CO515: Advanced Computer Networks: Selected Topics - Lab 01

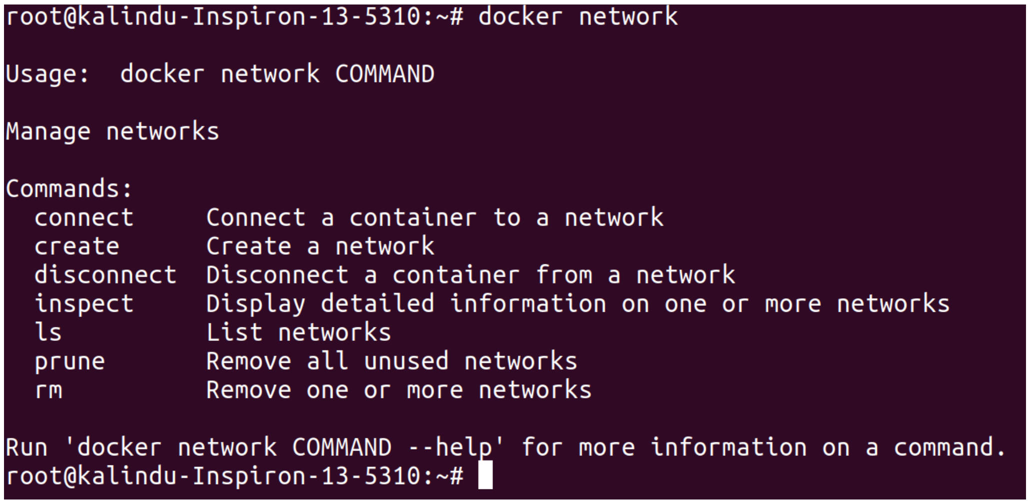
E/19/446: Wijerathna I.M.K.D.I.

28/03/2024

Lab Task 01

1. Getting Familiar with basic docker networking commands

1.1. Run docker network command in the first terminal. Observe the output.

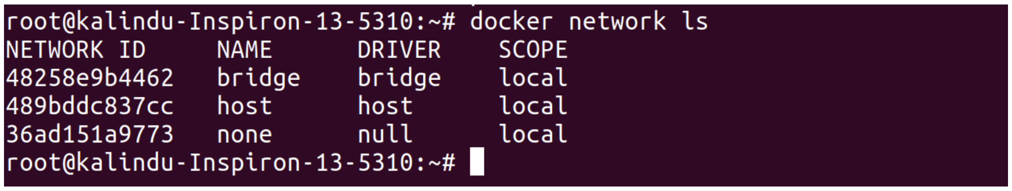


```
root@kalindu-Inspiron-13-5310:~# docker network

Usage:  docker network COMMAND

Manage networks

Commands:
  connect     Connect a container to a network
  create      Create a network
  disconnect  Disconnect a container from a network
  inspect     Display detailed information on one or more networks
  ls          List networks
  prune       Remove all unused networks
  rm          Remove one or more networks

Run 'docker network COMMAND --help' for more information on a command.
root@kalindu-Inspiron-13-5310:~#
```

This command gives an overview of how to use 'docker network' commands to manage networks.

1.2. Use subcommand on docker network command from the list of commands to

1.2.1. Obtain the list of available networks



```
root@kalindu-Inspiron-13-5310:~# docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
48258e9b4462   bridge    bridge    local
489bddc837cc   host      host      local
36ad151a9773   none      null      local
root@kalindu-Inspiron-13-5310:~#
```

The list consists of 4 columns.

1. NETWORK ID*: This column shows the unique identifier (ID) for each Docker network. Docker assigns a random ID to each network when it is created.*
2. NAME: *This column displays the names of the Docker networks. Names are assigned either automatically by Docker or specified by the user when creating the network.*

3. DRIVER: *The driver column specifies the driver type used by each network. Docker supports different network drivers for different use cases. In this output:*
   - *'bridge': This network driver is used to create networks that allow containers on the same Docker host to communicate with each other. It provides a bridge network that connects containers to each other and to the host network.*
   - *'host': This network driver removes network isolation between the container and the Docker host, allowing containers to directly access the host network interfaces.*
   - *'none': This network driver creates containers without networking. Containers created with this network mode have no network interfaces.*

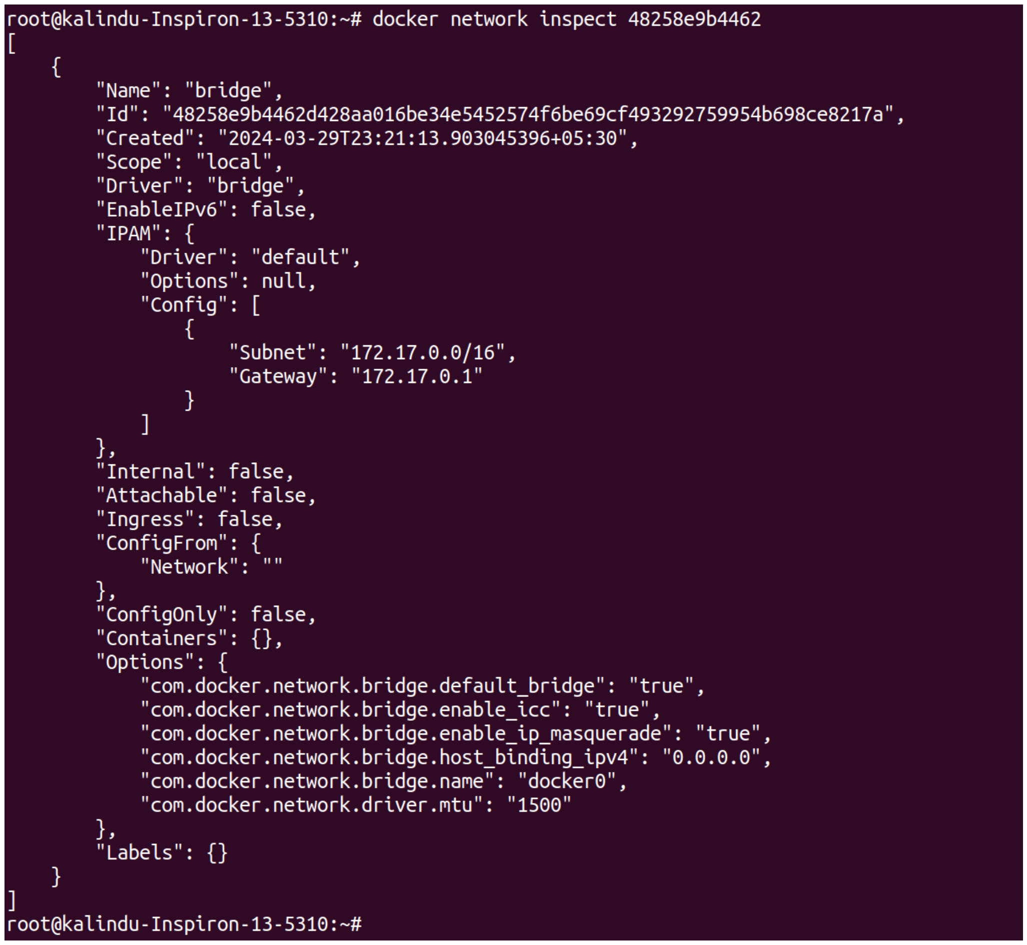
4. *SCOPE: This column indicates the scope of the network,*

   local: The network is only accessible within the Docker host where it was created. It is not visible to containers running on other hosts or external networks.

1.2.2. Obtain the configuration detail of the available networks. List the details which can be obtained using that command.

There are 3 networks available.

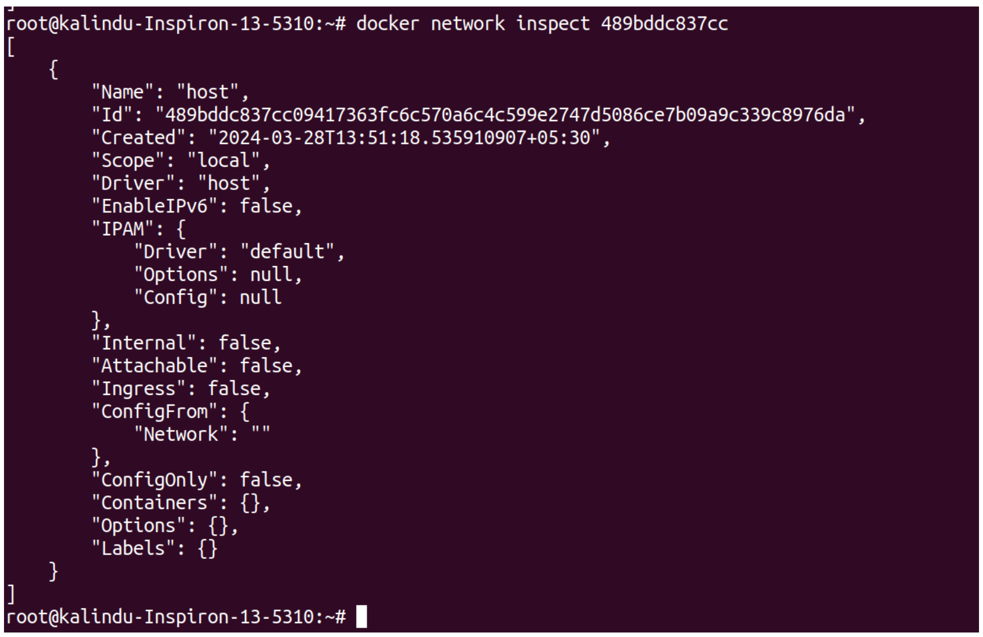
i. Config details of network => '48258e9b4462'



```
root@kalindu-Inspiron-13-5310:~# docker network inspect 48258e9b4462
[
    {
        "Name": "bridge",
        "Id": "48258e9b4462d428aa016be34e5452574f6be69cf493292759954b698ce8217a",
        "Created": "2024-03-29T23:21:13.903045396+05:30",
        "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": {},
        "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@kalindu-Inspiron-13-5310:~#
```

