Experiment 8

Aim- Implement Networking Commands in Linux (NIYATI SAVANT)

Commands and their explanation

ifconfig

The ‘ifconfig’ command in Linux is used to manage network interfaces. When executed without arguments, it displays information about all active network interfaces, including their IP addresses and MAC addresses. You can also use it to enable or disable interfaces, set IP addresses, and configure netmasks. While 'ifconfig' is still available on many systems, modern Linux distributions often recommend using the 'ip' command for more advanced networking tasks.

```
practicalexampc29@LAB306PC32:~$ ifconfig
enp1s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.31.47  netmask 255.255.255.0  broadcast 192.168.31.255
        inet6 fe80::f49a:1df0:4642:17d7  prefixlen 64  scopeid 0x20<link>
        ether a4:ae:12:84:83:16  txqueuelen 1000  (Ethernet)
        RX packets 9353  bytes 10583209 (10.5 MB)
        RX errors 0  dropped 8  overruns 0  frame 0
        TX packets 3081  bytes 249128 (249.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 342  bytes 34799 (34.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 342  bytes 34799 (34.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

virbr0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 192.168.122.1  netmask 255.255.255.0  broadcast 192.168.122.255
        ether 52:54:00:2a:e9:6f  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

ping

The ping command in Linux is used to test the connectivity between your computer and another host on a network, typically using the Internet Control Message Protocol (ICMP). When you run ping followed by a host or IP address, it sends a series of ICMP echo request packets to the specified host. If the host is reachable and responsive, it will reply with ICMP echo reply packets. This command is commonly used to check network connectivity, measure network latency (ping time), and troubleshoot network issues. By

default, ping sends a series of packets and displays statistics about the packets sent and received, helping you assess the quality of the network connection.

```
practicalexampc29@LAB306PC32:~$ ping google.com
PING google.com (142.250.182.206) 56(84) bytes of data.
64 bytes from bom07s28-in-f14.1e100.net (142.250.182.206): icmp_seq=1 ttl=58
time=3.50 ms
64 bytes from bom07s28-in-f14.1e100.net (142.250.182.206): icmp_seq=2 ttl=58
time=3.70 ms
64 bytes from bom07s28-in-f14.1e100.net (142.250.182.206): icmp_seq=3 ttl=58
time=3.58 ms
64 bytes from bom07s28-in-f14.1e100.net (142.250.182.206): icmp_seq=4 ttl=58
time=3.51 ms
^C
--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 3.499/3.572/3.699/0.080 ms


practicalexampc29@LAB306PC32:~$ ping 142.250.182.206
PING 142.250.182.206 (142.250.182.206) 56(84) bytes of data.
64 bytes from 142.250.182.206: icmp_seq=1 ttl=58 time=3.59 ms
64 bytes from 142.250.182.206: icmp_seq=2 ttl=58 time=3.65 ms
64 bytes from 142.250.182.206: icmp_seq=3 ttl=58 time=4.47 ms
64 bytes from 142.250.182.206: icmp_seq=4 ttl=58 time=3.53 ms
^C
--- 142.250.182.206 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 3.532/3.810/4.470/0.383 ms
```

ping -c

The ping -c command is an extension of the standard ping command with the -c flag used to specify the number of ICMP echo request packets to send. When you use ping -c followed by a number, such as ping -c 5, it instructs ping to send a specific count of ICMP echo request packets to the target host or IP address. After sending the specified number of packets, ping will display a summary of the results, including statistics like packet loss, round-trip time (ping time), and more. This option is useful for running a specific number of ping tests to assess network connectivity or diagnose issues.

```
practicalexampc29@LAB306PC32:~$ ping -c 4 142.250.182.206
PING 142.250.182.206 (142.250.182.206) 56(84) bytes of data.
64 bytes from 142.250.182.206: icmp_seq=1 ttl=58 time=3.85 ms
64 bytes from 142.250.182.206: icmp_seq=2 ttl=58 time=3.47 ms
64 bytes from 142.250.182.206: icmp_seq=3 ttl=58 time=3.60 ms
64 bytes from 142.250.182.206: icmp_seq=4 ttl=58 time=3.39 ms

--- 142.250.182.206 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 3.392/3.578/3.850/0.173 ms
```

ping -i5

The ping -i command is used to specify the interval between sending ICMP echo request packets when using the ping command. When you use ping -i followed by a number, such as ping -i 5, it determines the time interval (in seconds) between sending each ICMP echo request packet to the target host or IP address. Setting the interval with -i can be helpful when you want to space out the ping requests, especially in situations where you don't want to flood the network with too many requests in a short period. Adjusting the interval allows you to control the rate at which ping requests are sent, making it useful for network troubleshooting or monitoring.

