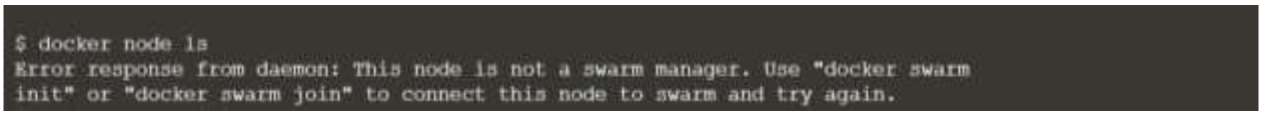
EXPERIMENT - 7 (Docker Swarm)

Name: Ankur Sehrawat
Roll No: R171218023

Open two terminals one for being the master and the other for being the slave (node), check if the swarm is active or not by the command ‘docker node ls’. If the swarm is active the command will list the master and the slaves but if the swarm isn’t active the command will throw an error.

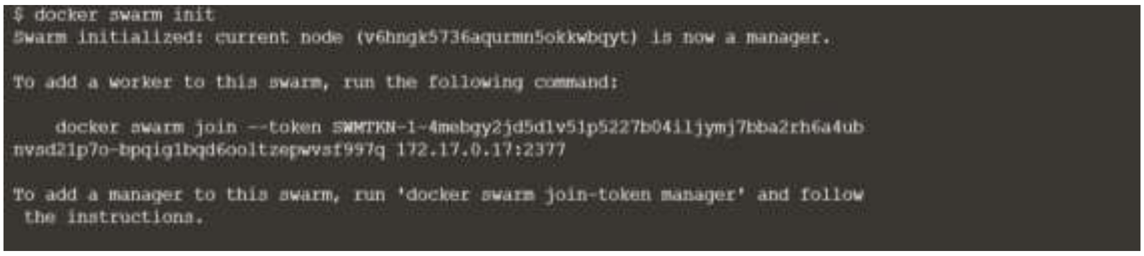
Terminal 1 -



```
$ docker node ls
Error response from daemon: This node is not a swarm manager. Use "docker swarm
init" or "docker swarm join" to connect this node to swarm and try again.
```

Activate the swarm using the command ‘docker swarm init’, the output of the command will have a command having a unique token and the IP address. Through this command, we can connect the other terminal as a worker to the master by running the command in the terminal.
The basic syntax of the command from the output is ‘docker swarm join –token <token> <ip and port no.>’

Terminal 1 -



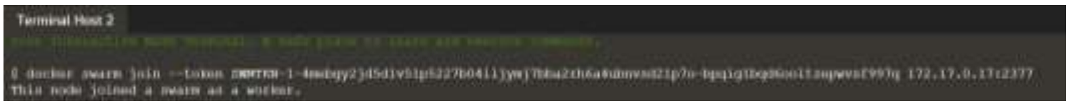
```
$ docker swarm init
Swarm initialized: current node (v6hngk5736aqurmn5okkwbqyt) is now a manager.

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

    docker swarm join --token SWMTKN-1-4mebgy2jd5dlv51p5227b04iljymj7bba2rh6a4ub
nvsd21p7o-bpqig1bqd6ooltzepwvsf997q 172.17.0.17:2377

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

Terminal 2 -

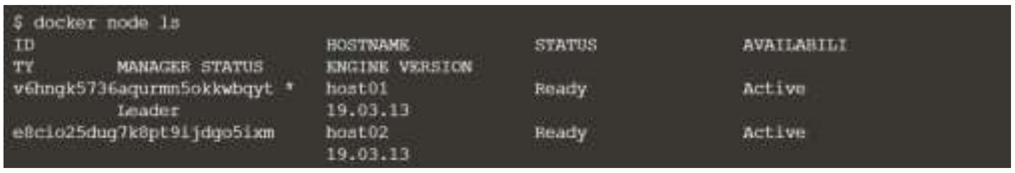


```
Terminal Host 2
[illegible]

