🧭 Upgrading your network tools Workshop

Workshop-style deck (vertical stacks for deep dives, now including `netstat` → `ss`)

Agenda

- Why move to `ip`
- Core syntax & mental model
- Old → new command mapping
- Hands-on: interfaces, addresses, routes, neighbors
- From `netstat` → `ss`
- Scripting & JSON
- Advanced: policy routing, VRFs/tunnels
- Migration tips + cheat sheet

Why move to `ip` — Overview

- Legacy tools: `ifconfig`, `route`, `arp`, `netstat`
- Fragmented features & syntax
- Weak/awkward IPv6 support
- `iproute2` unifies & modernizes

--

Why move to `ip` — Pain points with legacy

- Different flags & output formats
- Aliases (eth0:1) vs real multi-addressing
- Some tools unmaintained
- Parsing output is brittle

--

Why move to `ip` — What you gain

- One consistent CLI: `ip [object] [command]`
- First-class IPv6
- JSON output (`ip -j`) for automation
- Policy routing, VRFs, advanced features

What is `iproute2` ?

- Suite that replaces several legacy tools
- Ships on all modern distros
- Key binaries: `ip` , `ss` , `tc`
- Man pages: `man 8 ip` , `man 8 ss`

--

Replace this with that

- `ifconfig` → `ip addr` , `ip link`
- `route` → `ip route`
- `arp` → `ip neigh`
- `netstat` → `ss`

Core syntax & mental model

```
ip [OBJECT] [COMMAND] [OPTIONS]
```

- Objects: `link` , `addr` , `route` , `neigh` , `rule` , `maddr` , `tunnel` , ...
- Commands: `show` , `add` , `del` , `flush` , `get` , `set`
- Options: `dev` , `via` , `src` , `metric` , `table` , ...

--

Common examples

```
ip addr show
ip link set eth0 up
ip route add 10.0.0.0/24 via 192.168.1.1
ip neigh show
```

From `netstat` → `ss` — Modern Socket Inspection

Why replace `netstat`?

- `netstat` is deprecated and slow — parses `/proc` directly.
- Limited IPv6 visibility.
- `ss` (socket statistics) is part of `iproute2` and far more capable.

--

Common equivalents

`netstat`	`ss`
`netstat -tuln`	`ss -tuln`
`netstat -tanp`	`ss -tanp`
`netstat -rn`	`ip route show`
`netstat -s`	`ss -s`

--

Hands-on basics

```
ss -tuln          # TCP/UDP listening ports
ss -t state estab # Active connections
ss -uap           # UDP + processes
```

💡 Use filters like `state`, `src`, `dst`, `dport` for precision.

--

Filtering examples

```
ss -t state listening
ss -u src 192.168.1.10
ss -t dst 10.0.0.5:22
ss -tan 'sport = :443' | grep ESTAB
```

--

Statistics & monitoring

```
ss -s
```

Shows TCP/UDP totals, retransmits, queue sizes. Use with `watch -n1 ss -s` to view socket dynamics live.

--

Practical exercise

1. Run `ss -tuln` on your system.
2. Open a browser, then run `ss -t state estab`.
3. Identify your outbound web connection and process.
4. Bonus: Try `ss -tanp | grep :443`.

Interfaces — inspect (basic)

```
ip link show
ip -brief link
ip addr show
ip -brief addr
```

--

Interfaces — control

```
ip link set eth0 up
ip link set eth0 down
ip link set dev eth0 mtu 1400
ip link set eth0 address 02:11:22:33:44:55
ip link set eth0 name lan0
```

Scripting — stable text & JSON

- Predictable text output (`-brief`)
- JSON for tooling:

  ```
  ip -j addr show | jq .
  ss -j state established | jq .
  ```

- Great for CI, health checks, automation pipelines

Quick cheat sheet

```
ip -brief addr         # IP summary
ip -brief link         # Link summary
ip route show          # Routes
ip route get <IP>      # Path selection
ip neigh show          # ARP/ND
ss -tuln               # Listening sockets
ss -s                  # Socket statistics
```

Wrap-up

- Unified toolset: `ip` + `ss`
- Consistent syntax, IPv6-first
- Scriptable (JSON), automation-ready
- Ready for advanced routing and diagnostics