```
practicalexampc29@LAB306PC32:~$ ping -i5 142.250.182.206
PING 142.250.182.206 (142.250.182.206) 56(84) bytes of data.
64 bytes from 142.250.182.206: icmp_seq=1 ttl=58 time=3.57 ms
64 bytes from 142.250.182.206: icmp_seq=2 ttl=58 time=3.66 ms
64 bytes from 142.250.182.206: icmp_seq=3 ttl=58 time=3.76 ms
^C
--- 142.250.182.206 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 10012ms
rtt min/avg/max/mdev = 3.571/3.663/3.757/0.075 ms
```

ping -R

The 'ping -R' command is used to enable record route functionality in the 'ping' command. When you use 'ping -R', it instructs the 'ping' command to include the record route option in the ICMP echo request packets it sends to the target host or IP address. This option is typically used for debugging and network analysis. When the record route option is enabled, each ICMP echo request packet includes a list of routers (hops) that the packet traverses on its way to the destination. The routers' IP addresses are recorded in the packet, allowing you to see the path that the packet takes through the network.

```
practicalexampc29@LAB306PC32:~$ ping -R 142.250.182.206
PING 142.250.182.206 (142.250.182.206) 56(124) bytes of data.
^C
--- 142.250.182.206 ping statistics ---
26 packets transmitted, 0 received, 100% packet loss, time 25592ms
```

ping -w6

The ping -w command is used to specify a timeout period for the ping command. When you use ping -w followed by a number, such as ping -w 6, it sets a timeout period in seconds for each ICMP echo request packet sent to the target host or IP address. In this example, ping will wait for up to 6 seconds for a response from the destination. If a response is not received within that time frame, ping will consider the packet lost and report it as such.Setting a timeout period with -w can be useful for controlling how long ping should wait for a response before moving on to the next packet. It's often used to adjust the behavior of the ping command based on the network conditions or specific testing requirements.