- Name: The name of the Docker network (bridge in this case).
- Id: The unique identifier (ID) of the Docker network.
- Created: The timestamp indicating when the network was created.
- Scope: The scope of the network (local in this case).
- Driver: The driver used by the network (bridge driver).
- EnableIPv6: Indicates whether IPv6 is enabled for the network (in this case, false).
- IPAM: IP Address Management configuration for the network, including:
    - Driver: The IPAM driver (default).
    - Config: IP address configuration for the network, including subnet and gateway information.
- Internal: Indicates whether the network is internal (false in this case).
- Attachable: Indicates whether the network is attachable (false in this case).
- Ingress: Indicates whether the network is used for Ingress (false in this case).
- ConfigFrom: Information about the source of the network configuration.
- ConfigOnly: Indicates whether the network is created from a configuration (false in this case).
- Containers: Information about containers attached to the network (empty in this case).
- Options: Additional options and configuration settings for the network, including:
    - com.docker.network.bridge.default_bridge
    - com.docker.network.bridge.enable_icc
    - com.docker.network.bridge.enable_ip_masquerade
    - com.docker.network.bridge.host_binding_ipv4
    - com.docker.network.bridge.name
    - com.docker.network.driver.mtu
- Labels: Custom metadata labels associated with the network (empty in this case).

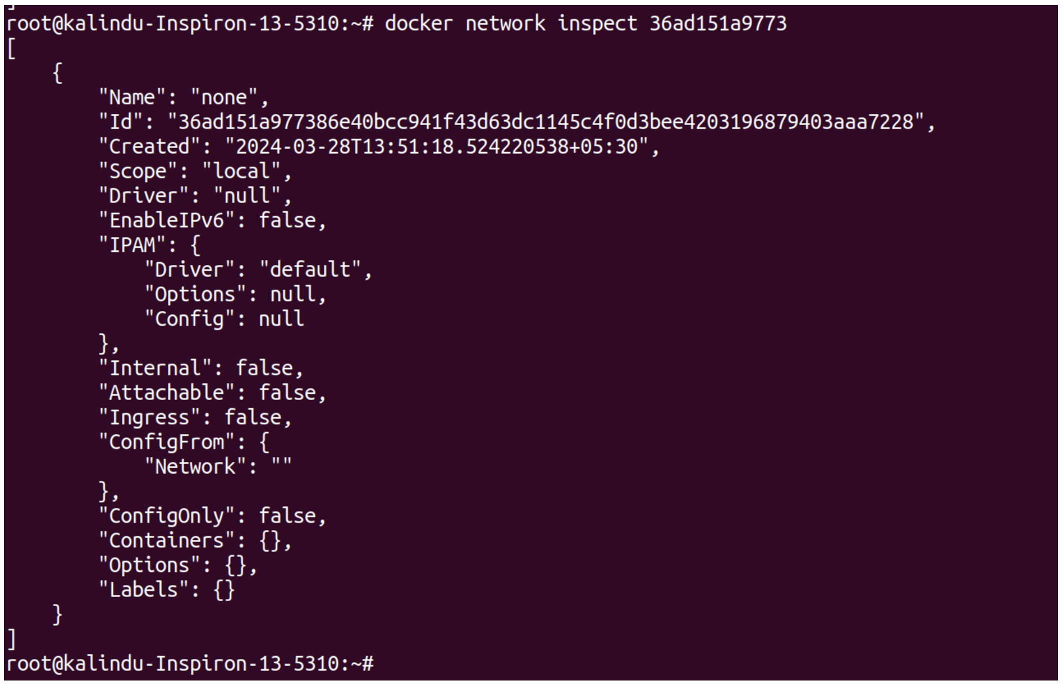
ii. Config details of network => ‘489bddc837cc’



```
root@kalindu-Inspiron-13-5310:~# docker network inspect 489bddc837cc
[
    {
        "Name": "host",
        "Id": "489bddc837cc09417363fc6c570a6c4c599e2747d5086ce7b09a9c339c8976da",
        "Created": "2024-03-28T13:51:18.535910907+05:30",
        "Scope": "local",
        "Driver": "host",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": null
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
root@kalindu-Inspiron-13-5310:~#
```

- *Name: The name of the Docker network (host in this case).*
- *Id: The unique identifier (ID) of the Docker network.*
- *Created: The timestamp indicating when the network was created.*
- *Scope: The scope of the network (local in this case).*
- *Driver: The driver used by the network (host driver).*
- *EnableIPv6: Indicates whether IPv6 is enabled for the network (in this case, false).*
- *IPAM: IP Address Management configuration for the network, which is not applicable in the case of the host network.*
- *Internal: Indicates whether the network is internal (false in this case).*
- *Attachable: Indicates whether the network is attachable (false in this case).*
- *Ingress: Indicates whether the network is used for Ingress (false in this case).*
- *ConfigFrom: Information about the source of the network configuration.*
- *ConfigOnly: Indicates whether the network is created from a configuration (false in this case).*
- *Containers: Information about containers attached to the network (empty in this case).*
- *Options: Additional options and configuration settings for the network, which are empty in this case because the host network doesn't have any specific configuration options.*
- *Labels: Custom metadata labels associated with the network (empty in this case).*

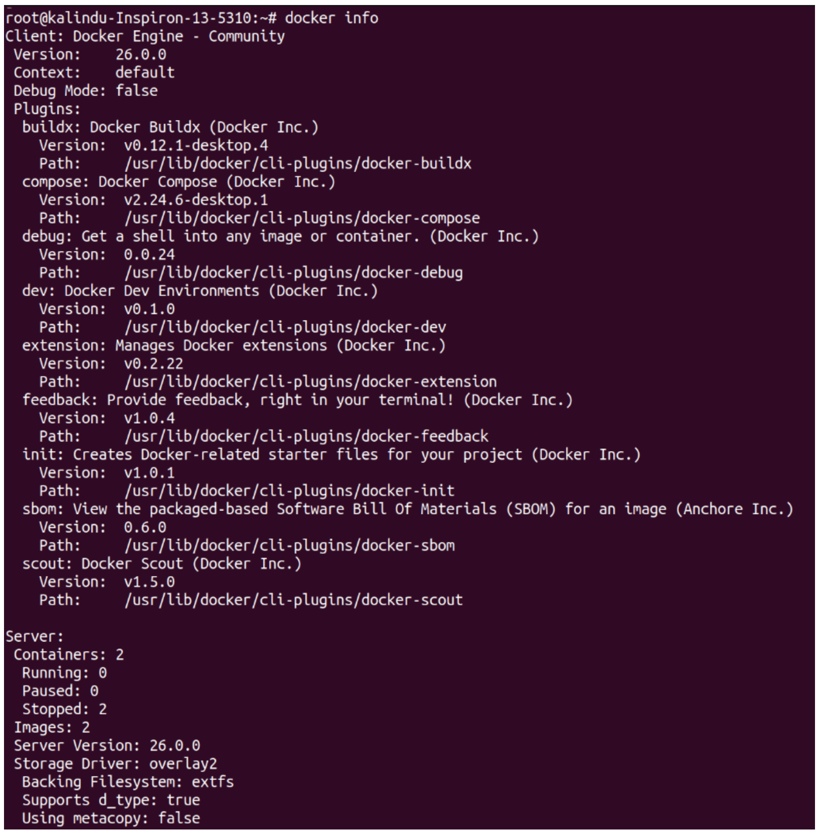
iii. Config details of network => '36ad151a9773'



