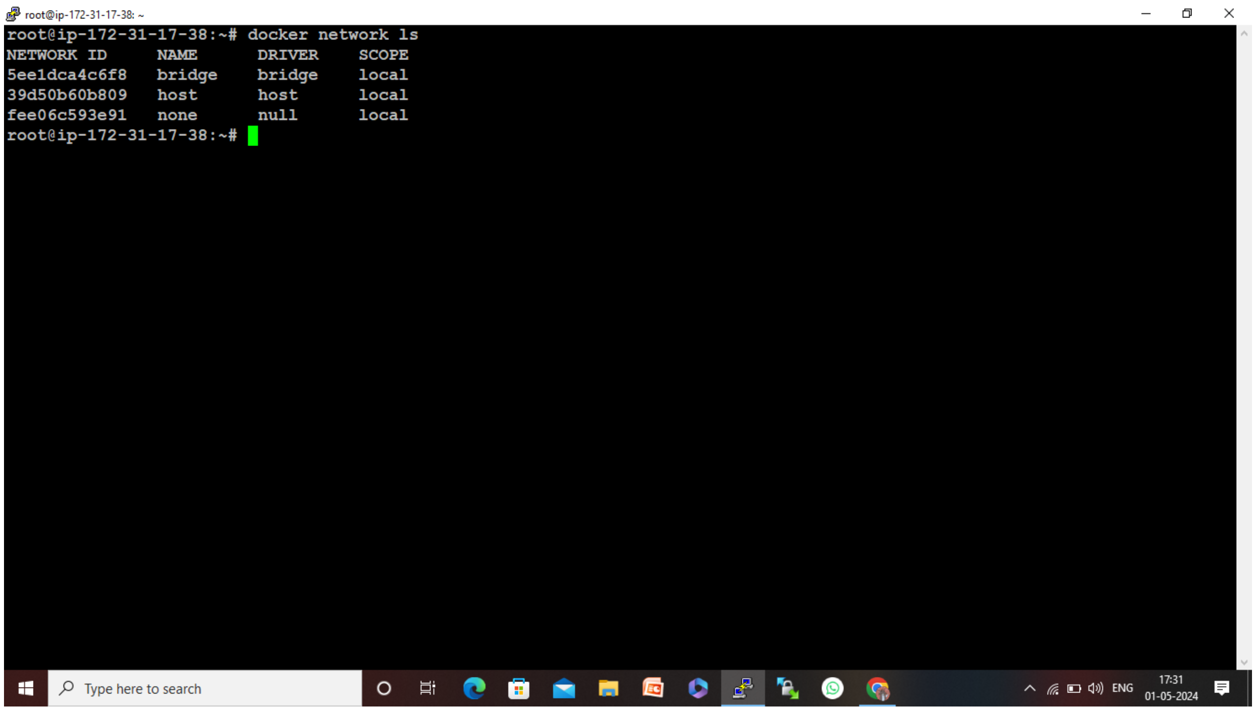
DOCKER NETWORKS

BRIDGE NETWORK: Bridge networks are the default network type in Docker. They allow containers to communicate with each other on the same host.

1.docker netwok ls - listing the network in docker host



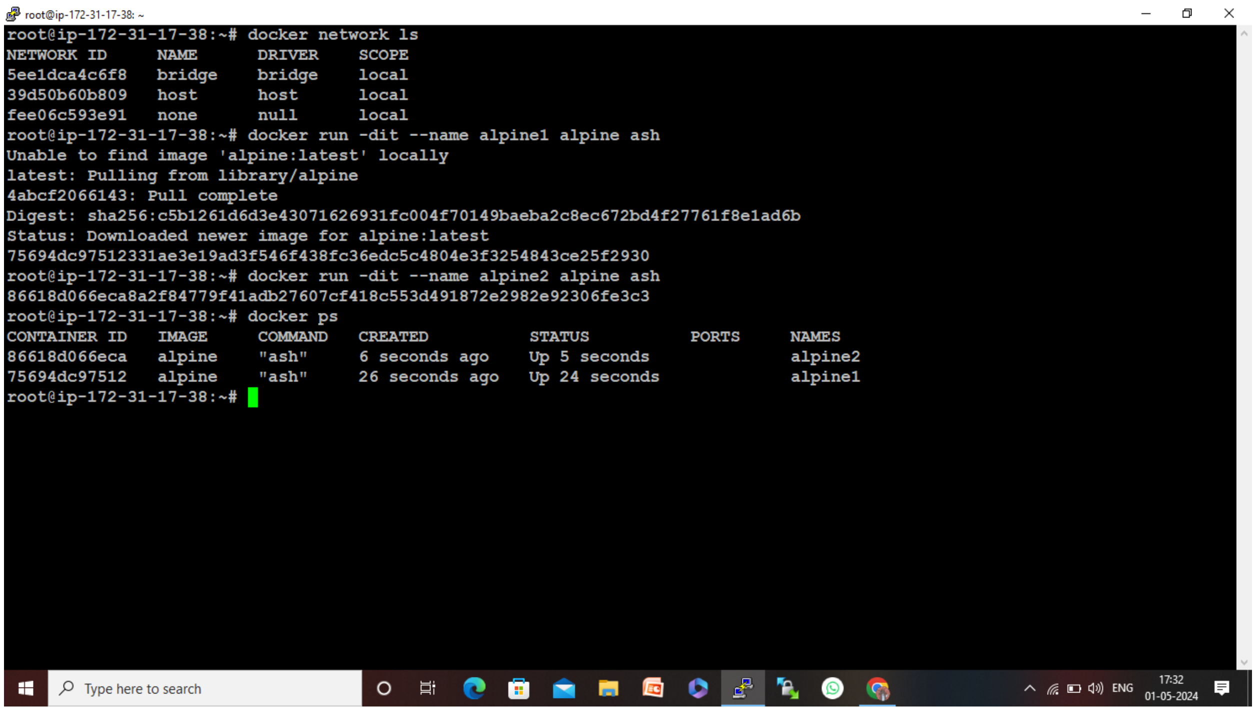
```
root@ip-172-31-17-38:~# docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
5ee1dca4c6f8   bridge    bridge    local
39d50b60b809   host      host      local
fee06c593e91   none      null      local
root@ip-172-31-17-38:~#
```

2.Start two alpine containers running ash, which is Alpine's default shell rather than bash. The -dit flags mean to start the container detached

Not specified any --network flags, the containers connect to the default bridge network.

docker run -dit --name alpine1 alpine ash

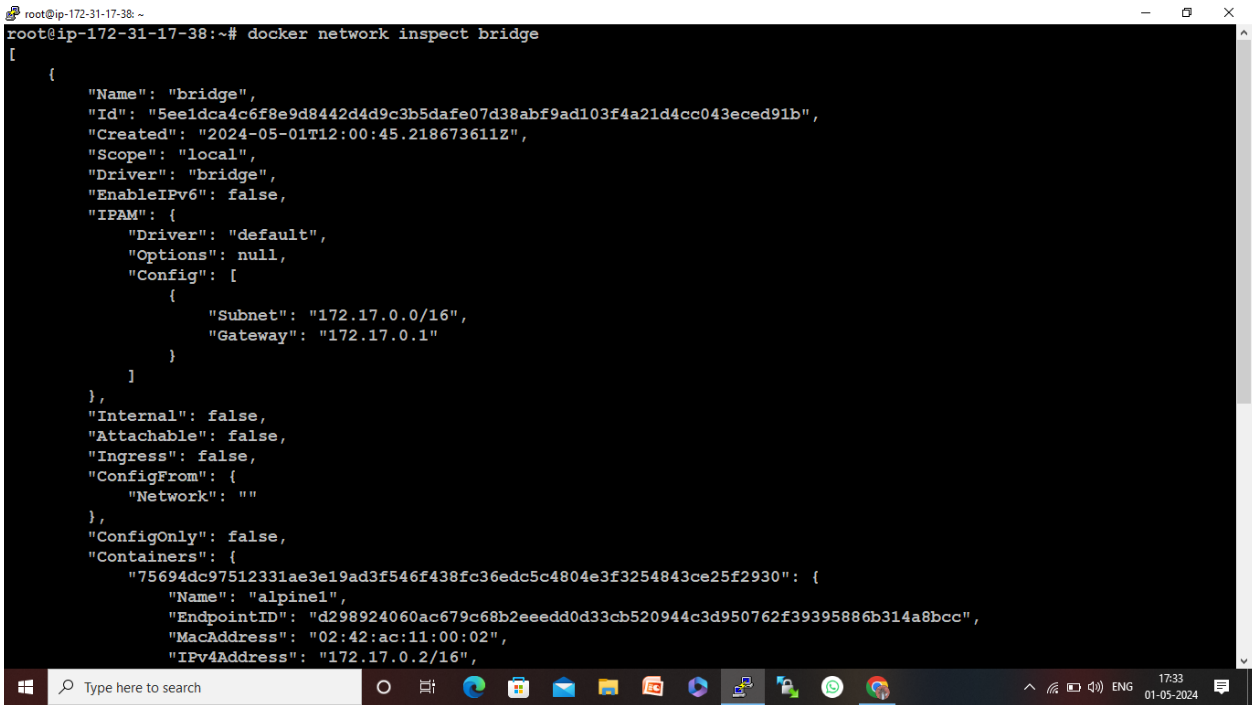
docker run -dit --name alpine2 alpine ash



root@ip-172-31-17-38: ~

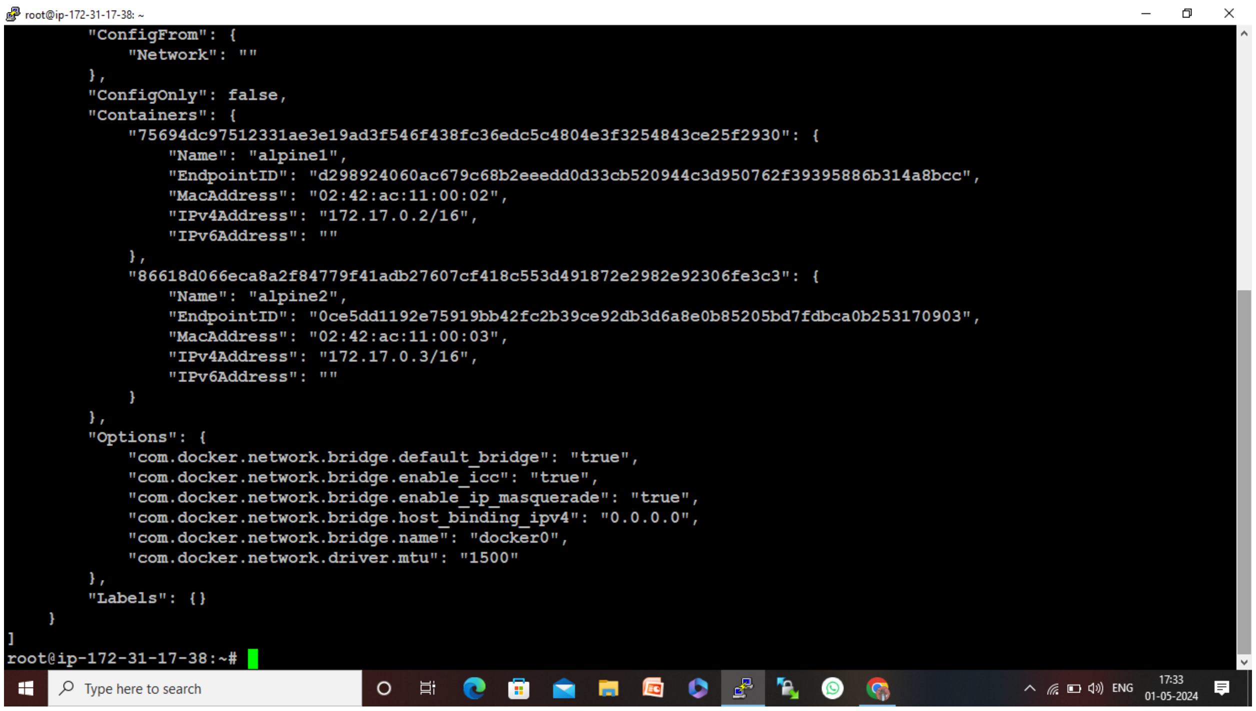
```
root@ip-172-31-17-38:~# docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
5ee1dca4c6f8   bridge    bridge    local
39d50b60b809   host      host      local
fee06c593e91   none      null      local
root@ip-172-31-17-38:~# docker run -dit --name alpine1 alpine ash
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
4abcf2066143: Pull complete
Digest: sha256:c5b1261d6d3e43071626931fc004f70149baeba2c8ec672bd4f27761f8e1ad6b
Status: Downloaded newer image for alpine:latest
75694dc97512331ae3e19ad3f546f438fc36edc5c4804e3f3254843ce25f2930
root@ip-172-31-17-38:~# docker run -dit --name alpine2 alpine ash
86618d066eca8a2f84779f41adb27607cf418c553d491872e2982e92306fe3c3
root@ip-172-31-17-38:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS          PORTS     NAMES
86618d066eca   alpine    "ash"     6 seconds ago    Up 5 seconds              alpine2
75694dc97512   alpine    "ash"     26 seconds ago   Up 24 seconds             alpine1
root@ip-172-31-17-38:~#
```

3. Inspect the bridge network to see what containers are connected to it.



root@ip-172-31-17-38: ~

```
root@ip-172-31-17-38:~# docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "5ee1dca4c6f8e9d8442d4d9c3b5dafe07d38abf9ad103f4a21d4cc043eced91b",
        "Created": "2024-05-01T12:00:45.218673611Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "75694dc97512331ae3e19ad3f546f438fc36edc5c4804e3f3254843ce25f2930": {
                "Name": "alpine1",
                "EndpointID": "d298924060ac679c68b2eeedd0d33cb520944c3d950762f39395886b314a8bcc",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
```



```
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "75694dc97512331ae3e19ad3f546f438fc36edc5c4804e3f3254843ce25f2930": {
                "Name": "alpine1",
                "EndpointID": "d298924060ac679c68b2eeedd0d33cb520944c3d950762f39395886b314a8bcc",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
            },
            "86618d066eca8a2f84779f41adb27607cf418c553d491872e2982e92306fe3c3": {
                "Name": "alpine2",
                "EndpointID": "0ce5dd1192e75919bb42fc2b39ce92db3d6a8e0b85205bd7fdbca0b253170903",
                "MacAddress": "02:42:ac:11:00:03",
                "IPv4Address": "172.17.0.3/16",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
    }
]
root@ip-172-31-17-38:~# 
```

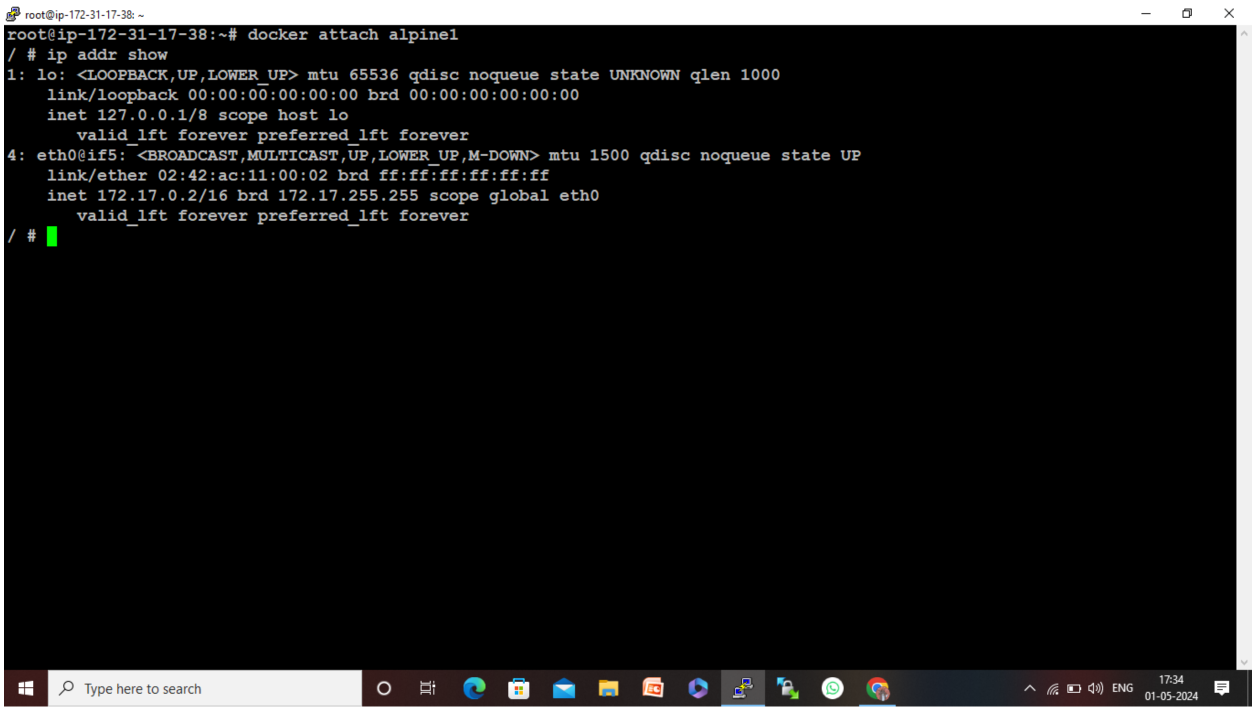
4. The containers are running in the background. Use the docker attach command to connect to alpine1

The prompt changes to # to indicate that you are the root user within the container.

