🌐Linux Networking & Firewall Management

📘 Scenario Overview

A critical internal web portal became inaccessible, triggering user reports and operational impact. As the assigned **Network Navigator**, I was responsible for performing a structured network diagnosis—from interface status to firewall configuration—to identify the root cause and restore service availability.

This lab simulates **real-world network troubleshooting**, where methodical verification prevents unnecessary changes and downtime.

🎯 Objectives

- Verify network interface availability
- Confirm IP address configuration
- Test external connectivity
- Inspect listening application ports
- Configure firewall rules securely using UFW

🔌 Step 1: Check Network Interface Status

Purpose

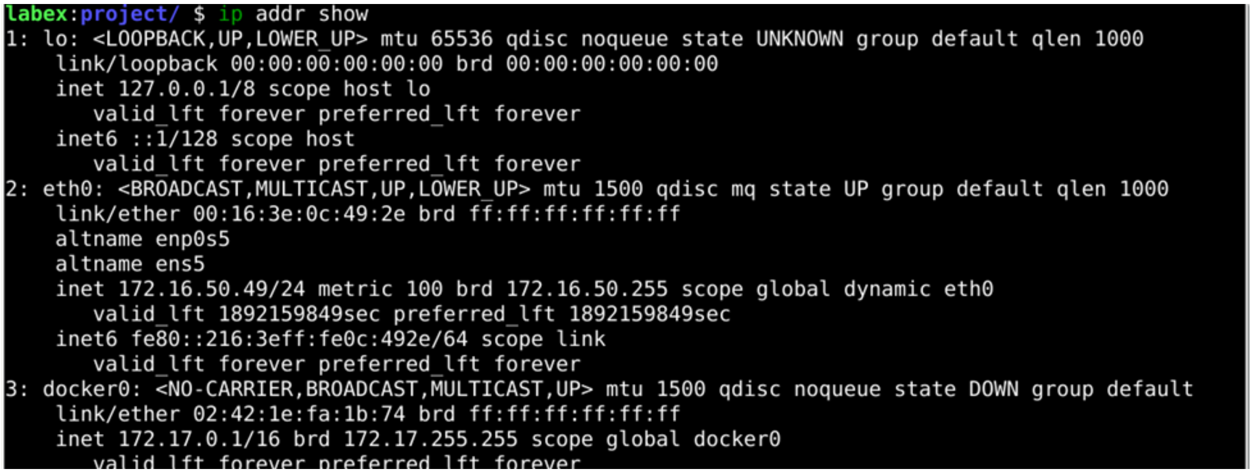
Ensure the system recognizes network hardware and interfaces are active.

Command Used

```
ip addr
```

Result

- Displayed all network interfaces
- Confirmed interface state (`UP`)
- Verified link-layer and IP configuration presence



```
labex:project/ $ 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
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:16:3e:0c:49:2e brd ff:ff:ff:ff:ff:ff
    altname enp0s5
    altname ens5
    inet 172.16.50.49/24 metric 100 brd 172.16.50.255 scope global dynamic eth0
       valid_lft 1892159849sec preferred_lft 1892159849sec
    inet6 fe80::216:3eff:fe0c:492e/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 02:42:1e:fa:1b:74 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
```