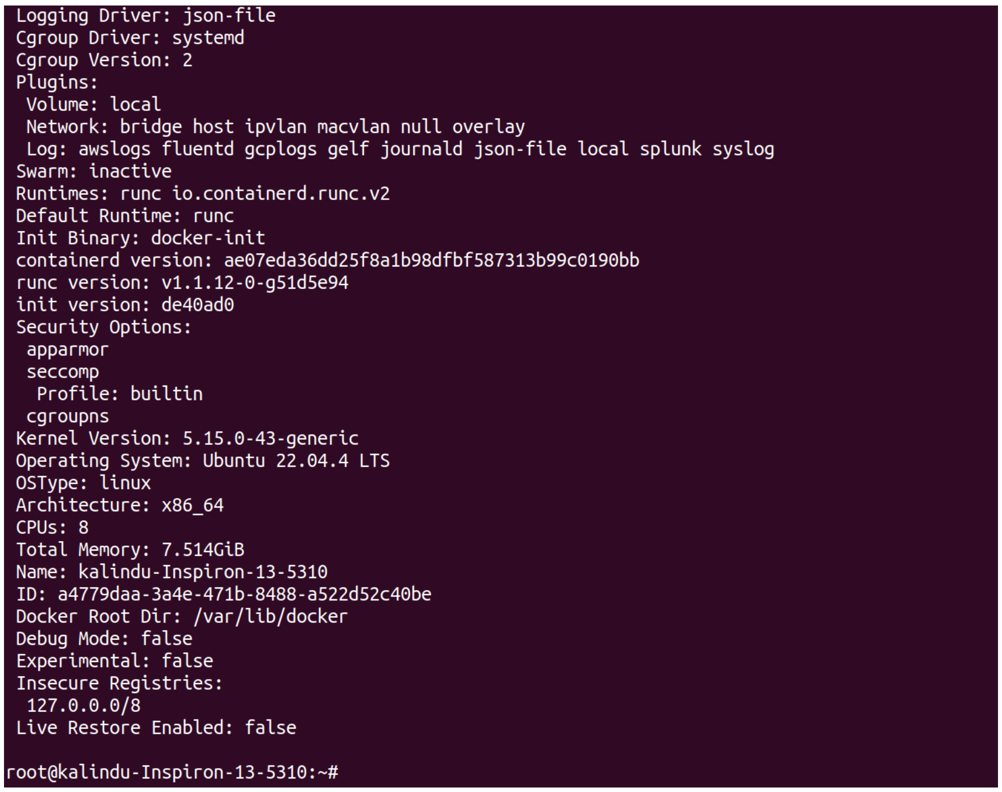
```
root@kalindu-Inspiron-13-5310:~# docker network inspect 36ad151a9773
[
    {
        "Name": "none",
        "Id": "36ad151a977386e40bcc941f43d63dc1145c4f0d3bee4203196879403aaa7228",
        "Created": "2024-03-28T13:51:18.524220538+05:30",
        "Scope": "local",
        "Driver": "null",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": null
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
root@kalindu-Inspiron-13-5310:~#
```

- *Name: The name of the Docker network (none in this case).*
- *Id: The unique identifier (ID) of the Docker network.*
- *Created: The timestamp indicating when the network was created.*
- *Scope: The scope of the network (local in this case).*
- *Driver: The driver used by the network (null driver).*
- *EnableIPv6: Indicates whether IPv6 is enabled for the network (in this case, false).*
- *IPAM: IP Address Management configuration for the network, which is not applicable in the case of the none network.*
- *Internal: Indicates whether the network is internal (false in this case).*
- *Attachable: Indicates whether the network is attachable (false in this case).*
- *Ingress: Indicates whether the network is used for Ingress (false in this case).*
- *ConfigFrom: Information about the source of the network configuration.*
- *ConfigOnly: Indicates whether the network is created from a configuration (false in this case).*
- *Containers: Information about containers attached to the network (empty in this case).*
- *Options: Additional options and configuration settings for the network, which are empty in this case because the none network doesn't have any specific configuration options.*
- *Labels: Custom metadata labels associated with the network (empty in this case).*

1.3. Use docker info command and observe the output.



```
root@kalindu-Inspiron-13-5310:~# docker info
Client: Docker Engine - Community
 Version:    26.0.0
 Context:    default
 Debug Mode: false
 Plugins:
  buildx: Docker Buildx (Docker Inc.)
    Version:  v0.12.1-desktop.4
    Path:     /usr/lib/docker/cli-plugins/docker-buildx
  compose: Docker Compose (Docker Inc.)
    Version:  v2.24.6-desktop.1
    Path:     /usr/lib/docker/cli-plugins/docker-compose
  debug: Get a shell into any image or container. (Docker Inc.)
    Version:  0.0.24
    Path:     /usr/lib/docker/cli-plugins/docker-debug
  dev: Docker Dev Environments (Docker Inc.)
    Version:  v0.1.0
    Path:     /usr/lib/docker/cli-plugins/docker-dev
  extension: Manages Docker extensions (Docker Inc.)
    Version:  v0.2.22
    Path:     /usr/lib/docker/cli-plugins/docker-extension
  feedback: Provide feedback, right in your terminal! (Docker Inc.)
    Version:  v1.0.4
    Path:     /usr/lib/docker/cli-plugins/docker-feedback
  init: Creates Docker-related starter files for your project (Docker Inc.)
    Version:  v1.0.1
    Path:     /usr/lib/docker/cli-plugins/docker-init
  sbom: View the packaged-based Software Bill Of Materials (SBOM) for an image (Anchore Inc.)
    Version:  0.6.0
    Path:     /usr/lib/docker/cli-plugins/docker-sbom
  scout: Docker Scout (Docker Inc.)
    Version:  v1.5.0
    Path:     /usr/lib/docker/cli-plugins/docker-scout

Server:
 Containers: 2
  Running: 0
  Paused: 0
  Stopped: 2
 Images: 2
 Server Version: 26.0.0
 Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Using metacopy: false
```



```
 Logging Driver: json-file
 Cgroup Driver: systemd
 Cgroup Version: 2
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local splunk syslog
 Swarm: inactive
 Runtimes: runc io.containerd.runc.v2
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: ae07eda36dd25f8a1b98dfbf587313b99c0190bb
 runc version: v1.1.12-0-g51d5e94
 init version: de40ad0
 Security Options:
  apparmor
  seccomp
   Profile: builtin
  cgroupns
 Kernel Version: 5.15.0-43-generic
 Operating System: Ubuntu 22.04.4 LTS
 OSType: linux
 Architecture: x86_64
 CPUs: 8
 Total Memory: 7.514GiB
 Name: kalindu-Inspiron-13-5310
 ID: a4779daa-3a4e-471b-8488-a522d52c40be
 Docker Root Dir: /var/lib/docker
 Debug Mode: false
 Experimental: false
 Insecure Registries:
  127.0.0.0/8
 Live Restore Enabled: false

root@kalindu-Inspiron-13-5310:~#
```

This outputs the client and server info of the docker installation in my pc. To elaborate:

<u>*Client Information:*</u>

- *Version: The version of the Docker client installed (26.0.0).*
- *Context: The current context used by the Docker client (default).*
- *Debug Mode: Indicates whether debug mode is enabled (false).*
- *Installed Plugins:The list of installed Docker plugins along with their versions and paths.*
    - *Buildx*
    - *Compose*
    - *Debug*
    - *Dev*
    - *Init*
    - *Sbom*
    - *Scout*
    - *Extention*
    - *Feedback*

<u>*Server Information:*</u>

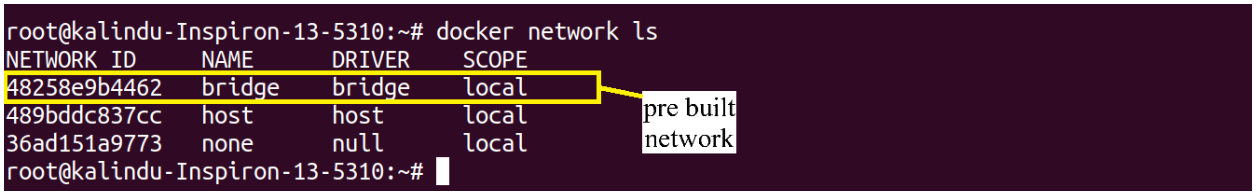
- *Containers: Number of containers on the Docker server (2).*
- *Running: Number of running containers (0).*
- *Paused: Number of paused containers (0).*

- *Stopped: Number of stopped containers (2).*
- *Images: Number of images on the Docker server (2).*
- *Server Version: The version of the Docker server (26.0.0).*
- *Storage Driver: Details about the storage driver used by Docker (overlay2).*
- *Logging Driver: The logging driver used by Docker (json-file).*
- *Cgroup Driver: The control group driver used by Docker (systemd).*
- *Swarm: Indicates whether Docker Swarm mode is active (inactive).*
- *Runtimes: List of container runtimes supported by Docker.*
- *Default Runtime: The default container runtime (runc.v2).*
- *Init Binary: The Docker initialization binary used.*
- *Security Options: Details about security options configured for Docker.*
- *Kernel Version: The kernel version used by Docker (5.15.0-43-generic).*
- *Operating System: The operating system used by Docker (Ubuntu 22.04.4 LTS).*
- *Architecture: The architecture of the system (x86_64).*
- *CPUs: Number of CPUs available to Docker (8).*
- *Total Memory: Total memory available to Docker (7.514GiB).*
- *Docker Root Dir: The directory where Docker stores its data (/var/lib/docker).*
- *HTTP Proxy: HTTP proxy settings configured for Docker.*
- *HTTPS Proxy: HTTPS proxy settings configured for Docker.*
- *No Proxy: No proxy settings configured for Docker.*
- *Experimental: Indicates whether experimental features are enabled (false).*
- *Insecure Registries: List of insecure registries configured for Docker.*
- *Live Restore Enabled: Indicates whether live restore is enabled for Docker (false).*