Use the ip addr show command to show the network interfaces for alpine1.

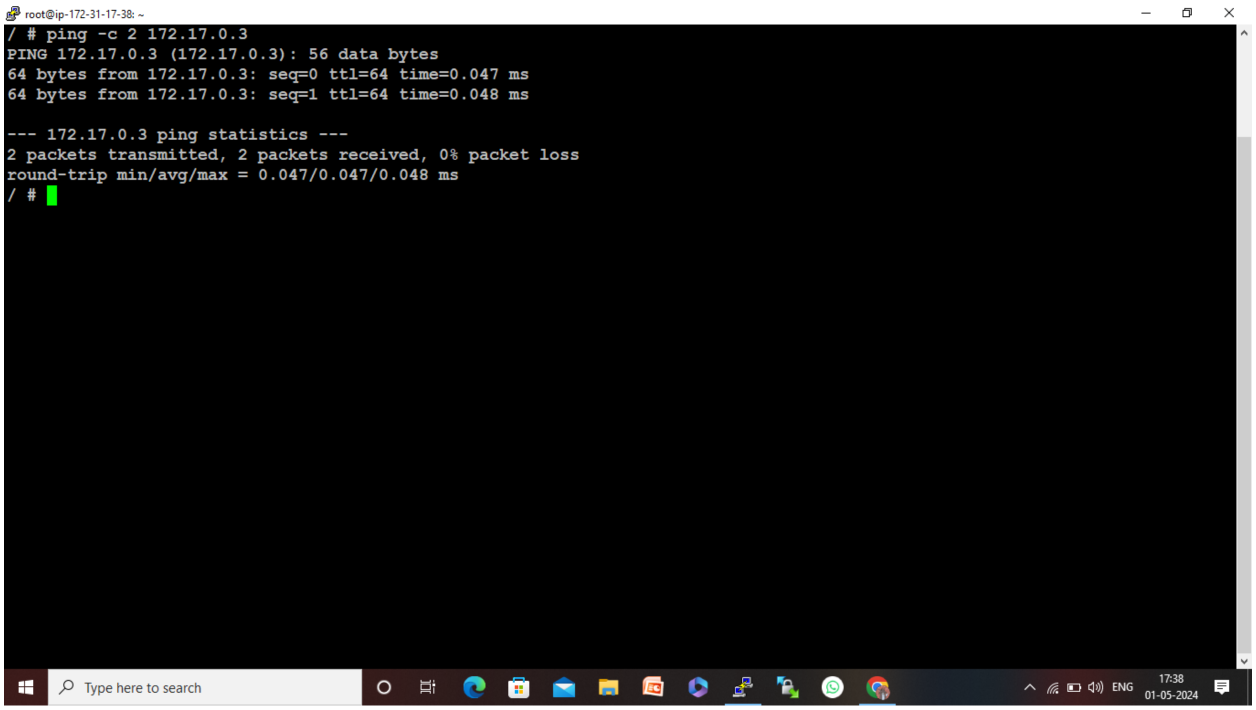
The first interface is the loopback device .

the second interface has the IP address 172.17.0.2, which is the same address shown for alpine1.



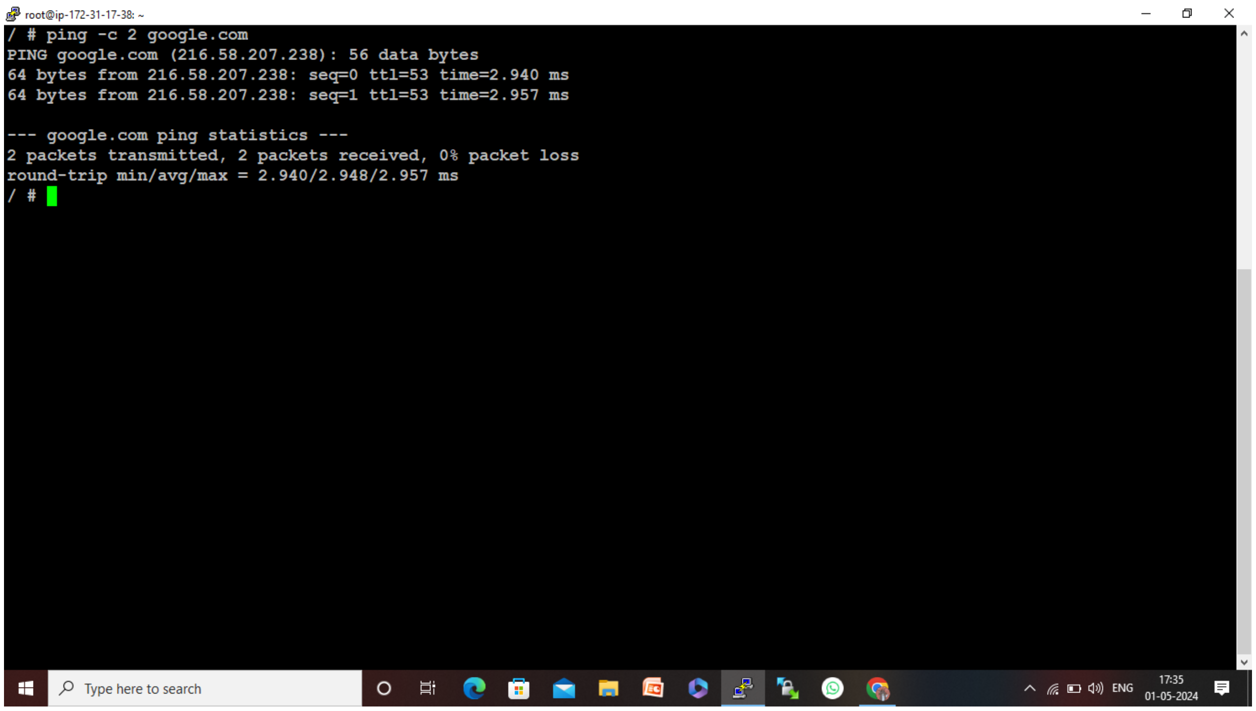
```
root@ip-172-31-17-38:~# docker attach alpine1
/ # ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
4: eth0@if5: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
/ #
```

5. try to ping the second container. First, ping it by its IP address, 172.17.0.3:



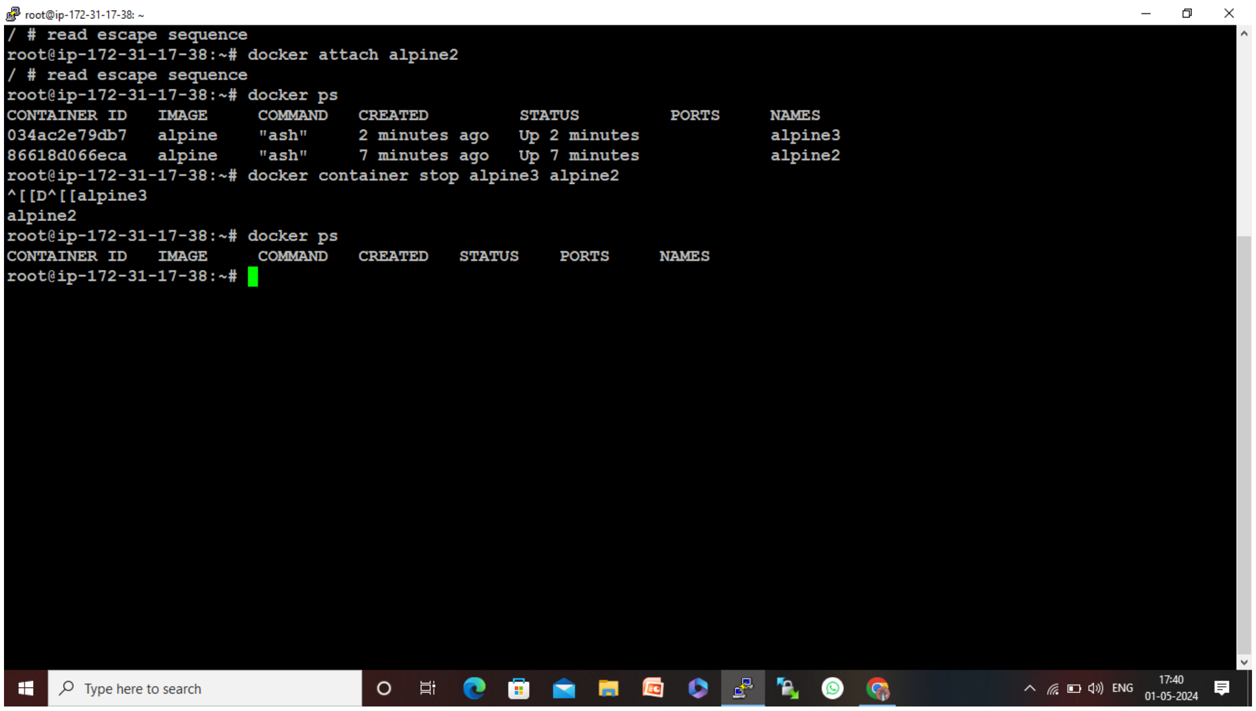
```
/ # ping -c 2 172.17.0.3
PING 172.17.0.3 (172.17.0.3): 56 data bytes
64 bytes from 172.17.0.3: seq=0 ttl=64 time=0.047 ms
64 bytes from 172.17.0.3: seq=1 ttl=64 time=0.048 ms

--- 172.17.0.3 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.047/0.047/0.048 ms
/ #
```

6. Now try within alpine1, make sure you can connect to the internet by pinging google.com. The -c 2 flag limits the command to two ping attempts.



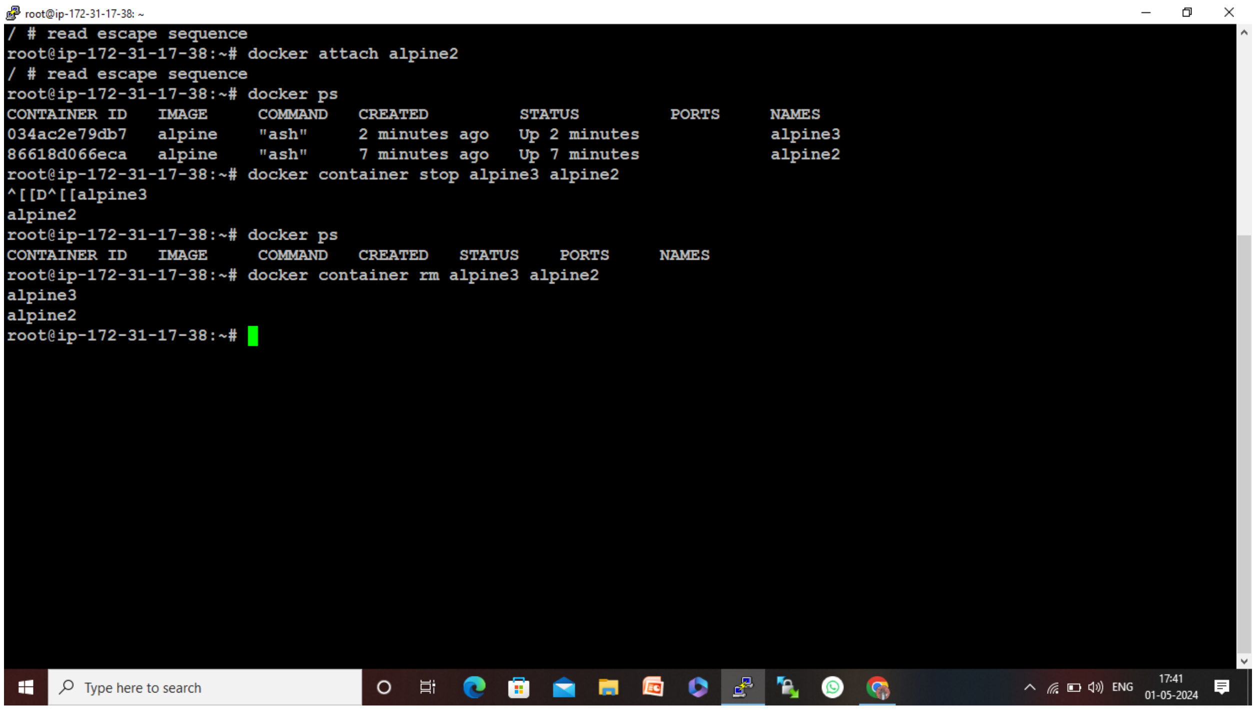
```
/ # ping -c 2 google.com
PING google.com (216.58.207.238): 56 data bytes
64 bytes from 216.58.207.238: seq=0 ttl=53 time=2.940 ms
64 bytes from 216.58.207.238: seq=1 ttl=53 time=2.957 ms

--- google.com ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 2.940/2.948/2.957 ms
/ #
```

7. Detach from alpine1 without stopping it by using the detach sequence, CTRL + p CTRL + q (hold down CTRL and type p followed by q).



```
/ # read escape sequence
root@ip-172-31-17-38:~# docker attach alpine2
/ # read escape sequence
root@ip-172-31-17-38:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS          PORTS     NAMES
034ac2e79db7   alpine    "ash"     2 minutes ago    Up 2 minutes              alpine3
86618d066eca   alpine    "ash"     7 minutes ago    Up 7 minutes              alpine2
root@ip-172-31-17-38:~# docker container stop alpine3 alpine2
^[[D^[[alpine3
alpine2
root@ip-172-31-17-38:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@ip-172-31-17-38:~#
```

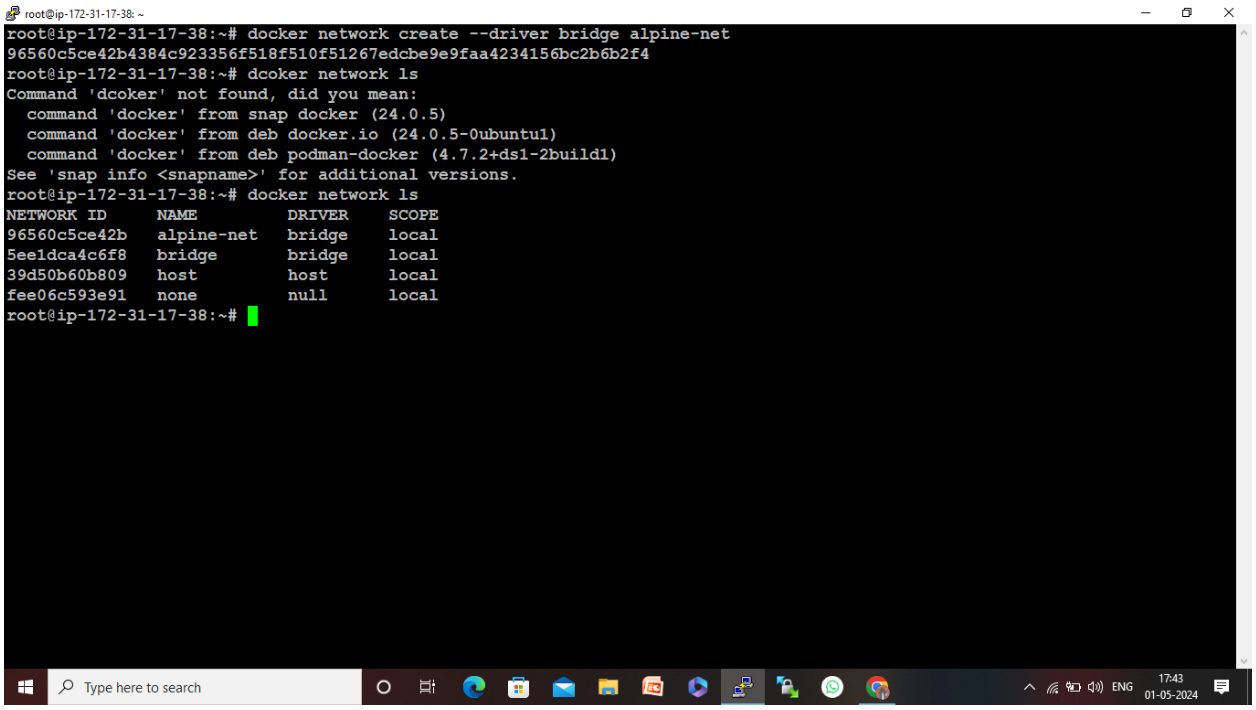
8. Stop and remove both containers.



```
/ # read escape sequence
root@ip-172-31-17-38:~# docker attach alpine2
/ # read escape sequence
root@ip-172-31-17-38:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED         STATUS         PORTS     NAMES
034ac2e79db7   alpine    "ash"     2 minutes ago   Up 2 minutes             alpine3
86618d066eca   alpine    "ash"     7 minutes ago   Up 7 minutes             alpine2
root@ip-172-31-17-38:~# docker container stop alpine3 alpine2
^[[D^[[alpine3
alpine2
root@ip-172-31-17-38:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@ip-172-31-17-38:~# docker container rm alpine3 alpine2
alpine3
alpine2
root@ip-172-31-17-38:~#
```

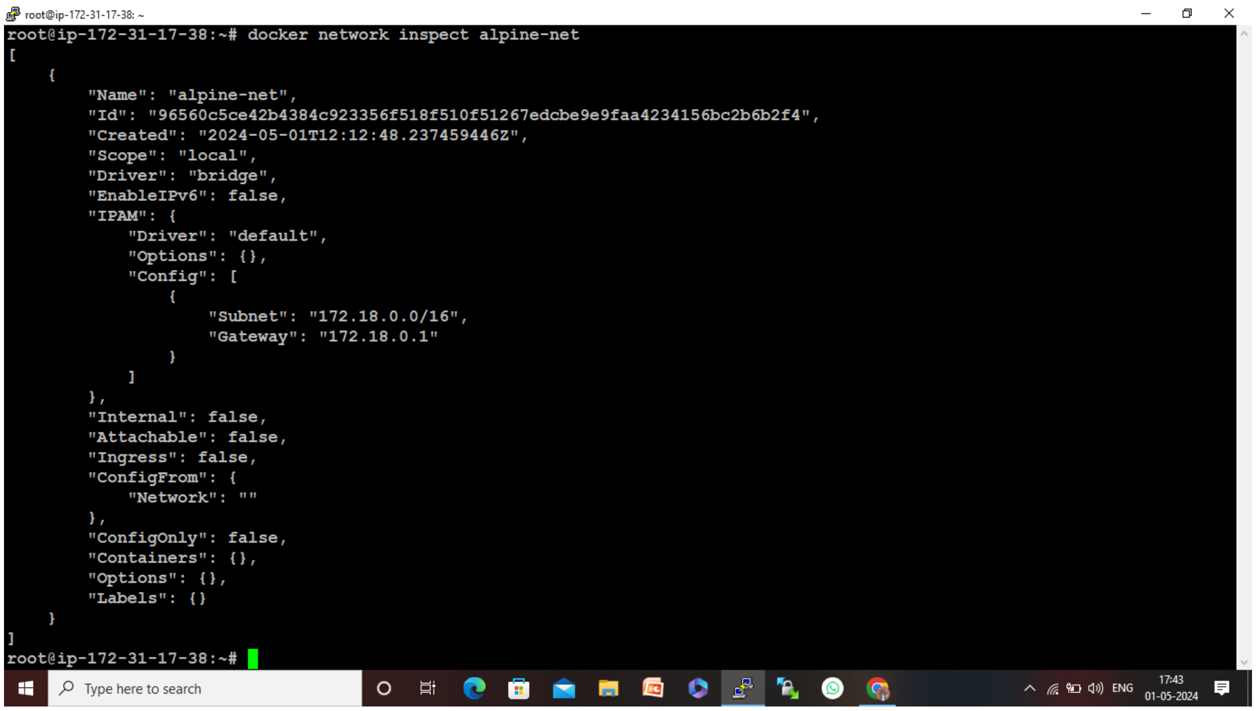
Use user-defined bridge networks:

1. Create the alpine-net network. You do not need the --driver bridge flag since it's the default, but this example shows how to specify it.



