Multi-Node Docker Swarm (Overlay Networking)

Submitted by:

1. Dileep Kumar (ERP: 18255)
2. Muhammad Arsalan Mubeen (ERP: 23394)

Multi-Node Docker Swarm (Overlay Networking)

Contents:

Architecture

We have implemented two cases in multi-node docker swarm each with different architecture.

Architecture 1:

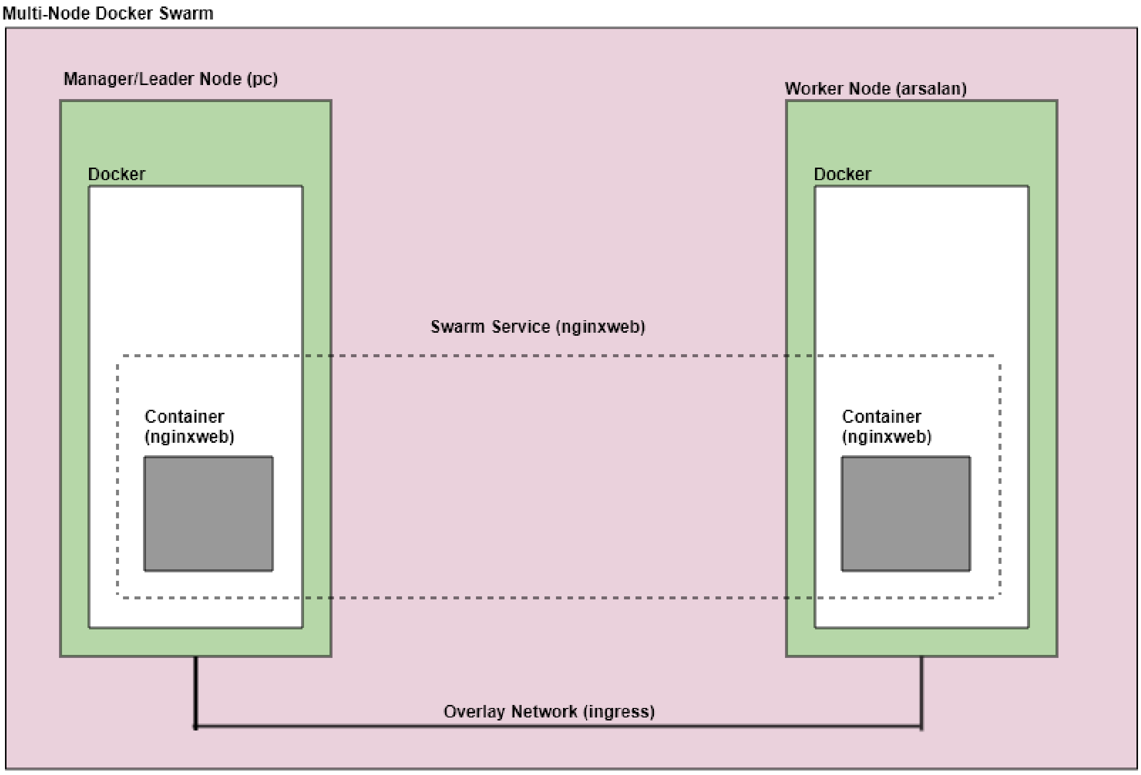
In the first case, we have created multi-node docker swarm consisting of two nodes (Manager/Leader and Worker). Then we created a docker service named **nginxweb** from **nginx** image with 2/2 replicas, each replica on the separate node.

This architecture is implemented in following steps, further recorded in this documents:

Step 1: Creating and Verifying Multi-Node Docker Swarm

Step 2: Overlay networking in Multi-Node Docker Swarm

Step 3: Managing services in Multi-Node Docker Swarm

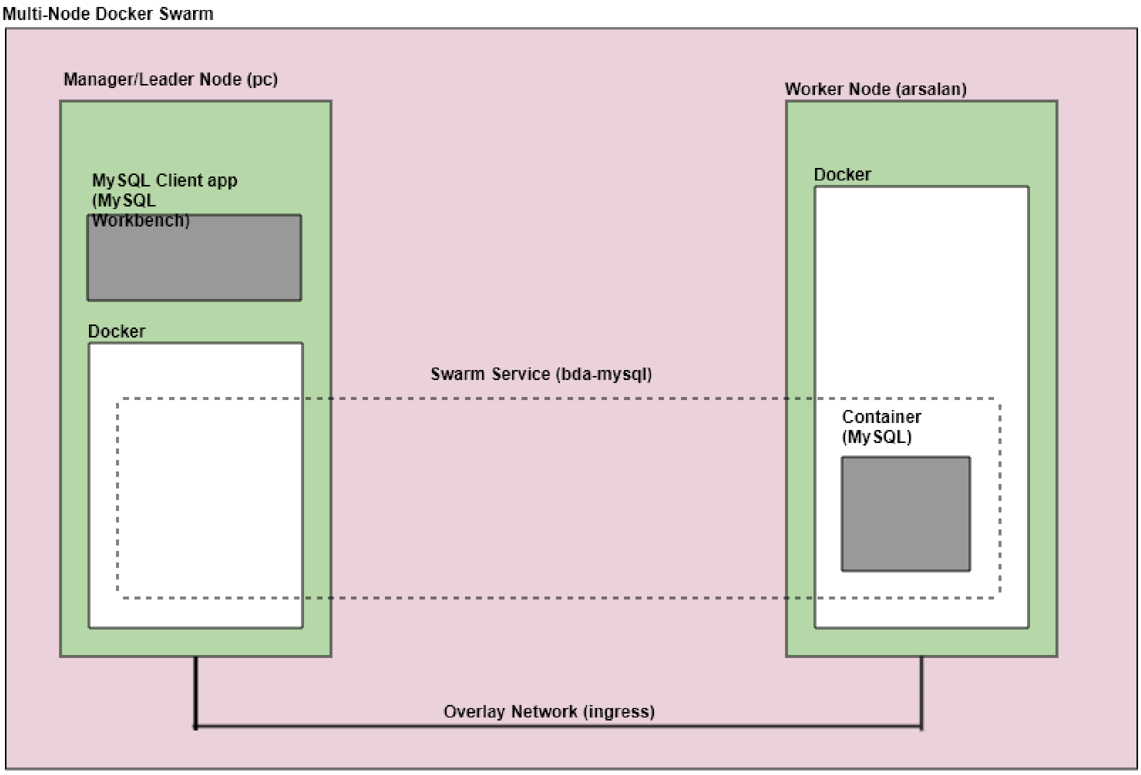


Architecture 2:

In the second case, we have created multi-node docker swarm consisting of two nodes (Manager/Leader and Worker). Then we created a docker service named **bda-mysql** from **mysql** image with 1/1 replica on Worker node. MySQL client app (**MySQL Workbench**) is installed on Manager Node. This database client app is used to query and upload dataset (**bda-dataset.csv**) to mysql container running on Worker Node.

This architecture is implemented in following step recorded in this documents:

Step 4: Dataset querying on Multi Node Docker Swarm

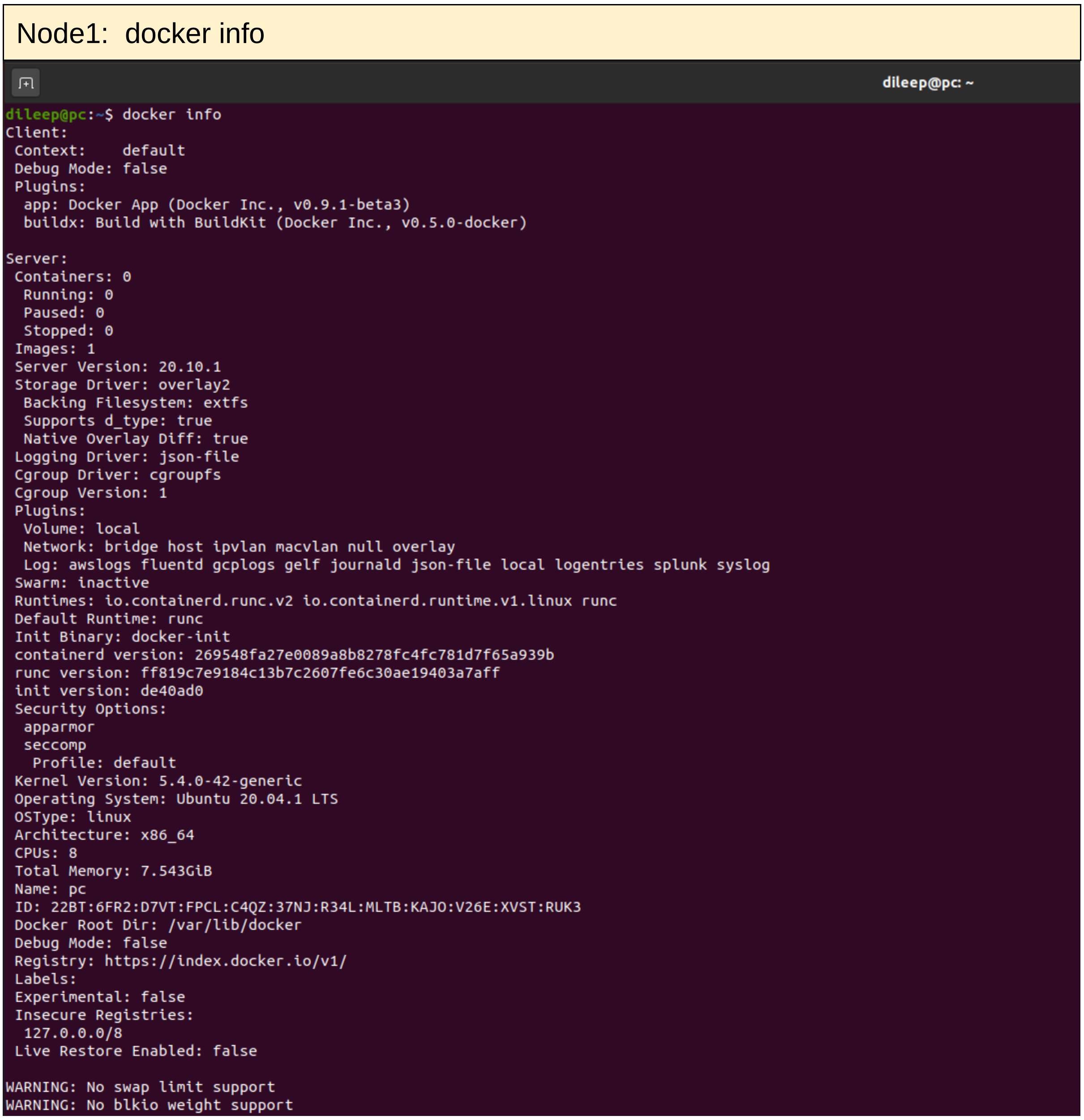


Step 1: Creating and Verifying Multi-Node Docker Swarm

In this section we have created 2 node docker swarm (1 Manager, and 1 Worker).

1. Verify initially swarm feature is inactive and experimental feature is false by default on Node1.

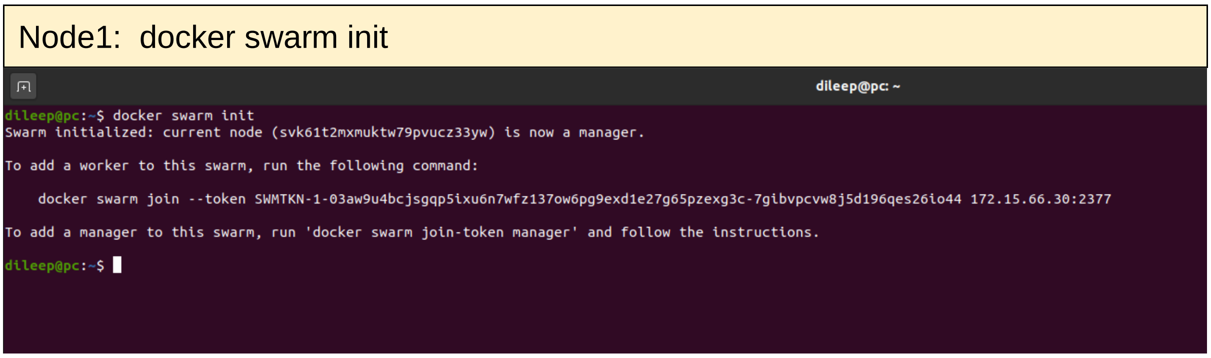
Node1: docker info



```
dileep@pc:~$ docker info
Client:
 Context:    default
 Debug Mode: false
 Plugins:
  app: Docker App (Docker Inc., v0.9.1-beta3)
  buildx: Build with BuildKit (Docker Inc., v0.5.0-docker)

Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 1
 Server Version: 20.10.1
 Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Native Overlay Diff: true
 Logging Driver: json-file
 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: inactive
 Runtimes: io.containerd.runc.v2 io.containerd.runtime.v1.linux runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: 269548fa27e0089a8b8278fc4fc781d7f65a939b
 runc version: ff819c7e9184c13b7c2607fe6c30ae19403a7aff
 init version: de40ad0
 Security Options:
  apparmor
  seccomp
   Profile: default
 Kernel Version: 5.4.0-42-generic
 Operating System: Ubuntu 20.04.1 LTS
 OSType: linux
 Architecture: x86_64
 CPUs: 8
 Total Memory: 7.543GiB
 Name: pc
 ID: 22BT:6FR2:D7VT:FPCL:C4QZ:37NJ:R34L:MLTB:KAJO:V26E:XVST:RUK3
 Docker Root Dir: /var/lib/docker
 Debug Mode: false
 Registry: https://index.docker.io/v1/
 Labels:
 Experimental: false
 Insecure Registries:
  127.0.0.0/8
 Live Restore Enabled: false

WARNING: No swap limit support
WARNING: No blkio weight support
```

2. Initialize docker swarm mode on Node 1. It will make this node manager (leader) by default.

Node1: docker swarm init



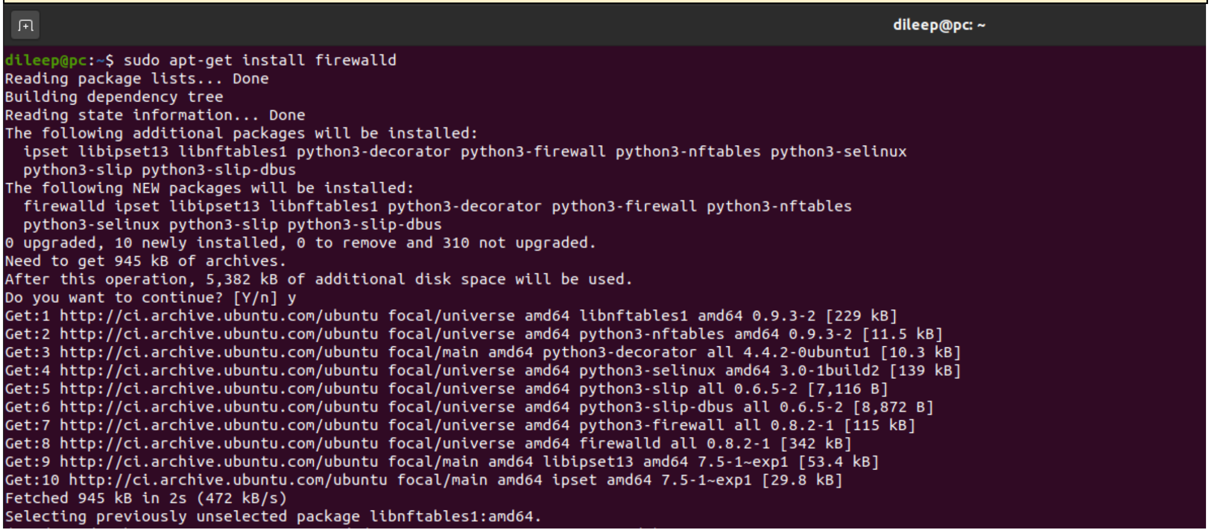
```
dileep@pc:~$ docker swarm init
Swarm initialized: current node (svk61t2mxmuktw79pvucz33yw) is now a manager.

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

    docker swarm join --token SWMTKN-1-03aw9u4bcjsgqp5ixu6n7wfz137ow6pg9exd1e27g65pzexg3c-7gibvpcvw8j5d196qes26io44 172.15.66.30:2377

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

dileep@pc:~$ 
```

3. Verify docker swarm feature is now active on Node 1 by repeating step1.

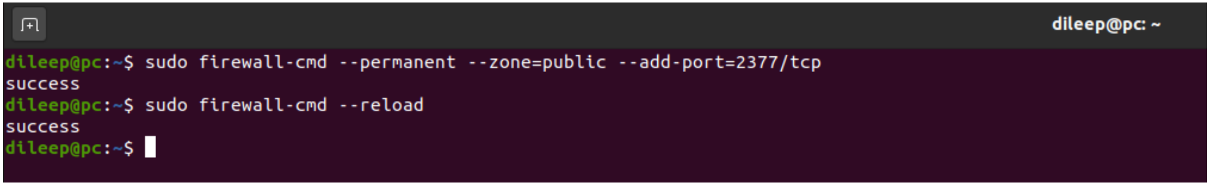
Node1: docker info

4. Allow firewall to open port number on Node 1. To do so:
   a. First install firewall app
   b. Allow firewall to permit the port number
   c. Reload the firewall to take effect