```
practicalexampc29@LAB306PC32:~$ ping -w6 142.250.182.206
PING 142.250.182.206 (142.250.182.206) 56(84) bytes of data.
64 bytes from 142.250.182.206: icmp_seq=1 ttl=58 time=3.64 ms
64 bytes from 142.250.182.206: icmp_seq=2 ttl=58 time=3.65 ms
```

64 bytes from 142.250.182.206: icmp_seq=3 ttl=58 time=3.58 ms
64 bytes from 142.250.182.206: icmp_seq=4 ttl=58 time=3.30 ms
64 bytes from 142.250.182.206: icmp_seq=5 ttl=58 time=3.41 ms
64 bytes from 142.250.182.206: icmp_seq=6 ttl=58 time=3.53 ms

--- 142.250.182.206 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5009ms
rtt min/avg/max/mdev = 3.303/3.517/3.652/0.125 ms

ping -w6 -c8

The 'ping -w' and 'ping -c' options can be used together to set both a timeout period and a specific count of ICMP echo request packets to send. 'ping -w6 -c8', will send 8 ICMP echo request packets to the target, waiting for a response for up to 6 seconds for each packet. After sending these 8 packets or when the 6-second timeout is reached for each, 'ping' will display a summary of the results, including statistics like packet loss, round-trip time (ping time), and more. This command can be useful for controlled testing and network troubleshooting with a specified timeout and packet count.

practicalexampc29@LAB306PC32:~$ ping -w6 -c8 142.250.182.206
PING 142.250.182.206 (142.250.182.206) 56(84) bytes of data.
64 bytes from 142.250.182.206: icmp_seq=1 ttl=58 time=3.60 ms
64 bytes from 142.250.182.206: icmp_seq=2 ttl=58 time=3.63 ms
64 bytes from 142.250.182.206: icmp_seq=3 ttl=58 time=3.57 ms
64 bytes from 142.250.182.206: icmp_seq=4 ttl=58 time=3.43 ms
64 bytes from 142.250.182.206: icmp_seq=5 ttl=58 time=3.78 ms
64 bytes from 142.250.182.206: icmp_seq=6 ttl=58 time=3.79 ms

--- 142.250.182.206 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5009ms
rtt min/avg/max/mdev = 3.425/3.631/3.785/0.125 ms

traceroute

The 'traceroute' command is used to trace the route that packets take from your computer to a destination host or IP address on a network. It displays a list of all the hops (routers or gateways) that the packets traverse, along with the round-trip times (RTT) for each hop. This tool is essential for network troubleshooting, diagnosing network issues, and understanding the path that data follows across the network.

practicalexampc29@LAB306PC32:~$ traceroute www.google.com
traceroute to www.google.com (142.250.183.196), 30 hops max, 60 byte packets
 1 _gateway (192.168.31.1) 1.039 ms 0.977 ms 0.948 ms
 2 203.212.25.1 (203.212.25.1) 2.229 ms 2.202 ms 2.178 ms
 3 203.212.24.53 (203.212.24.53) 2.153 ms 2.129 ms 2.104 ms
 4 10.10.226.153 (10.10.226.153) 3.313 ms * *
 5 72.14.196.213 (72.14.196.213) 5.647 ms 3.840 ms 5.599 ms
 6 108.170.248.209 (108.170.248.209) 4.392 ms 4.090 ms 4.027 ms
 7 142.251.64.11 (142.251.64.11) 3.420 ms 142.251.64.9 (142.251.64.9) 3.287 ms
3.228 ms
 8 bom07s33-in-f4.1e100.net (142.250.183.196) 3.203 ms 3.179 ms 3.727 ms

nslookup
The 'nslookup' command is used to query Domain Name System (DNS) servers to obtain information about domain names, IP addresses, and other DNS-related records. It is a tool for DNS-related tasks, such as looking up IP addresses associated with domain names (forward lookup) or finding domain names linked to IP addresses (reverse lookup). It's commonly used for DNS troubleshooting and verifying DNS configurations.

```
practicalexampc29@LAB306PC32:~$ nslookup www.google.com
Server:         127.0.0.53
Address:        127.0.0.53#53
Non-authoritative answer:
Name:www.google.com
Address: 142.250.183.196
Name:www.google.com
Address: 2404:6800:4009:826::2004
```

netstat -a
The 'netstat -a' command displays all active network connections and listening ports on a Linux system. It provides information about local and remote IP addresses, port numbers, and connection states, making it useful for network monitoring and troubleshooting. The '-a' option stands for "all" and shows both TCP and UDP connections along with listening ports.

```
practicalexampc29@LAB306PC32:~$ netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 localhost:mysql         0.0.0.0:*               LISTEN
tcp        0      0 localhost:domain        0.0.0.0:*               LISTEN
tcp        0      0 LAB306PC32:domain       0.0.0.0:*               LISTEN
tcp        0      0 localhost:33060         0.0.0.0:*               LISTEN
tcp        0      0 localhost:ipp           0.0.0.0:*               LISTEN
tcp6       0      0 ip6-localhost:ipp       [::]:*                  LISTEN
udp        0      0 LAB306PC32:domain       0.0.0.0:*
udp        0      0 localhost:domain        0.0.0.0:*
udp        0      0 0.0.0.0:bootps          0.0.0.0:*
udp        0      0 LAB306PC32:bootpc       _gateway:bootps         ESTABLISHED
udp        0      0 0.0.0.0:631             0.0.0.0:*
udp        0      0 0.0.0.0:mdns            0.0.0.0:*
udp        0      0 0.0.0.0:43145           0.0.0.0:*
udp6       0      0 [::]:40047              [::]:*
udp6       0      0 [::]:mdns               [::]:*
raw6       0      0 [::]:ipv6-icmp          [::]:*                  7
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ACC ]     STREAM     LISTENING     19696    /run/acpid.socket
unix  2      [ ACC ]     STREAM     LISTENING     19698    /run/avahi-daemon/socket
unix  2      [ ACC ]     STREAM     LISTENING     19702    /run/dbus/system_bus_socket
unix  2      [ ACC ]     STREAM     LISTENING     19704    /run/libvirt/libvirt-sock
unix  2      [ ACC ]     STREAM     LISTENING     19706    /run/uuidd/request
unix  2      [ ACC ]     STREAM     LISTENING     19708    /run/libvirt/virtlockd-sock
```

