip-172-31-4-217.ap-south-1.compute.internal    Ready     Active                          20.10.17
2ua786v8aypnjkskwobnktw7s     ip-172-31-36-104.ap-south-1.compute.internal   Ready     Active         Leader           20.10.17
2dm7sl3wx2fsfnjz7pnxgwebv *   ip-172-31-39-233.ap-south-1.compute.internal   Ready     Active         Reachable        20.10.17
[root@ip-172-31-39-233 ~]#
```

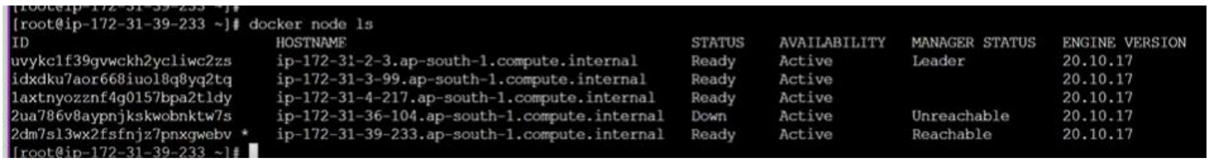
- Scaling with the docker service update command
  - Command: - **docker service update –replicas=5 (service name)**



```
[root@ip-172-31-39-233 ~]# docker service  update  --replicas=5   mypyapp_web
mypyapp_web
overall progress: 1 out of 5 tasks
1/5: running   [==================================================>]
2/5: No such image: 127.0.0.1:5000/stackdemo:latest
3/5: No such image: 127.0.0.1:5000/stackdemo:latest
4/5: No such image: 127.0.0.1:5000/stackdemo:latest
5/5: No such image: 127.0.0.1:5000/stackdemo:latest
```

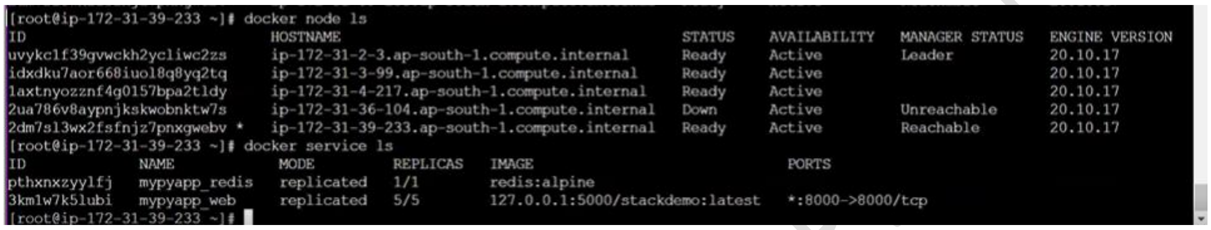
- In the docker service update command we can change replicas of one particular service at one point in time but in the docker service scale command we can scale multiple services at one point in time

- If the leader node goes down, so according to the formula of fault tolerance we have three nodes and if one of the nods goes down then our cluster can tolerate



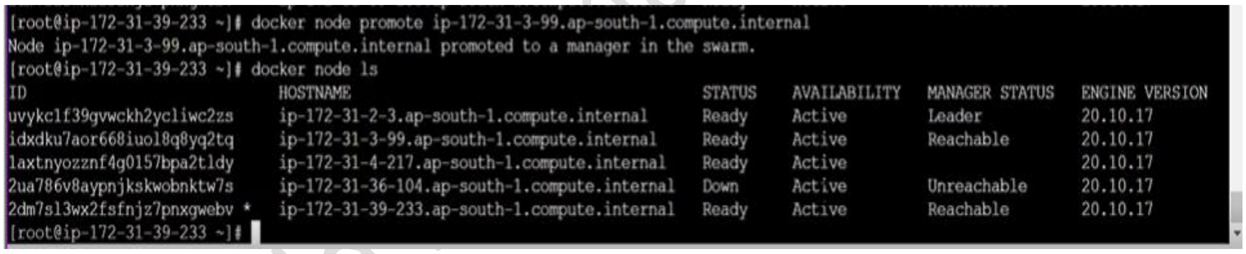
```
[root@ip-172-31-39-233 ~]# docker node ls
ID                            HOSTNAME                                    STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
uvykc1f39gvwckh2ycliwc2zs     ip-172-31-2-3.ap-south-1.compute.internal   Ready     Active         Leader           20.10.17
idxdku7aor668iuol8q8yq2tq     ip-172-31-3-99.ap-south-1.compute.internal  Ready     Active                          20.10.17
1axtnyozznf4g0157bpa2tldy     ip-172-31-4-217.ap-south-1.compute.internal Ready     Active                          20.10.17
2ua786v8aypnjkskwobnktw7s     ip-172-31-36-104.ap-south-1.compute.internal Down     Active         Unreachable      20.10.17
2dm7sl3wx2fsfnjz7pnxgwebv *   ip-172-31-39-233.ap-south-1.compute.internal Ready    Active         Reachable        20.10.17
[root@ip-172-31-39-233 ~]#
```

- Now we can list the nodes from another active leader because our cluster has one fault tolerance



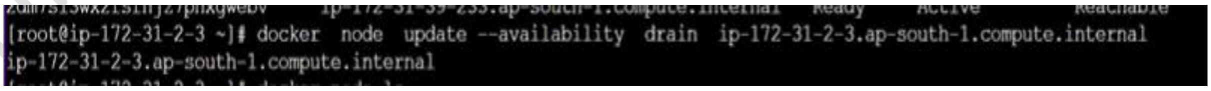
```
[root@ip-172-31-39-233 ~]# docker node ls
ID                            HOSTNAME                                    STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
uvykc1f39gvwckh2ycliwc2zs     ip-172-31-2-3.ap-south-1.compute.internal   Ready     Active         Leader           20.10.17
idxdku7aor668iuol8q8yq2tq     ip-172-31-3-99.ap-south-1.compute.internal  Ready     Active                          20.10.17
1axtnyozznf4g0157bpa2tldy     ip-172-31-4-217.ap-south-1.compute.internal Ready     Active                          20.10.17
2ua786v8aypnjkskwobnktw7s     ip-172-31-36-104.ap-south-1.compute.internal Down     Active         Unreachable      20.10.17
2dm7sl3wx2fsfnjz7pnxgwebv *   ip-172-31-39-233.ap-south-1.compute.internal Ready    Active         Reachable        20.10.17
[root@ip-172-31-39-233 ~]# docker service ls
ID             NAME            MODE         REPLICAS   IMAGE                               PORTS
pthxnxzyylfj   mypyapp_redis   replicated   1/1        redis:alpine
3km1w7k5lubi   mypyapp_web     replicated   5/5        127.0.0.1:5000/stackdemo:latest     *:8000->8000/tcp
[root@ip-172-31-39-233 ~]#
```

- We can change any node to a master or worker node at any point in time it is called promotion or demotion
- Promoting worker node to manager node
  - Command:- **docker node promote (Hostname)**