Node1: sudo apt-get install firewalld
Node1: sudo firewall-cmd –permanent –zone=public –add-port=2377/tcp
Node1: sudo firewall-cmd –reload



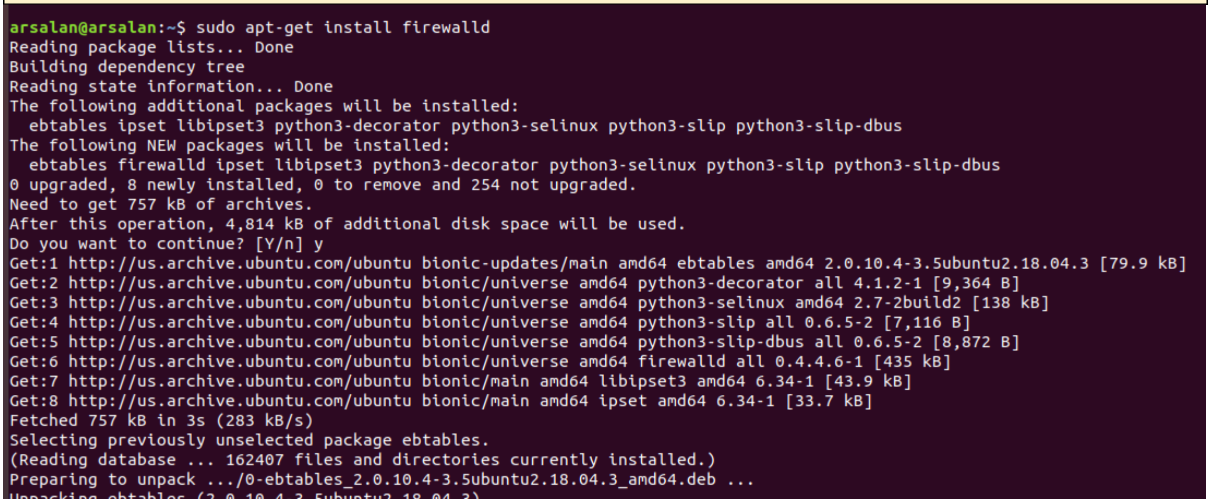
```
dileep@pc:~$ sudo apt-get install firewalld
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  ipset libipset13 libnftables1 python3-decorator python3-firewall python3-nftables python3-selinux
  python3-slip python3-slip-dbus
The following NEW packages will be installed:
  firewalld ipset libipset13 libnftables1 python3-decorator python3-firewall python3-nftables
  python3-selinux python3-slip python3-slip-dbus
0 upgraded, 10 newly installed, 0 to remove and 310 not upgraded.
Need to get 945 kB of archives.
After this operation, 5,382 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ci.archive.ubuntu.com/ubuntu focal/universe amd64 libnftables1 amd64 0.9.3-2 [229 kB]
Get:2 http://ci.archive.ubuntu.com/ubuntu focal/universe amd64 python3-nftables amd64 0.9.3-2 [11.5 kB]
Get:3 http://ci.archive.ubuntu.com/ubuntu focal/main amd64 python3-decorator all 4.4.2-0ubuntu1 [10.3 kB]
Get:4 http://ci.archive.ubuntu.com/ubuntu focal/universe amd64 python3-selinux amd64 3.0-1build2 [139 kB]
Get:5 http://ci.archive.ubuntu.com/ubuntu focal/universe amd64 python3-slip all 0.6.5-2 [7,116 B]
Get:6 http://ci.archive.ubuntu.com/ubuntu focal/universe amd64 python3-slip-dbus all 0.6.5-2 [8,872 B]
Get:7 http://ci.archive.ubuntu.com/ubuntu focal/universe amd64 python3-firewall all 0.8.2-1 [115 kB]
Get:8 http://ci.archive.ubuntu.com/ubuntu focal/universe amd64 firewalld all 0.8.2-1 [342 kB]
Get:9 http://ci.archive.ubuntu.com/ubuntu focal/main amd64 libipset13 amd64 7.5-1~exp1 [53.4 kB]
Get:10 http://ci.archive.ubuntu.com/ubuntu focal/main amd64 ipset amd64 7.5-1~exp1 [29.8 kB]
Fetched 945 kB in 2s (472 kB/s)
Selecting previously unselected package libnftables1:amd64.
```



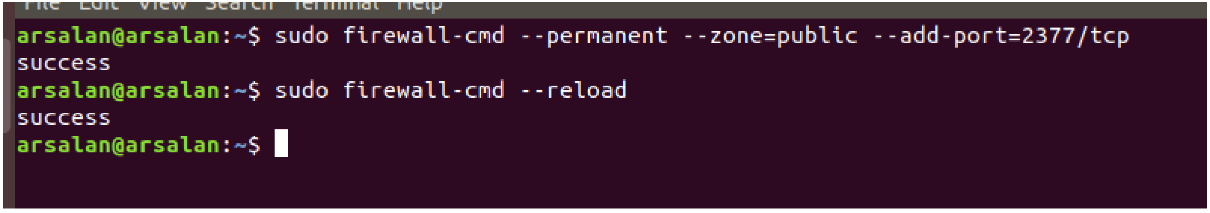
```
dileep@pc:~$ sudo firewall-cmd --permanent --zone=public --add-port=2377/tcp
success
dileep@pc:~$ sudo firewall-cmd --reload
success
dileep@pc:~$ 
```

5. Allow firewall to open port number on Node 2. To do so, repeat step 4 on node 2.

Node2: sudo apt-get install firewalld
Node2: sudo firewall-cmd –permanent –zone=public –add-port=2377/tcp



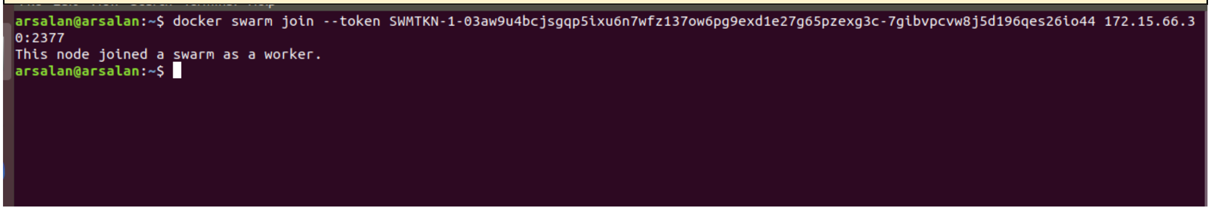
```
arsalan@arsalan:~$ sudo apt-get install firewalld
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  ebtables ipset libipset3 python3-decorator python3-selinux python3-slip python3-slip-dbus
The following NEW packages will be installed:
  ebtables firewalld ipset libipset3 python3-decorator python3-selinux python3-slip python3-slip-dbus
0 upgraded, 8 newly installed, 0 to remove and 254 not upgraded.
Need to get 757 kB of archives.
After this operation, 4,814 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 ebtables amd64 2.0.10.4-3.5ubuntu2.18.04.3 [79.9 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 python3-decorator all 4.1.2-1 [9,364 B]
Get:3 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 python3-selinux amd64 2.7-2build2 [138 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 python3-slip all 0.6.5-2 [7,116 B]
Get:5 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 python3-slip-dbus all 0.6.5-2 [8,872 B]
Get:6 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 firewalld all 0.4.4.6-1 [435 kB]
Get:7 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 libipset3 amd64 6.34-1 [43.9 kB]
Get:8 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 ipset amd64 6.34-1 [33.7 kB]
Fetched 757 kB in 3s (283 kB/s)
Selecting previously unselected package ebtables.
(Reading database ... 162407 files and directories currently installed.)
Preparing to unpack .../0-ebtables_2.0.10.4-3.5ubuntu2.18.04.3_amd64.deb ...
```



```
arsalan@arsalan:~$ sudo firewall-cmd --permanent --zone=public --add-port=2377/tcp
success
arsalan@arsalan:~$ sudo firewall-cmd --reload
success
arsalan@arsalan:~$
```

6. Add Node2 to swarm as worker, to do so, first get swarm-token-id from node 1 (manager node) using commands **docker swarm join-token worker.**

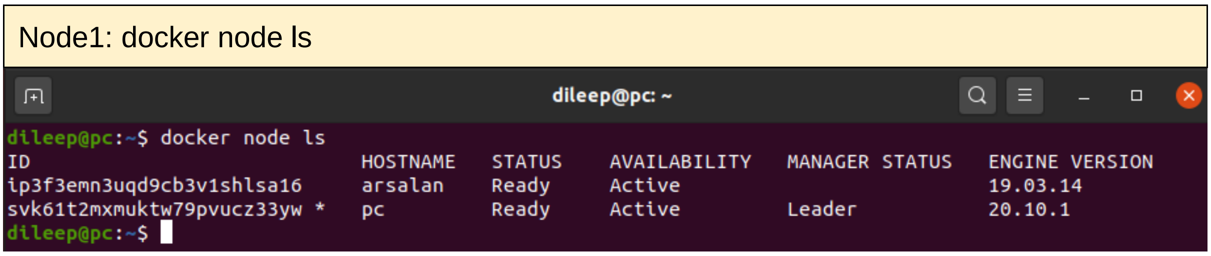
Node2: docker swarm join –token <swarm-token-id> ip:port



```
arsalan@arsalan:~$ docker swarm join --token SWMTKN-1-03aw9u4bcjsgqp5ixu6n7wfz137ow6pg9exd1e27g65pzexg3c-7gibvpcvw8j5d196qes26io44 172.15.66.3
0:2377
This node joined a swarm as a worker.
arsalan@arsalan:~$
```

7. Verify Node1 (as leader) and Node2 (as worker) has joined the swarm.

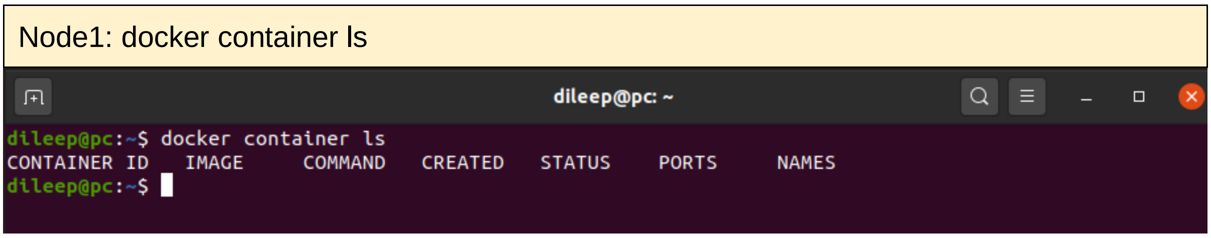
Node1: docker node ls



```
dileep@pc:~$ docker node ls
ID                            HOSTNAME   STATUS   AVAILABILITY   MANAGER STATUS   ENGINE VERSION
ip3f3emn3uqd9cb3v1shlsa16     arsalan    Ready    Active                          19.03.14
svk61t2mxmuktw79pvucz33yw *   pc         Ready    Active         Leader           20.10.1
dileep@pc:~$
```

8. Verify there is no container running on host initially.

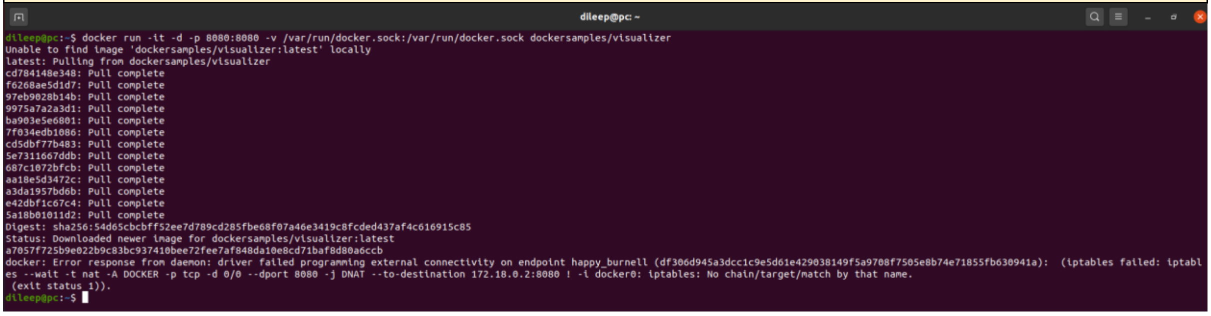
Node1: docker container ls



```
dileep@pc:~$ docker container ls
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
dileep@pc:~$ 
```

9. Running **docker-swarm-visualizer** container on the node 1. This container will let us visualize all manager and worker node in the swarm, and all service containers running on the swarm nodes.

Node1: docker run -it -d -p 8080:8080 -v /var/run/docker.sock:/var/run/docker.sock dockersamples/visualizer



```
dileep@pc:~$ docker run -it -d -p 8080:8080 -v /var/run/docker.sock:/var/run/docker.sock dockersamples/visualizer
Unable to find image 'dockersamples/visualizer:latest' locally
latest: Pulling from dockersamples/visualizer
cd784148e348: Pull complete
f6268ae5d1d7: Pull complete
97eb9028b14b: Pull complete
9975a7a2a3d1: Pull complete
ba903e5e6801: Pull complete
7f034edb1086: Pull complete
cd5dbf77b483: Pull complete
5e7311667ddb: Pull complete
687c1072bfcb: Pull complete
aa18e5d3472c: Pull complete
a3da1957bd6b: Pull complete
e42dbf1c67c4: Pull complete
5a18b01011d2: Pull complete
Digest: sha256:54d65cbcbff52ee7d789cd285fbe68f07a46e3419c8fcded437af4c616915c85
Status: Downloaded newer image for dockersamples/visualizer:latest
a7057f725b9e022b9c83bc937410bee72fee7af848da10e8cd71baf8d80a6ccb
docker: Error response from daemon: driver failed programming external connectivity on endpoint happy_burnell (df306d945a3dcc1c9e5d61e429038149f5a9708f7505e8b74e71855fb630941a):  (iptables failed: iptabl
es --wait -t nat -A DOCKER -p tcp -d 0/0 --dport 8080 -j DNAT --to-destination 172.18.0.2:8080 ! -i docker0: iptables: No chain/target/match by that name.
 (exit status 1)).
dileep@pc:~$ 
```

10. Only if there is iptables related error occurs in step 9, use following commands to fix it.

1. Clear all chains:

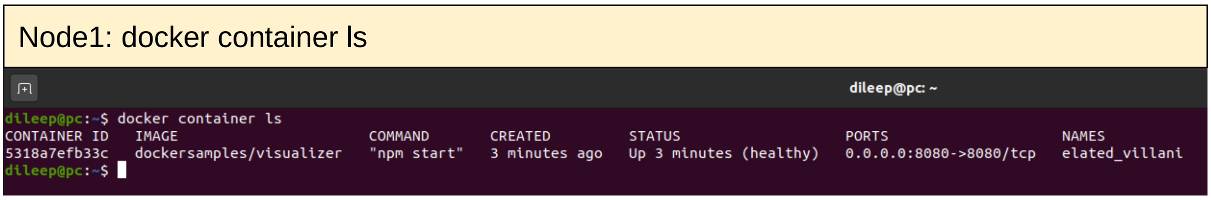
```
sudo iptables -t filter -F
sudo iptables -t filter -X
```

2. Then restart Docker Service:

```
systemctl restart docker
```

11. Verify the visualizer container is created

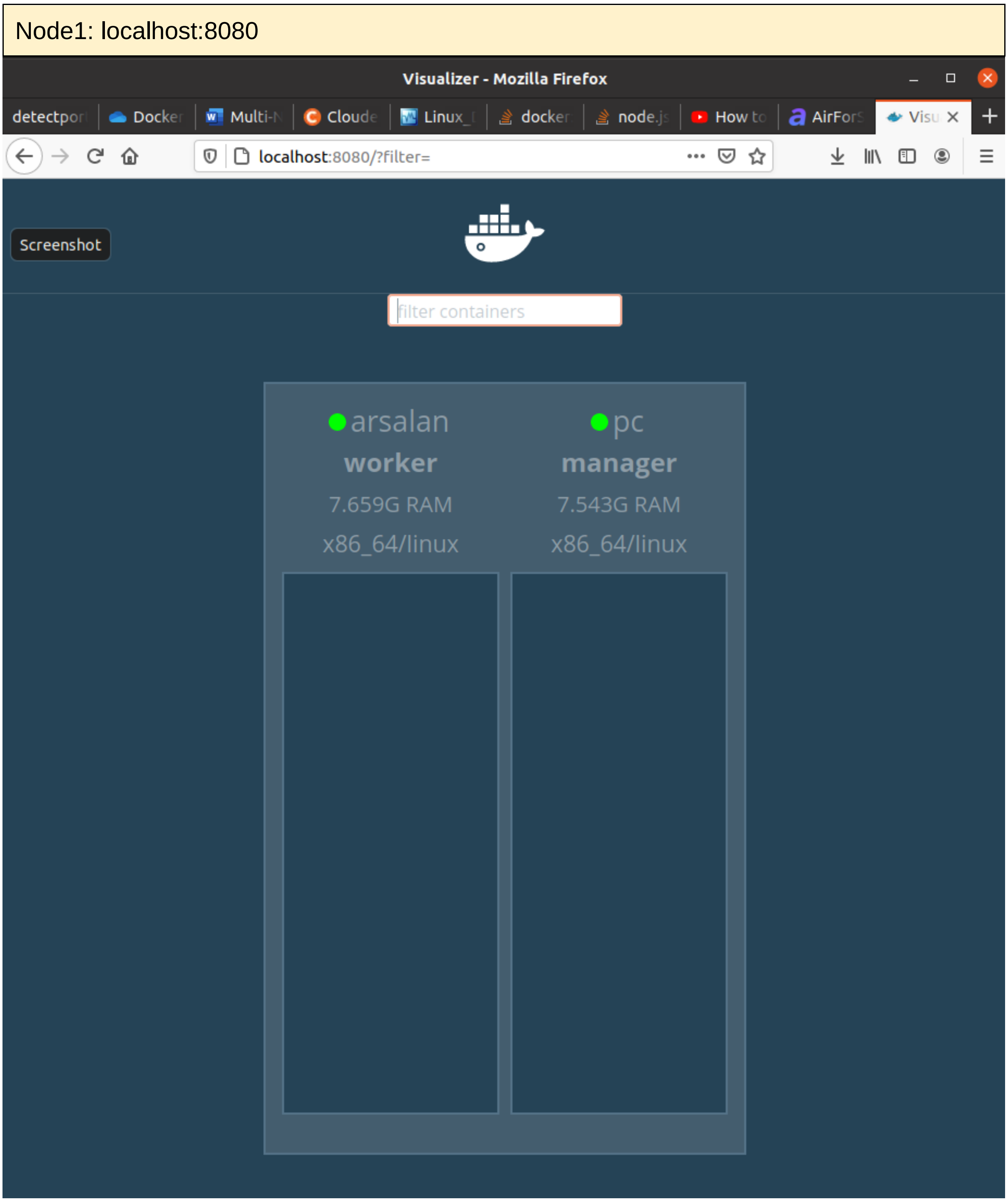
Node1: docker container ls



```
dileep@pc:~$ docker container ls
CONTAINER ID   IMAGE                      COMMAND       CREATED         STATUS                   PORTS                    NAMES
5318a7efb33c   dockersamples/visualizer   "npm start"   3 minutes ago   Up 3 minutes (healthy)   0.0.0.0:8080->8080/tcp   elated_villani
dileep@pc:~$ 
```