```
unix  2      [ ACC ]     STREAM     LISTENING     29645    /var/run/mysqld/mysqlx.sock
unix  2      [ ACC ]     STREAM     LISTENING     19710    /run/libvirt/virtlockd-admin-sock
unix  2      [ ACC ]     STREAM     LISTENING     19712    /run/libvirt/virtlogd-sock
unix  2      [ ACC ]     STREAM     LISTENING     29647    /var/run/mysqld/mysqld.sock
unix  2      [ ACC ]     STREAM     LISTENING     31767    /tmp/.X11-unix/X0
unix  2      [ ACC ]     STREAM     LISTENING     19714    /run/libvirt/virtlogd-admin-sock
unix  2      [ ACC ]     STREAM     LISTENING     19719    /run/libvirt/libvirt-admin-sock
unix  2      [ ACC ]     STREAM     LISTENING     19721    /run/libvirt/libvirt-sock-ro
unix  2      [ ACC ]     STREAM     LISTENING     21703
```

netstat -ap

The 'netstat -ap' command displays a list of all active network connections and listening ports on a Linux system, along with the associated process names. It provides information about local and remote IP addresses, port numbers, connection states, and the processes that are using those ports. This command is particularly helpful for identifying which processes are responsible for specific network connections or services running on the system. The '-a' option shows all connections, and the '-p' option displays the associated processes.

```
practicalexampc29@LAB306PC32:~$ netstat -ap
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program
name
tcp        0      0 localhost:mysql         0.0.0.0:*               LISTEN      -
tcp        0      0 localhost:domain        0.0.0.0:*               LISTEN      -
tcp        0      0 LAB306PC32:domain       0.0.0.0:*               LISTEN      -
tcp        0      0 localhost:33060         0.0.0.0:*               LISTEN      -
tcp        0      0 localhost:ipp           0.0.0.0:*               LISTEN      -
tcp6       0      0 ip6-localhost:ipp       [::]:*                  LISTEN      -
udp        0      0 LAB306PC32:domain       0.0.0.0:*                           -
udp        0      0 localhost:domain        0.0.0.0:*                           -
udp        0      0 0.0.0.0:bootps          0.0.0.0:*                           -
udp        0      0 LAB306PC32:bootpc       _gateway:bootps         ESTABLISHED -
udp        0      0 0.0.0.0:631             0.0.0.0:*                           -
udp        0      0 0.0.0.0:mdns            0.0.0.0:*                           -
udp        0      0 0.0.0.0:43145           0.0.0.0:*                           -
udp6       0      0 [::]:40047              [::]:*                              -
udp6       0      0 [::]:mdns               [::]:*                              -
raw6       0      0 [::]:ipv6-icmp          [::]:*                  7           -
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   PID/Program name     Path
unix  2      [ ACC ]     STREAM     LISTENING     19696    -                    /run/acpid.socket
unix  2      [ ACC ]     STREAM     LISTENING     19698    -
/run/avahi-daemon/socket
unix  2      [ ACC ]     STREAM     LISTENING     19702    -
/run/dbus/system_bus_socket
```

```
unix  2      [ ACC ]     STREAM     LISTENING     19704    -
/run/libvirt/libvirt-sock
unix  2      [ ACC ]     STREAM     LISTENING     19706    -
```

netstat -au

The 'netstat -au' command lists all active UDP (User Datagram Protocol) network connections on a Linux system. It displays information about local and remote IP addresses, port numbers, and connection states for UDP-based connections. This command is useful for monitoring and troubleshooting UDP-based network communication. The '-a' option shows all connections, and the '-u' option filters the output to display only UDP connections.