Lab Task 02

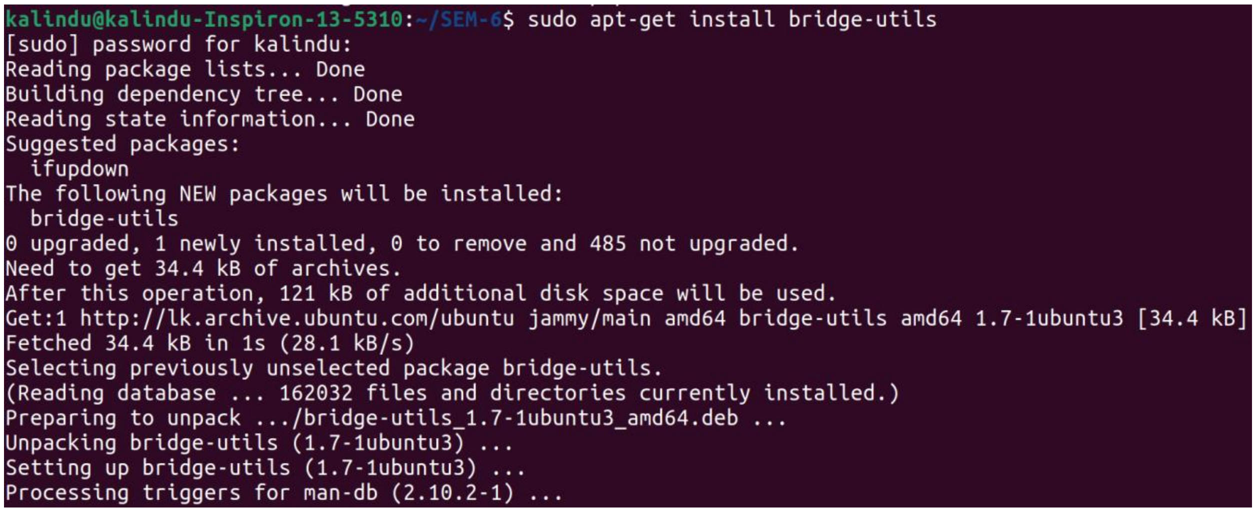
1. Every clean installation of Docker comes with a pre-built network called bridge.

1.1. You can verify this with the docker network ls command.



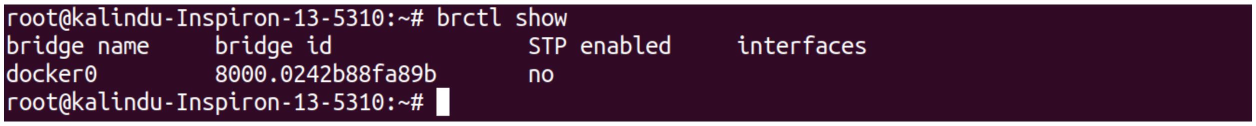
```
root@kalindu-Inspiron-13-5310:~# docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
48258e9b4462   bridge    bridge    local
489bddc837cc   host      host      local
36ad151a9773   none      null      local
root@kalindu-Inspiron-13-5310:~#
```

1.2. Install the brctl command and use it to list the Linux bridges on your Docker host. You can do this by running sudo apt-get install bridge-utils



```
kalindu@kalindu-Inspiron-13-5310:~/SEM-6$ sudo apt-get install bridge-utils
[sudo] password for kalindu:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  ifupdown
The following NEW packages will be installed:
  bridge-utils
0 upgraded, 1 newly installed, 0 to remove and 485 not upgraded.
Need to get 34.4 kB of archives.
After this operation, 121 kB of additional disk space will be used.
Get:1 http://lk.archive.ubuntu.com/ubuntu jammy/main amd64 bridge-utils amd64 1.7-1ubuntu3 [34.4 kB]
Fetched 34.4 kB in 1s (28.1 kB/s)
Selecting previously unselected package bridge-utils.
(Reading database ... 162032 files and directories currently installed.)
Preparing to unpack .../bridge-utils_1.7-1ubuntu3_amd64.deb ...
Unpacking bridge-utils (1.7-1ubuntu3) ...
Setting up bridge-utils (1.7-1ubuntu3) ...
Processing triggers for man-db (2.10.2-1) ...
```

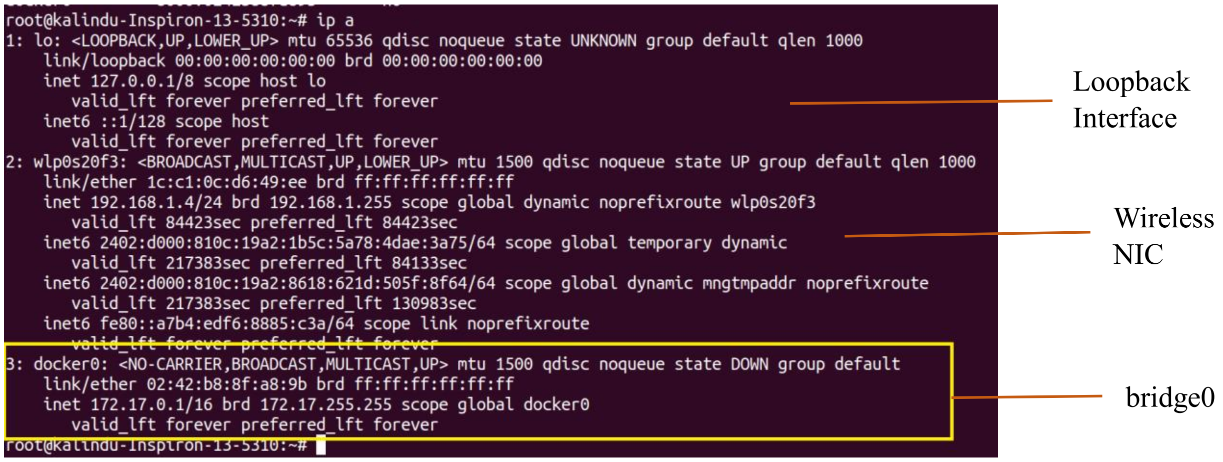
1.3. Use brctl show command to list the bridges on your Docker host. The output above shows a single Linux bridge called docker0. This is the bridge that was automatically created for the bridge network. You can see that it has no interfaces currently connected to it.



```
root@kalindu-Inspiron-13-5310:~# brctl show
bridge name     bridge id               STP enabled     interfaces
docker0         8000.0242b88fa89b       no
root@kalindu-Inspiron-13-5310:~#
```

- *This indicates that 'docker0' bridge is present in the system*
- *Since no Docker containers have been created, 'docker0' is currently not paired with an interface.*
- *Spanning Tree Protocol is disabled by default.*

1.4. use the ip a command to view details of the docker0 bridge

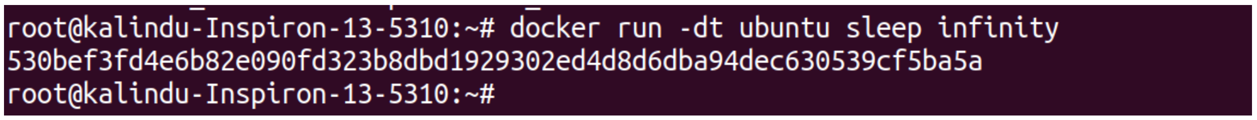


```
root@kalindu-Inspiron-13-5310:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: wlp0s20f3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 1c:c1:0c:d6:49:ee brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.4/24 brd 192.168.1.255 scope global dynamic noprefixroute wlp0s20f3
       valid_lft 84423sec preferred_lft 84423sec
    inet6 2402:d000:810c:19a2:1b5c:5a78:4dae:3a75/64 scope global temporary dynamic
       valid_lft 217383sec preferred_lft 84133sec
    inet6 2402:d000:810c:19a2:8618:621d:505f:8f64/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 217383sec preferred_lft 130983sec
    inet6 fe80::a7b4:edf6:8885:c3a/64 scope link noprefixroute
       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:b8:8f:a8:9b 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
root@kalindu-Inspiron-13-5310:~#
```

- *Status: DOWN (NO-CARRIER) – This specifies that the interface is not functioning at the moment because there is no active network connection detected on it.*
- *MAC Address: 02:42:b8:8f:a8:9b – The OUI '02:42:XX' indicates that this is a virtual interface belong to Docker*
- *IP Address: 172.17.0.1/16*
- *Valid life time and Preferred lifetime: Forever*