🧾 Step 2: Verify IP Address Configuration

Purpose

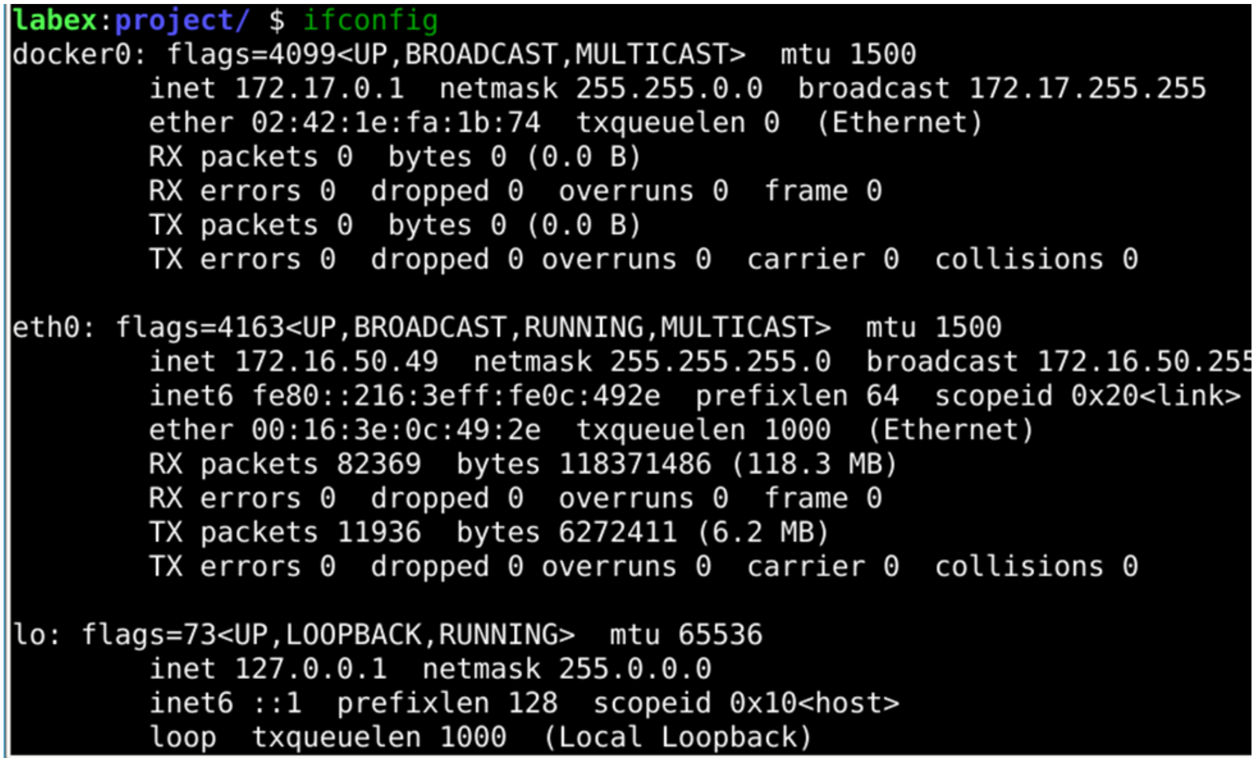
Double-check IP assignment using a classic networking tool.

Command Used

```
ifconfig
```

Why This Matters

- Confirms the interface has a valid IP address
- Provides redundancy by using multiple diagnostic tools
- Useful on legacy and modern systems



```
labex:project/ $ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:1e:fa:1b:74  txqueuelen 0  (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

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.50.49  netmask 255.255.255.0  broadcast 172.16.50.255
        inet6 fe80::216:3eff:fe0c:492e  prefixlen 64  scopeid 0x20<link>
        ether 00:16:3e:0c:49:2e  txqueuelen 1000  (Ethernet)
        RX packets 82369  bytes 118371486 (118.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 11936  bytes 6272411 (6.2 MB)
        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)
```