```
practicalexampc29@LAB306PC32:~$ netstat -au
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 LAB306PC32:domain       0.0.0.0:*
udp        0      0 localhost:domain        0.0.0.0:*
udp        0      0 0.0.0.0:bootps          0.0.0.0:*
udp        0      0 LAB306PC32:bootpc       _gateway:bootps         ESTABLISHED
udp        0      0 0.0.0.0:631             0.0.0.0:*
udp        0      0 0.0.0.0:mdns            0.0.0.0:*
udp        0      0 0.0.0.0:43145           0.0.0.0:*
udp6       0      0 [::]:40047              [::]:*
udp6       0      0 [::]:mdns               [::]:*
```

netstat -tnl

The 'netstat -tnl' command displays a list of all listening TCP (Transmission Control Protocol) network ports on a Linux system. It provides information about local IP addresses and port numbers for services that are actively listening for incoming connections. This command is useful for identifying which network services are running and listening for incoming connections. The '-t' option filters the output to show only TCP connections, and the '-n' option displays numerical IP addresses and port numbers instead of resolving them to hostnames and service names.

```
practicalexampc29@LAB306PC32:~$ netstat -tnl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN
tcp        0      0 192.168.122.1:53        0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:33060         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp6       0      0 ::1:631                 :::*                    LISTEN
```

netstat -s

The 'netstat -s' command provides a summary of various network statistics on a Linux system. It displays cumulative statistics for different network protocols, including TCP, UDP, ICMP, and others. This command is valuable for monitoring network performance

and diagnosing network-related issues. It presents detailed information about network errors, packet types, and various protocol-specific statistics, allowing administrators to gain insights into the network's health and performance.

```
practicalexampc29@LAB306PC32:~$ netstat -s
Ip:
    Forwarding: 1
    6684 total packets received
    4 with invalid addresses
    0 forwarded
    0 incoming packets discarded
    6160 incoming packets delivered
    3576 requests sent out
    20 outgoing packets dropped
Icmp:
    403 ICMP messages received
    233 input ICMP message failed
    ICMP input histogram:
        destination unreachable: 68
        timeout in transit: 37
        echo requests: 233
        echo replies: 65
    230 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
        destination unreachable: 48
        echo requests: 182
IcmpMsg:
        InType0: 65
        InType3: 68
        InType8: 233
        InType11: 37
        OutType3: 48
        OutType8: 182
Tcp:
    28 active connection openings
    0 passive connection openings
    6 failed connection attempts
    0 connection resets received
    0 connections established
    4086 segments received
    2745 segments sent out
    18 segments retransmitted
    0 bad segments received
    3 resets sent
Udp:
    788 packets received
    48 packets to unknown port received
    0 packet receive errors
    638 packets sent
```

```
    0 receive buffer errors
    0 send buffer errors
    IgnoredMulti: 811
UdpLite:
TcpExt:
    12 TCP sockets finished time wait in fast timer
    3 delayed acks sent
    Quick ack mode was activated 1 times
    3805 packet headers predicted
    49 acknowledgments not containing data payload received
    8 predicted acknowledgments
    TCPLostRetransmit: 14
    TCPTimeouts: 18
    TCPLossProbes: 2
    TCPDSACKOldSent: 1
    2 connections aborted due to timeout
    TCPRcvCoalesce: 39
    TCPOFOQueue: 13
    TCPAutoCorking: 1
    TCPOrigDataSent: 90
    TCPDelivered: 110
    TCPAckCompressed: 6
    TcpTimeoutRehash: 16
IpExt:
    InNoRoutes: 2
    InMcastPkts: 342
    OutMcastPkts: 95
    InBcastPkts: 1044
    InOctets: 10322907
    OutOctets: 252812
    InMcastOctets: 87492
    OutMcastOctets: 19560
    InBcastOctets: 111165
    InNoECTPkts: 9609
    InECT1Pkts: 3
    InECT0Pkts: 3
MPTcpExt:
```

netstat -rn

The 'netstat -rn' command displays the routing table on a Linux system. It shows the routing information, including the destination network or host, gateway (next hop), and the network interface through which traffic is routed. This command is essential for viewing the current network routing configuration, helping administrators understand how network packets are directed within the system and where they will be forwarded.

```
practicalexampc29@LAB306PC32:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.31.1    0.0.0.0         UG        0 0          0 enp1s0
```

```
169.254.0.0     0.0.0.0         255.255.0.0     U         0 0          0 virbr0
192.168.31.0    0.0.0.0         255.255.255.0   U         0 0          0 enp1s0
192.168.122.0   0.0.0.0         255.255.255.0   U         0 0          0 virbr0
```

netstat -i

The 'netstat -i' command provides a listing of network interfaces on a Linux system along with various statistics associated with each interface. It displays information such as the interface name, packets transmitted and received, errors, drops, and more. This command is useful for monitoring network interface activity, identifying potential network issues, and assessing network performance at a per-interface level.