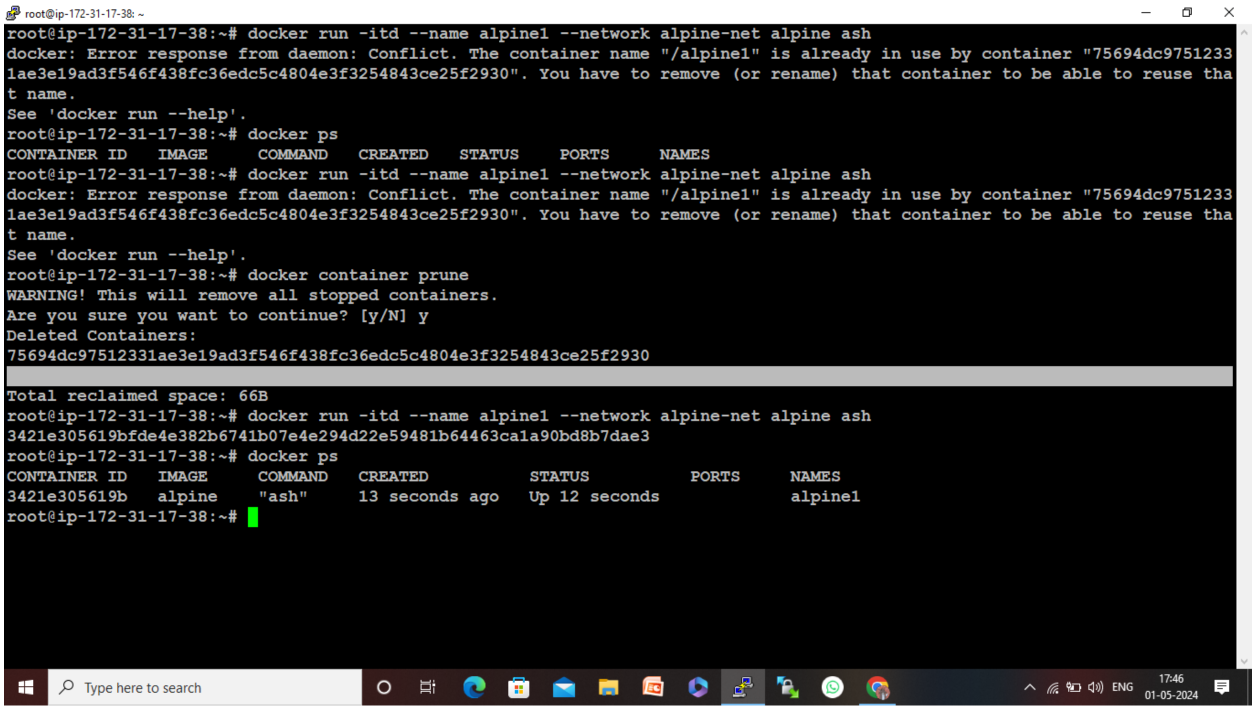
```
root@ip-172-31-17-38:~# docker network create --driver bridge alpine-net
96560c5ce42b4384c923356f518f510f51267edcbe9e9faa4234156bc2b6b2f4
root@ip-172-31-17-38:~# dcoker network ls
Command 'dcoker' not found, did you mean:
  command 'docker' from snap docker (24.0.5)
  command 'docker' from deb docker.io (24.0.5-0ubuntu1)
  command 'docker' from deb podman-docker (4.7.2+ds1-2build1)
See 'snap info <snapname>' for additional versions.
root@ip-172-31-17-38:~# docker network ls
NETWORK ID     NAME         DRIVER    SCOPE
96560c5ce42b   alpine-net   bridge    local
5ee1dca4c6f8   bridge       bridge    local
39d50b60b809   host         host      local
fee06c593e91   none         null      local
root@ip-172-31-17-38:~#
```

2. Inspect the alpine-net network. This shows you its IP address and the fact that no containers are connected to it:



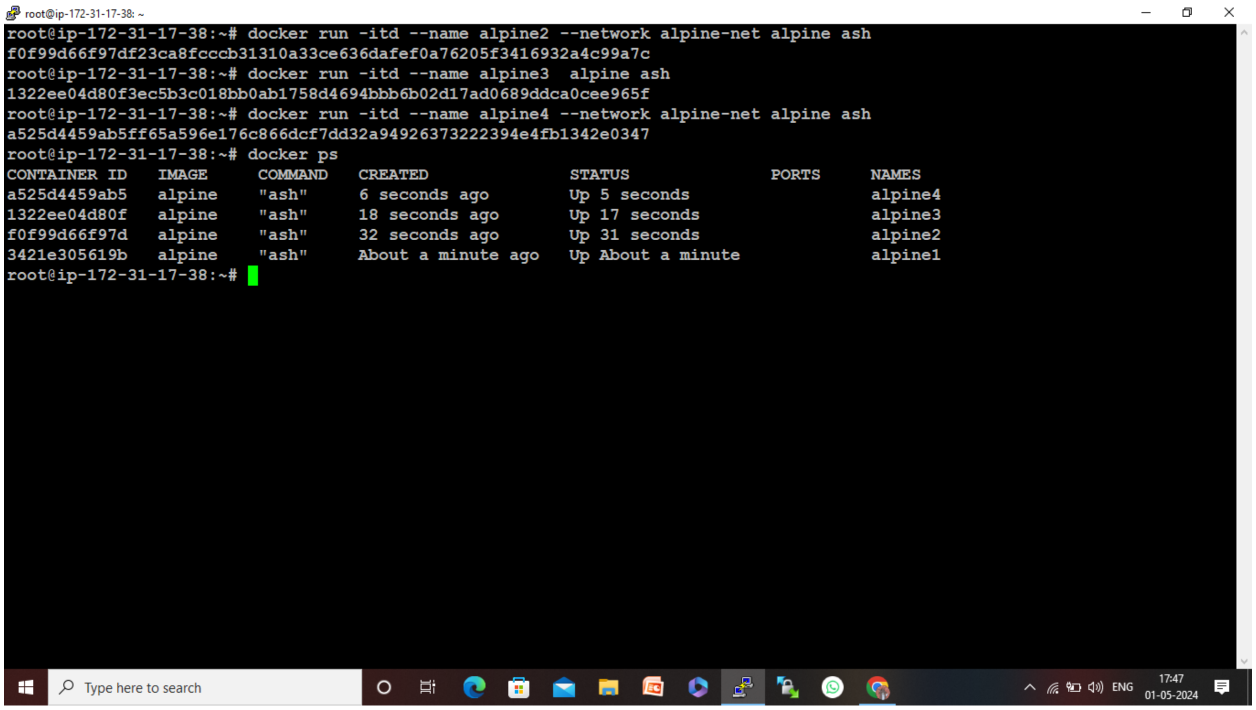
```
root@ip-172-31-17-38:~# docker network inspect alpine-net
[
    {
        "Name": "alpine-net",
        "Id": "96560c5ce42b4384c923356f518f510f51267edcbe9e9faa4234156bc2b6b2f4",
        "Created": "2024-05-01T12:12:48.237459446Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
root@ip-172-31-17-38:~#
```

3. Create your four containers. using--network flags. You can only connect to one network during the docker run command.



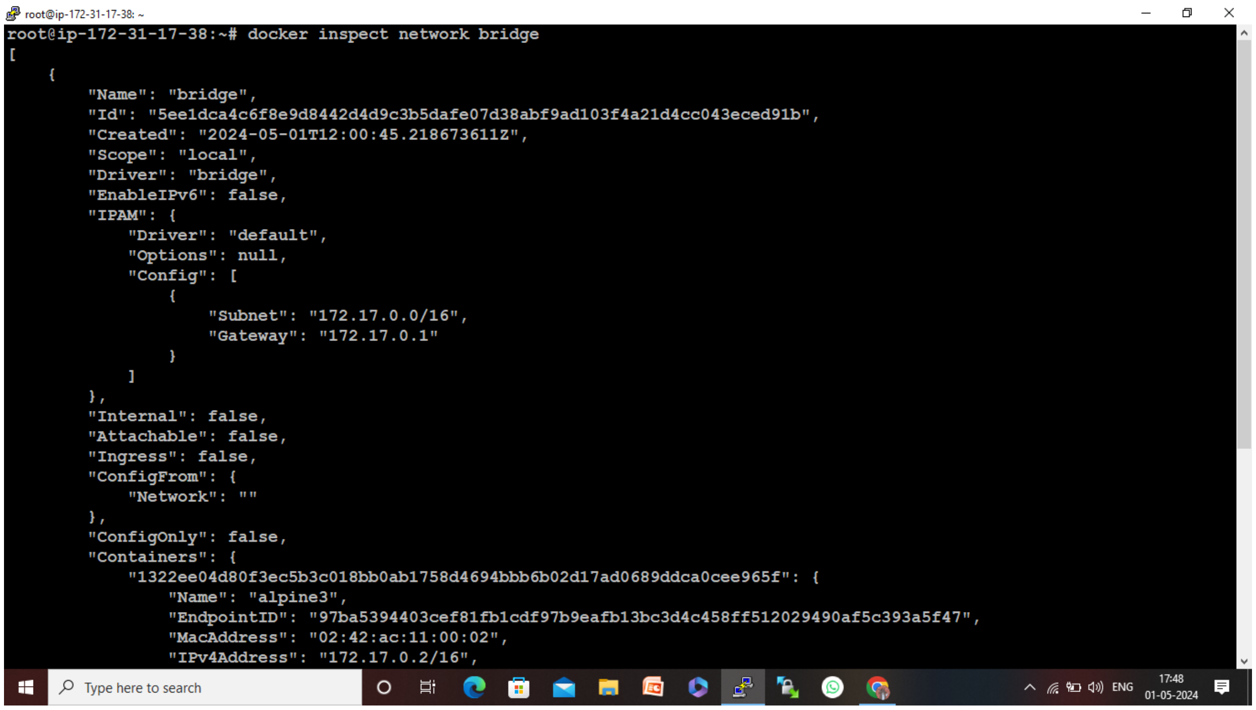
```
root@ip-172-31-17-38:~# docker run -itd --name alpine1 --network alpine-net alpine ash
docker: Error response from daemon: Conflict. The container name "/alpine1" is already in use by container "75694dc9751233
1ae3e19ad3f546f438fc36edc5c4804e3f3254843ce25f2930". You have to remove (or rename) that container to be able to reuse tha
t name.
See 'docker run --help'.
root@ip-172-31-17-38:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@ip-172-31-17-38:~# docker run -itd --name alpine1 --network alpine-net alpine ash
docker: Error response from daemon: Conflict. The container name "/alpine1" is already in use by container "75694dc9751233
1ae3e19ad3f546f438fc36edc5c4804e3f3254843ce25f2930". You have to remove (or rename) that container to be able to reuse tha
t name.
See 'docker run --help'.
root@ip-172-31-17-38:~# docker container prune
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] y
Deleted Containers:
75694dc97512331ae3e19ad3f546f438fc36edc5c4804e3f3254843ce25f2930

Total reclaimed space: 66B
root@ip-172-31-17-38:~# docker run -itd --name alpine1 --network alpine-net alpine ash
3421e305619bfde4e382b6741b07e4e294d22e59481b64463ca1a90bd8b7dae3
root@ip-172-31-17-38:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS          PORTS     NAMES
3421e305619b   alpine    "ash"     13 seconds ago   Up 12 seconds             alpine1
root@ip-172-31-17-38:~#
```

Verify that all containers are running:



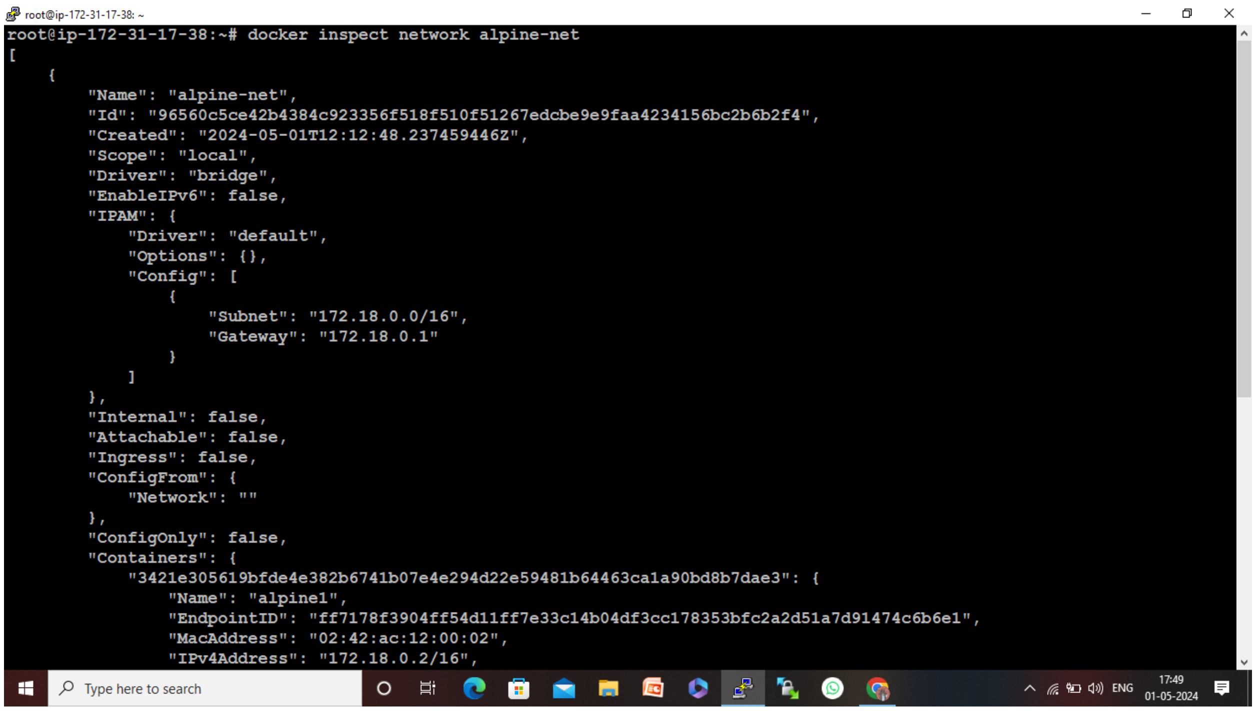
```
root@ip-172-31-17-38:~# docker run -itd --name alpine2 --network alpine-net alpine ash
f0f99d66f97df23ca8fcccb31310a33ce636dafef0a76205f3416932a4c99a7c
root@ip-172-31-17-38:~# docker run -itd --name alpine3  alpine ash
1322ee04d80f3ec5b3c018bb0ab1758d4694bbb6b02d17ad0689ddca0cee965f
root@ip-172-31-17-38:~# docker run -itd --name alpine4 --network alpine-net alpine ash
a525d4459ab5ff65a596e176c866dcf7dd32a94926373222394e4fb1342e0347
root@ip-172-31-17-38:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED              STATUS               PORTS     NAMES
a525d4459ab5   alpine    "ash"     6 seconds ago        Up 5 seconds                   alpine4
1322ee04d80f   alpine    "ash"     18 seconds ago       Up 17 seconds                  alpine3
f0f99d66f97d   alpine    "ash"     32 seconds ago       Up 31 seconds                  alpine2
3421e305619b   alpine    "ash"     About a minute ago   Up About a minute              alpine1
root@ip-172-31-17-38:~#
```

4. Inspect the bridge network and the alpine-net network again:



```
root@ip-172-31-17-38:~# docker inspect network bridge
[
    {
        "Name": "bridge",
        "Id": "5ee1dca4c6f8e9d8442d4d9c3b5dafe07d38abf9ad103f4a21d4cc043eced91b",
        "Created": "2024-05-01T12:00:45.218673611Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "1322ee04d80f3ec5b3c018bb0ab1758d4694bbb6b02d17ad0689ddca0cee965f": {
                "Name": "alpine3",
                "EndpointID": "97ba5394403cef81fb1cdf97b9eafb13bc3d4c458ff512029490af5c393a5f47",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
```

Containers alpine3 connected to the bridge network.