12. To view swarm nodes and services on the web browser use visualizer container.

Node1: localhost:8080



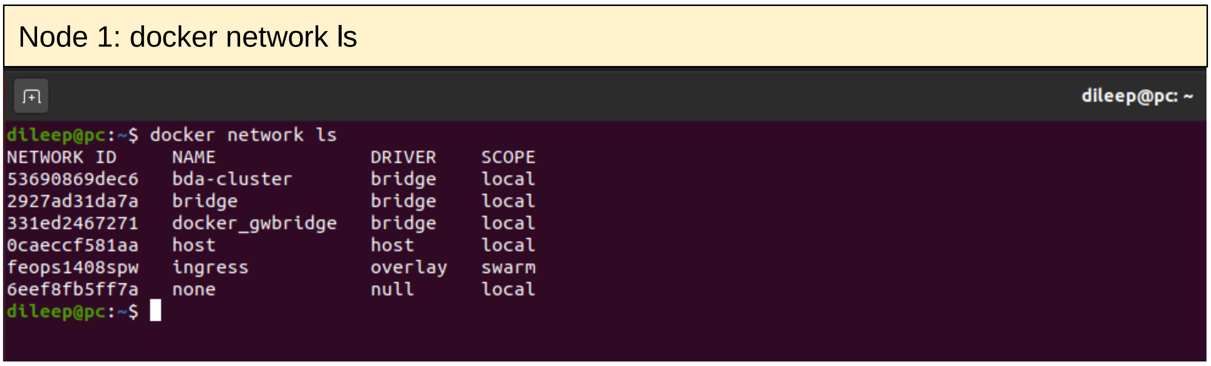
●arsalan
worker
7.659G RAM
x86_64/linux

●pc
manager
7.543G RAM
x86_64/linux

Step 2: Overlay networking in Multi-Node Docker Swarm

13. List available networks on manager node. We can see ingress named network is created by default using overlay driver by swarm. Because we haven’t yet created any such overlay network.

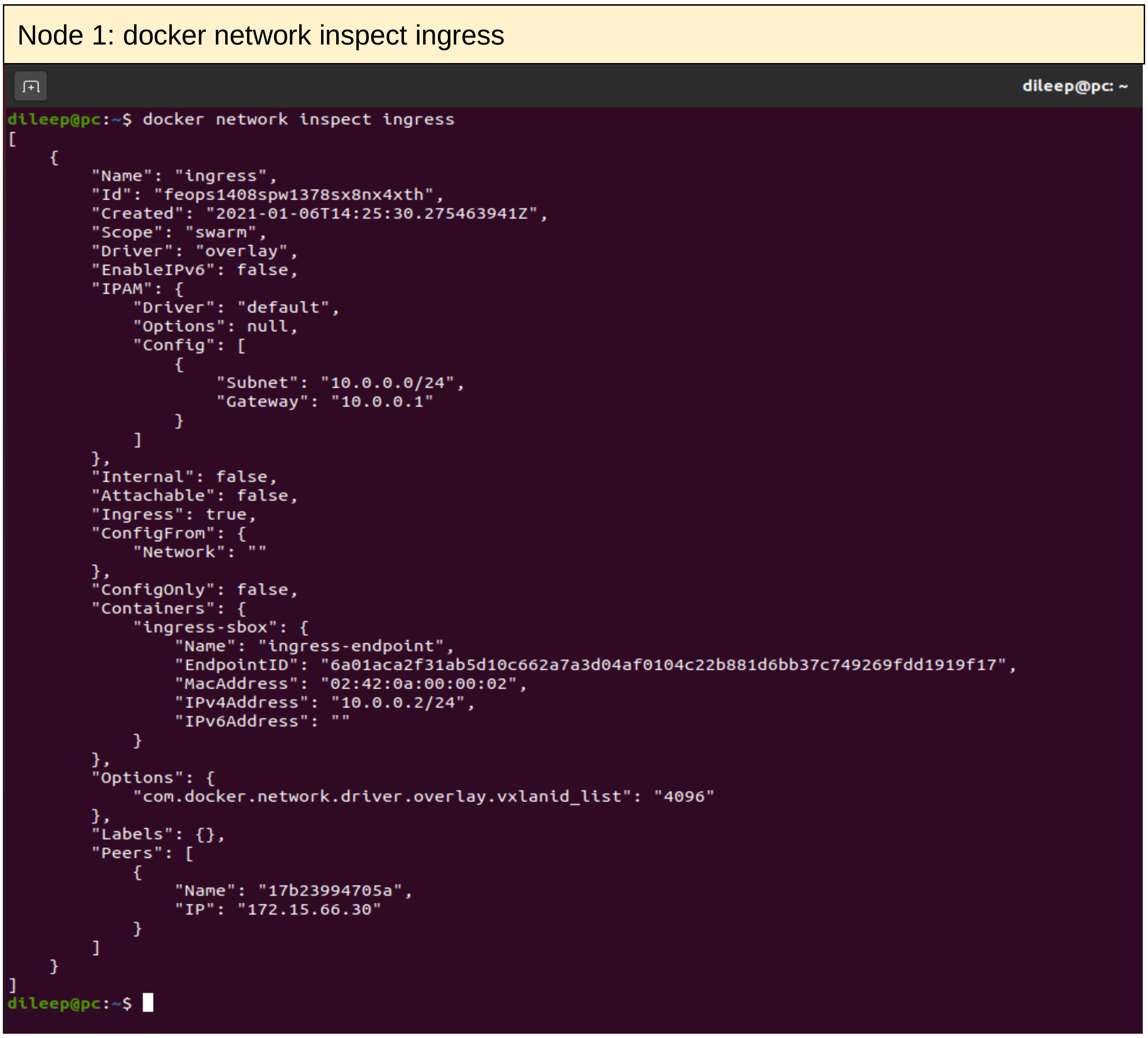
Node 1: docker network ls



```
dileep@pc:~$ docker network ls
NETWORK ID     NAME             DRIVER    SCOPE
53690869dec6   bda-cluster      bridge    local
2927ad31da7a   bridge           bridge    local
331ed2467271   docker_gwbridge  bridge    local
0caeccf581aa   host             host      local
feops1408spw   ingress          overlay   swarm
6eef8fb5ff7a   none             null      local
dileep@pc:~$ 
```

14. View detailed information of overlay network (named ingress)

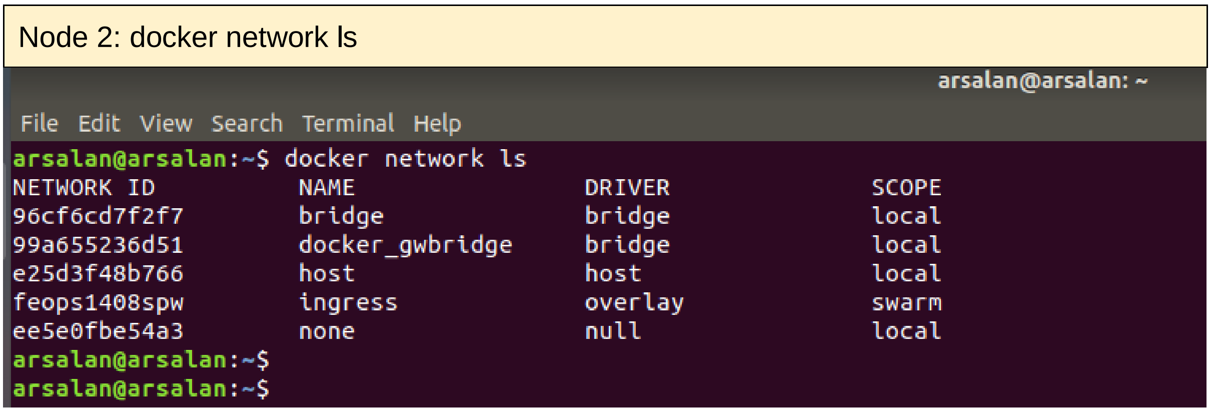
Node 1: docker network inspect ingress



```
dileep@pc:~$ docker network inspect ingress
[
    {
        "Name": "ingress",
        "Id": "feops1408spw1378sx8nx4xth",
        "Created": "2021-01-06T14:25:30.275463941Z",
        "Scope": "swarm",
        "Driver": "overlay",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "10.0.0.0/24",
                    "Gateway": "10.0.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": true,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "ingress-sbox": {
                "Name": "ingress-endpoint",
                "EndpointID": "6a01aca2f31ab5d10c662a7a3d04af0104c22b881d6bb37c749269fdd1919f17",
                "MacAddress": "02:42:0a:00:00:02",
                "IPv4Address": "10.0.0.2/24",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.driver.overlay.vxlanid_list": "4096"
        },
        "Labels": {},
        "Peers": [
            {
                "Name": "17b23994705a",
                "IP": "172.15.66.30"
            }
        ]
    }
]
dileep@pc:~$ 
```

15. List available networks on worker node
You can see ingress named network is created by default using overlay driver when Node 2 has joined the swarm.

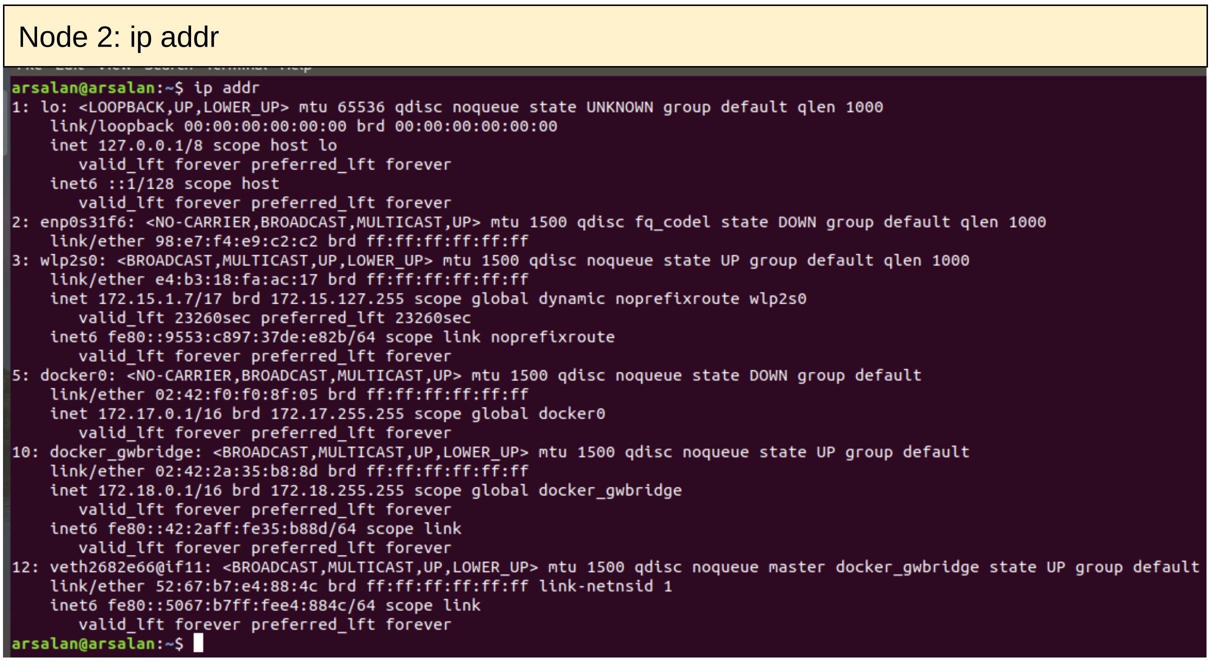
Node 2: docker network ls



```
arsalan@arsalan:~$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
96cf6cd7f2f7        bridge              bridge              local
99a655236d51        docker_gwbridge     bridge              local
e25d3f48b766        host                host                local
feops1408spw        ingress             overlay             swarm
ee5e0fbe54a3        none                null                local
arsalan@arsalan:~$
arsalan@arsalan:~$
```

16. View IP address of Node 2 to ping it from Node 1.

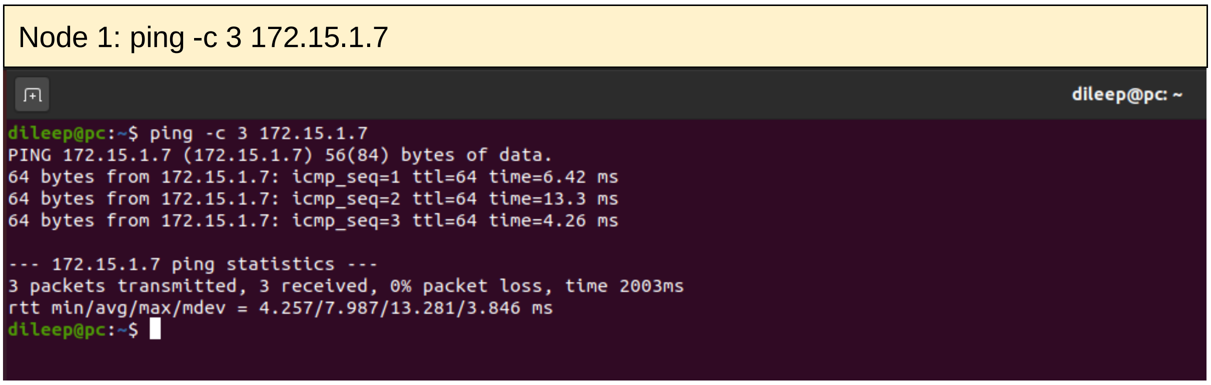
Node 2: ip addr



```
arsalan@arsalan:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s31f6: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether 98:e7:f4:e9:c2:c2 brd ff:ff:ff:ff:ff:ff
3: wlp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether e4:b3:18:fa:ac:17 brd ff:ff:ff:ff:ff:ff
    inet 172.15.1.7/17 brd 172.15.127.255 scope global dynamic noprefixroute wlp2s0
       valid_lft 23260sec preferred_lft 23260sec
    inet6 fe80::9553:c897:37de:e82b/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
5: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:f0:f0:8f:05 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
10: docker_gwbridge: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:2a:35:b8:8d brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.1/16 brd 172.18.255.255 scope global docker_gwbridge
       valid_lft forever preferred_lft forever
    inet6 fe80::42:2aff:fe35:b88d/64 scope link
       valid_lft forever preferred_lft forever
12: veth2682e66@if11: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge state UP group default
    link/ether 52:67:b7:e4:88:4c brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet6 fe80::5067:b7ff:fee4:884c/64 scope link
       valid_lft forever preferred_lft forever
arsalan@arsalan:~$
```