🌍 Step 3: Test Internet Connectivity

Purpose

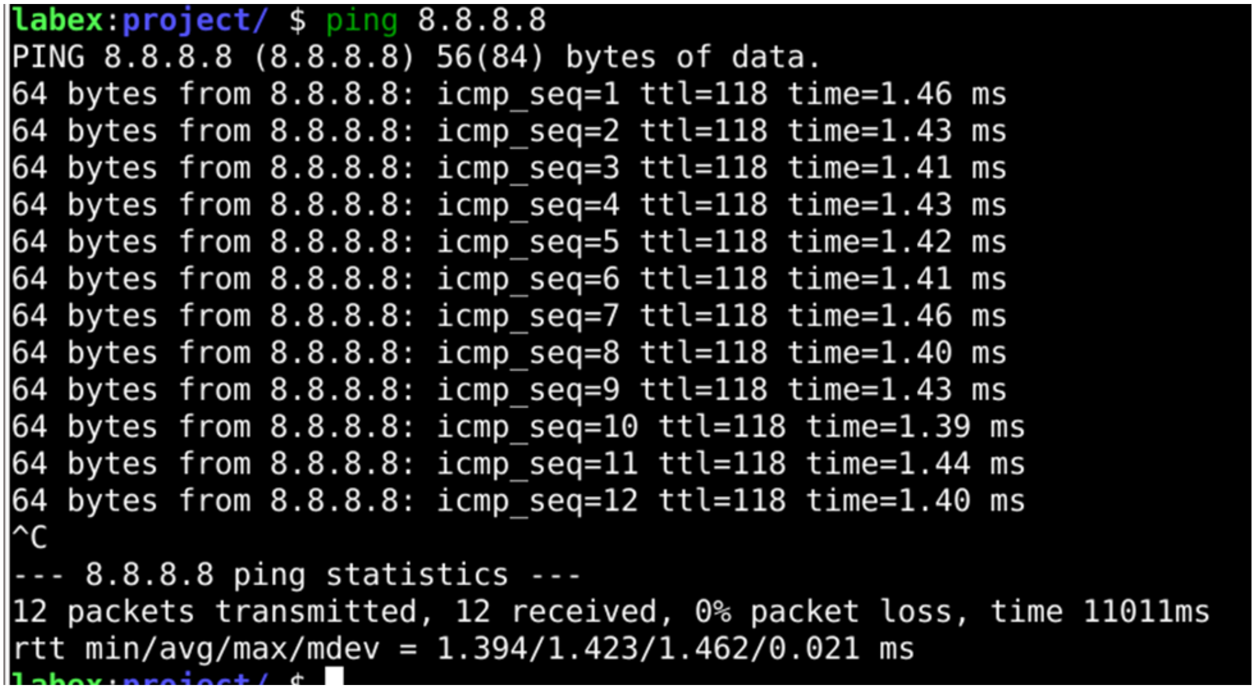
Validate external network reachability and routing.