```
practicalexampc29@LAB306PC32:~$ netstat -i
Kernel Interface table
Iface      MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR
Flg
enp1s0    1500    10774      0      8 0          3304      0      0      0 BMRU
lo       65536      425      0      0 0           425      0      0      0 LRU
virbr0    1500        0      0      0 0             0      0      0      0 BMU
```

route -n

The 'route -n' command displays the routing table in a concise numeric format on a Linux system. It provides information about the network routes, including destination networks or hosts, gateway addresses, network masks, and interface names, all displayed in numerical form (IP addresses and numbers). This command is used to view the routing configuration and can be helpful for understanding how network traffic is routed within the system without hostname resolution.

```
practicalexampc29@LAB306PC32:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.31.1    0.0.0.0         UG    100    0        0 enp1s0
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 virbr0
192.168.31.0    0.0.0.0         255.255.255.0   U     100    0        0 enp1s0
192.168.122.0   0.0.0.0         255.255.255.0   U     0      0        0 virbr0
```

route add default gw

The 'route add default gw' command is used to manually add a default gateway on a Linux system. The "default gateway" is the router or gateway that your Linux system uses to forward network traffic when the destination is outside of your local network. It's the route taken for all non-local traffic. The 'route add' part of the command is used to add a new route to the routing table. The 'default' specifies that this route is the default route, used when no other specific route matches the destination. 'gw' Indicates that you're specifying a gateway address.

```
practicalexampc29@LAB306PC32:~$ sudo su root
root@LAB306PC32:/home/practicalexampc29# route add default gw 192.168.31.253
root@LAB306PC32:/home/practicalexampc29# route -n
Kernel IP routing table
```

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.31.253  0.0.0.0         UG    0      0        0 enp1s0
0.0.0.0         192.168.31.1    0.0.0.0         UG    100    0        0 enp1s0
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 virbr0
192.168.31.0    0.0.0.0         255.255.255.0   U     100    0        0 enp1s0
192.168.122.0   0.0.0.0         255.255.255.0   U     0      0        0 virbr0
```

route add -net

The 'route add -net' command is used to add a specific network route in a Linux system's routing table. It signals the addition of a network route and is followed by the destination network that you want to reach through this route, the network mask for the destination network, and the gateway (next hop) for reaching the specified destination network. Lastly, we specify the network interface (eth0) through which this route should be applied.

```
root@LAB306PC32:/home/practicalexampc29# route add -net 192.168.31.0 netmask
255.255.255.0 gw 192.168.31.253 eth 0
Usage: inet_route [-vF] del {-host|-net} Target[/prefix] [gw Gw] [metric M] [[dev] If]
      inet_route [-vF] add {-host|-net} Target[/prefix] [gw Gw] [metric M]
                        [netmask N] [mss Mss] [window W] [irtt I]
                        [mod] [dyn] [reinstate] [[dev] If]
      inet_route [-vF] add {-host|-net} Target[/prefix] [metric M] reject
      inet_route [-FC] flush      NOT supported
```

arp -a

The 'arp -a' command is used to display the ARP (Address Resolution Protocol) cache table on a Linux system. The ARP cache is a table that stores mappings between IP addresses and MAC (Media Access Control) addresses on your local network. It helps your system quickly resolve IP addresses to MAC addresses, a process necessary for proper network communication. When you run this command, it shows a list of entries in the ARP cache, including the IP addresses and corresponding MAC addresses of devices that your system has recently communicated with. This information is used for efficient data transfer within your local network.

```
root@LAB306PC32:/home/practicalexampc29# arp -a
_gateway (192.168.31.253) at <incomplete> on enp1s0
? (192.168.31.6) at d0:67:e5:1a:23:05 [ether] on enp1s0
? (192.168.31.27) at a4:ae:12:84:80:e1 [ether] on enp1s0
_gateway (192.168.31.1) at 9c:53:22:05:6a:19 [ether] on enp1s0
? (192.168.31.48) at a4:ae:12:84:81:df [ether] on enp1s0
```