17. Verify connectivity from Node 1 to Node 2 using ping utility.

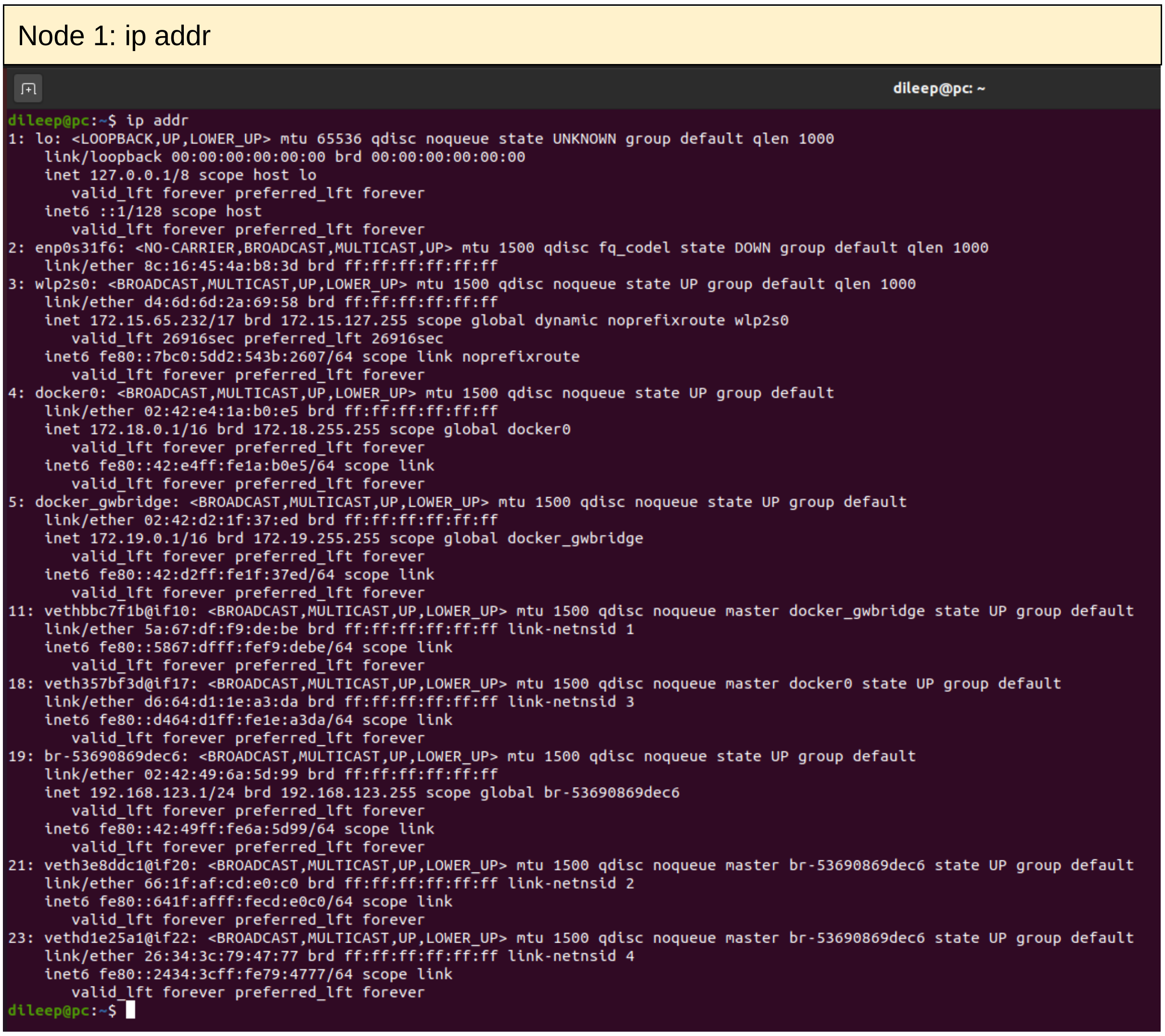
Node 1: ping -c 3 172.15.1.7



```
dileep@pc:~$ ping -c 3 172.15.1.7
PING 172.15.1.7 (172.15.1.7) 56(84) bytes of data.
64 bytes from 172.15.1.7: icmp_seq=1 ttl=64 time=6.42 ms
64 bytes from 172.15.1.7: icmp_seq=2 ttl=64 time=13.3 ms
64 bytes from 172.15.1.7: icmp_seq=3 ttl=64 time=4.26 ms

--- 172.15.1.7 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 4.257/7.987/13.281/3.846 ms
dileep@pc:~$
```

18. View IP address of Node 1 to ping it from Node 2.

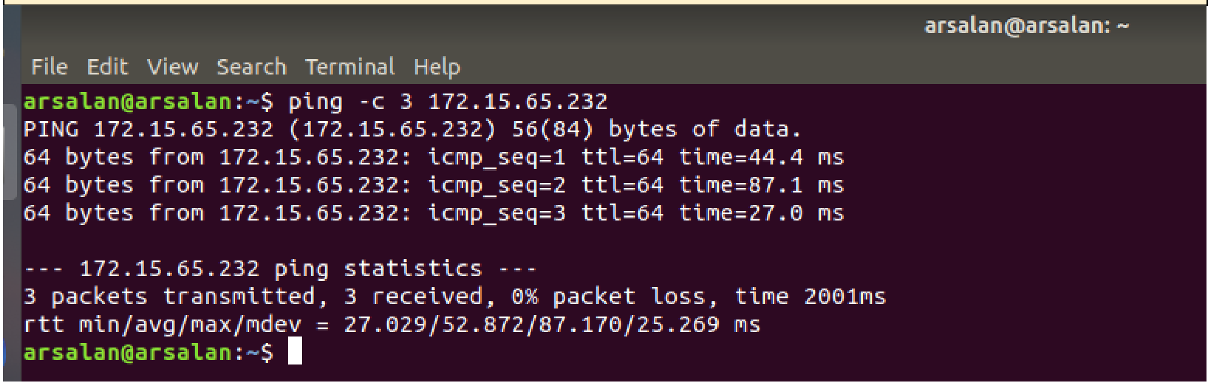
Node 1: ip addr



```
dileep@pc:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s31f6: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether 8c:16:45:4a:b8:3d brd ff:ff:ff:ff:ff:ff
3: wlp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether d4:6d:6d:2a:69:58 brd ff:ff:ff:ff:ff:ff
    inet 172.15.65.232/17 brd 172.15.127.255 scope global dynamic noprefixroute wlp2s0
       valid_lft 26916sec preferred_lft 26916sec
    inet6 fe80::7bc0:5dd2:543b:2607/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
4: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:e4:1a:b0:e5 brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.1/16 brd 172.18.255.255 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:e4ff:fe1a:b0e5/64 scope link
       valid_lft forever preferred_lft forever
5: docker_gwbridge: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:d2:1f:37:ed brd ff:ff:ff:ff:ff:ff
    inet 172.19.0.1/16 brd 172.19.255.255 scope global docker_gwbridge
       valid_lft forever preferred_lft forever
    inet6 fe80::42:d2ff:fe1f:37ed/64 scope link
       valid_lft forever preferred_lft forever
11: vethbbc7f1b@if10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge state UP group default
    link/ether 5a:67:df:f9:de:be brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet6 fe80::5867:dfff:fef9:debe/64 scope link
       valid_lft forever preferred_lft forever
18: veth357bf3d@if17: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP group default
    link/ether d6:64:d1:1e:a3:da brd ff:ff:ff:ff:ff:ff link-netnsid 3
    inet6 fe80::d464:d1ff:fe1e:a3da/64 scope link
       valid_lft forever preferred_lft forever
19: br-53690869dec6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:49:6a:5d:99 brd ff:ff:ff:ff:ff:ff
    inet 192.168.123.1/24 brd 192.168.123.255 scope global br-53690869dec6
       valid_lft forever preferred_lft forever
    inet6 fe80::42:49ff:fe6a:5d99/64 scope link
       valid_lft forever preferred_lft forever
21: veth3e8ddc1@if20: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-53690869dec6 state UP group default
    link/ether 66:1f:af:cd:e0:c0 brd ff:ff:ff:ff:ff:ff link-netnsid 2
    inet6 fe80::641f:afff:fecd:e0c0/64 scope link
       valid_lft forever preferred_lft forever
23: vethd1e25a1@if22: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-53690869dec6 state UP group default
    link/ether 26:34:3c:79:47:77 brd ff:ff:ff:ff:ff:ff link-netnsid 4
    inet6 fe80::2434:3cff:fe79:4777/64 scope link
       valid_lft forever preferred_lft forever
dileep@pc:~$
```

19. Verify connectivity from Node 2 to Node 1 using ping utility.

Node 2: ping -c 3 172.15.65.232



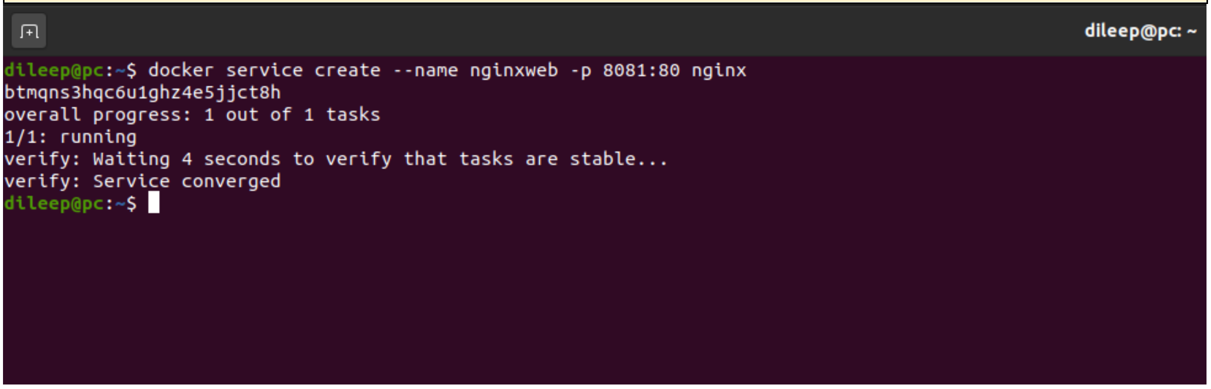
```
arsalan@arsalan:~$ ping -c 3 172.15.65.232
PING 172.15.65.232 (172.15.65.232) 56(84) bytes of data.
64 bytes from 172.15.65.232: icmp_seq=1 ttl=64 time=44.4 ms
64 bytes from 172.15.65.232: icmp_seq=2 ttl=64 time=87.1 ms
64 bytes from 172.15.65.232: icmp_seq=3 ttl=64 time=27.0 ms

--- 172.15.65.232 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 27.029/52.872/87.170/25.269 ms
arsalan@arsalan:~$
```

Step 3: Managing services in Multi-Node Docker Swarm

In this step, to demonstrate how services are managed in a multi-node docker swarm, we will be creating a nginx service.

20. Create a nginx service on the manager node.

Node 1: docker service create --name nginxweb -p 80801:80 nginx



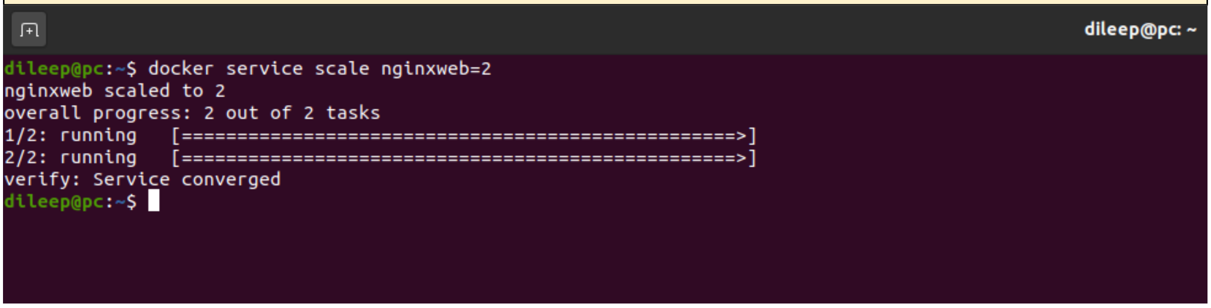
```
dileep@pc:~$ docker service create --name nginxweb -p 8081:80 nginx
btmqns3hqc6u1ghz4e5jjct8h
overall progress: 1 out of 1 tasks
1/1: running
verify: Waiting 4 seconds to verify that tasks are stable...
verify: Service converged
dileep@pc:~$
```

21. Verify that the nginxweb service is created.

Node 1: docker service ls

22. Scale-up the nginxweb service.
    This will create 2 instances (containers) of the nginxweb service.
    Docker swarm load-balancer will distribute by default each instance of the service on different nodes of the swarm.

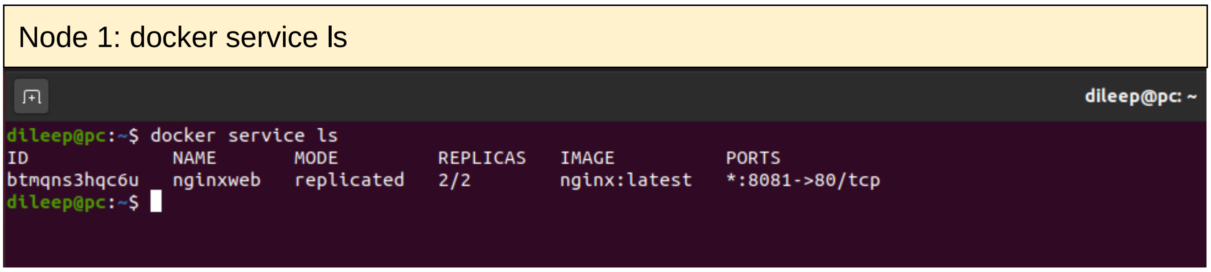
Node 1: docker service scale nginxweb=2



```
dileep@pc:~$ docker service scale nginxweb=2
nginxweb scaled to 2
overall progress: 2 out of 2 tasks
1/2: running   [==================================================>]
2/2: running   [==================================================>]
verify: Service converged
dileep@pc:~$
```

23. Verify the services scaled-up.

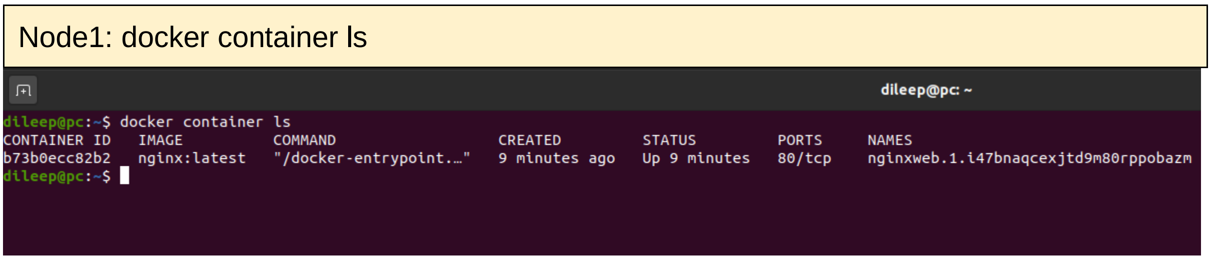
Node 1: docker service ls



```
dileep@pc:~$ docker service ls
ID             NAME       MODE         REPLICAS   IMAGE          PORTS
btmqns3hqc6u   nginxweb   replicated   2/2        nginx:latest   *:8081->80/tcp
dileep@pc:~$
```

24. Verify one container of the nginxweb service is created on Node 1.

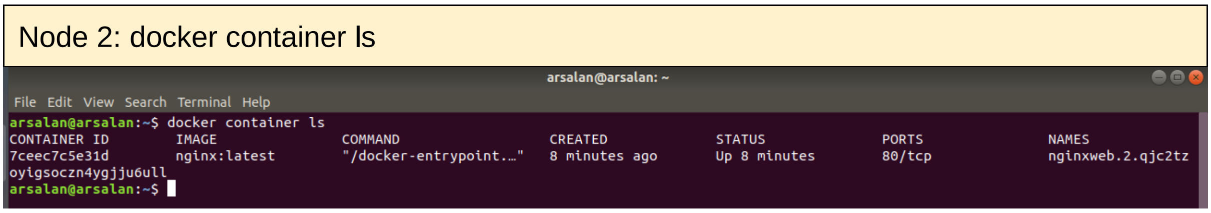
Node1: docker container ls



```
dileep@pc:~$ docker container ls
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS     NAMES
b73b0ecc82b2   nginx:latest   "/docker-entrypoint.…"   9 minutes ago   Up 9 minutes   80/tcp    nginxweb.1.i47bnaqcexjtd9m80rppobazm
dileep@pc:~$
```