$ docker swarm join --token SWMTKN-1-4mebgy2jd5dlv51p5227b04iljymj7bba2rh6a4ubnvsd21p7o-bpqig1bqd6ooltzepwvsf997q 172.17.0.17:2377
This node joined a swarm as a worker.
```

Again run the command ‘docker node ls’ to view all the nodes in the swarm we just activated, the nodes which are acting as a worker will be written directly but the node which is acting as a manager will have ‘Leader’ written with it and a ‘*’ mark on it.

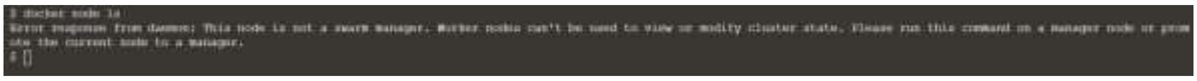
Terminal 1 -



```
$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILI
TY          MANAGER STATUS    ENGINE VERSION
v6hngk5736aqurmn5okkwbqyt *   host01              Ready               Active
            Leader            19.03.13
e8cio25dug7k8pt9ijdgo5ixm     host02              Ready               Active
                              19.03.13
```

Only the manager can access the information regarding the number of rows and nodes in the swarm so if we try to run the command in a worker node terminal, it will fail.

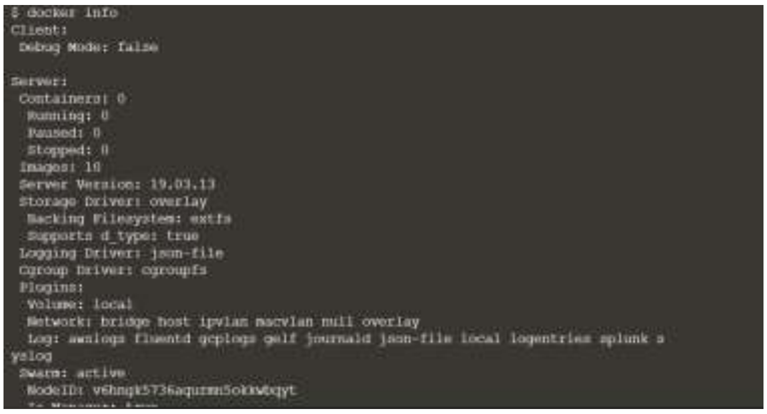
Terminal 2 -



```
$ docker node ls
Error response from daemon: This node is not a swarm manager. Worker nodes can't be used to view or modify cluster state. Please run this command on a manager node or prom
ote the current node to a manager.
$ []
```

We can check all the details of the swarm and the nodes by running the command ‘docker info’.

Terminal 1 -

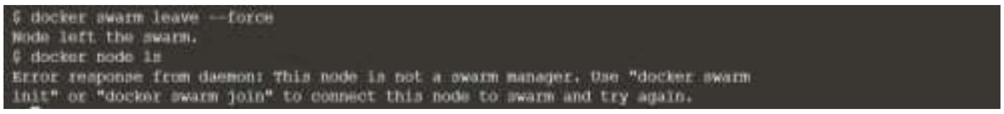


```
$ docker info
Client:
 Debug Mode: false

Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 10
 Server Version: 19.03.13
 Storage Driver: overlay
  Backing Filesystem: extfs
  Supports d_type: true
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk s
yslog
 Swarm: active
  NodeID: v6hnqk5736aqurmn5okkwbqyt
```

When we want to remove a node from the swarm we use the command ‘docker swarm leave --force’ if the node is a manager and ‘docker swarm leave’ if it’s a worker node. If a manager leaves the swarm, the manager properties are automatically inherited by a worker node and it becomes the new manager

Terminal 1 -



```
$ docker swarm leave --force
Node left the swarm.
$ docker node ls
Error response from daemon: This node is not a swarm manager. Use "docker swarm
init" or "docker swarm join" to connect this node to swarm and try again.
```