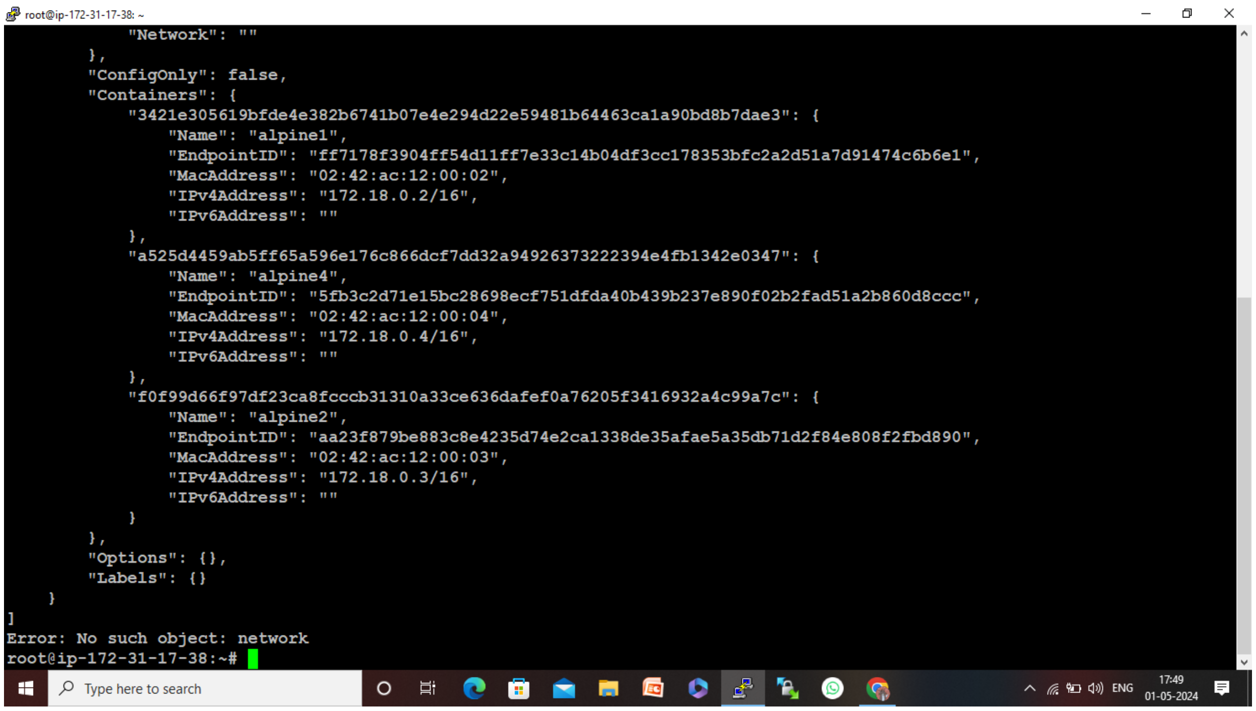

root@ip-172-31-17-38: ~

```
root@ip-172-31-17-38:~# docker inspect network alpine-net
[
    {
        "Name": "alpine-net",
        "Id": "96560c5ce42b4384c923356f518f510f51267edcbe9e9faa4234156bc2b6b2f4",
        "Created": "2024-05-01T12:12:48.237459446Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "3421e305619bfde4e382b6741b07e4e294d22e59481b64463ca1a90bd8b7dae3": {
                "Name": "alpine1",
                "EndpointID": "ff7178f3904ff54d11ff7e33c14b04df3cc178353bfc2a2d51a7d91474c6b6e1",
                "MacAddress": "02:42:ac:12:00:02",
                "IPv4Address": "172.18.0.2/16",
```

Type here to search

ENG 17:49 01-05-2024

Containers alpine1, alpine2, and alpine4 are connected to the alpine-net network.


root@ip-172-31-17-38: ~

```
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "3421e305619bfde4e382b6741b07e4e294d22e59481b64463ca1a90bd8b7dae3": {
                "Name": "alpine1",
                "EndpointID": "ff7178f3904ff54d11ff7e33c14b04df3cc178353bfc2a2d51a7d91474c6b6e1",
                "MacAddress": "02:42:ac:12:00:02",
                "IPv4Address": "172.18.0.2/16",
                "IPv6Address": ""
            },
            "a525d4459ab5ff65a596e176c866dcf7dd32a94926373222394e4fb1342e0347": {
                "Name": "alpine4",
                "EndpointID": "5fb3c2d71e15bc28698ecf751dfda40b439b237e890f02b2fad51a2b860d8ccc",
                "MacAddress": "02:42:ac:12:00:04",
                "IPv4Address": "172.18.0.4/16",
                "IPv6Address": ""
            },
            "f0f99d66f97df23ca8fcccb31310a33ce636dafef0a76205f3416932a4c99a7c": {
                "Name": "alpine2",
                "EndpointID": "aa23f879be883c8e4235d74e2ca1338de35afae5a35db71d2f84e808f2fbd890",
                "MacAddress": "02:42:ac:12:00:03",
                "IPv4Address": "172.18.0.3/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
Error: No such object: network
root@ip-172-31-17-38:~#
```

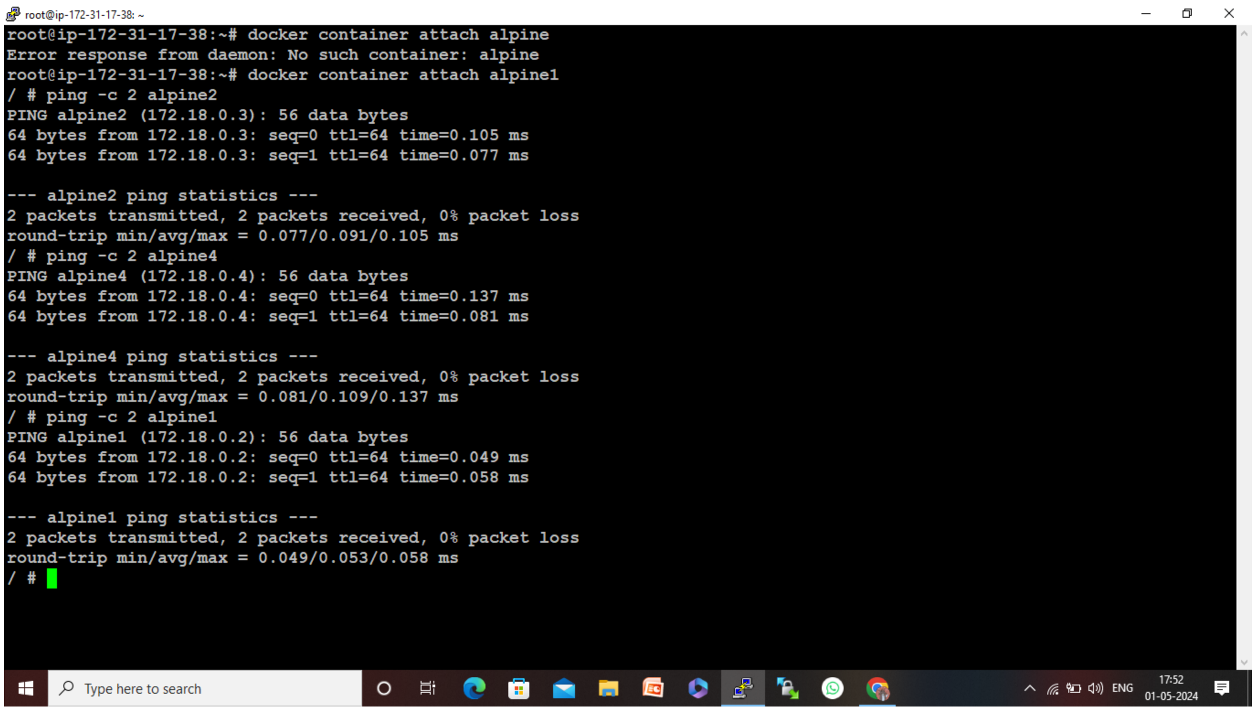
Type here to search

ENG 17:49 01-05-2024

5. On user-defined networks like alpine-net, containers can not only communicate by IP address, but can also resolve a container name to an IP address.

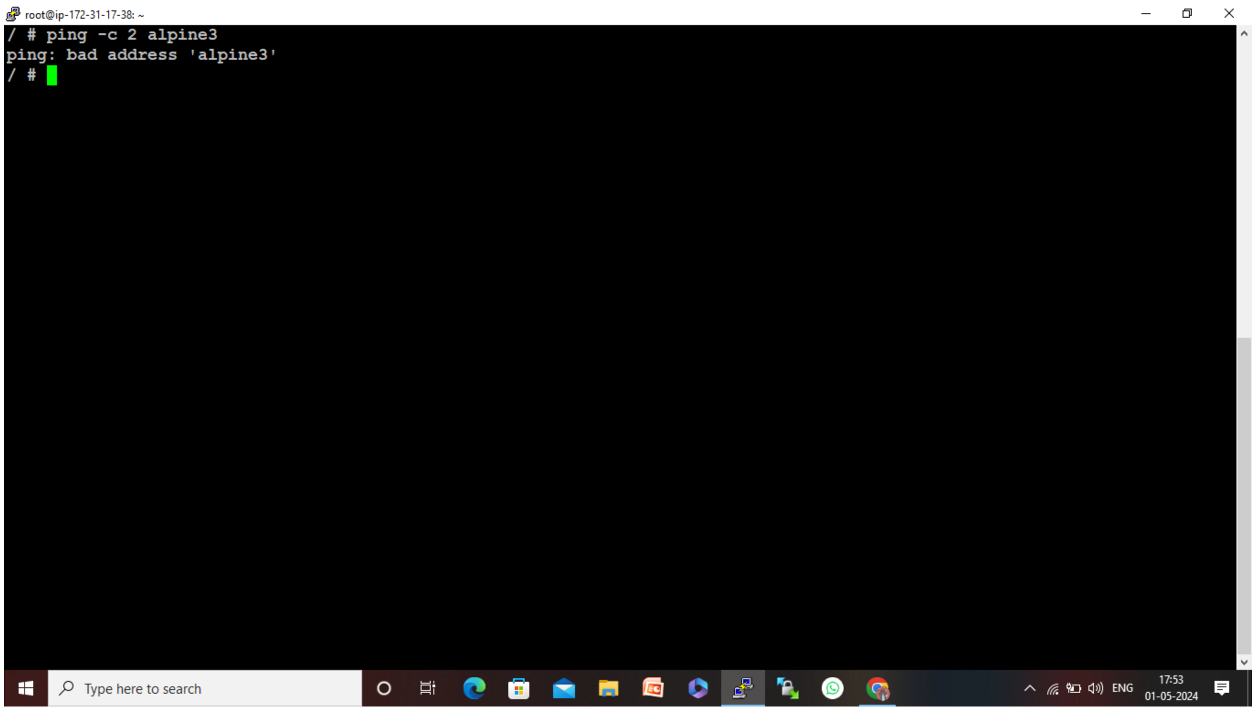
This capability is called automatic service discovery. Let's connect to alpine1 and test this out. alpine1 should be able to resolve alpine2 and alpine4 (and alpine1, itself) to IP addresses

Automatic service discovery can only resolve custom container names, not default automatically generated container names,



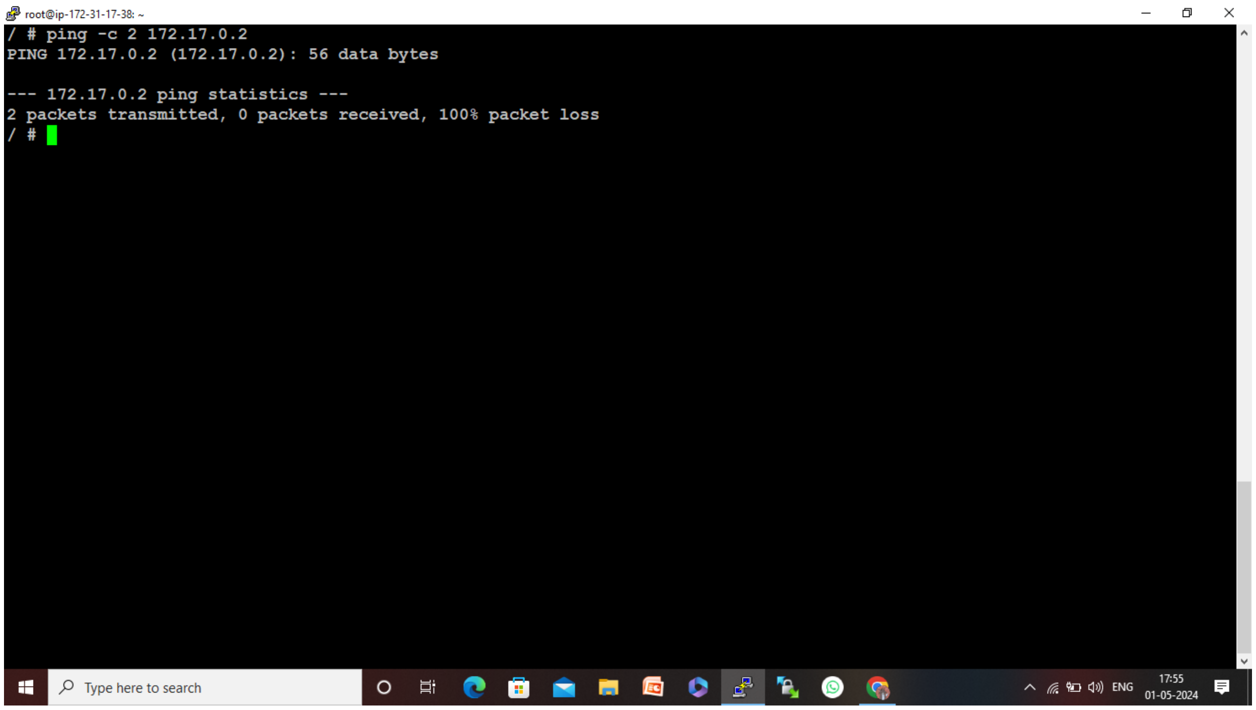
```
root@ip-172-31-17-38:~# docker container attach alpine
Error response from daemon: No such container: alpine
root@ip-172-31-17-38:~# docker container attach alpine1
/ # ping -c 2 alpine2
PING alpine2 (172.18.0.3): 56 data bytes
64 bytes from 172.18.0.3: seq=0 ttl=64 time=0.105 ms
64 bytes from 172.18.0.3: seq=1 ttl=64 time=0.077 ms

--- alpine2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.077/0.091/0.105 ms
/ # ping -c 2 alpine4
PING alpine4 (172.18.0.4): 56 data bytes
64 bytes from 172.18.0.4: seq=0 ttl=64 time=0.137 ms
64 bytes from 172.18.0.4: seq=1 ttl=64 time=0.081 ms

--- alpine4 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.081/0.109/0.137 ms
/ # ping -c 2 alpine1
PING alpine1 (172.18.0.2): 56 data bytes
64 bytes from 172.18.0.2: seq=0 ttl=64 time=0.049 ms
64 bytes from 172.18.0.2: seq=1 ttl=64 time=0.058 ms

--- alpine1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.049/0.053/0.058 ms
/ #
```