2. Connecting a Container

2.1. Create a new container by running docker run -dt ubuntu sleep infinity

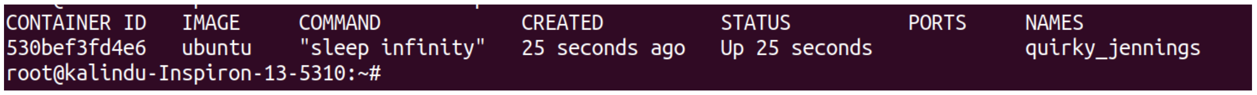

```
root@kalindu-Inspiron-13-5310:~# docker run -dt ubuntu sleep infinity
530bef3fd4e6b82e090fd323b8dbd1929302ed4d8d6dba94dec630539cf5ba5a
root@kalindu-Inspiron-13-5310:~#
```

This creates a new container with

- *'-dt': in detached mode (-d) and allocate a pseudo-TTY (-t)*
- *Ubuntu image*
- *'sleep' – Do nothing*
- *'infinity' – Remain running indefinitely*

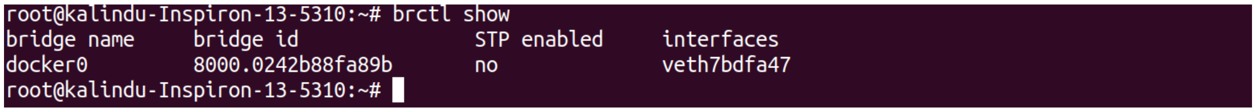
Then, the new container ID is returned. ('530bef3fd……')

2.2. Verify that the new container is up by running docker ps


```
CONTAINER ID   IMAGE     COMMAND            CREATED          STATUS          PORTS     NAMES
530bef3fd4e6   ubuntu    "sleep infinity"   25 seconds ago   Up 25 seconds             quirky_jennings
root@kalindu-Inspiron-13-5310:~#
```

- *The container details are shown and it indicates that it has been running for 25 seconds till present.*
- *Name of the container is auto generated as 'quirky_jennings'.*
- *No ports have been exposed to the outside from the container to the host*
- *'sleep infinity' command has been executing inside the container*

2.3. As no network was specified on the docker run command, the container will be added to the bridge network. Run the brctl show command again.

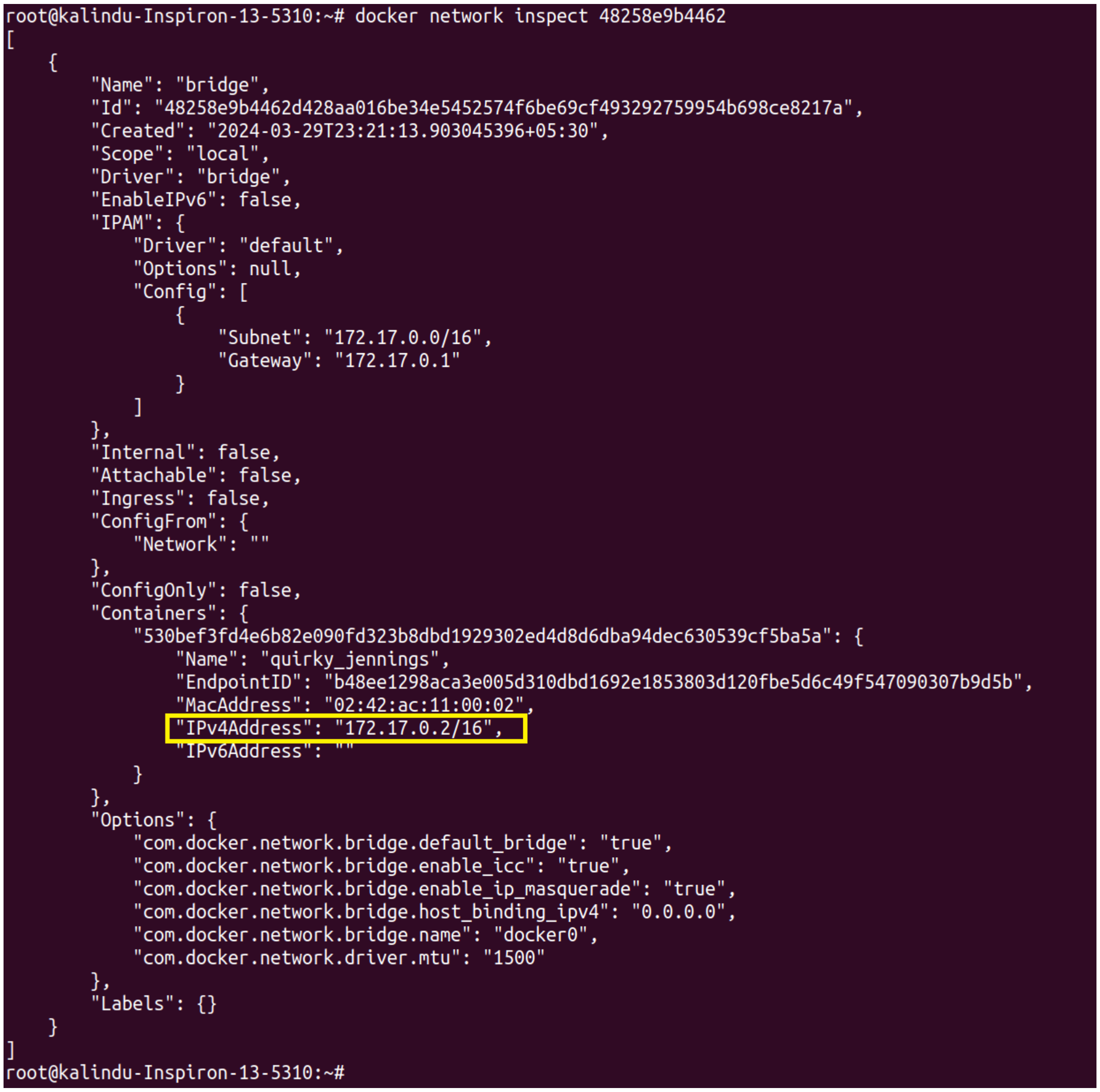

```
root@kalindu-Inspiron-13-5310:~# brctl show
bridge name     bridge id               STP enabled     interfaces
docker0         8000.0242b88fa89b       no              veth7bdfa47
root@kalindu-Inspiron-13-5310:~#
```

Now, 'docker0' is paired with a virtual ethernet interface named 'veth7bdfa47'.

The 'veth7bdfa47' interface is a virtual Ethernet pair associated with a Docker container, where one end is inside the container namespace ('veth7bdfa47') and the other end is connected to the bridge ('docker0'). This setup allows communication between the container and the host network.

2.4. Notice how the docker0 bridge now has an interface connected.Inspect the bridge network again using appropriate command. What is the ip address of the new container?

The IP address of the new container's virtual network interface is: 172.17.0.2/16 as shown below.

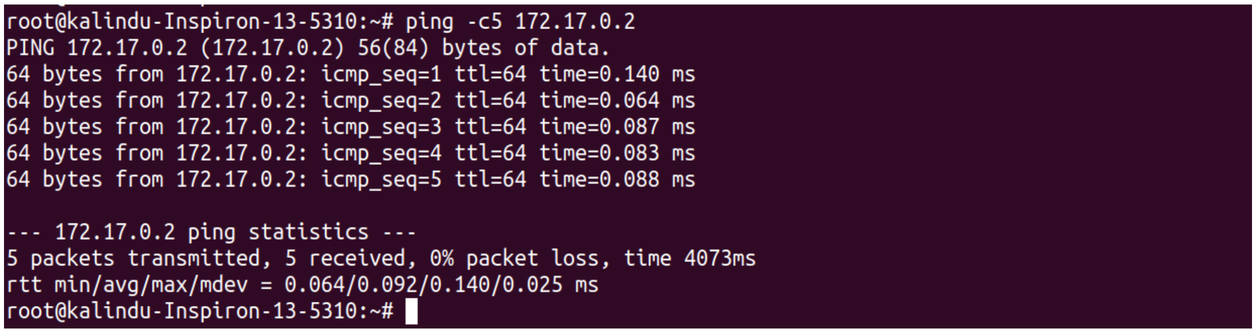


```
root@kalindu-Inspiron-13-5310:~# docker network inspect 48258e9b4462
[
    {
        "Name": "bridge",
        "Id": "48258e9b4462d428aa016be34e5452574f6be69cf493292759954b698ce8217a",
        "Created": "2024-03-29T23:21:13.903045396+05:30",
        "Scope": "local",
        "Driver": "bridge",
        "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": {
            "530bef3fd4e6b82e090fd323b8dbd1929302ed4d8d6dba94dec630539cf5ba5a": {
                "Name": "quirky_jennings",
                "EndpointID": "b48ee1298aca3e005d310dbd1692e1853803d120fbe5d6c49f547090307b9d5b",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "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@kalindu-Inspiron-13-5310:~#
```

3. Test network connectivity

3.1. Ping the IP address of the container from the shell prompt of your Docker host by running ping -c5 .Observe the replies.

