Day3_Materials_EN

1. General Info

Date: 21.08.2025

Topic: Introduction to networking concepts.

Daily goal: Learn basic networking concepts and commands for diagnosing and configuring networks in Linux.

| Commands: `ifconfig`, `ip addr`, `ip link`, `ip route` |
| Connectivity testing commands: `ping`, `traceroute`, `curl`, `wget` |
| Working with `/etc/hosts` and checking DNS via `dig`, `nslookup` |
| Mini-lab: network diagnostics in Ubuntu |

---

2. Warm-up

1. Navigation

```
leprecha@Ubuntu-DevOps:~$ pwd
/home/leprecha
leprecha@Ubuntu-DevOps:~$ ls -l
drwxr-xr-x 4 leprecha sysadmin 4096 Aug 21 17:38 Desktop
drwxr-xr-x 5 leprecha sysadmin 4096 Aug 20 20:28 DevOps
drwxr-xr-x 2 leprecha sysadmin 4096 Aug 20 20:17 Documents
```

2. Working with files

```
leprecha@Ubuntu-DevOps:~$ touch file.txt
leprecha@Ubuntu-DevOps:~$ cp file.txt copy.txt
leprecha@Ubuntu-DevOps:~$ mv copy.txt moved.txt
```

```
leprecha@Ubuntu-DevOps:~$ rm moved.txt
leprecha@Ubuntu-DevOps:~$ ls -l
-rw-r--r-- 1 leprecha sysadmin    0 Aug 21 20:56 file.txt
```

3. Permissions

```
leprecha@Ubuntu-DevOps:~$ chmod 644 file.txt
leprecha@Ubuntu-DevOps:~$ touch script.sh
leprecha@Ubuntu-DevOps:~$ chmod +x script.sh
leprecha@Ubuntu-DevOps:~$ sudo chown helpme file.txt
[sudo] password for leprecha:
leprecha@Ubuntu-DevOps:~$ ls -l
-rw-r--r-- 1 helpme   sysadmin    0 Aug 21 20:56 file.txt
-rwxr-xr-x 1 leprecha sysadmin    0 Aug 21 20:58 script.sh
```

3. Introduction to Networking Concepts

1. What is an IP Address

- **IPv4** — 4 numbers from 0 to 255 (example: `192.168.0.1`), 4.3 billion addresses.
- **IPv6** — long hexadecimal addresses (example: `2001:0db8::1`), trillions of addresses.
- Each network card (interface) can have one or more IP addresses.

2. Local and Global Addresses

- **Local (private)** — used inside networks, not visible from the internet:
  - `192.168.x.x`
  - `10.x.x.x`
  - `172.16.x.x` — `172.31.x.x`
- **Global (public)** — visible from the internet.

3. DNS (Domain Name System)

- Translates domain names (e.g., `google.com`) into IP addresses.
- `/etc/hosts` — a local file for manually defining mappings.

---

4. Routing

- A route is the path that packets take.
- A device maintains a routing table.

How it works step by step:

1. PC → Router (192.168.0.1)

- Your computer on the local network has a **private IP** (e.g., `192.168.0.42`).
- When you type `google.com` in the browser, the computer **doesn't know its IP yet** —only the name.
- The first thing the PC does is check:
  1. Whether the IP is in the DNS cache (operating system, browser).
  2. If not — sends a DNS query to the **DNS server** specified in the network settings.
- The packet with this query goes to your **router** (`192.168.0.1`), because it's your "gateway to the internet."

---

2. Router → Internet (Public IP)

- The router also has two IPs:
  - **LAN IP** (local) — `192.168.0.1`.
  - **WAN IP** (public, e.g., `93.184.216.34`).
- When your request goes out to the internet, the router performs **NAT** (Network Address Translation):
  - Replaces your private IP (`192.168.0.42`) with its public one.
  - Remembers that the reply needs to be returned specifically to your PC.

- The request then heads toward the ISP and further across the network.

3. Internet → DNS server

- The DNS query reaches a **DNS server** (most often the provider's, or for example Google DNS `8.8.8.8`).
- The DNS server looks up an IP address for the name `youtube.com`:
    1. First checks its own cache.
    2. If not found—asks other DNS servers (root, then the authoritative ones for the domain).
- In the end the server replies:
  `youtube.com → 173.194.69.91`

4. DNS server → Website

- Now your browser knows the site's IP and sends an HTTP/HTTPS request **to that IP**.
- The request again goes:
    - Through your router (NAT).
    - Through your ISP's network.
    - Across multiple routers on the internet (the route may be 5–20 hops).
- At IP `173.194.69.91` there's a web server (e.g., Apache or Nginx).
- It receives the request, processes it, and sends back HTML, CSS, images, etc.