25. Verify the second container of the nginxweb service is created on Node 2.

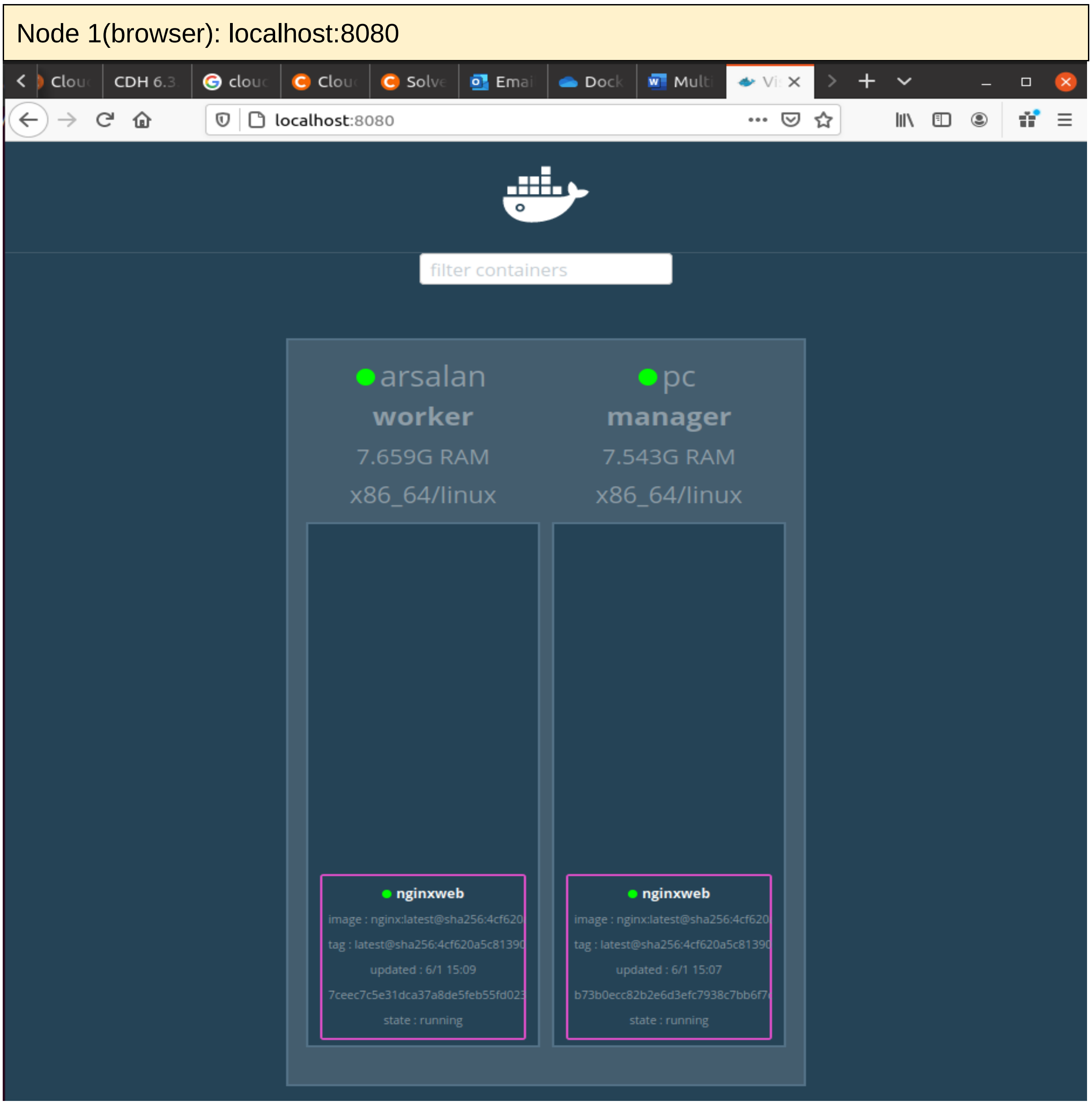
Node 2: docker container ls



```
arsalan@arsalan:~$ docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
7ceec7c5e31d        nginx:latest        "/docker-entrypoint.…"   8 minutes ago       Up 8 minutes        80/tcp              nginxweb.2.qjc2tz
oyigsoczn4ygjju6ull
arsalan@arsalan:~$
```

26. To visualize the swarm nodes and nginxweb service on the web browser use the Visualizer container.

Node 1(browser): localhost:8080



● arsalan
worker
7.659G RAM
x86_64/linux

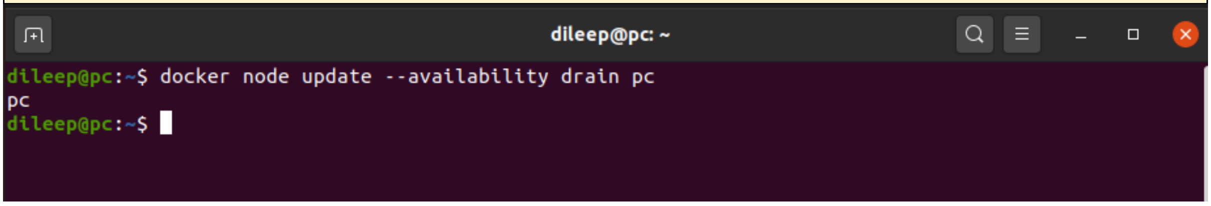
● nginxweb
image : nginx:latest@sha256:4cf620
tag : latest@sha256:4cf620a5c81390
updated : 6/1 15:09
7ceec7c5e31dca37a8de5feb55fd023
state : running

● pc
manager
7.543G RAM
x86_64/linux

● nginxweb
image : nginx:latest@sha256:4cf620
tag : latest@sha256:4cf620a5c81390
updated : 6/1 15:07
b73b0ecc82b2e6d3efc7938c7bb6f7
state : running

27. Drain the Node 1(manager node).
This will migrate all the containers running on Node 1 to Node 2. This will not affect the service absolutely.
We are doing this because we want Node 1 to act as manager only.

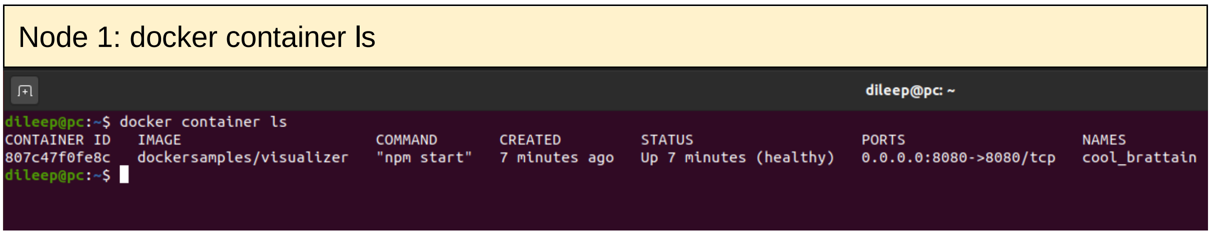
Node 1: docker node update --availability drain pc



```
dileep@pc:~$ docker node update --availability drain pc
pc
dileep@pc:~$ 
```

28. Verify that there is no nginxweb service container running on Node 1 anymore.

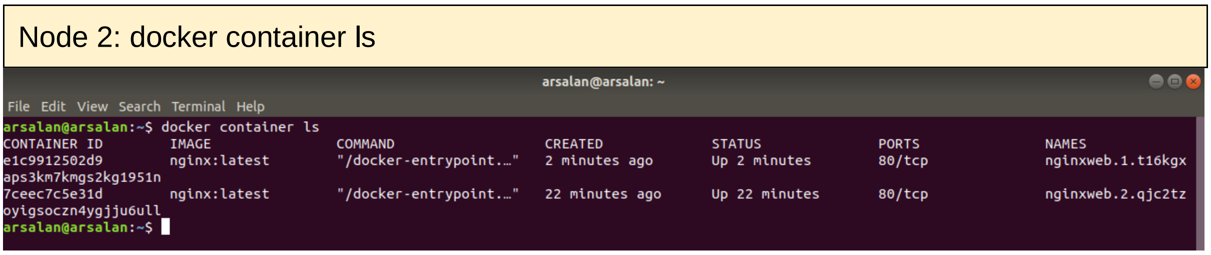
Node 1: docker container ls



```
dileep@pc:~$ docker container ls
CONTAINER ID   IMAGE                      COMMAND       CREATED         STATUS                   PORTS                    NAMES
807c47f0fe8c   dockersamples/visualizer   "npm start"   7 minutes ago   Up 7 minutes (healthy)   0.0.0.0:8080->8080/tcp   cool_brattain
dileep@pc:~$ 
```

29. Verify that both the containers of nginxweb service are running on Node 2.

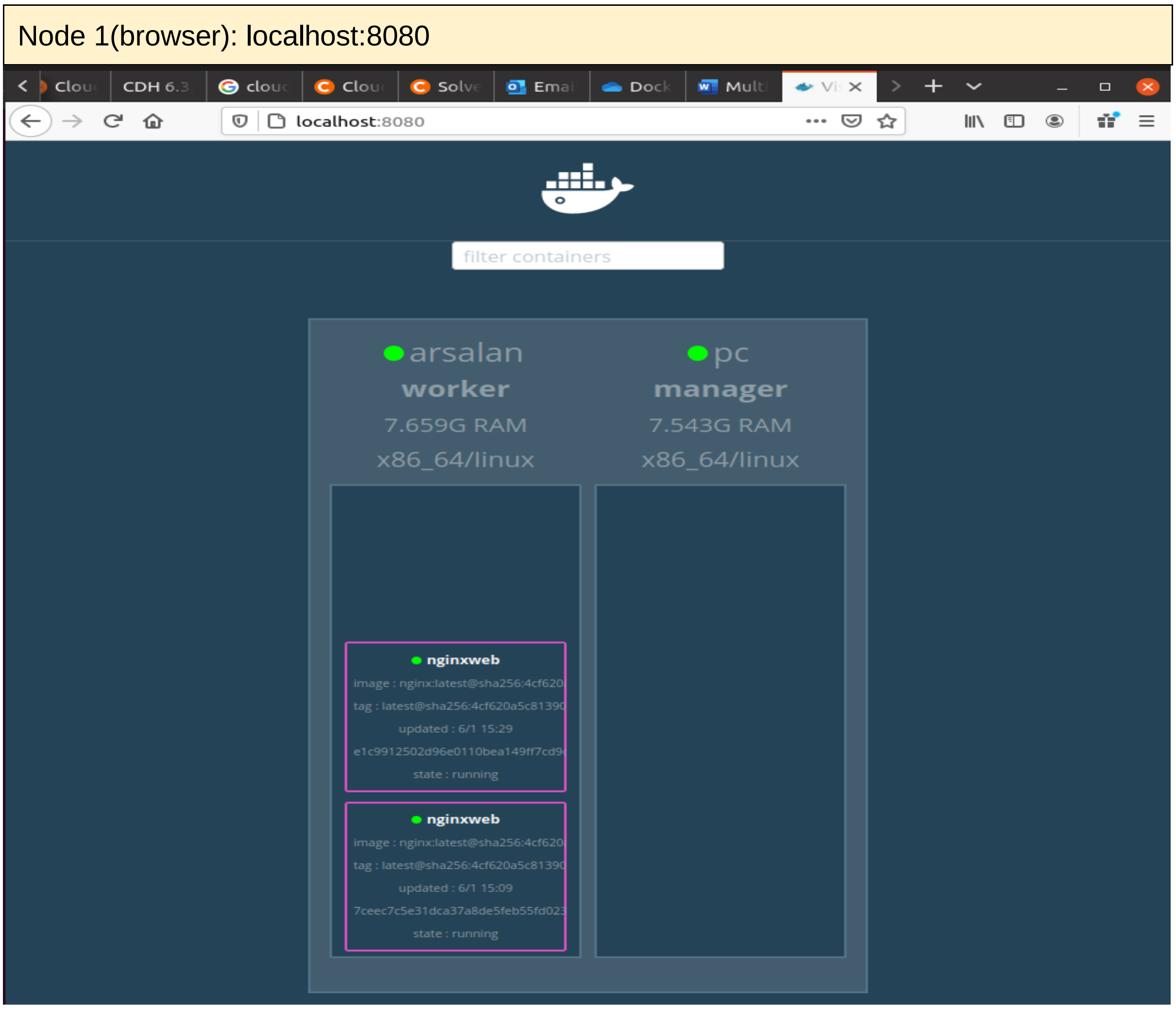
Node 2: docker container ls



```
arsalan@arsalan:~$ docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
e1c9912502d9        nginx:latest        "/docker-entrypoint.…"   2 minutes ago       Up 2 minutes        80/tcp              nginxweb.1.t16kgx
aps3km7kmgs2kg1951n
7ceec7c5e31d        nginx:latest        "/docker-entrypoint.…"   22 minutes ago      Up 22 minutes       80/tcp              nginxweb.2.qjc2tz
oyigsoczn4ygjju6ull
arsalan@arsalan:~$ 
```

30. Visualize that both the containers of nginxweb service are running on Node 2.

Node 1(browser): localhost:8080



arsalan
worker
7.659G RAM
x86_64/linux

pc
manager
7.543G RAM
x86_64/linux

nginxweb
image : nginx:latest@sha256:4cf620
tag : latest@sha256:4cf620a5c81390
updated : 6/1 15:29
e1c9912502d96e0110bea149ff7cd9
state : running

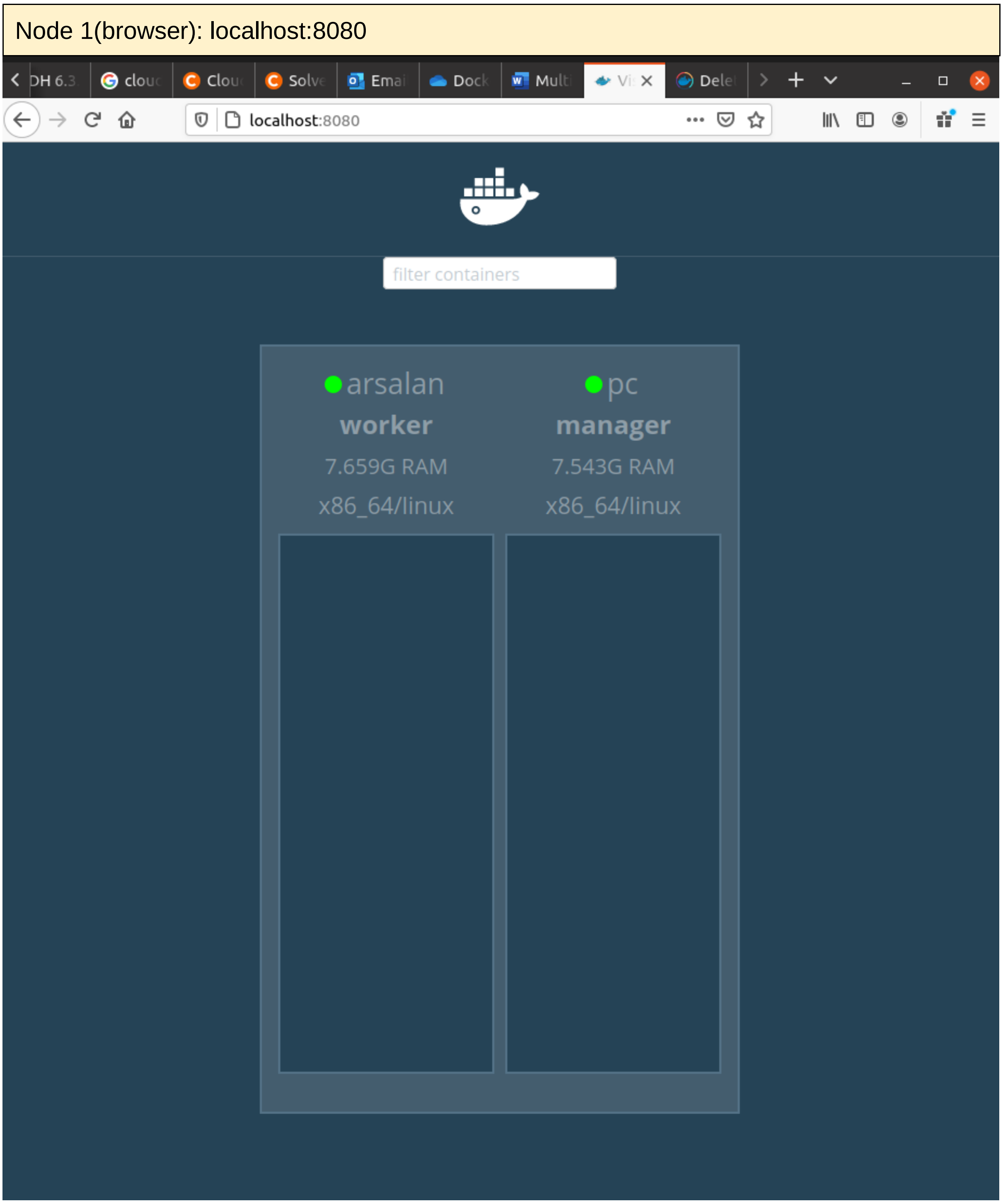
nginxweb
image : nginx:latest@sha256:4cf620
tag : latest@sha256:4cf620a5c81390
updated : 6/1 15:09
7ceec7c5e31dca37a8de5feb55fd023
state : running

31. Remove the nginxweb service from the docker swarm.

```
Node 1: docker service rm nginxweb
```

32. Verify by using the Visualizer container that service is removed.

```
Node 1(browser): localhost:8080
```

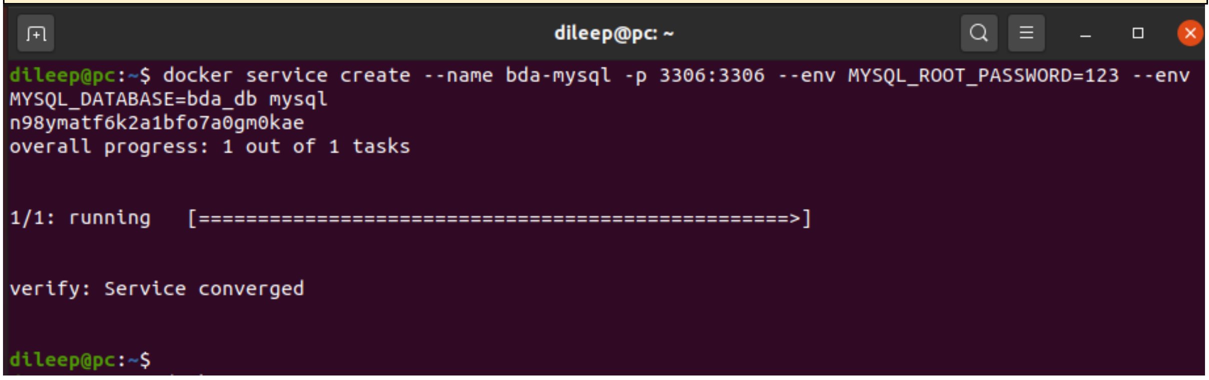


Step 4: Dataset querying on Multi Node Docker Swarm

In the scenario, we have created a MySQL service and migrated it to Swarm Worker Node, and Used MySQL-WorkBench on Swarm Manager Node. Connected both the nodes, and created Database, uploaded the dataset into the database, the Queried the database created on Swarm Worker Node from the Swarm Manager.