6. From alpine1, you should not be able to connect to alpine3 at all, since it is not on the alpine-net network.

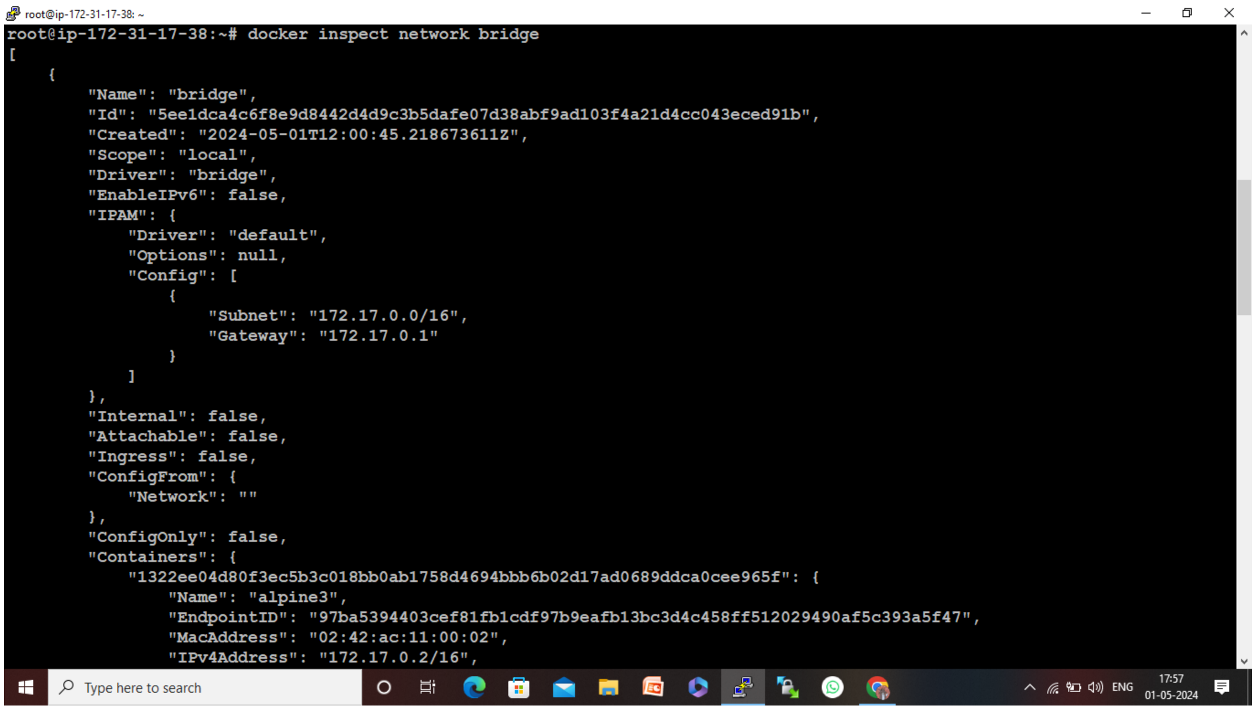


```
/ # ping -c 2 alpine3
ping: bad address 'alpine3'
/ #
```

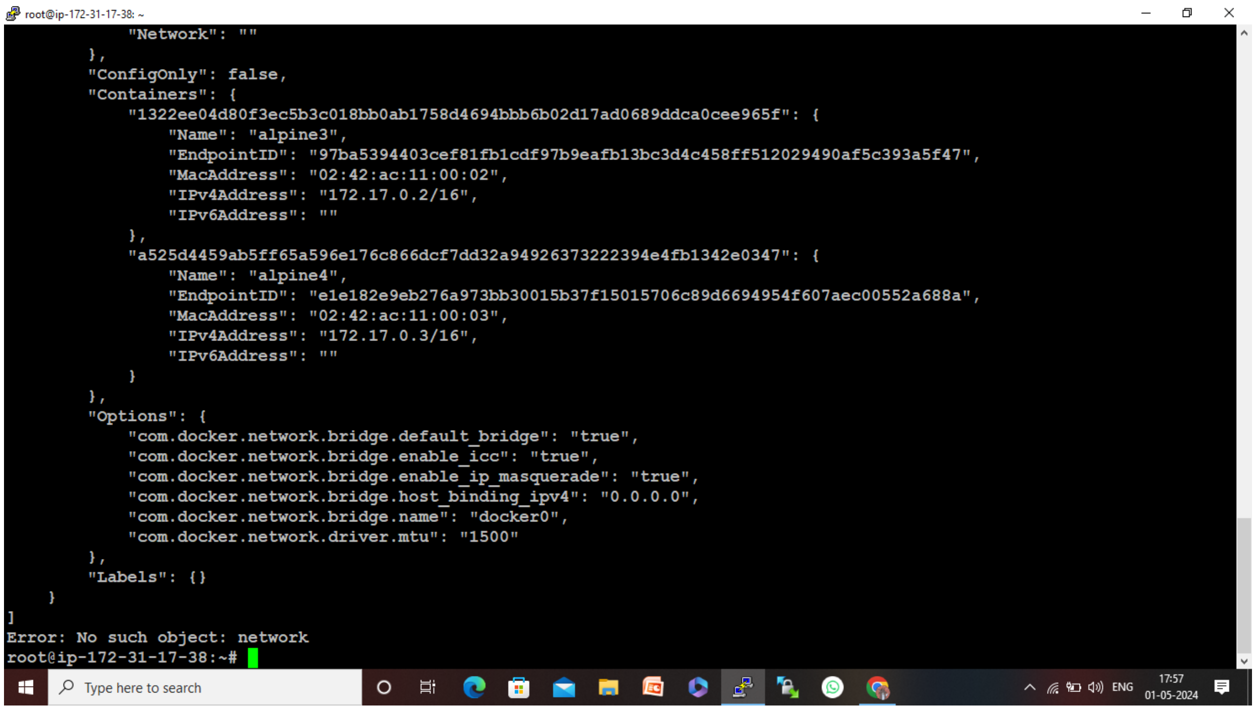
Not only that, but you can't connect to alpine3 from alpine1 by its IP address either. Look back at the docker network inspect output for the bridge network and find alpine3's IP address: 172.17.0.2 Try to ping it.



```
/ # ping -c 2 172.17.0.2
PING 172.17.0.2 (172.17.0.2): 56 data bytes

--- 172.17.0.2 ping statistics ---
2 packets transmitted, 0 packets received, 100% packet loss
/ #
```

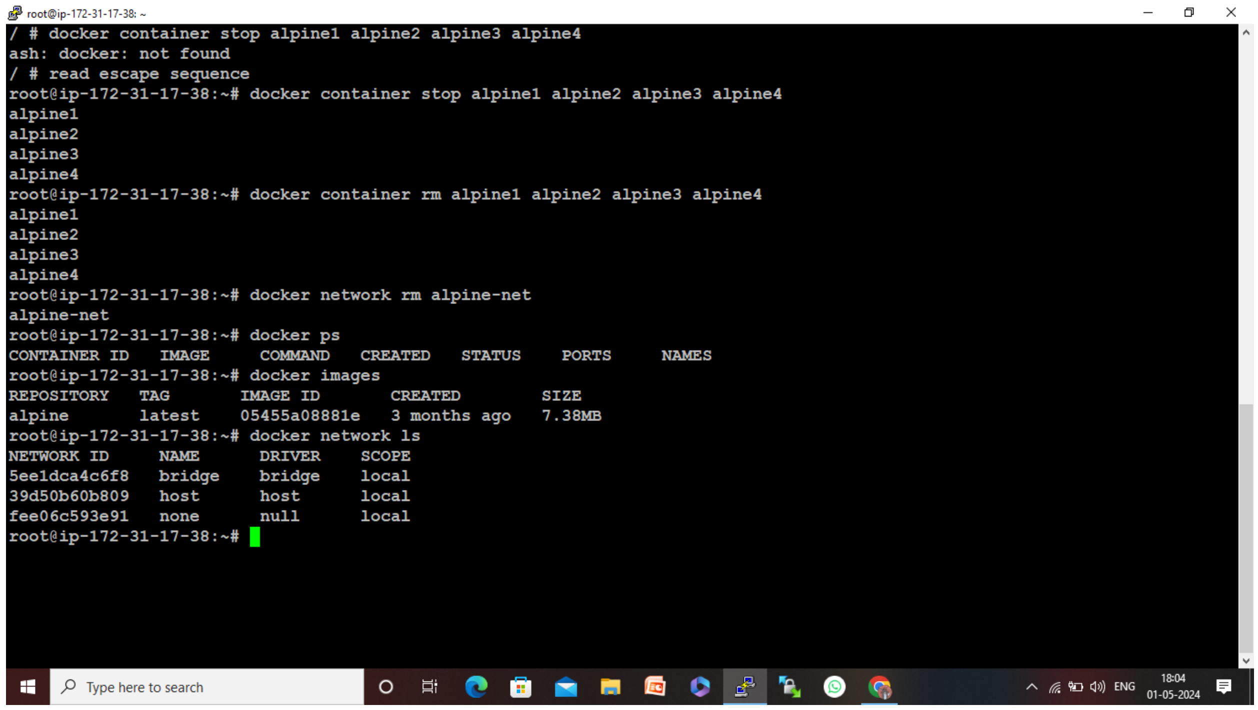

root@ip-172-31-17-38: ~

```
root@ip-172-31-17-38:~# docker inspect network bridge
[
    {
        "Name": "bridge",
        "Id": "5ee1dca4c6f8e9d8442d4d9c3b5dafe07d38abf9ad103f4a21d4cc043eced91b",
        "Created": "2024-05-01T12:00:45.218673611Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "1322ee04d80f3ec5b3c018bb0ab1758d4694bbb6b02d17ad0689ddca0cee965f": {
                "Name": "alpine3",
                "EndpointID": "97ba5394403cef81fb1cdf97b9eafb13bc3d4c458ff512029490af5c393a5f47",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
```


root@ip-172-31-17-38: ~

```
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "1322ee04d80f3ec5b3c018bb0ab1758d4694bbb6b02d17ad0689ddca0cee965f": {
                "Name": "alpine3",
                "EndpointID": "97ba5394403cef81fb1cdf97b9eafb13bc3d4c458ff512029490af5c393a5f47",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
            },
            "a525d4459ab5ff65a596e176c866dcf7dd32a94926373222394e4fb1342e0347": {
                "Name": "alpine4",
                "EndpointID": "e1e182e9eb276a973bb30015b37f15015706c89d6694954f607aec00552a688a",
                "MacAddress": "02:42:ac:11:00:03",
                "IPv4Address": "172.17.0.3/16",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
    }
]
Error: No such object: network
root@ip-172-31-17-38:~#
```

7. Stop and remove all containers and the alpine-net network.



```
/ # docker container stop alpine1 alpine2 alpine3 alpine4
ash: docker: not found
/ # read escape sequence
root@ip-172-31-17-38:~# docker container stop alpine1 alpine2 alpine3 alpine4
alpine1
alpine2
alpine3
alpine4
root@ip-172-31-17-38:~# docker container rm alpine1 alpine2 alpine3 alpine4
alpine1
alpine2
alpine3
alpine4
root@ip-172-31-17-38:~# docker network rm alpine-net
alpine-net
root@ip-172-31-17-38:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@ip-172-31-17-38:~# docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
alpine       latest    05455a08881e   3 months ago    7.38MB
root@ip-172-31-17-38:~# docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
5ee1dca4c6f8   bridge    bridge    local
39d50b60b809   host      host      local
fee06c593e91   none      null      local
root@ip-172-31-17-38:~#
```

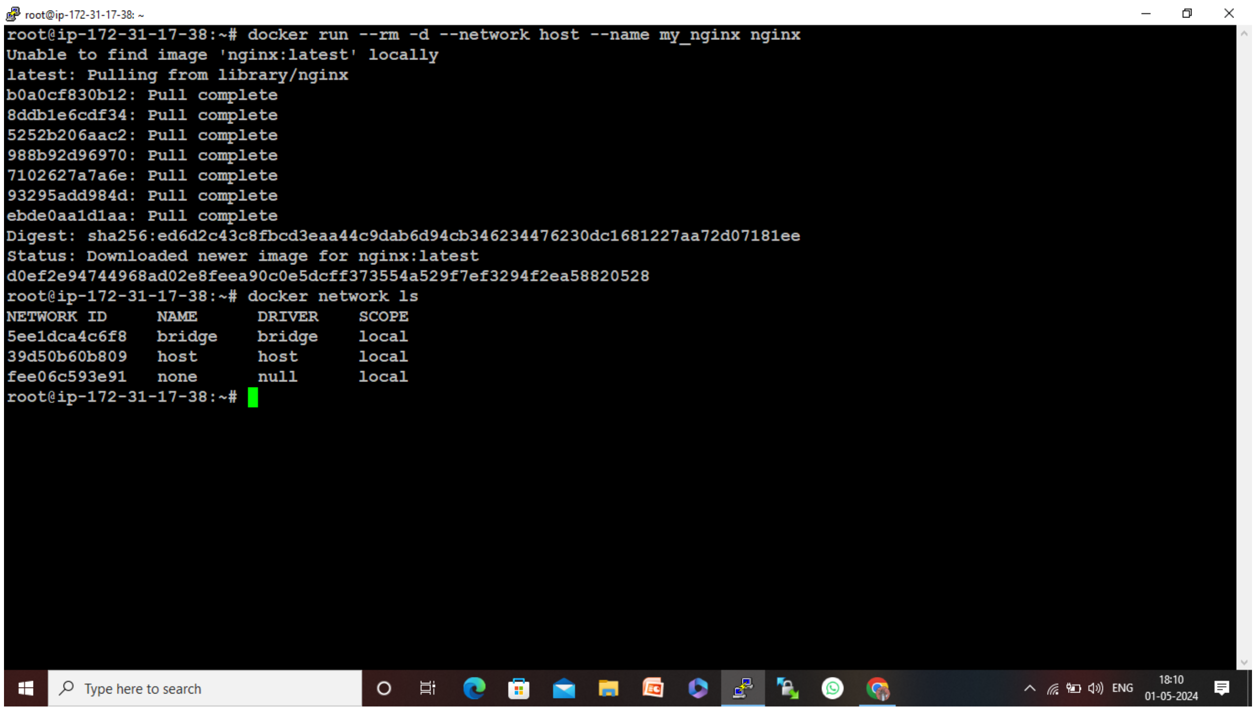
2.NETWORKING USING THE HOST NETWORK:

Host networks allow containers to share the host's network namespace, effectively bypassing Docker's network isolation.

1. Create and start the container as a detached process.

The --rm option means to remove the container once it exits/stops.

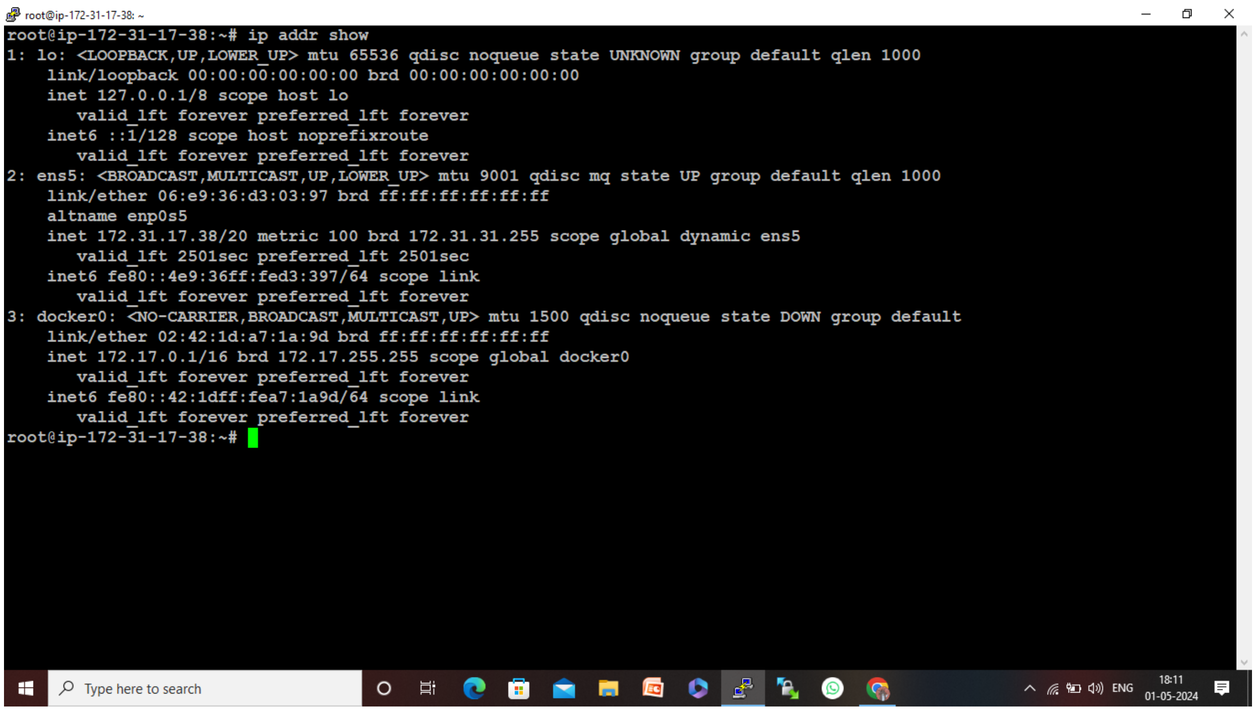
The -d flag means to start the container detached (in the background).



root@ip-172-31-17-38: ~

```
root@ip-172-31-17-38:~# docker run --rm -d --network host --name my_nginx nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
b0a0cf830b12: Pull complete
8ddb1e6cdf34: Pull complete
5252b206aac2: Pull complete
988b92d96970: Pull complete
7102627a7a6e: Pull complete
93295add984d: Pull complete
ebde0aa1d1aa: Pull complete
Digest: sha256:ed6d2c43c8fbcd3eaa44c9dab6d94cb346234476230dc1681227aa72d07181ee
Status: Downloaded newer image for nginx:latest
d0ef2e94744968ad02e8feea90c0e5dcff373554a529f7ef3294f2ea58820528
root@ip-172-31-17-38:~# docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
5ee1dca4c6f8   bridge    bridge    local
39d50b60b809   host      host      local
fee06c593e91   none      null      local
root@ip-172-31-17-38:~#
```