Command Used

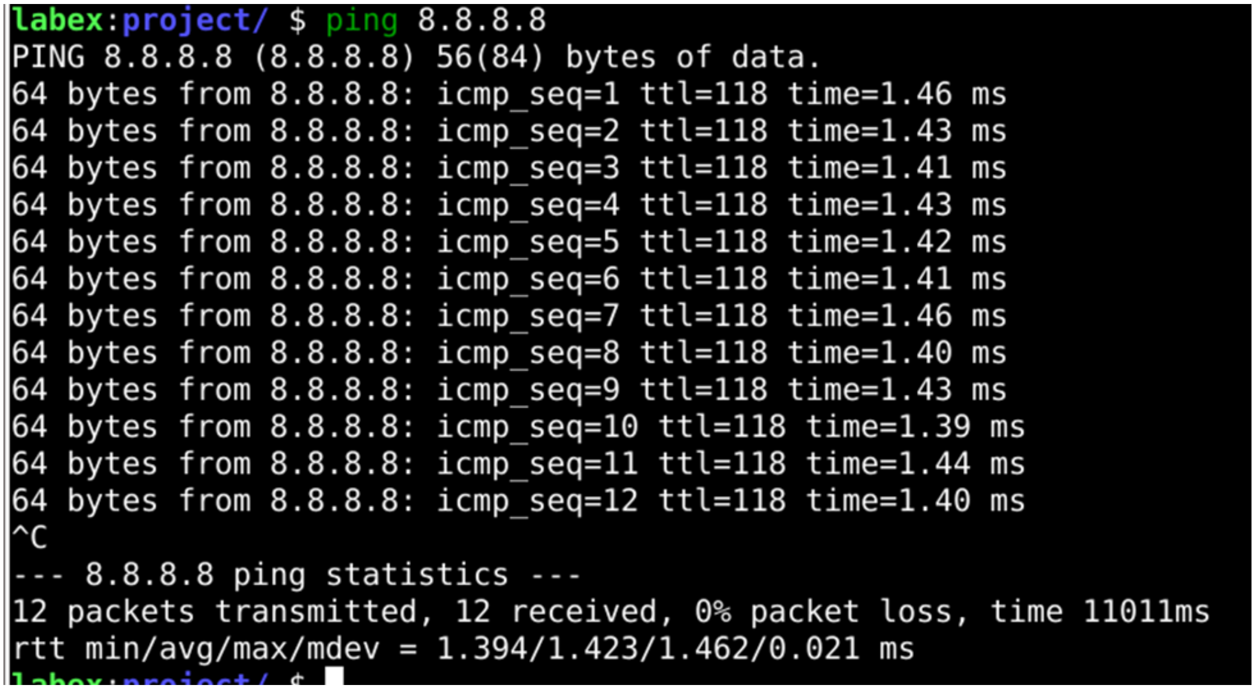
```
ping -c 3 8.8.8.8
```

Result

- Successfully sent exactly 3 ICMP packets
- Confirmed outbound connectivity and gateway functionality
- Eliminated DNS and routing as the issue



```
labex:project/ $ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=118 time=1.46 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=118 time=1.43 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=118 time=1.41 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=118 time=1.43 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=118 time=1.42 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=118 time=1.41 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=118 time=1.46 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=118 time=1.40 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=118 time=1.43 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=118 time=1.39 ms
64 bytes from 8.8.8.8: icmp_seq=11 ttl=118 time=1.44 ms
64 bytes from 8.8.8.8: icmp_seq=12 ttl=118 time=1.40 ms
^C
--- 8.8.8.8 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11011ms
rtt min/avg/max/mdev = 1.394/1.423/1.462/0.021 ms
labex:project/ $
```



```
labex:project/ $ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=118 time=1.46 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=118 time=1.43 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=118 time=1.41 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=118 time=1.43 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=118 time=1.42 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=118 time=1.41 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=118 time=1.46 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=118 time=1.40 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=118 time=1.43 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=118 time=1.39 ms
64 bytes from 8.8.8.8: icmp_seq=11 ttl=118 time=1.44 ms
64 bytes from 8.8.8.8: icmp_seq=12 ttl=118 time=1.40 ms
^C
--- 8.8.8.8 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11011ms
rtt min/avg/max/mdev = 1.394/1.423/1.462/0.021 ms
labex:project/ $
```

🔎 Step 4: Inspect Open Network Ports

Purpose

Confirm whether the internal portal is actively listening for connections.

Requirement

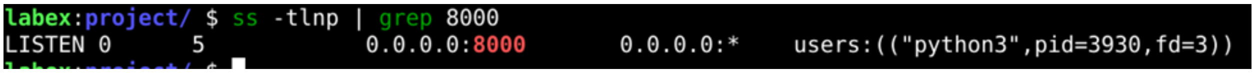
- Portal expected on TCP port **8000**

Command Used

```
ss -ltn | grep :8000
```

Result

- Verified a listening TCP socket on port 8000
- Confirmed application was running correctly



```
labex:project/ $ ss -tlnp | grep 8000
LISTEN 0      5            0.0.0.0:8000       0.0.0.0:*    users:(("python3",pid=3930,fd=3))
labex:project/ $
```

🔥 Step 5: Configure Firewall Rules with UFW

Diagnosis

Since networking and the application were functioning, the firewall was identified as the likely blocker.