33. Create a service of MySql database on Node 1 (Swarm Manager)

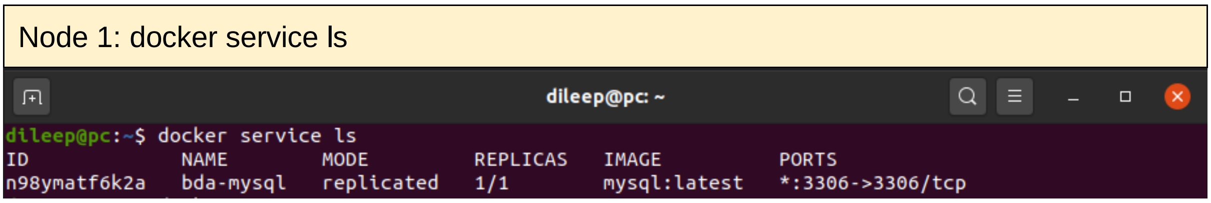
Node 1: docker service --name dba-mysql -p 3306:3306 –env MYSQL_ROOT_PASSWORD=123 --env MYSQL_DATABSE=bda_db mysql



```
dileep@pc:~$ docker service create --name bda-mysql -p 3306:3306 --env MYSQL_ROOT_PASSWORD=123 --env
MYSQL_DATABASE=bda_db mysql
n98ymatf6k2a1bfo7a0gm0kae
overall progress: 1 out of 1 tasks

1/1: running   [==================================================>]

verify: Service converged

dileep@pc:~$
```

34. Verify the MySql service is created

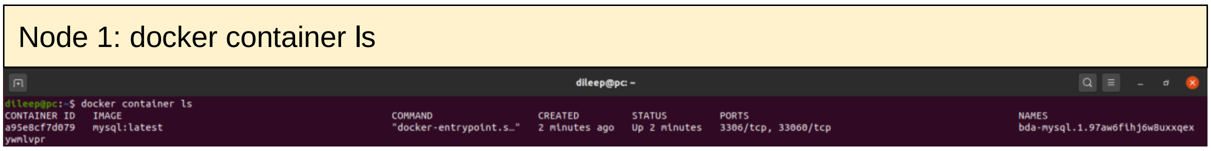
Node 1: docker service ls



```
dileep@pc:~$ docker service ls
ID             NAME        MODE         REPLICAS   IMAGE          PORTS
n98ymatf6k2a   bda-mysql   replicated   1/1        mysql:latest   *:3306->3306/tcp
```

35. Verify the container is created of the service

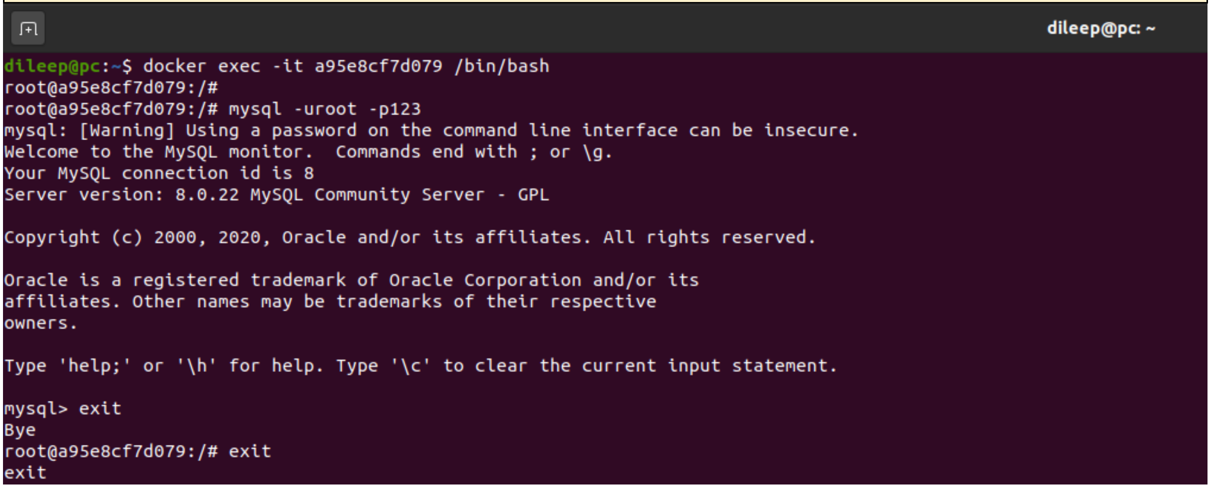
Node 1: docker container ls



```
dileep@pc:~$ docker container ls
CONTAINER ID   IMAGE                                                    COMMAND                  CREATED         STATUS         PORTS                                                      NAMES
a95e8cf7d079   mysql:latest                                             "docker-entrypoint.s…"   2 minutes ago   Up 2 minutes   3306/tcp, 33060/tcp                                        bda-mysql.1.97aw6fihj6w8uxxqex
ywmlvpr
```

36. Verify the container is running and exit from it

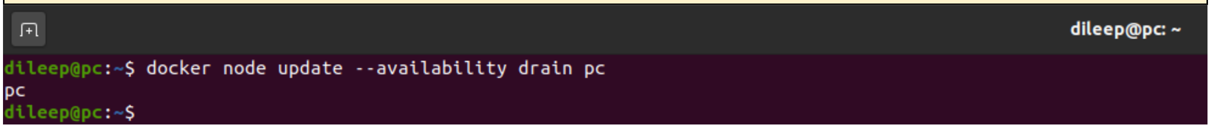
```
Node 1: docker exec -it <container-id> /bin/bash
Node 1: mysql -uroot -p123
Node 1: exit
```



```
dileep@pc:~$ docker exec -it a95e8cf7d079 /bin/bash
root@a95e8cf7d079:/#
root@a95e8cf7d079:/# mysql -uroot -p123
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.22 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> exit
Bye
root@a95e8cf7d079:/# exit
exit
```

37. Drain the Node 1 (Swarm Manager) to migrate the service to Node 2 (Swarm Worker)

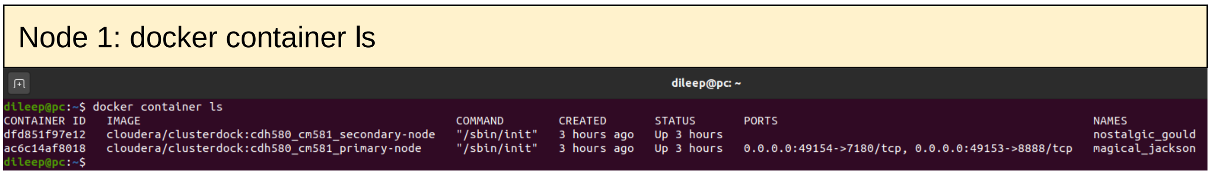
```
Node 1: docker node update  --availability drain pc
```



```
dileep@pc:~$ docker node update --availability drain pc
pc
dileep@pc:~$
```

38. Verify the container is no more running on Node 1 (Swarm Manager)

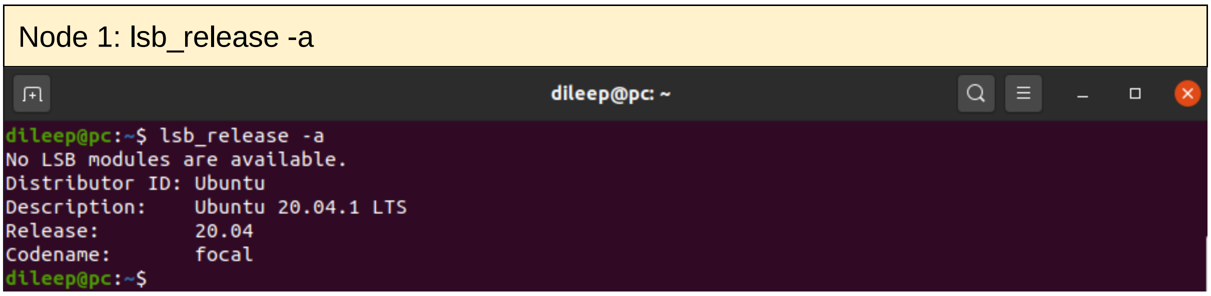
```
Node 1: docker container ls
```



```
dileep@pc:~$ docker container ls
CONTAINER ID   IMAGE                                          COMMAND         CREATED         STATUS         PORTS                                            NAMES
dfd851f97e12   cloudera/clusterdock:cdh580_cm581_secondary-node   "/sbin/init"   3 hours ago   Up 3 hours                                                      nostalgic_gould
ac6c14af8018   cloudera/clusterdock:cdh580_cm581_primary-node     "/sbin/init"   3 hours ago   Up 3 hours   0.0.0.0:49154->7180/tcp, 0.0.0.0:49153->8888/tcp   magical_jackson
dileep@pc:~$
```

39. Check the Release of Ubuntu on Node 1 (Swarm Manager) to download appropriate MySql Workbench installation package

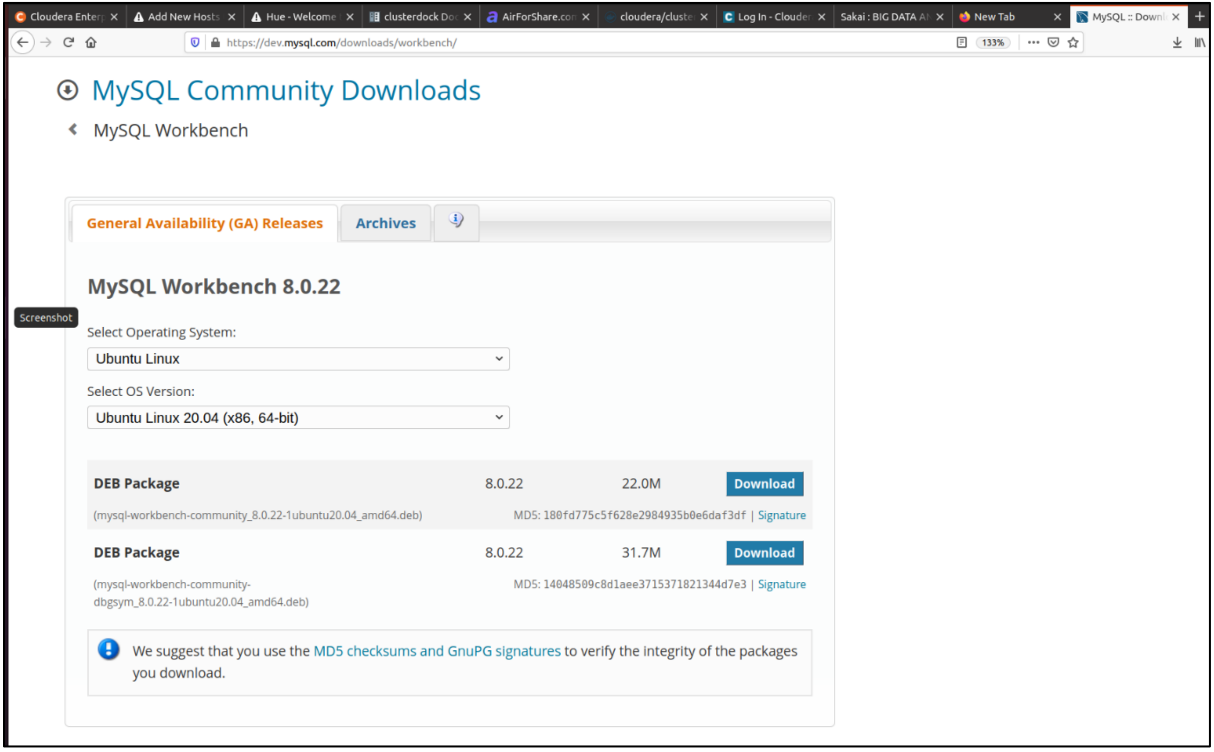
```
Node 1: lsb_release -a
```



```
dileep@pc:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.1 LTS
Release:        20.04
Codename:       focal
dileep@pc:~$
```

40. Download appropriate MySql Workbench installation package on Node 1 (Swarm Manager) from its official website

Node 1 (browser): https://dev.mysql.com/downloads/workbench/



MySQL Community Downloads

MySQL Workbench

General Availability (GA) Releases | Archives

MySQL Workbench 8.0.22

Select Operating System:
Ubuntu Linux

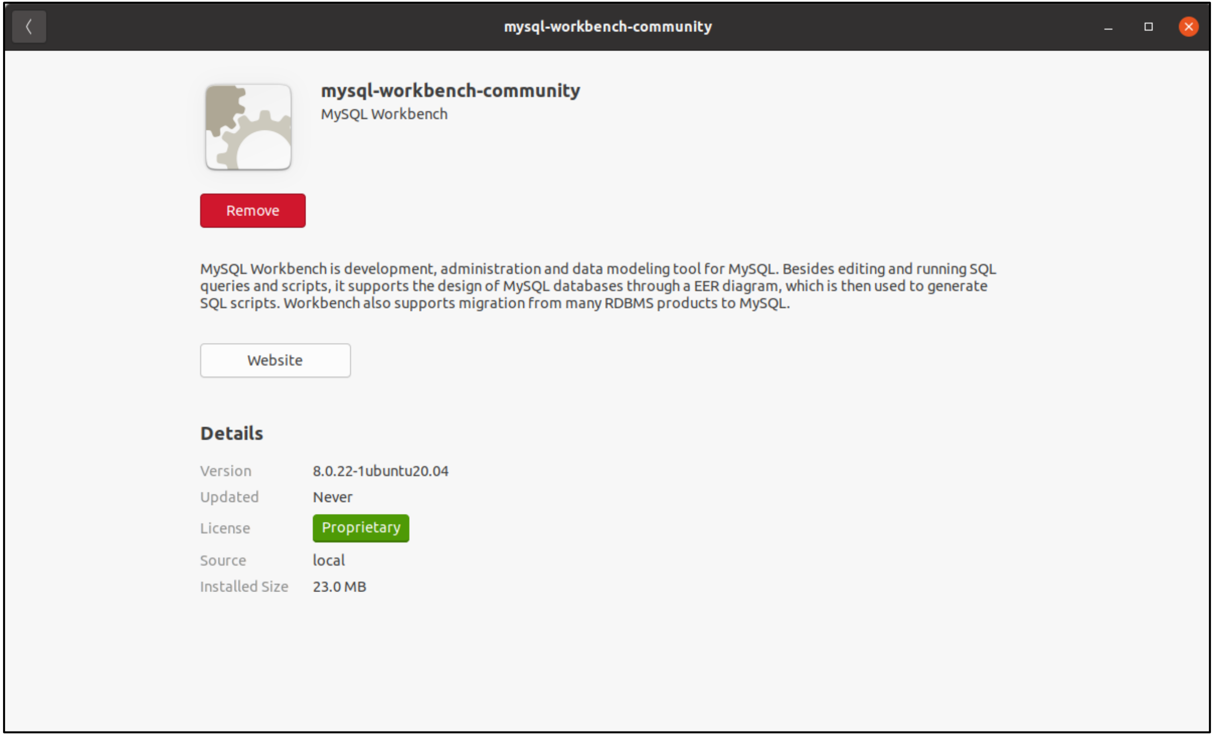
Select OS Version:
Ubuntu Linux 20.04 (x86, 64-bit)

DEB Package (mysql-workbench-community_8.0.22-1ubuntu20.04_amd64.deb)	8.0.22	22.0M	Download MD5: 180fd775c5f628e2984935b0e6daf3df | Signature
DEB Package (mysql-workbench-community-dbgsym_8.0.22-1ubuntu20.04_amd64.deb)	8.0.22	31.7M	Download MD5: 14048509c8d1aee3715371821344d7e3 | Signature

We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

41. Install the MySql Workbench package on Node 1 (Swarm Manager)

Node 1:



mysql-workbench-community
MySQL Workbench

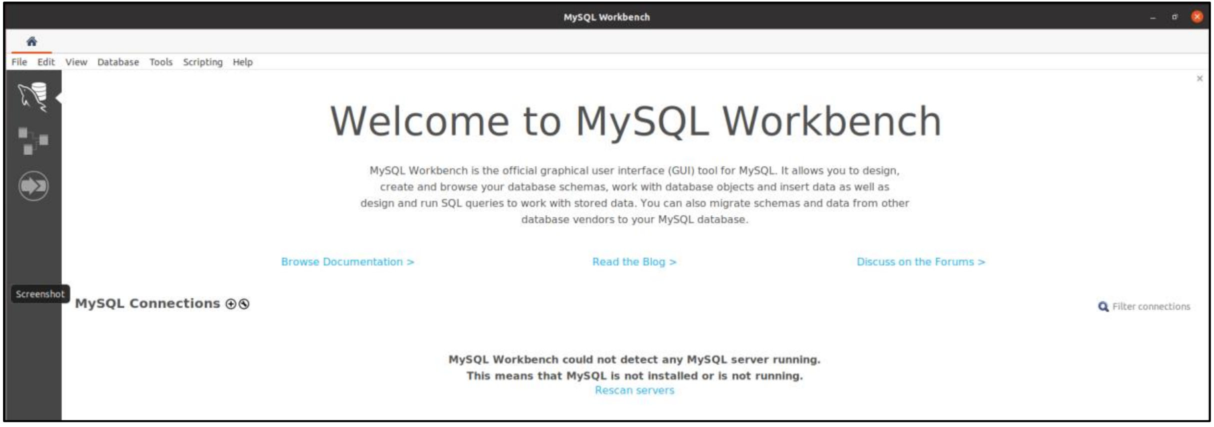
MySQL Workbench is development, administration and data modeling tool for MySQL. Besides editing and running SQL queries and scripts, it supports the design of MySQL databases through a EER diagram, which is then used to generate SQL scripts. Workbench also supports migration from many RDBMS products to MySQL.