Command Breakdown

`ip addr` — the modern replacement.

```
leprecha@Ubuntu-DevOps:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: enp44s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_
codel state DOWN group default qlen 1000
    link/ether e7:1b:25:52:22:7q brd ff:ff:ff:ff:ff:ff
3: wlo1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UP group default qlen 1000
    link/ether a4:0d:34:e2:4c:24 brd ff:ff:ff:ff:ff:ff
    altname wqp0s40n3
    inet 192.168.1.12/24 brd 192.168.1.255 scope global dynamic noprefixroute wl
o1
       valid_lft 247009sec preferred_lft 247009sec
    inet6 2002:bb5:a3c:7000:8fib:baw1:7f10:6a1d/64 scope global temporary d
ynamic
       valid_lft 3582sec preferred_lft 3582sec
    inet6 2002:bb5:a3c:7000:8fib:baw1:a67e:866f/64 scope global dynamic m
ngtmpaddr noprefixroute
       valid_lft 3582sec preferred_lft 3582sec
    inet6 fe84::d07:7dn3:941c:6b02/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

`ip link` — List of interfaces without IPs, only their state.

```


leprecha@Ubuntu-DevOps:~$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOW
N mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp44s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_
codel state DOWN mode DEFAULT group default qlen 1000
    link/ether e3:9c:45:72:21:7e brd ff:ff:ff:ff:ff:ff
3: wlo1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UP mode DORMANT group default qlen 1000
```

```
    link/ether e4:2d:56:e5:3f:14 brd ff:ff:ff:ff:ff:ff
    altname wlp0s30f5
```

`ip route` — Routing table, a “map” of where packets are sent.

```
leprecha@Ubuntu-DevOps:~$ ip route
default via 192.168.1.254 dev wlo1 proto dhcp src 192.168.1.12 metric 600
192.168.1.0/24 dev wlo1 proto kernel scope link src 192.168.1.12 metric 600
```

Connectivity Check Commands

`ping` — Checks if a host is reachable.

- `c 4` — send 4 packets, otherwise it will keep pinging indefinitely.

```
leprecha@Ubuntu-DevOps:~$ ping -c 4 google.com
PING google.com (2a00:1450:400b:c02::8a) 56 data bytes
64 bytes from dj-in-f138.1e100.net (2a00:1450:400b:c02::8a): icmp_seq=1 ttl=110 time=9.29 ms
64 bytes from dj-in-f138.1e100.net (2a00:1450:400b:c02::8a): icmp_seq=2 ttl=110 time=66.5 ms
64 bytes from dj-in-f138.1e100.net (2a00:1450:400b:c02::8a): icmp_seq=3 ttl=110 time=8.99 ms
64 bytes from dj-in-f138.1e100.net (2a00:1450:400b:c02::8a): icmp_seq=4 ttl=110 time=9.22 ms

--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 8.987/23.510/66.545/24.846 ms
```

`traceroute` — Shows the path (routers) that packets take to reach the destination.

```
leprecha@Ubuntu-DevOps:~$ traceroute google.com
traceroute to google.com (209.85.203.139), 30 hops max, 60 byte packets
 1  MyRouter.home (192.168.1.254)  5.063 ms  5.166 ms  5.291 ms
 2  95-44-248-1-dynamic.agg2.lky.bge-rtd.eircom.net (95.44.248.1)  6.695 ms  6.863 ms  7.054 ms
 3  lag-6.agg3.lky.bge-rtd.eircom.net (86.47.61.112)  7.300 ms  7.364 ms  8.860 ms
 4  eth-trunk107.hcore1.bge.core.eircom.net (86.43.59.104)  11.835 ms  13.258 ms  13.236 ms
 5  eth-trunk18.hcore1.bdt.core.eircom.net (86.43.12.253)  18.342 ms  18.539 ms  18.514 ms
 6  * * *
 7  * * *
 8  * * *
 9  209.85.244.230 (209.85.244.230)  9.225 ms 209.85.143.80 (209.85.143.80)  10.396 ms 209.85.243.216 (209.85.243.216)  8.827 ms
10  192.178.107.66 (192.178.107.66)  9.029 ms 192.178.107.96 (192.178.107.96)  10.695 ms 192.178.107.64 (192.178.107.64)  9.117 ms
11  72.14.236.167 (72.14.236.167)  9.314 ms 172.253.70.249 (172.253.70.249)  9.067 ms  7.824 ms
12  172.253.71.163 (172.253.71.163)  9.555 ms 172.253.71.160 (172.253.71.160)  8.025 ms 172.253.71.82 (172.253.71.82)  9.465 ms
13  209.85.142.137 (209.85.142.137)  9.191 ms 209.85.253.175 (209.85.253.175)  7.227 ms 172.253.69.127 (172.253.69.127)  8.069 ms
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  dh-in-f139.1e100.net (209.85.203.139)  9.791 ms  9.586 ms  9.768 ms
```

```
#The network is working, the route to Google is correct.
#Latency is low (7–18 ms).
#Asterisks * = nodes not responding, but traffic continues.
#Everything after the 9th hop belongs to the Google network.
```

`curl` — Downloads the contents of a webpage into the terminal.