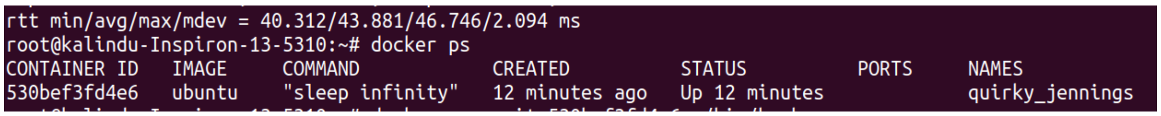
The ping was successful.



```
root@kalindu-Inspiron-13-5310:~# ping -c5 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.140 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.064 ms
64 bytes from 172.17.0.2: icmp_seq=3 ttl=64 time=0.087 ms
64 bytes from 172.17.0.2: icmp_seq=4 ttl=64 time=0.083 ms
64 bytes from 172.17.0.2: icmp_seq=5 ttl=64 time=0.088 ms

--- 172.17.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4073ms
rtt min/avg/max/mdev = 0.064/0.092/0.140/0.025 ms
root@kalindu-Inspiron-13-5310:~#
```

3.2. Verify that the container can connect to the outside world by ping to www.google.com.

3.2.1. Get the ID of the container using docker ps

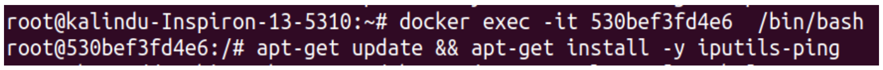


```
rtt min/avg/max/mdev = 40.312/43.881/46.746/2.094 ms
root@kalindu-Inspiron-13-5310:~# docker ps
CONTAINER ID   IMAGE     COMMAND            CREATED          STATUS          PORTS     NAMES
530bef3fd4e6   ubuntu    "sleep infinity"   12 minutes ago   Up 12 minutes             quirky_jennings
```

Therefore, the container ID is '530bef3fd4e6'

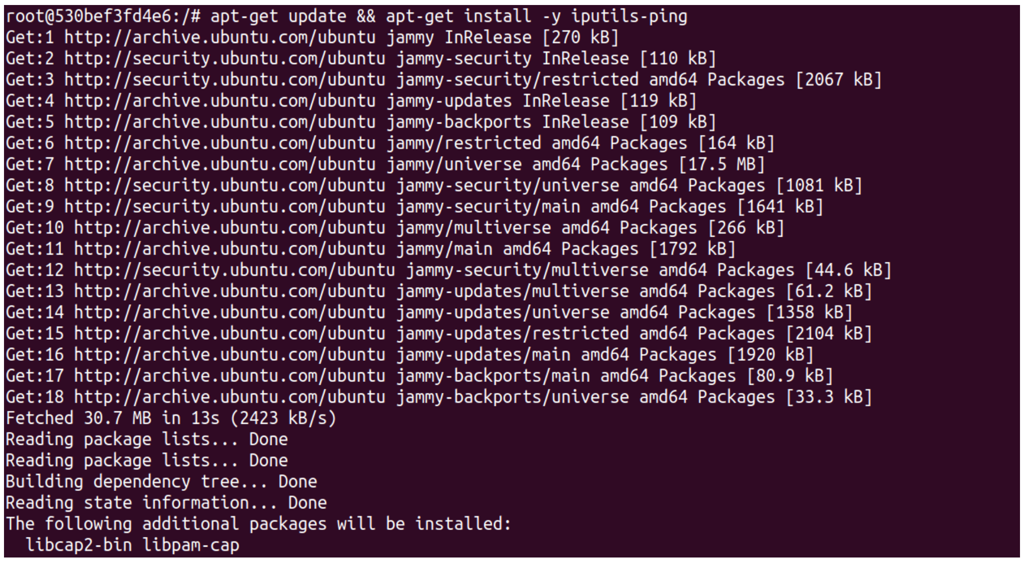
3.2.2. Run a shell inside that ubuntu container, by running docker exec -it /bin/bash

Entering the shell inside the container:



```
root@kalindu-Inspiron-13-5310:~# docker exec -it 530bef3fd4e6  /bin/bash
root@530bef3fd4e6:/# apt-get update && apt-get install -y iputils-ping
```

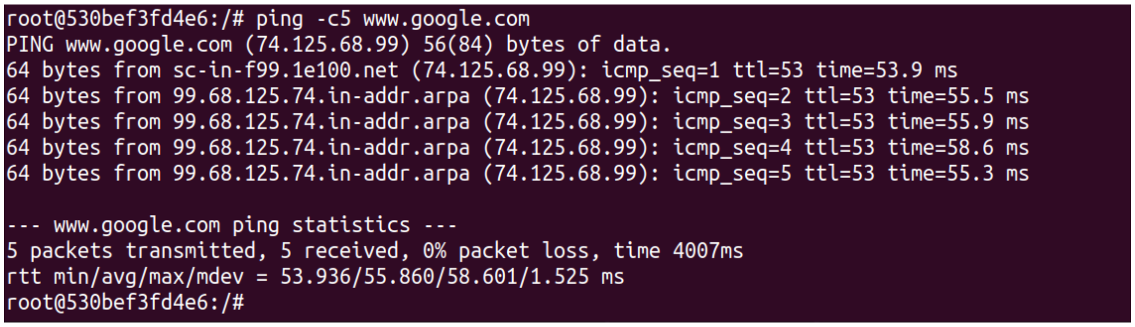
3.2.3. Run apt-get update && apt-get install -y iputils-ping to install the ping program



```
root@530bef3fd4e6:/# apt-get update && apt-get install -y iputils-ping
Get:1 http://archive.ubuntu.com/ubuntu jammy InRelease [270 kB]
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [2067 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy/restricted amd64 Packages [164 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [17.5 MB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [1081 kB]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1641 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [266 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy/main amd64 Packages [1792 kB]
Get:12 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [44.6 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [61.2 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1358 kB]
Get:15 http://archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [2104 kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1920 kB]
Get:17 http://archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [80.9 kB]
Get:18 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [33.3 kB]
Fetched 30.7 MB in 13s (2423 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libcap2-bin libpam-cap
```

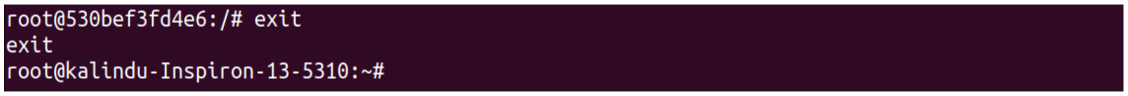
3.2.4. ping www.google.com by running ping -c5 www.google.com

Ping was successful. Hence, can communicate with outside networks.



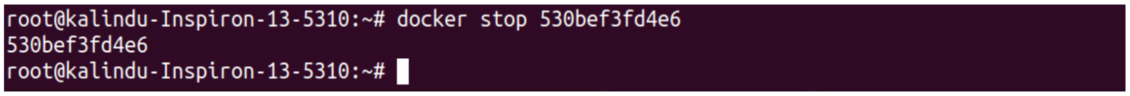
```
root@530bef3fd4e6:/# ping -c5 www.google.com
PING www.google.com (74.125.68.99) 56(84) bytes of data.
64 bytes from sc-in-f99.1e100.net (74.125.68.99): icmp_seq=1 ttl=53 time=53.9 ms
64 bytes from 99.68.125.74.in-addr.arpa (74.125.68.99): icmp_seq=2 ttl=53 time=55.5 ms
64 bytes from 99.68.125.74.in-addr.arpa (74.125.68.99): icmp_seq=3 ttl=53 time=55.9 ms
64 bytes from 99.68.125.74.in-addr.arpa (74.125.68.99): icmp_seq=4 ttl=53 time=58.6 ms
64 bytes from 99.68.125.74.in-addr.arpa (74.125.68.99): icmp_seq=5 ttl=53 time=55.3 ms

--- www.google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 53.936/55.860/58.601/1.525 ms
root@530bef3fd4e6:/#
```

3.2.5. Run exit to disconnect the shell from the container



```
root@530bef3fd4e6:/# exit
exit
root@kalindu-Inspiron-13-5310:~#
```

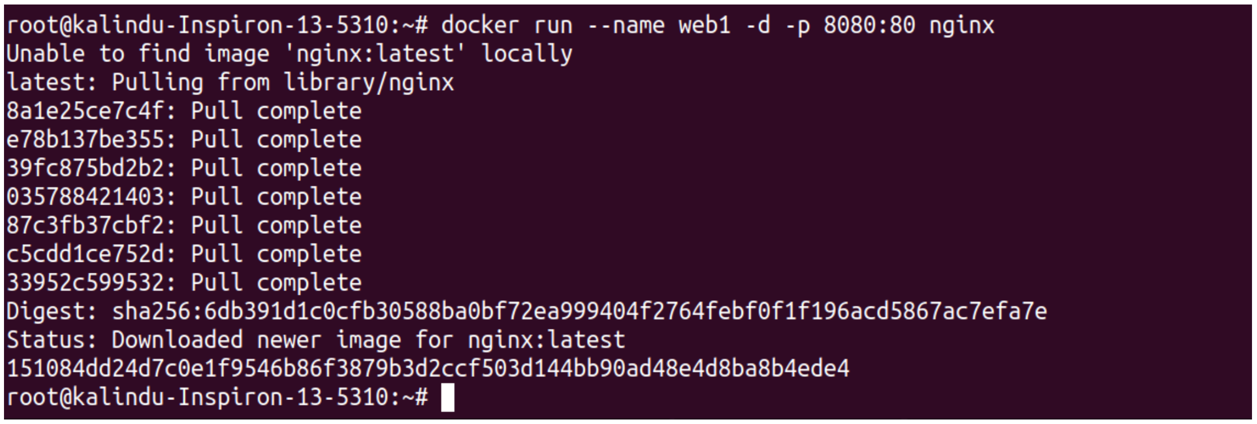
3.2.6. Stop this container by cleaning things up from this test, by running docker stop



```
root@kalindu-Inspiron-13-5310:~# docker stop 530bef3fd4e6
530bef3fd4e6
root@kalindu-Inspiron-13-5310:~#
```