```
[root@ip-172-31-39-233 ~]# docker node promote ip-172-31-3-99.ap-south-1.compute.internal
Node ip-172-31-3-99.ap-south-1.compute.internal promoted to a manager in the swarm.
[root@ip-172-31-39-233 ~]# docker node ls
ID                            HOSTNAME                                    STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
uvykc1f39gvwckh2ycliwc2zs     ip-172-31-2-3.ap-south-1.compute.internal   Ready     Active         Leader           20.10.17
idxdku7aor668iuol8q8yq2tq     ip-172-31-3-99.ap-south-1.compute.internal  Ready     Active         Reachable        20.10.17
1axtnyozznf4g0157bpa2tldy     ip-172-31-4-217.ap-south-1.compute.internal Ready     Active                          20.10.17
2ua786v8aypnjkskwobnktw7s     ip-172-31-36-104.ap-south-1.compute.internal Down     Active         Unreachable      20.10.17
2dm7sl3wx2fsfnjz7pnxgwebv *   ip-172-31-39-233.ap-south-1.compute.internal Ready    Active         Reachable        20.10.17
[root@ip-172-31-39-233 ~]#
```

- The manager works as a master node and worker node. The good practice is to make the manager node a master-only node
- Making the manager master-only node
  - Command :- **docker node update --availability drain (hostname)**
  - Drain means if any container running on the master node stop it and launch the container on a worker node



```
[root@ip-172-31-2-3 ~]# docker  node  update --availability  drain  ip-172-31-2-3.ap-south-1.compute.internal
ip-172-31-2-3.ap-south-1.compute.internal
```

- The drain is also used at the time of maintenance