```
leprecha@Ubuntu-DevOps:~$ curl https://google.com
<HTML><HEAD><meta http-equiv="content-type" content="text/html;charse
t=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="https://www.google.com/">here</A>.
</BODY></HTML>

# curl -I — returns only the headers without the body.
```

`wget` — Downloads a file.

```
leprecha@Ubuntu-DevOps:~$ wget http://speedtest.tele2.net/1MB.zip
--2025-08-21 21:19:13--  http://speedtest.tele2.net/1MB.zip
Resolving speedtest.tele2.net (speedtest.tele2.net)... 2a00:800:1010::1, 90.13
0.70.73
Connecting to speedtest.tele2.net (speedtest.tele2.net)|2a00:800:1010::1|:8
0... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1048576 (1.0M) [application/zip]
Saving to: ‘1MB.zip’

1MB.zip             100%[====================================⇒]
1.00M  2.50MB/s    in 0.4s

2025-08-21 21:19:14 (2.50 MB/s) - ‘1MB.zip’ saved [1048576/1048576]
```

```
#wget --spider — doesn’t download the file, just checks if the URL is accessible.
```

Working with `/etc/hosts` and checking DNS using `dig`, `nslookup`

1. `/etc/hosts` file

```
leprecha@Ubuntu-DevOps:~$ cat /etc/hosts
127.0.0.1 localhost
127.0.1.1 Ubuntu-DevOps

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Example — let’s add a test entry: `echo "1.2.3.4 mytest.local"`

```
Outputs the line: "1.2.3.4 mytest.local"
The format is: IP address domain\_name

leprecha@Ubuntu-DevOps:~$ echo "1.2.3.4 mytest.local"
1.2.3.4 mytest.local
```

> **`sudo tee -a /etc/hosts`**

- `|` — passes the output of the `echo` command to the next command (`tee`).
- `sudo` — runs `tee` as administrator, since `/etc/hosts` is a system file.
- `tee` — takes a string from the pipe and writes it to a file.

- `-a` — **append** (adds to the end of the file without overwriting).
- `/etc/hosts` — local file that the system checks before making DNS queries.

```
sysadmin@Ubuntu-DevOps:~$ echo "1.2.3.4 mytest.local" | sudo tee -a  /etc/hosts
[sudo] password for sysadmin:
1.2.3.4 mytest.local
leprecha@Ubuntu-DevOps:~$ ping -c 4 mytest.local
PING mytest.local (1.2.3.4) 56(84) bytes of data.

--- mytest.local ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3096ms
```

This is useful for:

- Testing websites before their DNS records are configured.
- Mapping a domain to a different IP (e.g., a local server).
- Blocking domains (by pointing them to `127.0.0.1`).

---

2. `dig` — detailed DNS query

`dig` is a utility for DNS queries (**Domain Information Groper**).

`dig` sends a request to a DNS server and shows a detailed response:

- which IP is associated with the domain (A/AAAA records),
- which DNS server is authoritative for the domain,
- record time-to-live (TTL),
- the full resolution path.

```
leprecha@Ubuntu-DevOps:~$ dig google.com

; <<>> DiG 9.18.30-0ubuntu0.24.04.2-Ubuntu <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13475
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;google.com.          IN    A

;; ANSWER SECTION:
google.com.      297  IN    A  209.85.203.138
google.com.      297  IN    A  209.85.203.101
google.com.      297  IN    A  209.85.203.102
google.com.      297  IN    A  209.85.203.139
google.com.      297  IN    A  209.85.203.113
google.com.      297  IN    A  209.85.203.100

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Thu Aug 21 21:32:22 IST 2025
;; MSG SIZE  rcvd: 135
```

```
leprecha@Ubuntu-DevOps:~$ dig +short
f.root-servers.net.
a.root-servers.net.
e.root-servers.net.

# Only IPs (**A records**)
```

```
leprecha@Ubuntu-DevOps:~$ dig google.com MX
;; ANSWER SECTION:
google.com.      3600   IN    MX  10 smtp.google.com.

# MX records (mail servers)
```

```
leprecha@Ubuntu-DevOps:~$ dig google.com NS
;; ANSWER SECTION:
google.com.      128655    IN    NS   ns2.google.com.

# NS records (domain name servers).
```

---

3. `nslookup` — simple DNS query (Name Server Lookup)

- Finds the IP of a domain (**A** or **AAAA** record).
- Finds the domain from an IP (reverse lookup).
- Can query a specific DNS server.

```
leprecha@Ubuntu-DevOps:~$ nslookup google.com
Server:       127.0.0.53
Address: 127.0.0.53#53

Non-authoritative answer:
Name: google.com
Address: 172.253.116.100
Name: google.com
Address: 172.253.116.101
Name: google.com
Address: 172.253.116.113
```

```
# nslookup is simpler and shorter, but provides fewer details.
```

Practice

Execute commands to view network configuration, test connections, and diagnose issues.