Details

Version	8.0.22-1ubuntu20.04
Updated	Never
License	Proprietary
Source	local
Installed Size	23.0 MB

42. Run the MySql Workbench, and crate a new connection

Node 1 > MySQL-Workbench > Home: click > MySQL Connections + sign



Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

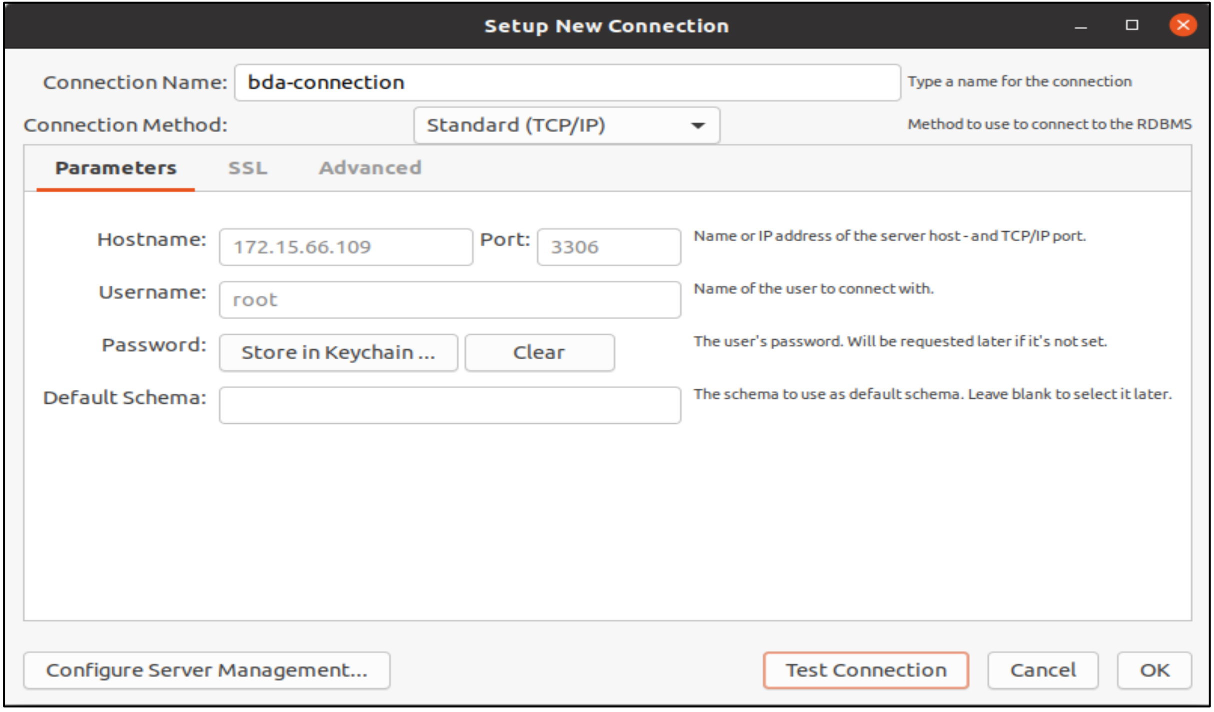
MySQL Connections ⊕⊗

MySQL Workbench could not detect any MySQL server running.
This means that MySQL is not installed or is not running.

43. Create a database connection to MySQL database container running on Node 2 (Swarm worker) by providing following values:

Node 1 > MySQL-Workbench:

Connection name: Used defined connection name
Hostname: IP-address-of-the-host-running-MySQL-Database
Port: Port address that is mapped to the MySQL-Database on the host. This port is defined while creating the service.

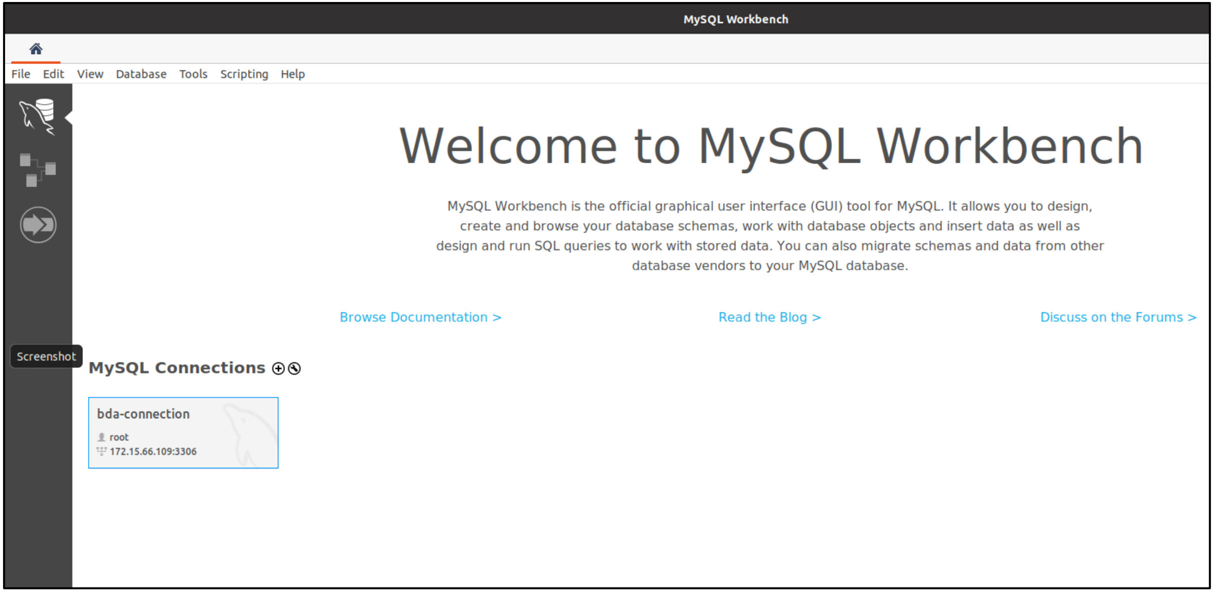


Connection Name: bda-connection — Type a name for the connection
Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

Hostname: 172.15.66.109 Port: 3306 — Name or IP address of the server host - and TCP/IP port.
Username: root — Name of the user to connect with.
Password: — The user's password. Will be requested later if it's not set.
Default Schema: — The schema to use as default schema. Leave blank to select it later.

44. Verify the connection (bda-connection) is created

Node 1:



Welcome to MySQL Workbench

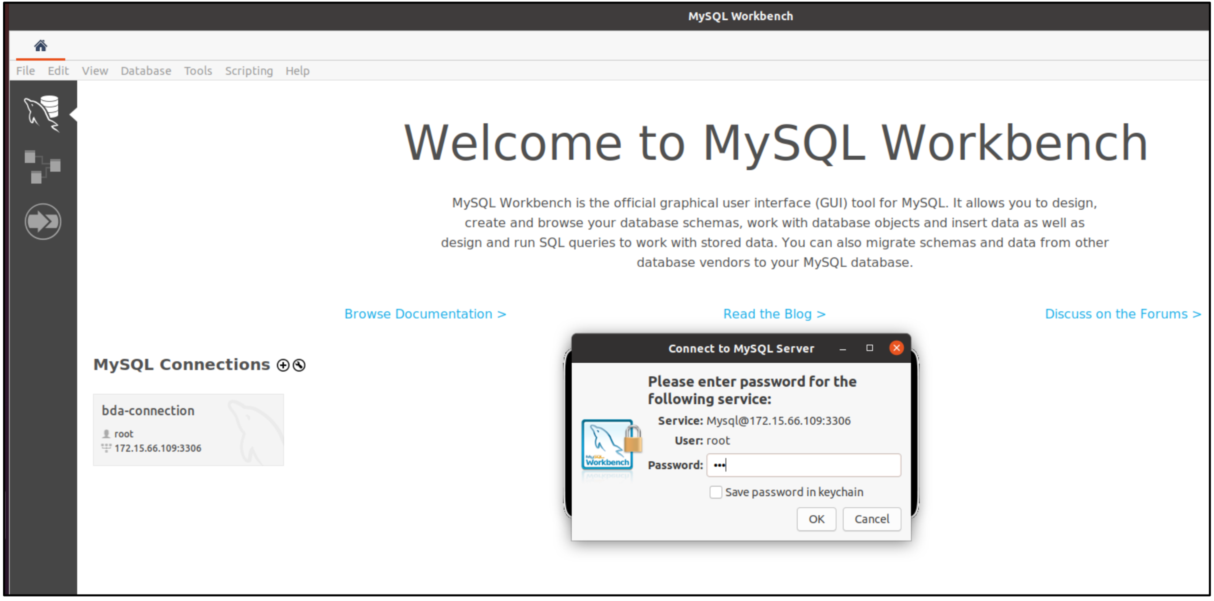
MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

MySQL Connections ⊕ ⊗

bda-connection
root
172.15.66.109:3306

45. Login to MySQL-Database via recently created connection, by providing password set while creating the MySQL-Database service.

Node 1: Password: 123



Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

MySQL Connections ⊕ ⊗

bda-connection
root
172.15.66.109:3306

Connect to MySQL Server

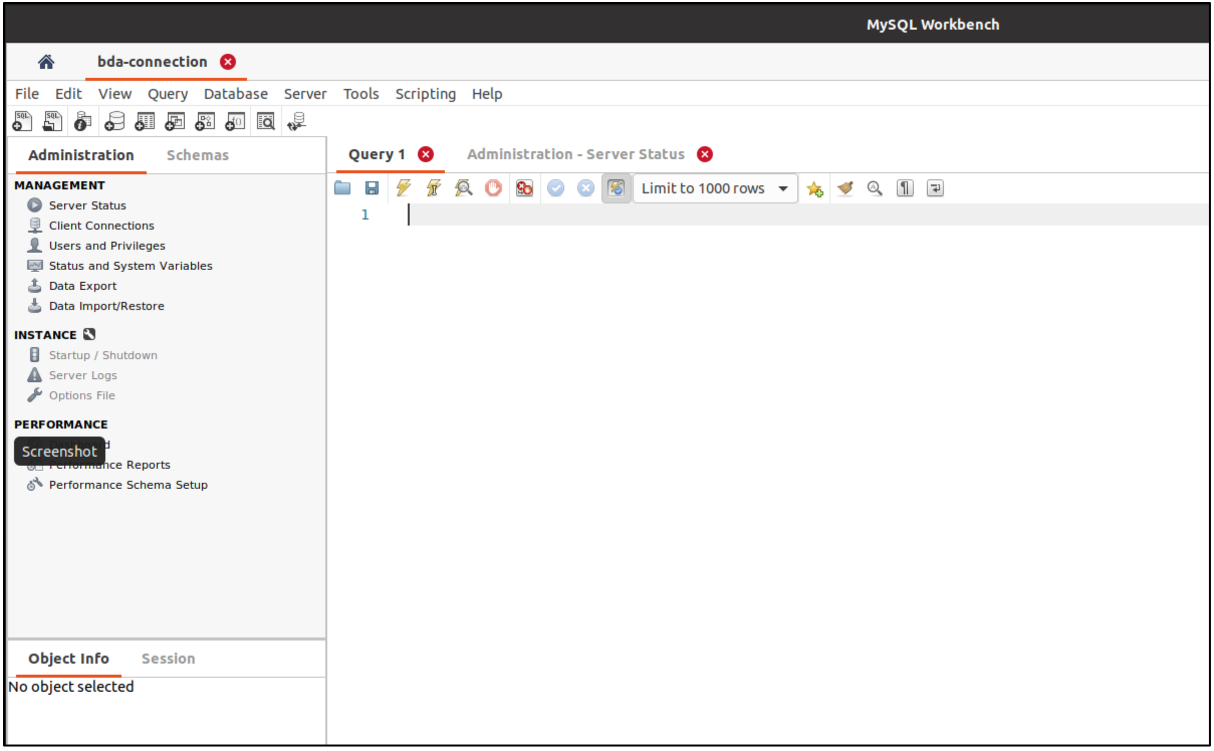
Please enter password for the following service:

Service: Mysql@172.15.66.109:3306
User: root
Password: •••

□ Save password in keychain

46. Verify the bda-connection is opened

Node 1:

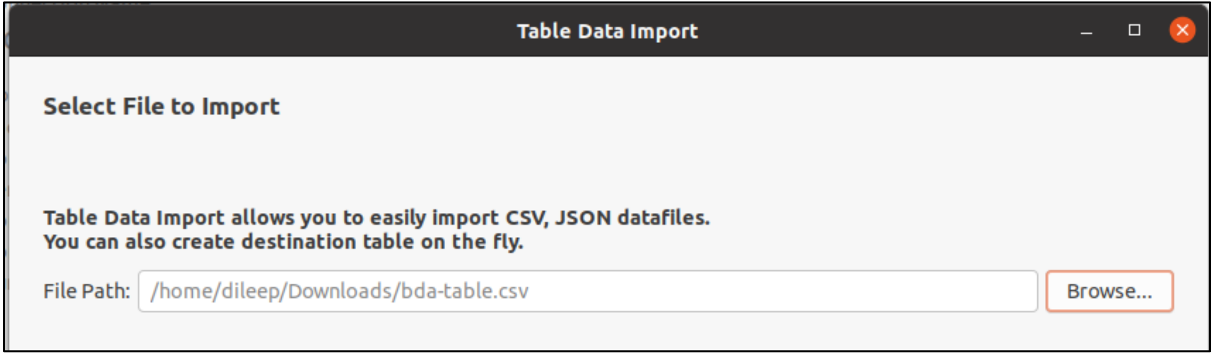


47. Access **Table Data Import Wizard** to import the **bda-table.csv** file into the MySQL-Database

Node 1 > MySQL-Workbench: bda-db > Tables > Right Click > Table Data Import Wizard

48. Provide path to **bda-table.csv** located on your hard disk.

Node 1:



Select File to Import

Table Data Import allows you to easily import CSV, JSON datafiles.
You can also create destination table on the fly.

File Path: /home/dileep/Downloads/bda-table.csv

49. Specify the database and table name to import the dataset into it.

Node 1:

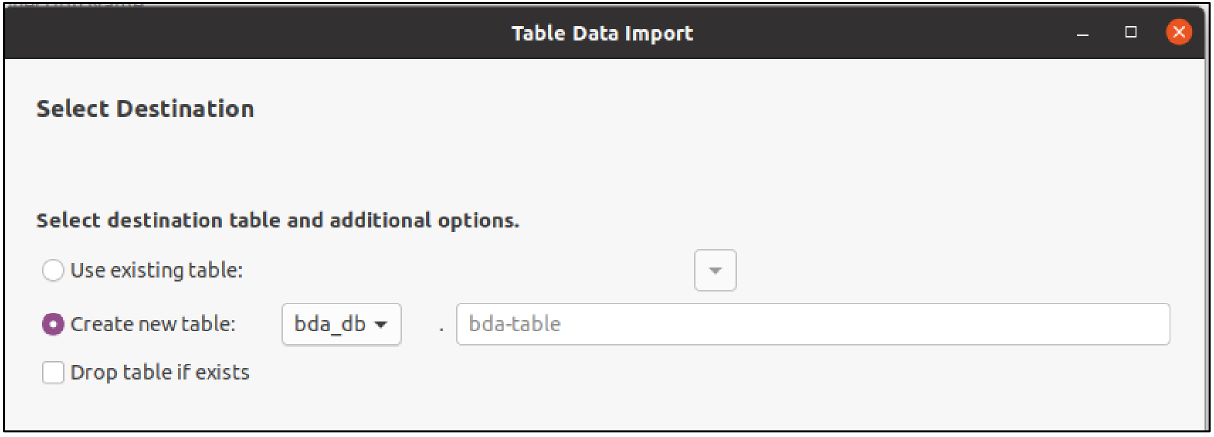


Table Data Import

Select Destination

Select destination table and additional options.