Commands Used

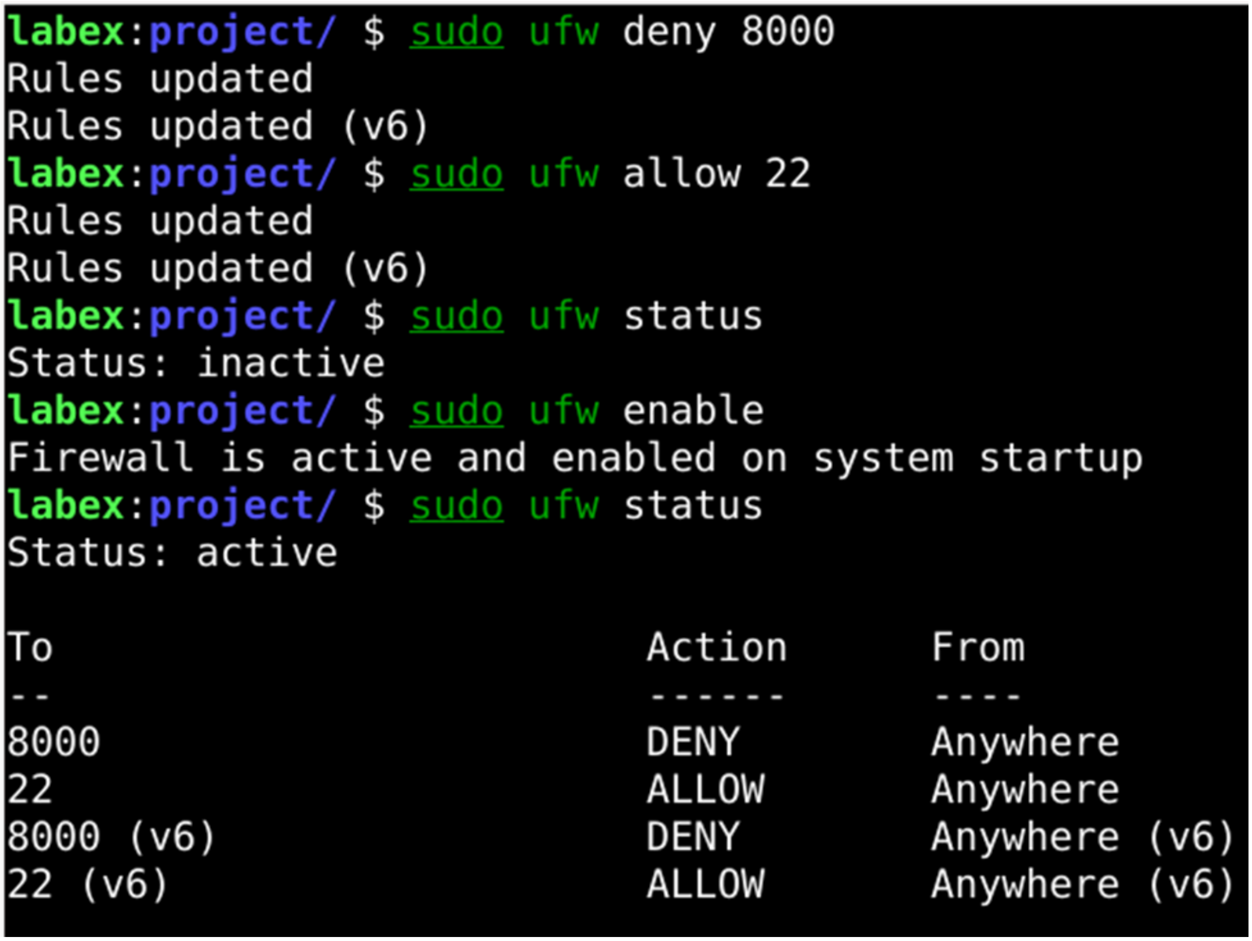
```
sudo ufw deny 8000
sudo ufw allow 22
sudo ufw enable
```

Verification

```
sudo ufw status
```

Outcome

- Incoming traffic on port 8000 denied
- SSH (port 22) explicitly allowed
- Firewall enabled and actively enforcing rules



```
labex:project/ $ sudo ufw deny 8000
Rules updated
Rules updated (v6)
labex:project/ $ sudo ufw allow 22
Rules updated
Rules updated (v6)
labex:project/ $ sudo ufw status
Status: inactive
labex:project/ $ sudo ufw enable
Firewall is active and enabled on system startup
labex:project/ $ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
8000                       DENY        Anywhere
22                         ALLOW       Anywhere
8000 (v6)                  DENY        Anywhere (v6)
22 (v6)                    ALLOW       Anywhere (v6)
```

🧠 Skills Demonstrated

- Network interface inspection (`ip addr`)
- IP configuration verification (`ifconfig`)
- Connectivity testing (`ping`)
- Port and socket inspection (`ss`)
- Firewall configuration with UFW
- Structured troubleshooting methodology
- Secure access control implementation