4. Configure NAT for external connectivity

4.1. Start a new container based off the official NGINX image by running docker run --name web1 -d -p 8080:80 nginx

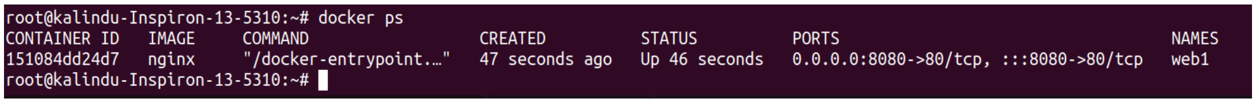


```
root@kalindu-Inspiron-13-5310:~# docker run --name web1 -d -p 8080:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
8a1e25ce7c4f: Pull complete
e78b137be355: Pull complete
39fc875bd2b2: Pull complete
035788421403: Pull complete
87c3fb37cbf2: Pull complete
c5cdd1ce752d: Pull complete
33952c599532: Pull complete
Digest: sha256:6db391d1c0cfb30588ba0bf72ea999404f2764febf0f1f196acd5867ac7efa7e
Status: Downloaded newer image for nginx:latest
151084dd24d7c0e1f9546b86f3879b3d2ccf503d144bb90ad48e4d8ba8b4ede4
root@kalindu-Inspiron-13-5310:~#
```

This creates a new container with

- *Name as 'web1'*
- *'-d': Runs the container in detached mode (runs in the background)*
- *'-p 8080:80': Maps port 8080 on the host to port 80 in the container, allowing access to the NGINX web server running inside the container.*
- *'nginx': using NGINX as the docker image*

4.2. Review the container status and port mappings by running docker ps

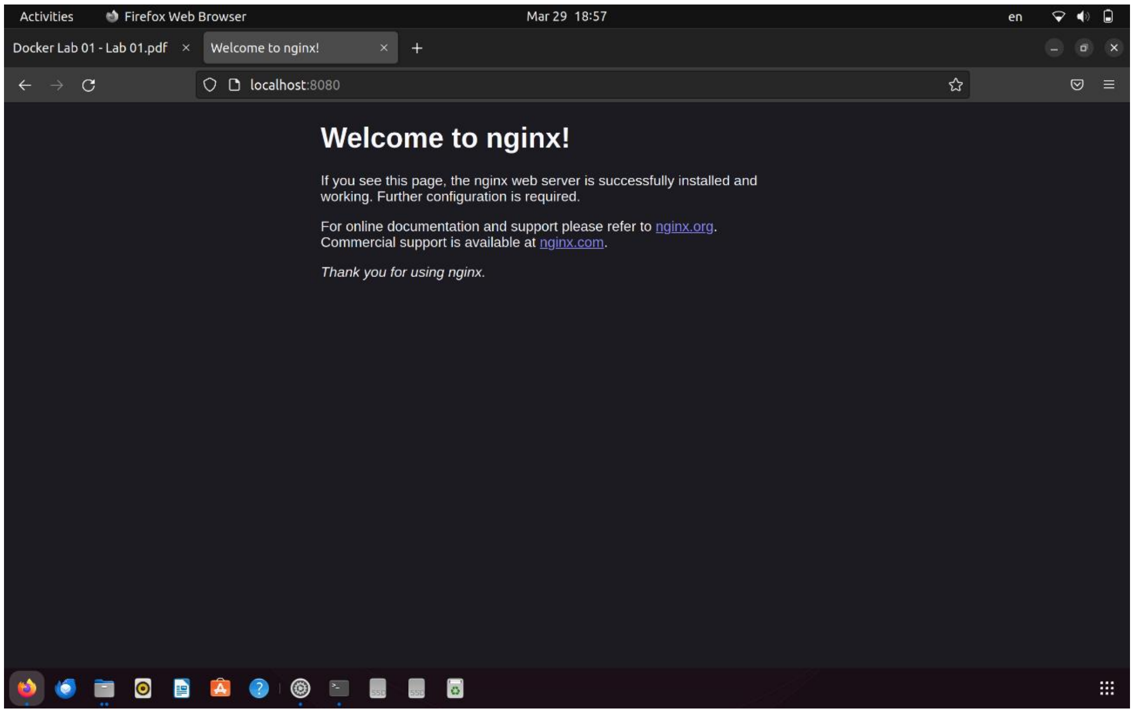


```
root@kalindu-Inspiron-13-5310:~# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                                   NAMES
151084dd24d7   nginx     "/docker-entrypoint.…"   47 seconds ago   Up 46 seconds   0.0.0.0:8080->80/tcp, :::8080->80/tcp   web1
root@kalindu-Inspiron-13-5310:~# 
```

This verifies the above configured details:

- *Port Mappings:*
  - *Host: 0.0.0.0:8080*
  - *Container: 80/tcp*
- *Name: 'web1'*
- *Docker image: NGINX*
- *Command: 'docker-entrypoint.sh' script is running which keeps running the NGINX web server*

Verification through local Web Browser:



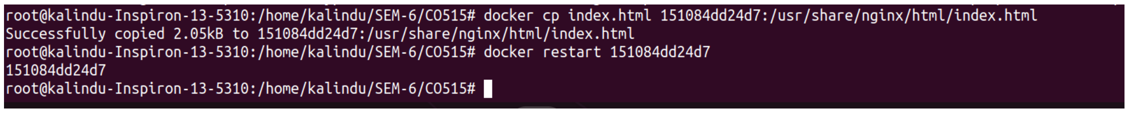
Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

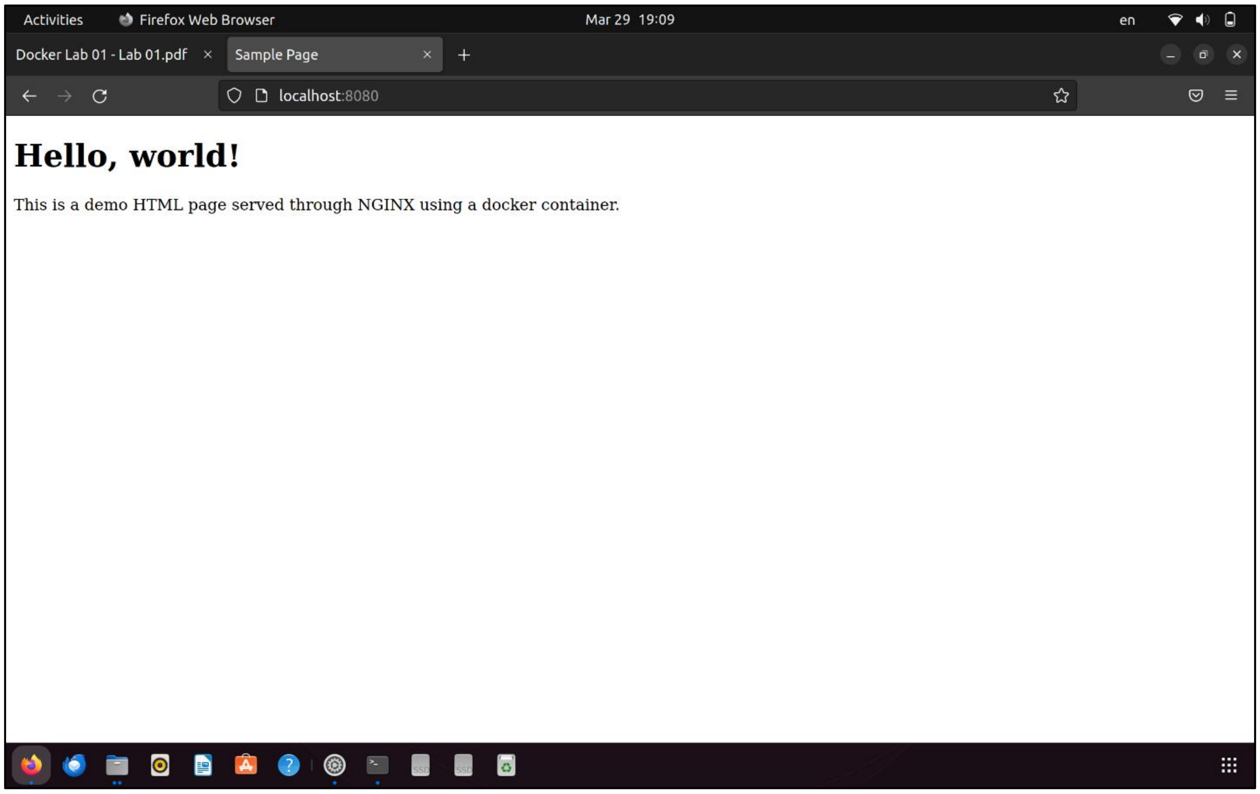
Thank you for using nginx.