○ Use existing table:

● Create new table: bda_db . bda-table

□ Drop table if exists

50. Verify the data-types of attributes of the dataset

Node 1 :

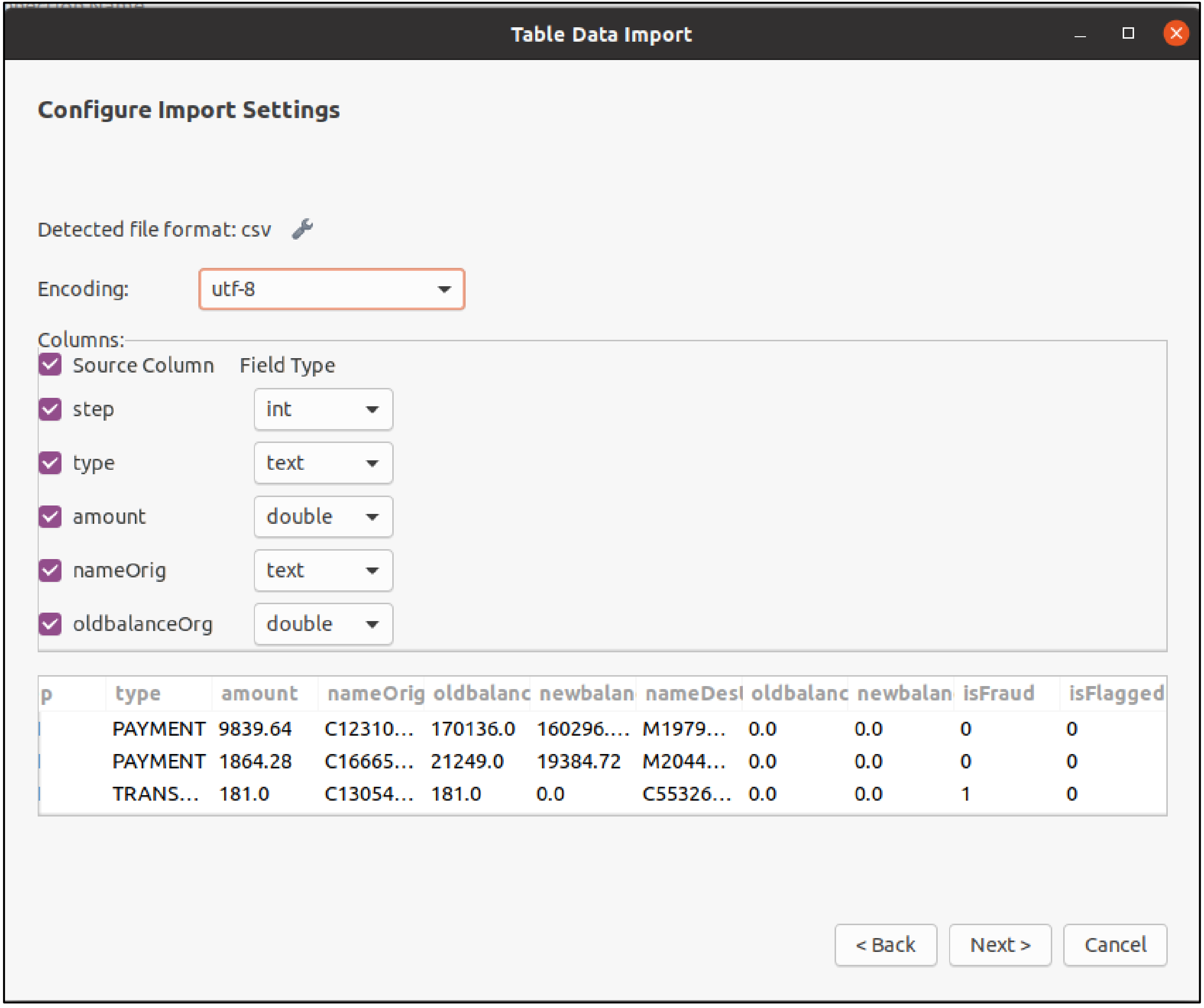


Table Data Import

Configure Import Settings

Detected file format: csv

Encoding: utf-8

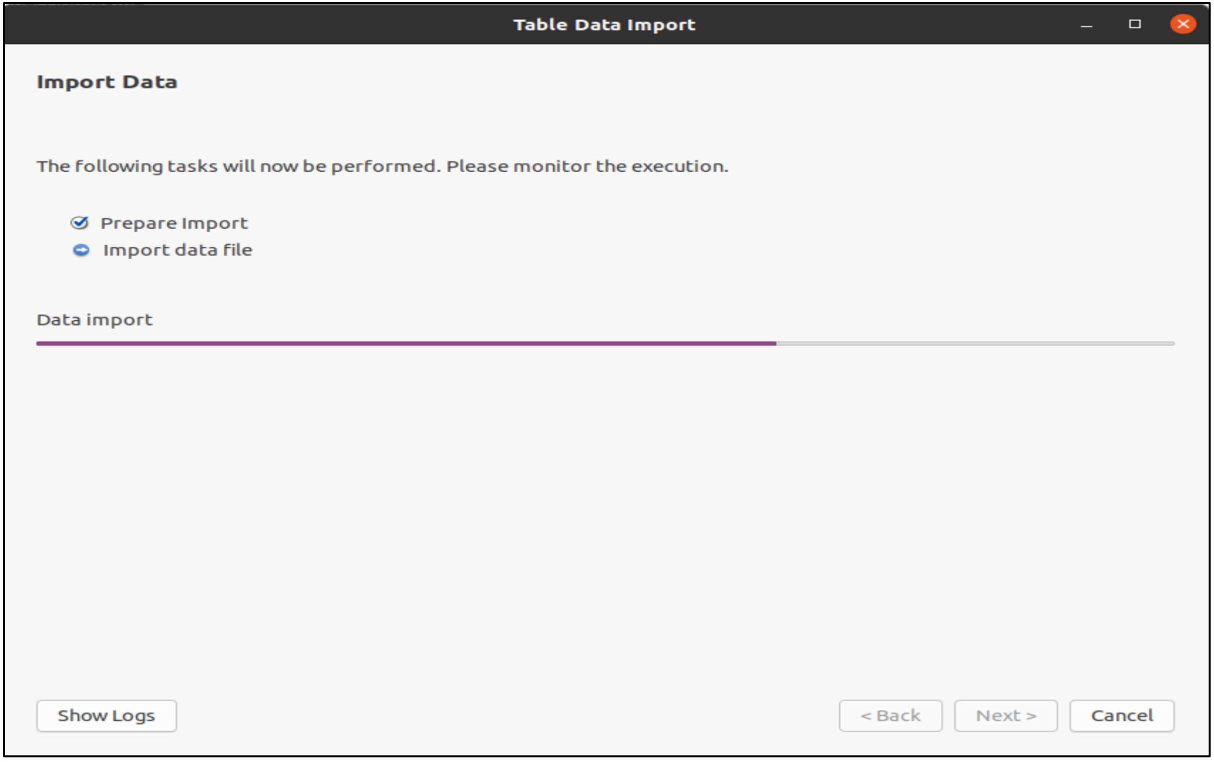
Columns:

☑ Source Column	Field Type
☑ step	int
☑ type	text
☑ amount	double
☑ nameOrig	text
☑ oldbalanceOrg	double

p	type	amount	nameOrig	oldbalanc	newbalan	nameDest	oldbalanc	newbalan	isFraud	isFlagged
	PAYMENT	9839.64	C12310...	170136.0	160296....	M1979...	0.0	0.0	0	0
	PAYMENT	1864.28	C16665...	21249.0	19384.72	M2044...	0.0	0.0	0	0
	TRANS...	181.0	C13054...	181.0	0.0	C55326...	0.0	0.0	1	0

51. Data is importing in progress

Node 1:



Import Data

The following tasks will now be performed. Please monitor the execution.

- ☑ Prepare Import
- ● Import data file

Data import

52. Verify the to **bda-table** is created

Node 1 (MySQL-Workbench): bda_db > Right click > Refresh all

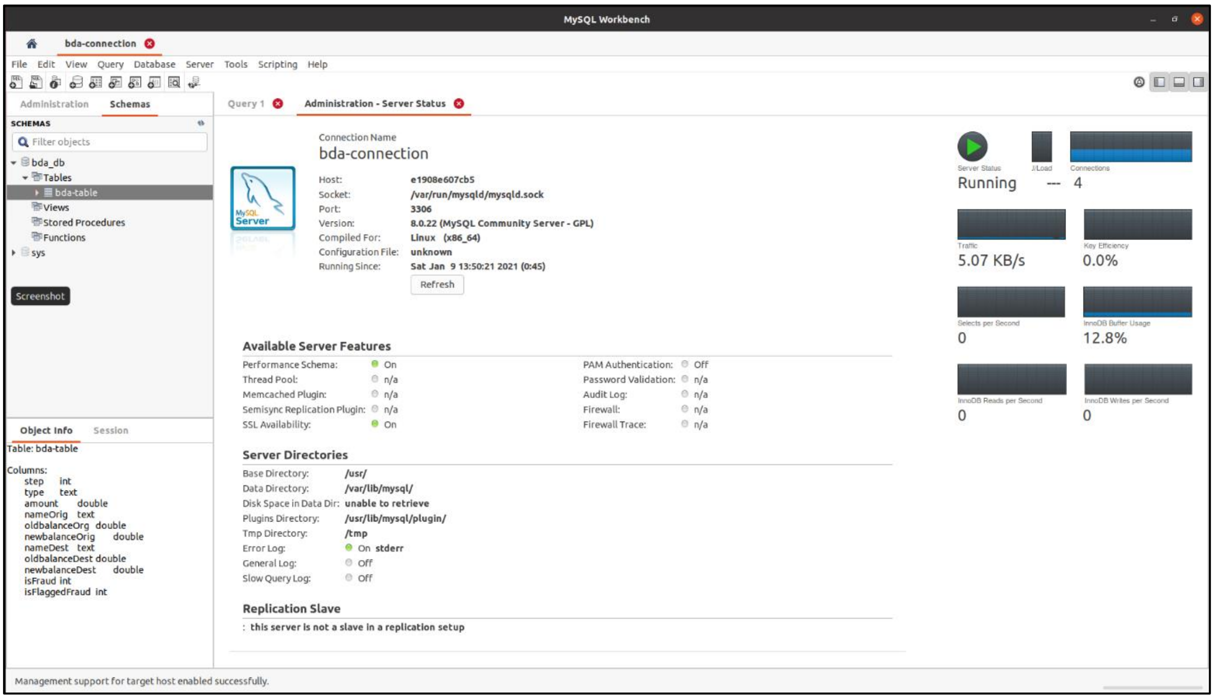


Table: bda-table

Columns:

```
    step    int
    type    text
    amount      double
    nameOrig   text
    oldbalanceOrg  double
    newbalanceOrig      double
    nameDest   text
    oldbalanceDest double
    newbalanceDest      double
    isFraud int
    isFlaggedFraud  int
```

Connection Name

bda-connection

Host:	e1908e607cb5
Socket:	/var/run/mysqld/mysqld.sock
Port:	3306
Version:	8.0.22 (MySQL Community Server - GPL)
Compiled For:	Linux (x86_64)
Configuration File:	unknown
Running Since:	Sat Jan 9 13:50:21 2021 (0:45)

Server Status: Running

J/Load: ---

Connections: 4

Traffic: 5.07 KB/s

Key Efficiency: 0.0%

Selects per Second: 0

InnoDB Buffer Usage: 12.8%

InnoDB Reads per Second: 0

InnoDB Writes per Second: 0

Available Server Features

Performance Schema:	● On	PAM Authentication:	○ Off
Thread Pool:	○ n/a	Password Validation:	○ n/a
Memcached Plugin:	○ n/a	Audit Log:	○ n/a
Semisync Replication Plugin:	○ n/a	Firewall:	○ n/a
SSL Availability:	● On	Firewall Trace:	○ n/a

Server Directories

Base Directory:	/usr/
Data Directory:	/var/lib/mysql/
Disk Space in Data Dir:	unable to retrieve
Plugins Directory:	/usr/lib/mysql/plugin/
Tmp Directory:	/tmp
Error Log:	● On stderr
General Log:	○ Off
Slow Query Log:	○ Off

Replication Slave

: this server is not a slave in a replication setup

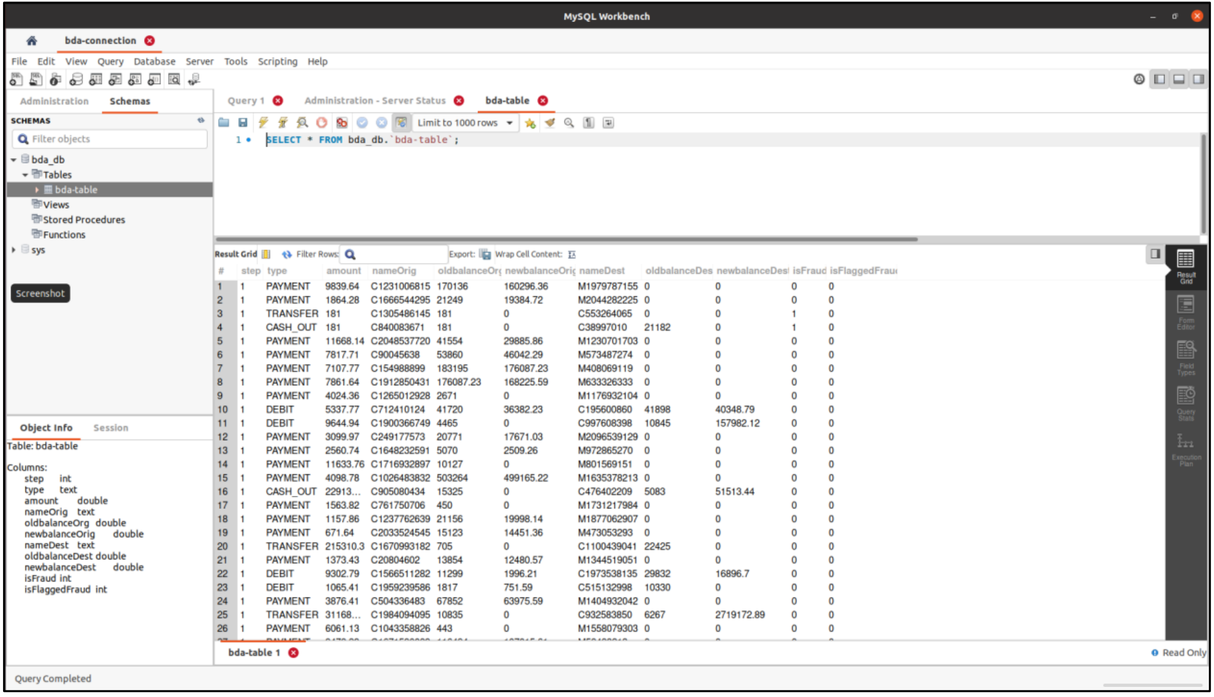
Management support for target host enabled successfully.

53. Run Select * Query

Node 1 (MySQL-Workbench): bda_db > Tables > bda-table > Right click > Select Row Limit 1000

54. Verify the output of Select * Query.
We can also write a query here and use other queries.

Node 1:



```
SELECT * FROM bda_db.`bda-table`;
```

Table: bda-table

Columns:
step int
type text
amount double
nameOrig text
oldbalanceOrg double
newbalanceOrig double
nameDest text
oldbalanceDest double
newbalanceDest double
isFraud int
isFlaggedFraud int

#	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDes	newbalanceDest	isFraud	isFlaggedFrau
1	1	PAYMENT	9839.64	C1231006815	170136	160296.36	M1979787155	0	0	0	0
2	1	PAYMENT	1864.28	C1666544295	21249	19384.72	M2044282225	0	0	0	0
3	1	TRANSFER	181	C1305486145	181	0	C553264065	0	0	1	0
4	1	CASH_OUT	181	C840083671	181	0	C38997010	21182	0	1	0
5	1	PAYMENT	11668.14	C2048537720	41554	29885.86	M1230701703	0	0	0	0
6	1	PAYMENT	7817.71	C90045638	53860	46042.29	M573487274	0	0	0	0
7	1	PAYMENT	7107.77	C154988899	183195	176087.23	M408069119	0	0	0	0
8	1	PAYMENT	7861.64	C1912850431	176087.23	168225.59	M633326333	0	0	0	0
9	1	PAYMENT	4024.36	C1265012928	2671	0	M1176932104	0	0	0	0
10	1	DEBIT	5337.77	C712410124	41720	36382.23	C195600860	41898	40348.79	0	0
11	1	DEBIT	9644.94	C1900366749	4465	0	C997608398	10845	157982.12	0	0
12	1	PAYMENT	3099.97	C249177573	20771	17671.03	M2096539129	0	0	0	0
13	1	PAYMENT	2560.74	C1648232591	5070	2509.26	M972865270	0	0	0	0
14	1	PAYMENT	11633.76	C1716932897	10127	0	M801569151	0	0	0	0
15	1	PAYMENT	4098.78	C1026483832	503264	499165.22	M1635378213	0	0	0	0
16	1	CASH_OUT	22913...	C905080434	15325	0	C476402209	5083	51513.44	0	0
17	1	PAYMENT	1563.82	C761750706	450	0	M1731217984	0	0	0	0
18	1	PAYMENT	1157.86	C1237762639	21156	19998.14	M1877062907	0	0	0	0
19	1	PAYMENT	671.64	C2033524545	15123	14451.36	M473053293	0	0	0	0
20	1	TRANSFER	215310.3	C1670993182	705	0	C1100439041	22425	0	0	0
21	1	PAYMENT	1373.43	C20804602	13854	12480.57	M1344519051	0	0	0	0
22	1	DEBIT	9302.79	C1566511282	11299	1996.21	C1973538135	29832	16896.7	0	0
23	1	DEBIT	1065.41	C1959239586	1817	751.59	C515132998	10330	0	0	0
24	1	PAYMENT	3876.41	C504336483	67852	63975.59	M1404932042	0	0	0	0
25	1	TRANSFER	31168...	C1984094095	10835	0	C932583850	6267	2719172.89	0	0
26	1	PAYMENT	6061.13	C1043358826	443	0	M1558079303	0	0	0	0