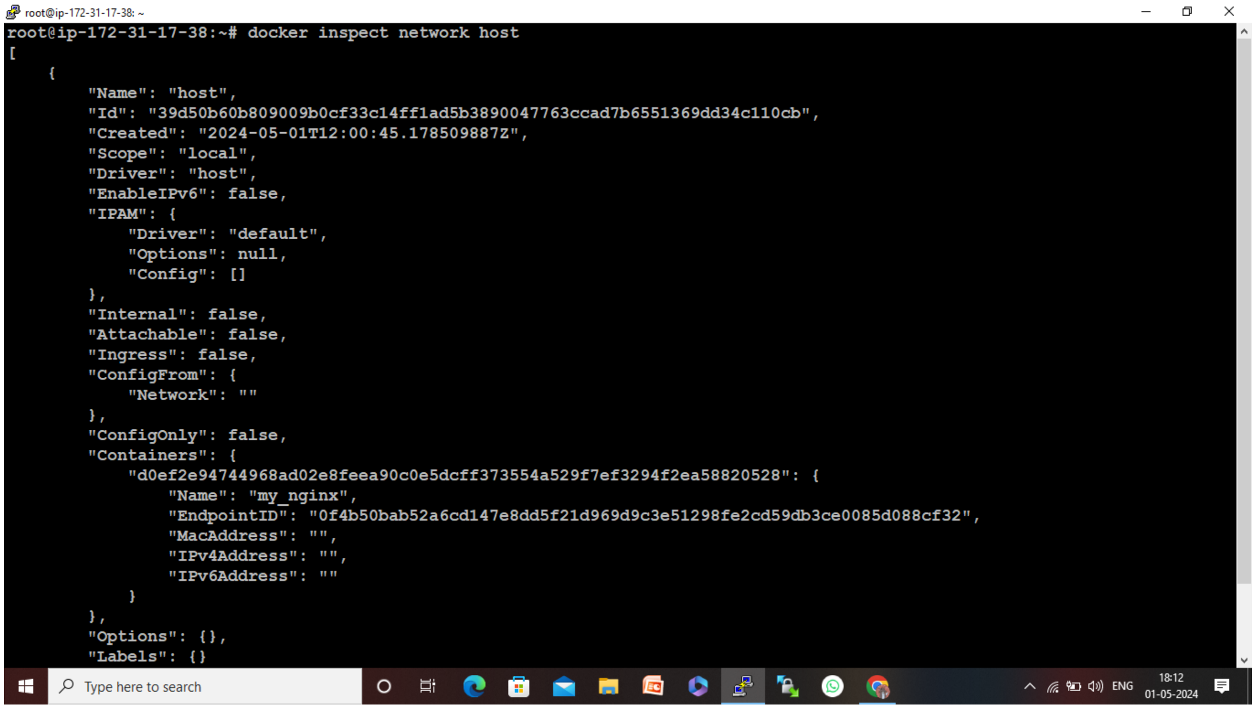
2. Examine all network interfaces and verify that a new one was not created.



root@ip-172-31-17-38: ~

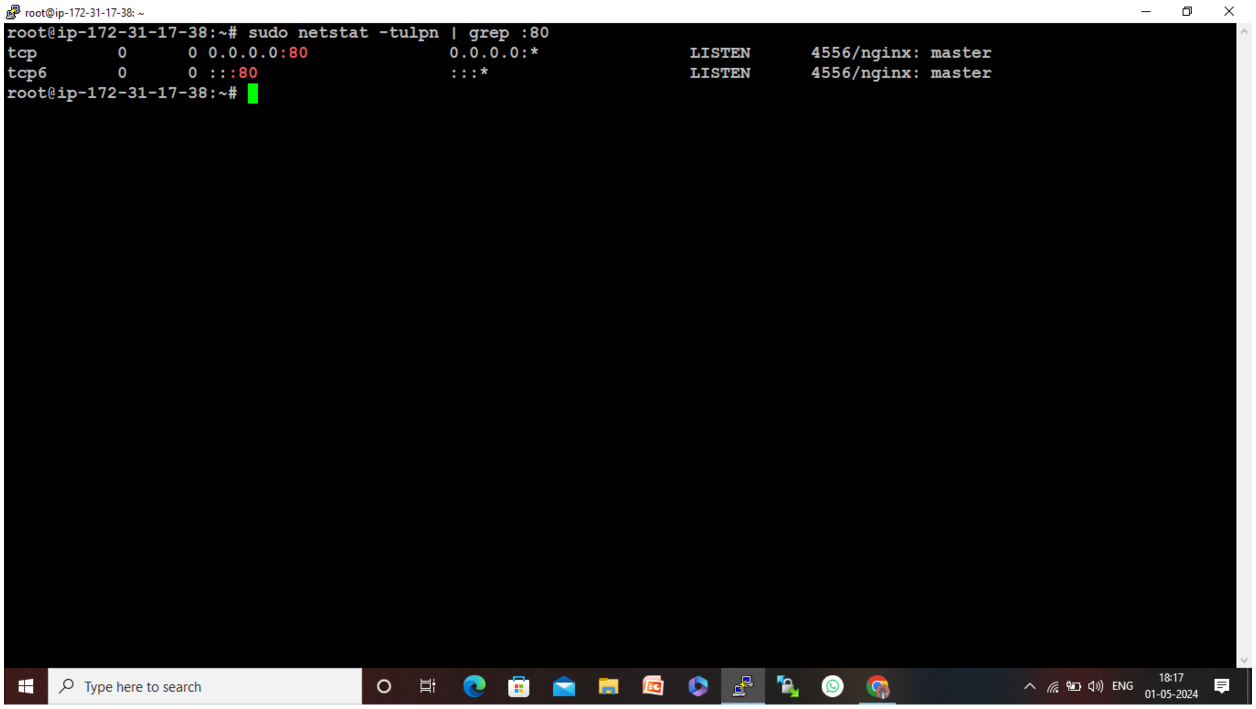
```
root@ip-172-31-17-38:~# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: ens5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc mq state UP group default qlen 1000
    link/ether 06:e9:36:d3:03:97 brd ff:ff:ff:ff:ff:ff
    altname enp0s5
    inet 172.31.17.38/20 metric 100 brd 172.31.31.255 scope global dynamic ens5
       valid_lft 2501sec preferred_lft 2501sec
    inet6 fe80::4e9:36ff:fed3:397/64 scope link
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:1d:a7:1a:9d 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
    inet6 fe80::42:1dff:fea7:1a9d/64 scope link
       valid_lft forever preferred_lft forever
root@ip-172-31-17-38:~#
```

3. Inspecting the host network.



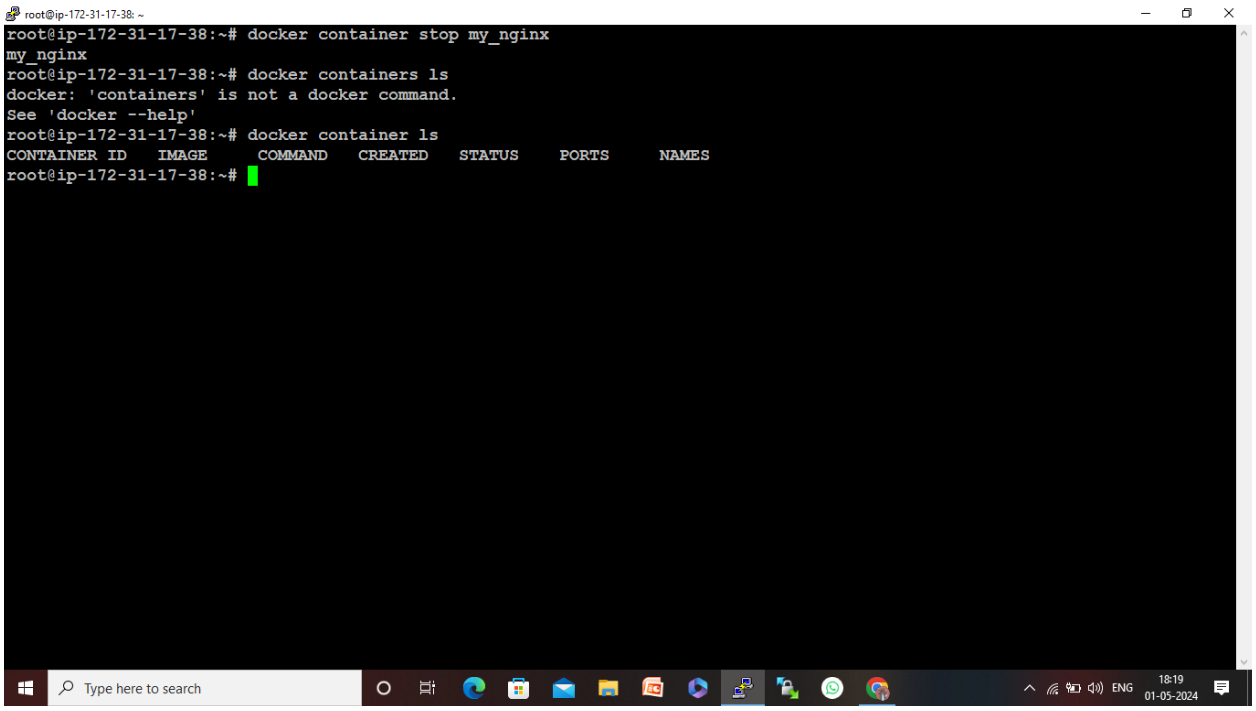
```
root@ip-172-31-17-38:~# docker inspect network host
[
    {
        "Name": "host",
        "Id": "39d50b60b809009b0cf33c14ff1ad5b3890047763ccad7b6551369dd34c110cb",
        "Created": "2024-05-01T12:00:45.178509887Z",
        "Scope": "local",
        "Driver": "host",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": []
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "d0ef2e94744968ad02e8feea90c0e5dcff373554a529f7ef3294f2ea58820528": {
                "Name": "my_nginx",
                "EndpointID": "0f4b50bab52a6cd147e8dd5f21d969d9c3e51298fe2cd59db3ce0085d088cf32",
                "MacAddress": "",
                "IPv4Address": "",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
```

4.Verify which process is bound to port 80, using the netstat command. You need to use sudo because the process is owned by the Docker daemon user and you otherwise won't be able to see its name or PID.



```
root@ip-172-31-17-38:~# sudo netstat -tulpn | grep :80
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      4556/nginx: master
tcp6       0      0 :::80                   :::*                    LISTEN      4556/nginx: master
root@ip-172-31-17-38:~#
```

4.Stop the container. It will be removed automatically as it was started using the --rm option.



```
root@ip-172-31-17-38:~# docker container stop my_nginx
my_nginx
root@ip-172-31-17-38:~# docker containers ls
docker: 'containers' is not a docker command.
See 'docker --help'
root@ip-172-31-17-38:~# docker container ls
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@ip-172-31-17-38:~#
```

3. NETWORKING WITH OVERLAY NETWORKS:

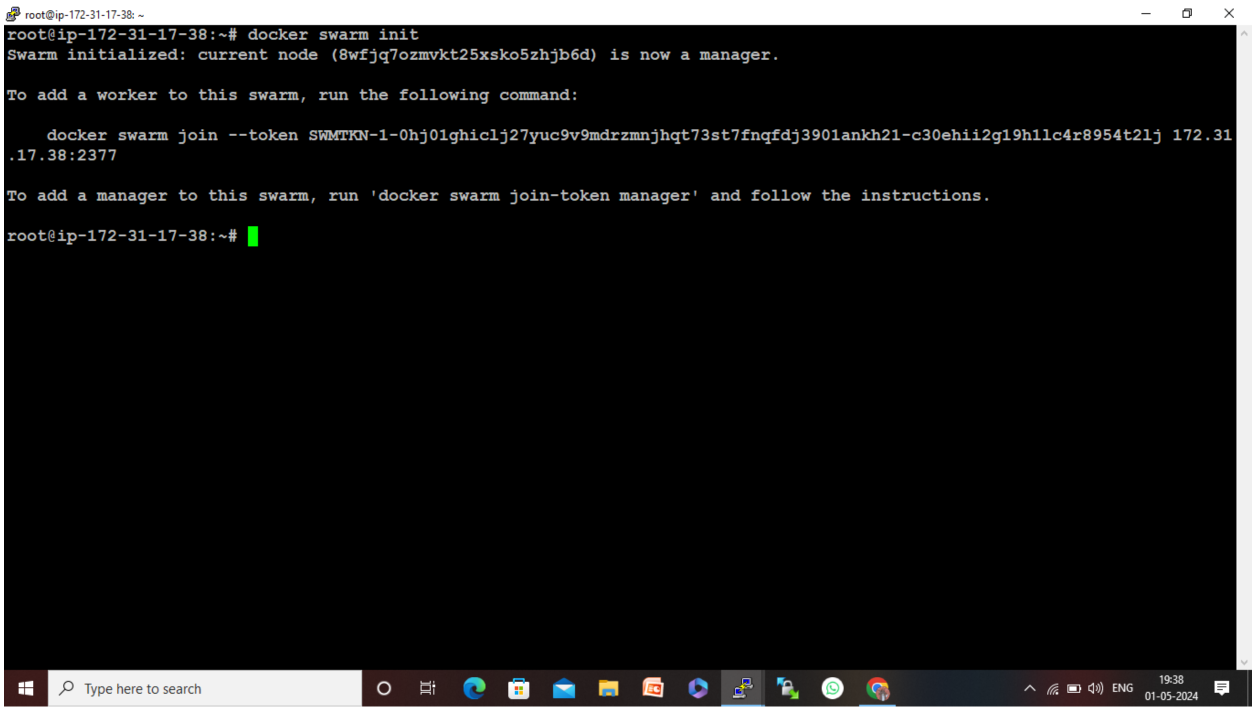
Overlay networks facilitate communication between containers across multiple Docker hosts or Swarm nodes.

1. On manager. initialize the swarm.

If the host only has one network interface,On manager.

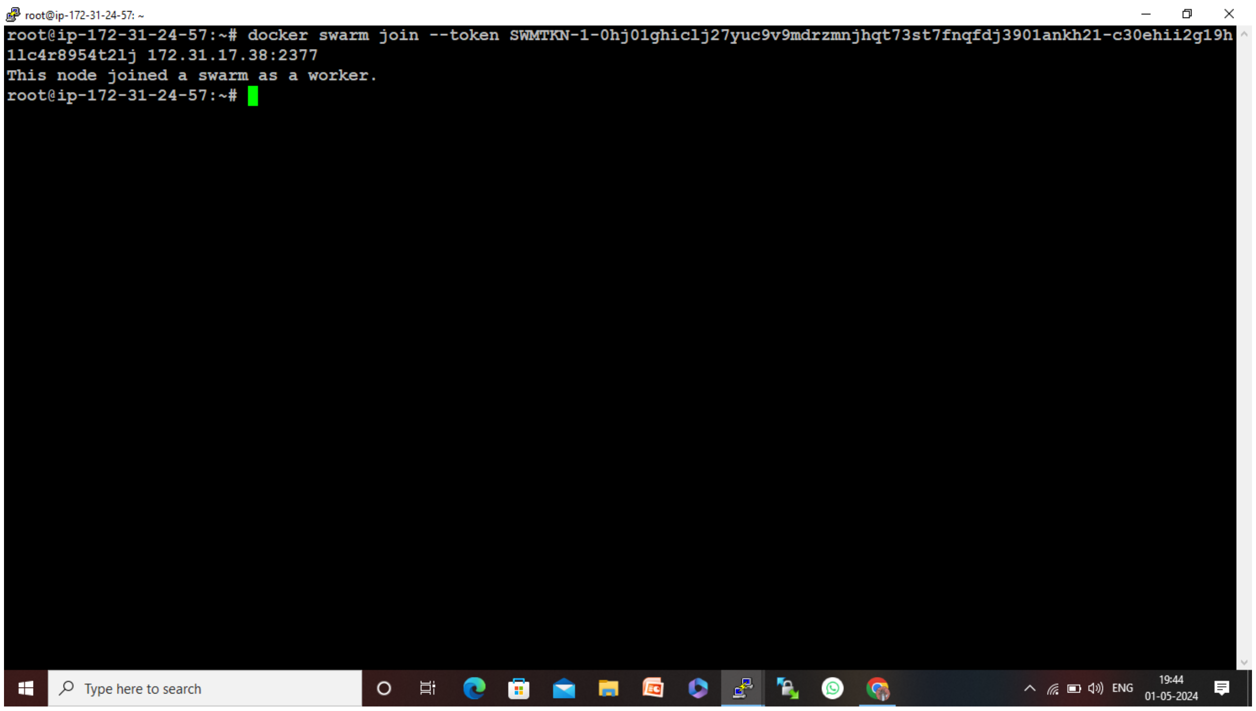
Make a note of the text that is printed, as this contains the token that you will use to join worker to the swarm.

