Setting up a Docker instance

- To verify the installation:
    1. Open the command-line interface
    2. Type in the command:

        docker --version

```
root@ip-172-31-17-73:~# apt install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
docker.io is already the newest version (18.09.7-0ubuntu1~18.04.3).
The following packages were automatically installed and are no longer required:
  apache2-bin apache2-data libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.2-0
Use 'apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 9 not upgraded.
root@ip-172-31-17-73:~# docker version
Client:
 Version:           18.09.7
 API version:       1.39
 Go version:        go1.10.1
 Git commit:        2d0083d
 Built:             Wed Jul  3 12:13:59 2019
 OS/Arch:           linux/amd64
 Experimental:      false

Server:
 Engine:
  Version:          18.09.7
  API version:      1.39 (minimum version 1.12)
  Go version:       go1.10.1
  Git commit:       2d0083d
  Built:            Mon Jul  1 19:31:12 2019
  OS/Arch:          linux/amd64
  Experimental:     false
root@ip-172-31-17-73:~#
```

Setting up Docker swarm with multiple nodes

- Edit the **/etc/hosts** file across the two nodes via **gedit** or **vim** and make the following changes:

```
172.31.17.73dockermanager
```

172.31.86.69dockerworker1

- After modifying the host file with the details mentioned above, check the connectivity with **ping** between all the nodes
    - From Docker Manager Host instance:

```
root@ip-172-31-17-73:~# ping dockerworker1
PING dockerworker1 (172.31.86.69) 56(84) bytes of data.
64 bytes from dockerworker1 (172.31.86.69): icmp_seq=1 ttl=64 time=0.637 ms
64 bytes from dockerworker1 (172.31.86.69): icmp_seq=2 ttl=64 time=0.727 ms
64 bytes from dockerworker1 (172.31.86.69): icmp_seq=3 ttl=64 time=0.673 ms
64 bytes from dockerworker1 (172.31.86.69): icmp_seq=4 ttl=64 time=5.00 ms
64 bytes from dockerworker1 (172.31.86.69): icmp_seq=5 ttl=64 time=0.674 ms
64 bytes from dockerworker1 (172.31.86.69): icmp_seq=6 ttl=64 time=0.647 ms
64 bytes from dockerworker1 (172.31.86.69): icmp_seq=7 ttl=64 time=0.751 ms
64 bytes from dockerworker1 (172.31.86.69): icmp_seq=8 ttl=64 time=0.663 ms
^C
--- dockerworker1 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7136ms
rtt min/avg/max/mdev = 0.637/1.222/5.005/1.430 ms
root@ip-172-31-17-73:~#
```

    - From Docker Worker Node instance:

```
root@ip-172-31-86-69:~# ping dockermanager
PING dockermanager (172.31.17.73) 56(84) bytes of data.
64 bytes from dockermanager (172.31.17.73): icmp_seq=1 ttl=64 time=0.669 ms
64 bytes from dockermanager (172.31.17.73): icmp_seq=2 ttl=64 time=0.693 ms
64 bytes from dockermanager (172.31.17.73): icmp_seq=3 ttl=64 time=0.693 ms
64 bytes from dockermanager (172.31.17.73): icmp_seq=4 ttl=64 time=0.713 ms
64 bytes from dockermanager (172.31.17.73): icmp_seq=5 ttl=64 time=0.697 ms
^C
--- dockermanager ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4100ms
rtt min/avg/max/mdev = 0.669/0.693/0.713/0.014 ms
root@ip-172-31-86-69:~#
```

- Initialize the Docker swarm mode by running the following docker command on the **dockermanager** node

docker swarm init --advertise-addr<manager node IP address>

docker swarm init --advertise-addr172.31.17.73

```
root@ip-172-31-17-73:~# docker swarm init --advertise-addr 172.31.17.73
Swarm initialized: current node (ba8j0ti2lols6f8pbxfyqy5lc) is now a manager.

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

    docker swarm join --token SWMTKN-1-2o9yesj2p0jk65wory232wthdrec38yeg1r037ryoxe6duuy4n-ant4lo3e6xkdociyk9ut5ky4j 172.31.17.73:2377

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

root@ip-172-31-17-73:~#
```

- Once the swarm cluster is initialized, allow the ports mentioned below in security groups

- While initializing the Docker swarm cluster, you will get docker swarm join command which can be executed on node manager to add node to swarm

```
root@ip-172-31-86-69:~# docker swarm join --token SWMTKN-1-2o9yesj2p0jk65wory232wthdrec38yeg1r037ryoxe6duuy4n-ant4lo3e6xkdociyk9ut5ky4j 172.31.17.73:2377
This node joined a swarm as a worker.
root@ip-172-31-86-69:~#
```

cluster

- Run the command below to see the node status

docker node ls

```
root@ip-172-31-17-73:~# docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
ba8j0ti2lols6f8pbxfyqy5lc *   ip-172-31-17-73     Ready               Active              Leader              18.09.7
sk9bkki9xk5gr39jj5j7kdstz     ip-172-31-86-69     Ready               Active                                  18.09.7
root@ip-172-31-17-73:~#
```

Deploying a custom Docker image to a Docker swarm cluster

- Create service in Docker swarm cluster

```
root@ip-172-31-17-73:~# docker service create --name webapp --publish 8080:8080 --replicas 2 docker.io/jocatalin/kubernetes-bootcamp:v1
oh0u8450pritrl1hwicjqbqjy
overall progress: 2 out of 2 tasks
1/2: running   [==================================================>]
2/2: running   [==================================================>]
verify: Service converged
root@ip-172-31-17-73:~# docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE                                  PORTS
oh0u8450prit        webapp              replicated          2/2                 jocatalin/kubernetes-bootcamp:v1       *:8080->8080/tcp
root@ip-172-31-17-73:~#
```

- You can now validate if Docker containers got deployed on both nodes or not using the command below

docker service pswebapp

```
root@ip-172-31-17-73:~# docker service ps webapp
ID                  NAME                IMAGE                              NODE                DESIRED STATE
S
kxlfdaa25vol        webapp.1            jocatalin/kubernetes-bootcamp:v1   ip-172-31-17-73     Running
wouv28ypnnje        webapp.2            jocatalin/kubernetes-bootcamp:v1   ip-172-31-86-69     Running
root@ip-172-31-17-73:~#
```

Please Note: We can validate the application using the **curl** command to see if the application is up and running.

```
root@ip-172-31-17-73:~# curl localhost:8080
Hello Kubernetes bootcamp! | Running on: dda6e7f30789 | v=1
root@ip-172-31-17-73:~#
```