Then, I created a sample index.html and placed it inside the container to ensure if it would correctly serve a simple web site.



```
root@kalindu-Inspiron-13-5310:/home/kalindu/SEM-6/CO515# docker cp index.html 151084dd24d7:/usr/share/nginx/html/index.html
Successfully copied 2.05kB to 151084dd24d7:/usr/share/nginx/html/index.html
root@kalindu-Inspiron-13-5310:/home/kalindu/SEM-6/CO515# docker restart 151084dd24d7
151084dd24d7
root@kalindu-Inspiron-13-5310:/home/kalindu/SEM-6/CO515# 
```

Verification of a Simple Hello World Page:

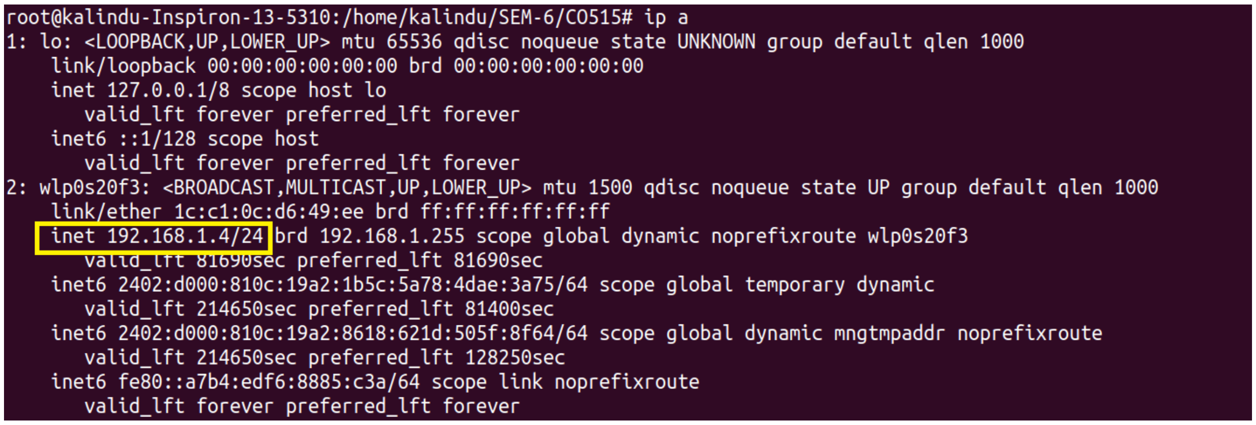


Hello, world!

This is a demo HTML page served through NGINX using a docker container.

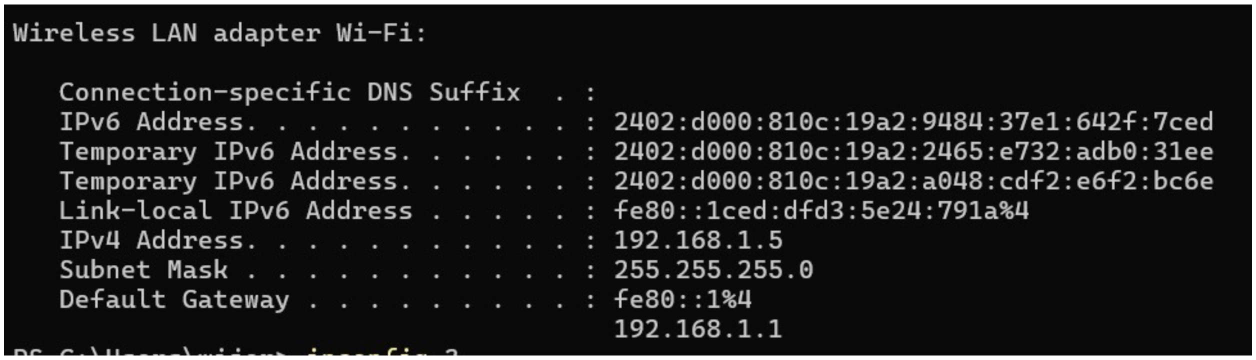
Ensuring whether the web page is visible from another device in the same network as the hosted PC:

Hosted PC's IP: 192.168.1.4/24 as shown in the following

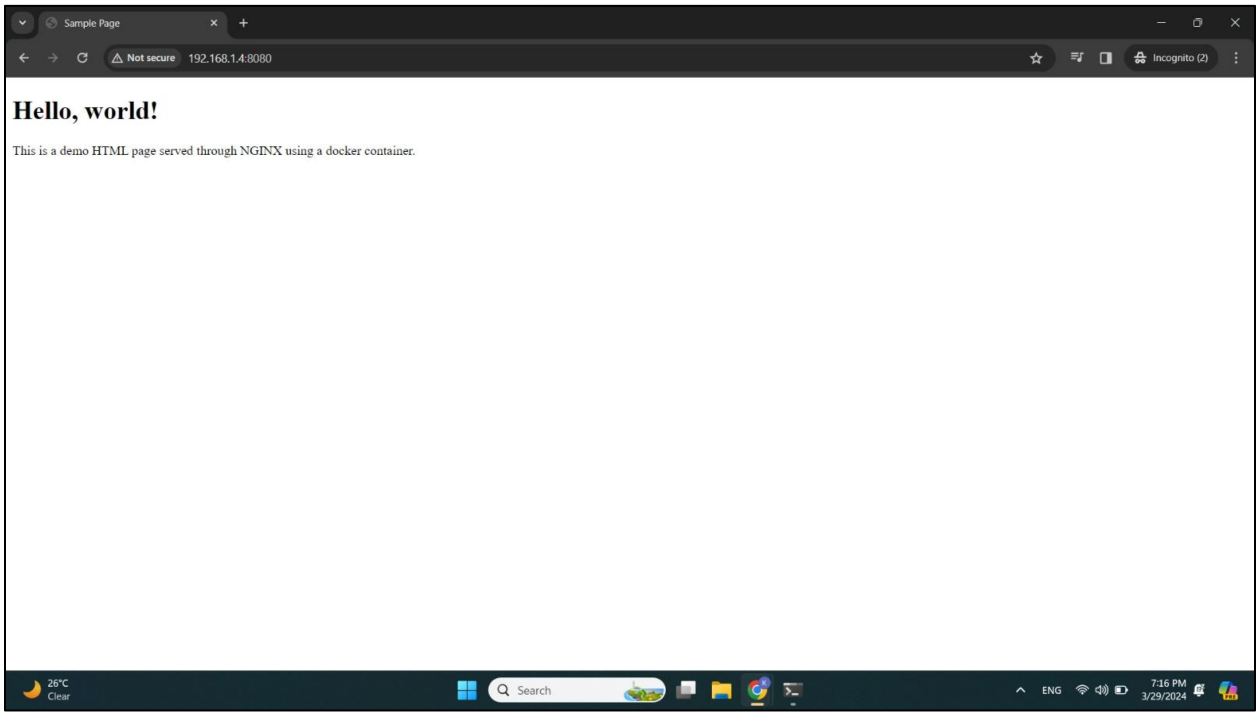


```
root@kalindu-Inspiron-13-5310:/home/kalindu/SEM-6/CO515# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: wlp0s20f3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 1c:c1:0c:d6:49:ee brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.4/24 brd 192.168.1.255 scope global dynamic noprefixroute wlp0s20f3
       valid_lft 81690sec preferred_lft 81690sec
    inet6 2402:d000:810c:19a2:1b5c:5a78:4dae:3a75/64 scope global temporary dynamic
       valid_lft 214650sec preferred_lft 81400sec
    inet6 2402:d000:810c:19a2:8618:621d:505f:8f64/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 214650sec preferred_lft 128250sec
    inet6 fe80::a7b4:edf6:8885:c3a/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Trying to access the web page using a device with IP address: 192.168.1.5/24 (which is on the same pvt network)



```
Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2402:d000:810c:19a2:9484:37e1:642f:7ced
   Temporary IPv6 Address. . . . . . : 2402:d000:810c:19a2:2465:e732:adb0:31ee
   Temporary IPv6 Address. . . . . . : 2402:d000:810c:19a2:a048:cdf2:e6f2:bc6e
   Link-local IPv6 Address . . . . . : fe80::1ced:dfd3:5e24:791a%4
   IPv4 Address. . . . . . . . . . . : 192.168.1.5
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::1%4
                                       192.168.1.1
```



Hello, world!

This is a demo HTML page served through NGINX using a docker container.

The Page was successfully displayed.