It is a good idea to store the token in a password manager.


root@ip-172-31-17-38: ~

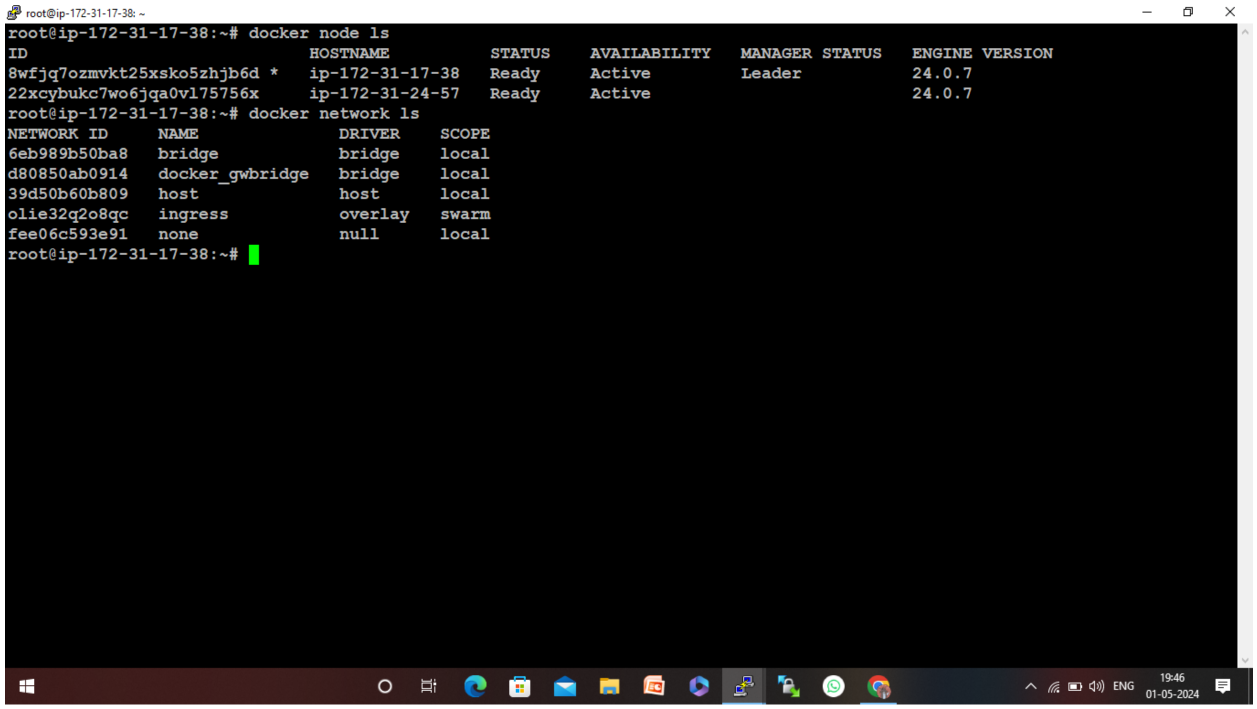
```
root@ip-172-31-17-38:~# docker swarm init
Swarm initialized: current node (8wfjq7ozmvkt25xsko5zhjb6d) is now a manager.

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

    docker swarm join --token SWMTKN-1-0hj01ghiclj27yuc9v9mdrzmnjhqt73st7fnqfdj3901ankh21-c30ehii2g19h1lc4r8954t21j 172.31
.17.38:2377

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

root@ip-172-31-17-38:~#
```

2. On worker, join the swarm. If the host only has one network interface.


root@ip-172-31-24-57: ~

```
root@ip-172-31-24-57:~# docker swarm join --token SWMTKN-1-0hj01ghiclj27yuc9v9mdrzmnjhqt73st7fnqfdj3901ankh21-c30ehii2g19h
1lc4r8954t21j 172.31.17.38:2377
This node joined a swarm as a worker.
root@ip-172-31-24-57:~#
```

3.On manager, list all the nodes. This command can only be done from a manager.



```
root@ip-172-31-17-38:~# docker node ls
ID                            HOSTNAME           STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
8wfjq7ozmvkt25xsko5zhjb6d *   ip-172-31-17-38    Ready     Active         Leader           24.0.7
22xcybukc7wo6jqa0vl75756x     ip-172-31-24-57    Ready     Active                          24.0.7
root@ip-172-31-17-38:~# docker network ls
NETWORK ID     NAME              DRIVER    SCOPE
6eb989b50ba8   bridge            bridge    local
d80850ab0914   docker_gwbridge   bridge    local
39d50b60b809   host              host      local
olie32q2o8qc   ingress           overlay   swarm
fee06c593e91   none              null      local
root@ip-172-31-17-38:~#
```

4. List the Docker networks on manager, worker

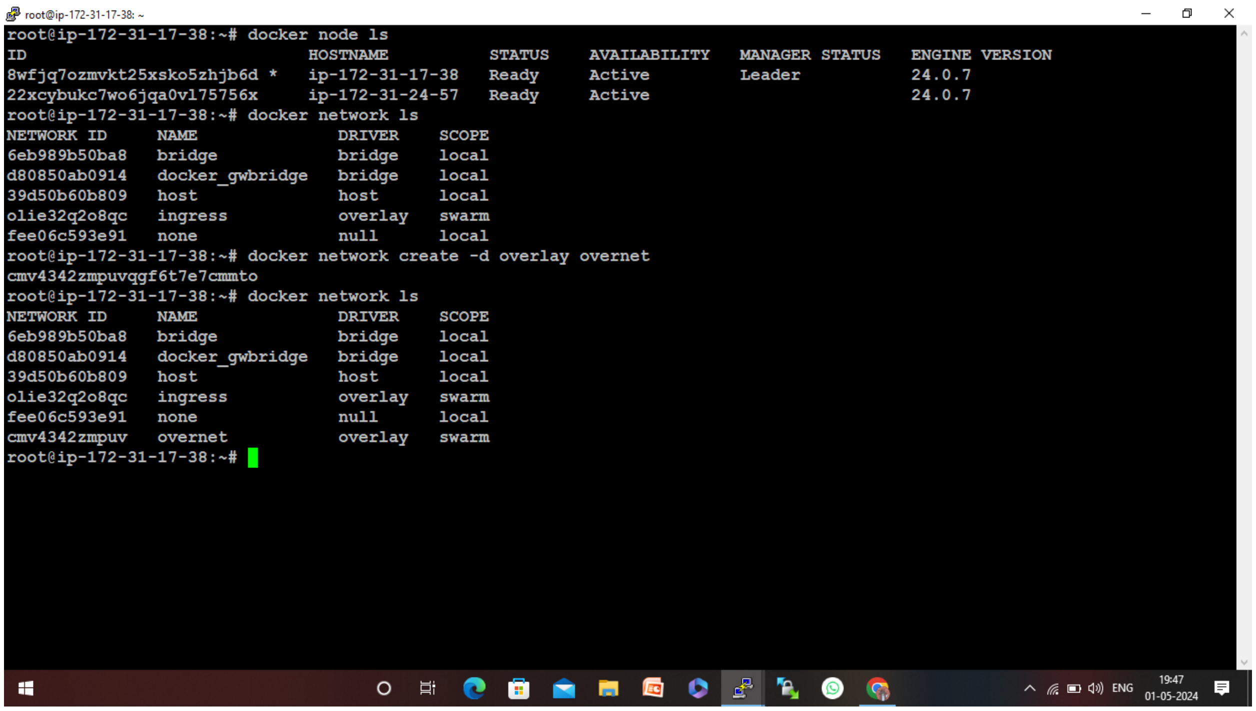
Notice that each of them now has an overlay network called ingress and a bridge network called docker_gwbridge.

Only the listing for manager is shown here.

CREATING THE SERVICES

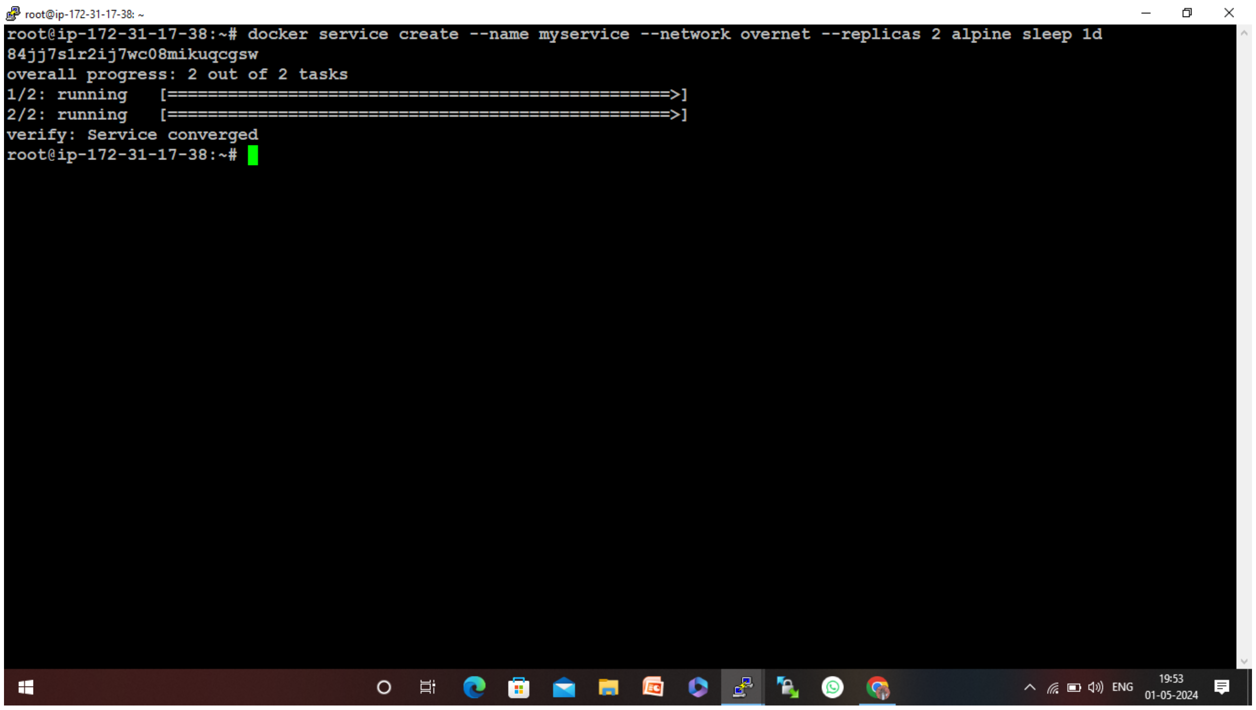
Services can only be created on a manager.

1. On manager, create a new overlay network called overnet



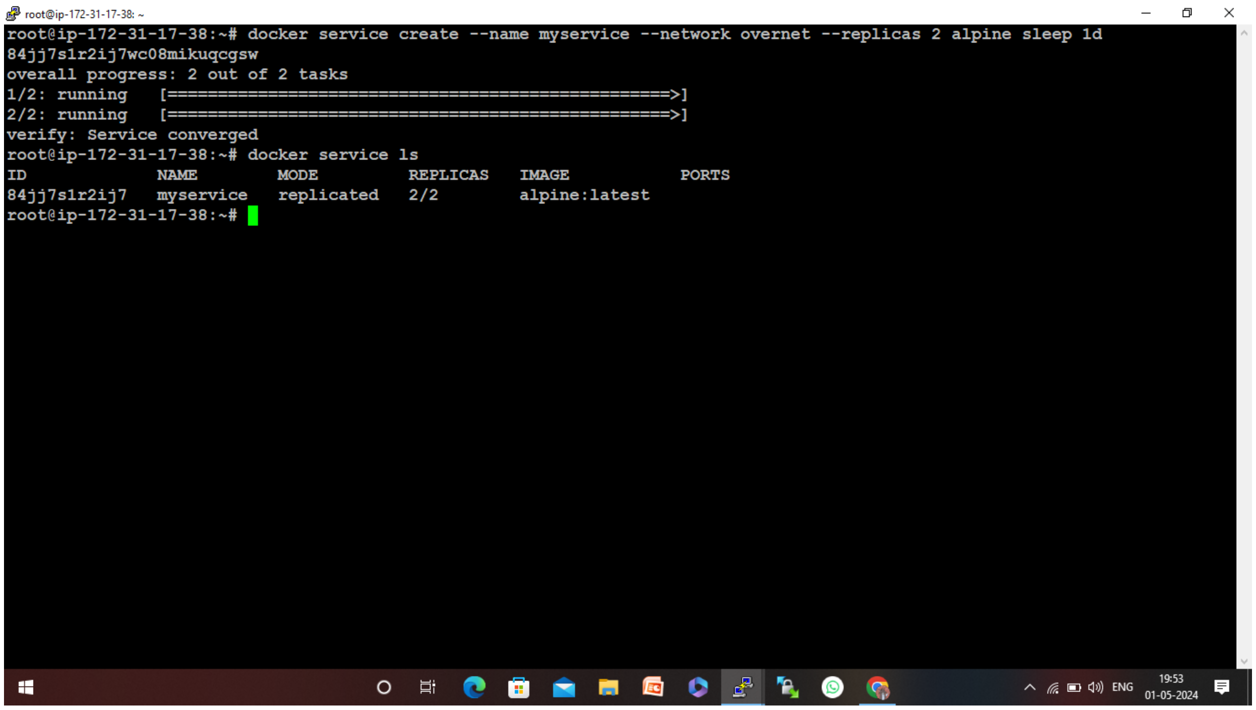
```
root@ip-172-31-17-38:~# docker node ls
ID                            HOSTNAME           STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
8wfjq7ozmvkt25xsko5zhjb6d *   ip-172-31-17-38    Ready     Active         Leader           24.0.7
22xcybukc7wo6jqa0vl75756x     ip-172-31-24-57    Ready     Active                          24.0.7
root@ip-172-31-17-38:~# docker network ls
NETWORK ID     NAME              DRIVER    SCOPE
6eb989b50ba8   bridge            bridge    local
d80850ab0914   docker_gwbridge   bridge    local
39d50b60b809   host              host      local
olie32q2o8qc   ingress           overlay   swarm
fee06c593e91   none              null      local
root@ip-172-31-17-38:~# docker network create -d overlay overnet
cmv4342zmpuvqgf6t7e7cmmto
root@ip-172-31-17-38:~# docker network ls
NETWORK ID     NAME              DRIVER    SCOPE
6eb989b50ba8   bridge            bridge    local
d80850ab0914   docker_gwbridge   bridge    local
39d50b60b809   host              host      local
olie32q2o8qc   ingress           overlay   swarm
fee06c593e91   none              null      local
cmv4342zmpuv   overnet           overlay   swarm
root@ip-172-31-17-38:~#
```

2. On manager, create a 2-replica alpine service connected to overnet.



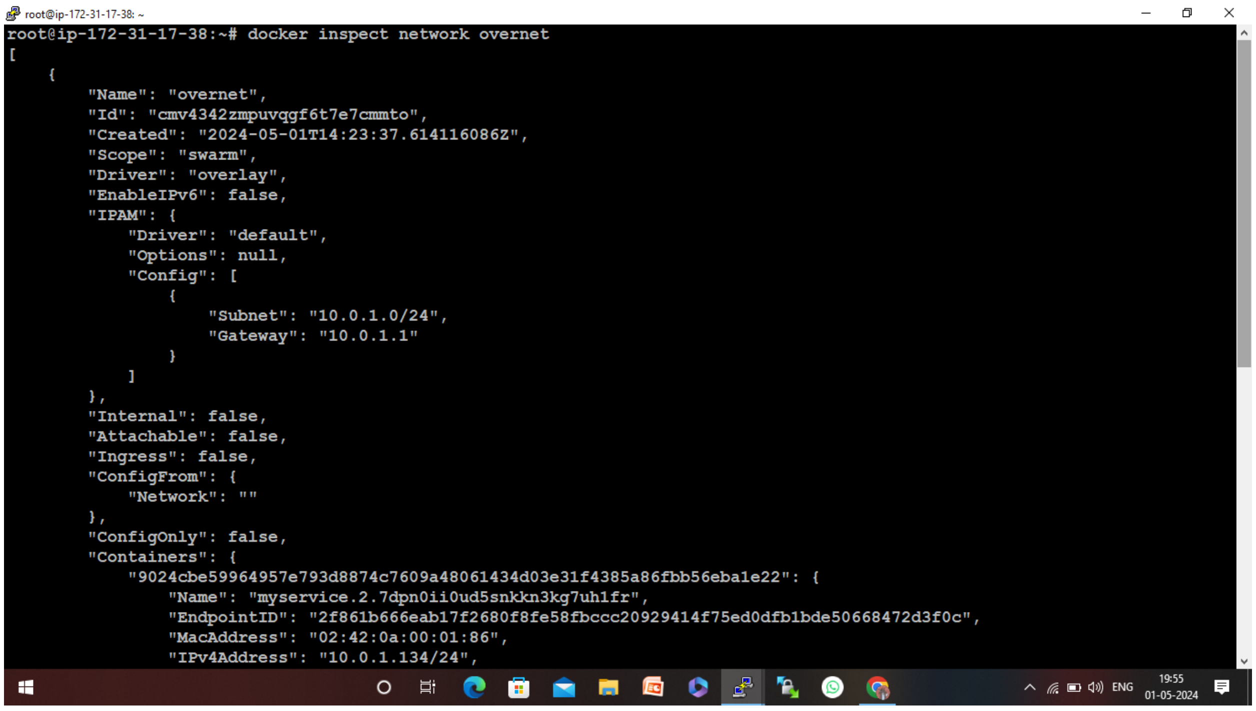
```
root@ip-172-31-17-38:~# docker service create --name myservice --network overnet --replicas 2 alpine sleep 1d
84jj7s1r2ij7wc08mikuqcgsw
overall progress: 2 out of 2 tasks
1/2: running   [==================================================>]
2/2: running   [==================================================>]
verify: Service converged
root@ip-172-31-17-38:~#
```

3.Docker service ls to list the service created.

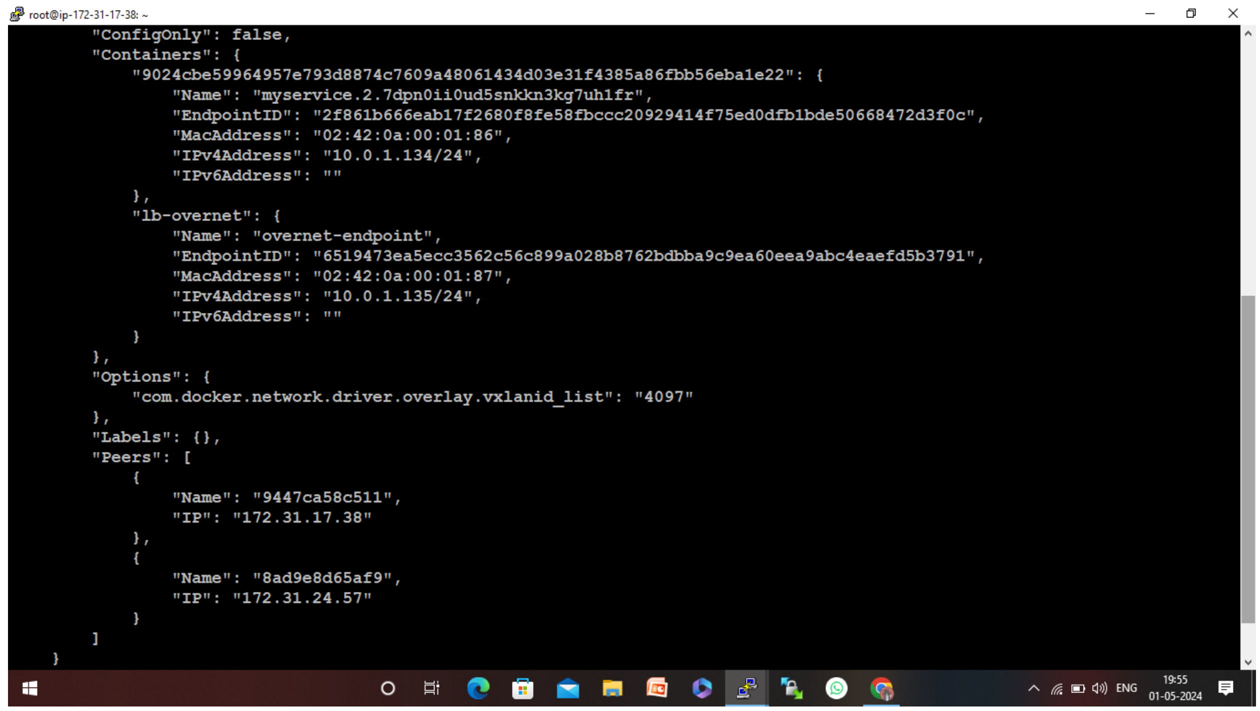


```
root@ip-172-31-17-38:~# docker service create --name myservice --network overnet --replicas 2 alpine sleep 1d
84jj7s1r2ij7wc08mikuqcgsw
overall progress: 2 out of 2 tasks
1/2: running   [==================================================>]
2/2: running   [==================================================>]
verify: Service converged
root@ip-172-31-17-38:~# docker service ls
ID             NAME        MODE         REPLICAS   IMAGE           PORTS
84jj7s1r2ij7   myservice   replicated   2/2        alpine:latest
root@ip-172-31-17-38:~#
```

4. Inspecting the manager network to know this ip to check in the worker container.

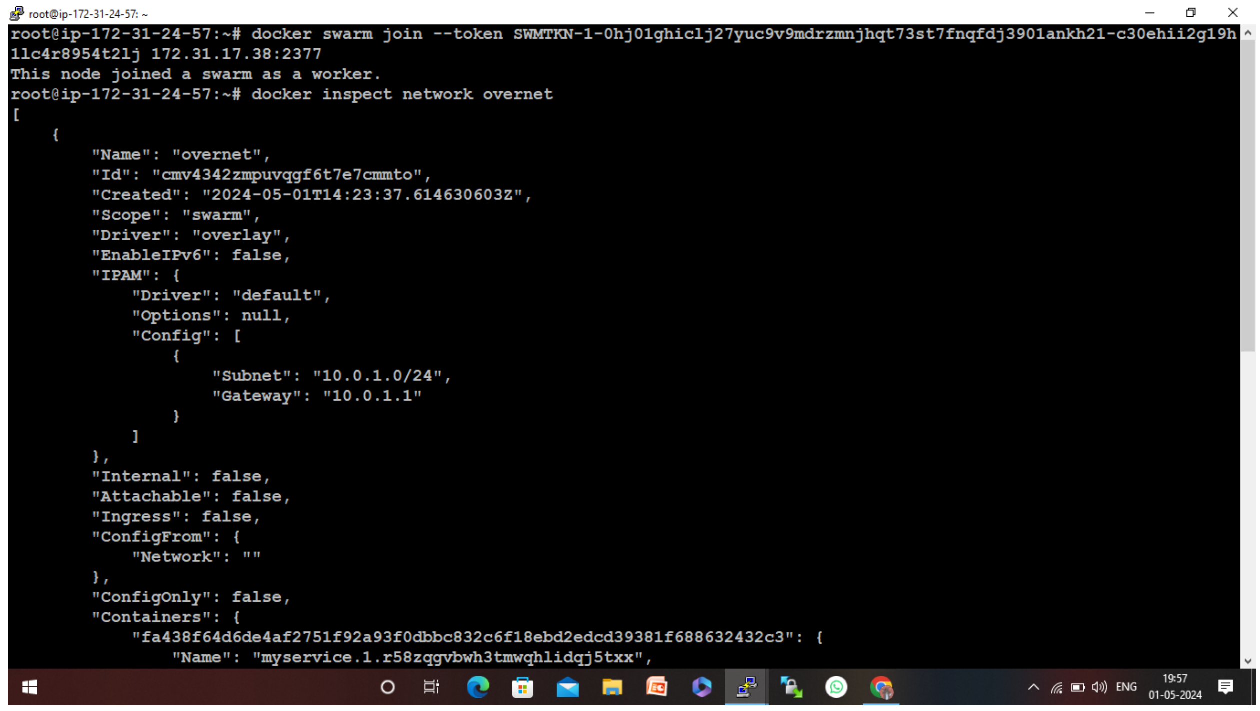


```
root@ip-172-31-17-38:~# docker inspect network overnet
[
    {
        "Name": "overnet",
        "Id": "cmv4342zmpuvqgf6t7e7cmmto",
        "Created": "2024-05-01T14:23:37.614116086Z",
        "Scope": "swarm",
        "Driver": "overlay",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "10.0.1.0/24",
                    "Gateway": "10.0.1.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "9024cbe59964957e793d8874c7609a48061434d03e31f4385a86fbb56eba1e22": {
                "Name": "myservice.2.7dpn0ii0ud5snkkn3kg7uh1fr",
                "EndpointID": "2f861b666eab17f2680f8fe58fbccc20929414f75ed0dfb1bde50668472d3f0c",
                "MacAddress": "02:42:0a:00:01:86",
                "IPv4Address": "10.0.1.134/24",
```

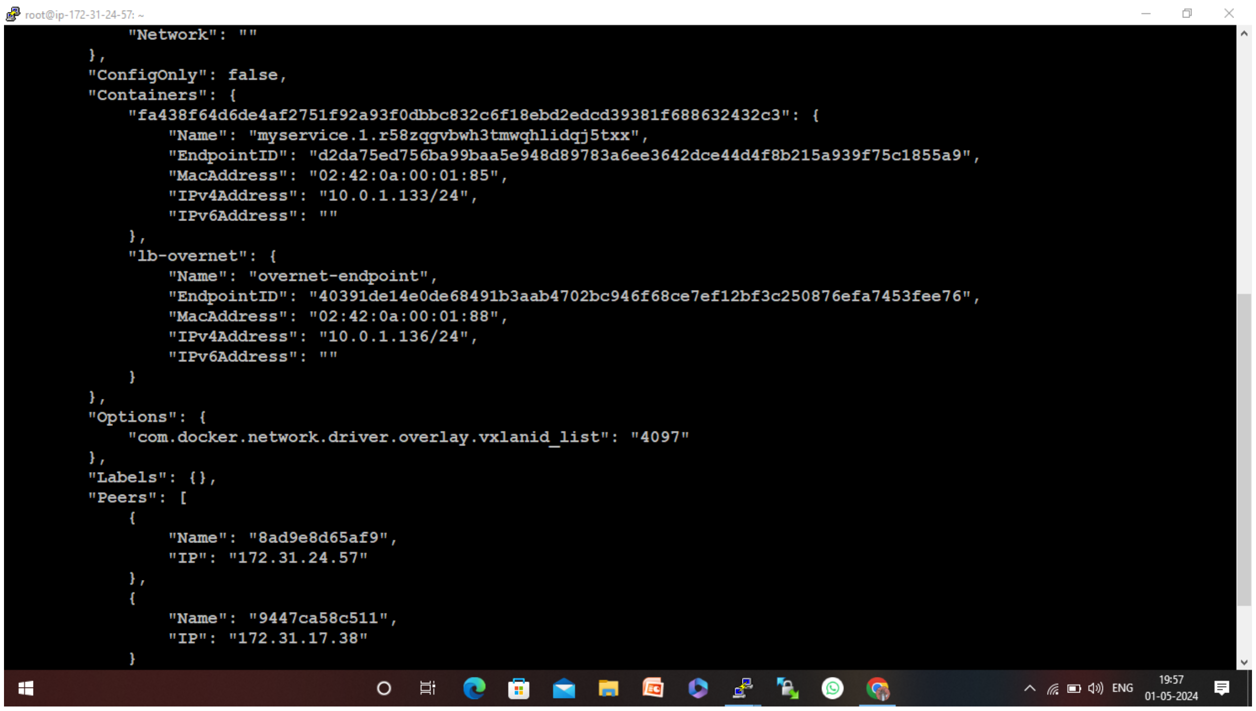


```
        "ConfigOnly": false,
        "Containers": {
            "9024cbe59964957e793d8874c7609a48061434d03e31f4385a86fbb56eba1e22": {
                "Name": "myservice.2.7dpn0ii0ud5snkkn3kg7uh1fr",
                "EndpointID": "2f861b666eab17f2680f8fe58fbccc20929414f75ed0dfb1bde50668472d3f0c",
                "MacAddress": "02:42:0a:00:01:86",
                "IPv4Address": "10.0.1.134/24",
                "IPv6Address": ""
            },
            "lb-overnet": {
                "Name": "overnet-endpoint",
                "EndpointID": "6519473ea5ecc3562c56c899a028b8762bdbba9c9ea60eea9abc4eaefd5b3791",
                "MacAddress": "02:42:0a:00:01:87",
                "IPv4Address": "10.0.1.135/24",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.driver.overlay.vxlanid_list": "4097"
        },
        "Labels": {},
        "Peers": [
            {
                "Name": "9447ca58c511",
                "IP": "172.31.17.38"
            },
            {
                "Name": "8ad9e8d65af9",
                "IP": "172.31.24.57"
            }
        ]
    }
```

5. Inspecting the worker network.



```
root@ip-172-31-24-57:~# docker swarm join --token SWMTKN-1-0hj01ghiclj27yuc9v9mdrzmnjhqt73st7fnqfdj3901ankh21-c30ehii2g19h
11c4r8954t21j 172.31.17.38:2377
This node joined a swarm as a worker.
root@ip-172-31-24-57:~# docker inspect network overnet
[
    {
        "Name": "overnet",
        "Id": "cmv4342zmpuvqgf6t7e7cmmto",
        "Created": "2024-05-01T14:23:37.614630603Z",
        "Scope": "swarm",
        "Driver": "overlay",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "10.0.1.0/24",
                    "Gateway": "10.0.1.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "fa438f64d6de4af2751f92a93f0dbbc832c6f18ebd2edcd39381f688632432c3": {
                "Name": "myservice.1.r58zqgvbwh3tmwqhlidqj5txx",
```



```
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "fa438f64d6de4af2751f92a93f0dbbc832c6f18ebd2edcd39381f688632432c3": {
                "Name": "myservice.1.r58zqgvbwh3tmwqhlidqj5txx",
                "EndpointID": "d2da75ed756ba99baa5e948d89783a6ee3642dce44d4f8b215a939f75c1855a9",
                "MacAddress": "02:42:0a:00:01:85",
                "IPv4Address": "10.0.1.133/24",
                "IPv6Address": ""
            },
            "lb-overnet": {
                "Name": "overnet-endpoint",
                "EndpointID": "40391de14e0de68491b3aab4702bc946f68ce7ef12bf3c250876efa7453fee76",
                "MacAddress": "02:42:0a:00:01:88",
                "IPv4Address": "10.0.1.136/24",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.driver.overlay.vxlanid_list": "4097"
        },
        "Labels": {},
        "Peers": [
            {
                "Name": "8ad9e8d65af9",
                "IP": "172.31.24.57"
            },
            {
                "Name": "9447ca58c511",
                "IP": "172.31.17.38"
            }
```

FIREWALL RULES FOR DOCKER DAEMONS USING OVERLAY NETWORKS

You need the following ports open to traffic to and from each Docker host participating on an overlay network:

TCP port 2377 for cluster management communications

TCP and UDP port 7946 for communication among nodes

UDP port 4789 for overlay network traffic

If you're using Ubuntu or another Debian-based Linux distribution,

you won't have access firewalld, as these are primarily used in Red Hat-based distributions like CentOS, RHEL, and Fedora.

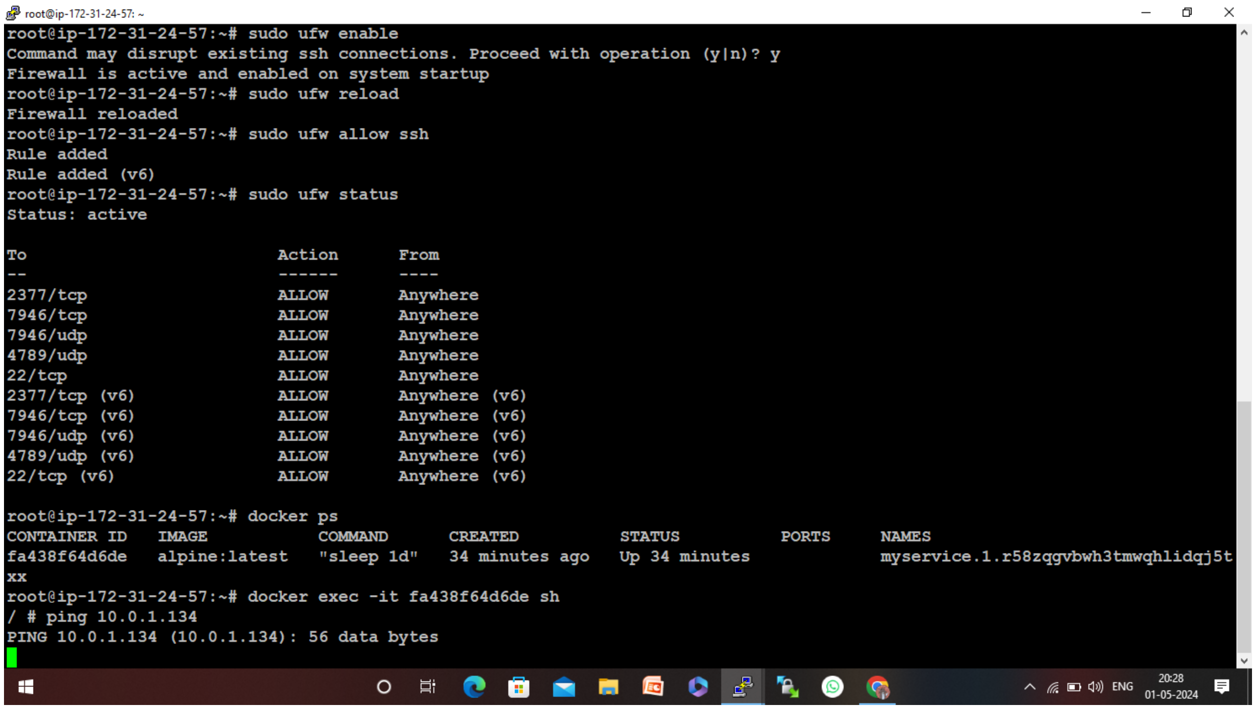
Instead, you'll use apt as the package manager and ufw (Uncomplicated Firewall) for managing firewall rules.

With ufw, you can create rules to allow or deny incoming and outgoing traffic based on your requirements.

It's a straightforward and easy-to-use firewall management tool for Ubuntu and other Debian-based systems.

6 Now in the worker host go into the service(container) and

ping the ip(10.0.1.134) of the manager service(container) it will works.



root@ip-172-31-24-57: ~

```
root@ip-172-31-24-57:~# sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
root@ip-172-31-24-57:~# sudo ufw reload
Firewall reloaded
root@ip-172-31-24-57:~# sudo ufw allow ssh
Rule added
Rule added (v6)
root@ip-172-31-24-57:~# sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
2377/tcp                   ALLOW       Anywhere
7946/tcp                   ALLOW       Anywhere
7946/udp                   ALLOW       Anywhere
4789/udp                   ALLOW       Anywhere
22/tcp                     ALLOW       Anywhere
2377/tcp (v6)              ALLOW       Anywhere (v6)
7946/tcp (v6)              ALLOW       Anywhere (v6)
7946/udp (v6)              ALLOW       Anywhere (v6)
4789/udp (v6)              ALLOW       Anywhere (v6)
22/tcp (v6)                ALLOW       Anywhere (v6)

root@ip-172-31-24-57:~# docker ps
CONTAINER ID   IMAGE           COMMAND      CREATED          STATUS          PORTS     NAMES
fa438f64d6de   alpine:latest   "sleep 1d"   34 minutes ago   Up 34 minutes             myservice.1.r58zqgvbwh3tmwqhlidqj5t
xx
root@ip-172-31-24-57:~# docker exec -it fa438f64d6de sh
/ # ping 10.0.1.134
PING 10.0.1.134 (10.0.1.134): 56 data bytes
```

ENG 20:28 01-05-2024