Table of Contents

01	`mtr`	Combine ping and traceroute in a single network diagnostic tool
02	`nc`	Networking utility for reading/writing across network connections
03	`netstat`	Print network connections, routing tables, and stats (legacy)
04	`nmap`	Network exploration tool and security scanner
05	`nmcli`	NetworkManager command-line client
06	`nslookup`	Query DNS name servers interactively
07	`ping`	Send ICMP echo requests to test network connectivity
08	`resolvectl`	DNS resolution and systemd-resolved management
09	`ss`	Show socket statistics (modern replacement for netstat)
10	`ss -tulnp`	Show listening TCP/UDP ports with process info

$ mtr

Intermediate

Combine ping and traceroute in a single network diagnostic tool

mtr (My Traceroute) combines the functionality of traceroute and ping into a single real-time diagnostic tool. It continuously probes each hop in the network path, showing packet loss and latency statistics.

mtr is far superior to traceroute for diagnosing network issues because it provides ongo...

Options & Flags

`-r`	Report mode (text output, then exit)
`-c`	Number of pings per hop
`-n`	No DNS resolution
`-w`	Wide report (show IPs and hostnames)
`-T`	Use TCP instead of ICMP
`-4/-6`	Force IPv4 or IPv6

Practical Examples

Example: Interactive trace

```
$ mtr example.com
```

Opens interactive display with real-time hop statistics.

Example: Report mode

```
$ mtr -r -c 100 example.com
HOST           Loss%   Snt   Avg  Best  Wrst
1. gateway      0.0%   100   1.2   0.8   3.4
2. isp-router   0.5%   100   5.6   4.2   12.1
```

Sends 100 pings per hop and outputs a text report.

Example: Wide report

```
$ mtr -rw -c 50 8.8.8.8
```

Wide report showing both hostnames and IPs.

Example: TCP mode

```
$ sudo mtr -T -P 443 example.com
```

Uses TCP port 443 - works through firewalls that block ICMP.

Example: No DNS

```
$ mtr -rn -c 30 example.com
```

Fast report without DNS resolution.

Tips & Best Practices

Pro Tip: Look at loss patterns: Loss at one hop but not the next is usually harmless (router deprioritizes ICMP). Loss that persists to the destination is real.

Note: Report for support tickets: mtr -rw -c 100 host gives a comprehensive report. Paste this when reporting network issues to your ISP or hosting provider.

Warning: Requires root for some modes: ICMP mode (default) often needs root. TCP mode (-T) always needs root. Use sudo mtr.

$ nc

Advanced

Networking utility for reading/writing across network connections

nc (netcat) is a versatile networking utility that reads and writes data across network connections. It can create TCP/UDP connections, listen for connections, port scan, transfer files, and serve as a network debugging tool.

nc is often called the "Swiss Army knife" of networking. It can functi...

Options & Flags

`-l`	Listen mode (server)
`-v`	Verbose output
`-z`	Scan mode (zero I/O, for port scanning)
`-w`	Timeout in seconds
`-u`	UDP mode
`-k`	Keep listening after client disconnects

Practical Examples

Example: Check if port is open

```
$ nc -zv example.com 443
Connection to example.com 443 port [tcp/https] succeeded!
```

Tests if port 443 is open on the remote host.

Example: Scan port range

```
$ nc -zv example.com 20-25
```

Scans ports 20-25 on the remote host.

Example: Simple chat server

```
$ nc -l 9000
```

Listens on port 9000 - anything typed on the other end appears here.

Example: Connect to chat

```
$ nc hostname 9000
```

Connects to the listening nc and enables text exchange.

Example: File transfer (receiver)

```
$ nc -l 9000 > received_file.tar.gz
```

Receives a file on port 9000.

Tips & Best Practices

Pro Tip: Quick port check: nc -zv host port is the fastest way to test if a remote port is open. Faster and simpler than telnet.

Warning: ncat vs nc variants: There are several nc variants: GNU netcat, OpenBSD netcat, ncat (from nmap). Options vary. Check nc -h for your version.

Note: Security tool: nc can create reverse shells and transfer data. It is a legitimate admin tool but can be used maliciously - use responsibly.

$ netstat

Intermediate

Print network connections, routing tables, and stats (legacy)

netstat displays network connections, routing tables, interface statistics, and protocol information. While deprecated in favor of ss, netstat remains widely used and available on most systems.

netstat is commonly used to check which ports are open, what processes are listening, and to verify ne...

Options & Flags

`-t`	Show TCP connections
`-u`	Show UDP connections
`-l`	Show only listening sockets
`-n`	Show numerical addresses (skip DNS resolution)
`-p`	Show process ID and name
`-r`	Show routing table
`-i`	Show network interface statistics
`-s`	Show protocol statistics

Practical Examples

Example: Show listening ports

```
$ sudo netstat -tulnp
Proto Local Address    State    PID/Program
tcp   0.0.0.0:22       LISTEN   1234/sshd
tcp   0.0.0.0:80       LISTEN   5678/nginx
```

The most common usage - shows all TCP/UDP listening ports with process names.

Example: Show all connections

```
$ netstat -an
```

Shows all connections with numerical addresses.

Example: Show routing table

```
$ netstat -rn
```

Displays the kernel routing table with IP addresses (not hostnames).

Example: Count connections by state

```
$ netstat -ant | awk '{print $6}' | sort | uniq -c | sort -rn
  145 ESTABLISHED\n   23 TIME_WAIT\n    5 LISTEN
```

Shows a summary of connection states (ESTABLISHED, TIME_WAIT, etc.).

Example: Find process using a port

```
$ sudo netstat -tlnp | grep :80
tcp  0.0.0.0:80  LISTEN  5678/nginx
```

Shows which process is listening on port 80.

Tips & Best Practices

Pro Tip: Use ss instead: ss is faster and more feature-rich than netstat. Equivalent command: ss -tulnp instead of netstat -tulnp.

Note: Requires sudo for -p: The -p flag (show process names) requires root privileges to see processes owned by other users.

Warning: Deprecated command: netstat is deprecated in favor of ss. It may not be installed by default on newer distributions. Install with: apt install net-tools.

$ nmap

Advanced

Network exploration tool and security scanner

nmap (Network Mapper) is the industry-standard network discovery and security auditing tool. It discovers hosts, services, operating systems, and vulnerabilities on a network.

nmap sends specially crafted packets and analyzes responses to determine what hosts are available, what services they ru...

Options & Flags

`-sS`	TCP SYN scan (stealth, default)
`-sV`	Version detection
`-O`	OS detection
`-p`	Specify ports
`-A`	Aggressive scan (OS, version, scripts, traceroute)
`-sn`	Ping scan (host discovery only)
`-Pn`	Skip host discovery (treat as online)
`--top-ports`	Scan most common N ports

Practical Examples

Example: Quick scan

```
$ nmap example.com
PORT    STATE  SERVICE
22/tcp  open   ssh
80/tcp  open   http
443/tcp open   https
```

Scans the 1000 most common ports.

Example: Scan specific ports

```
$ nmap -p 22,80,443,3306,5432 server.com
```

Checks only specific ports.

Example: Service versions

```
$ nmap -sV example.com
80/tcp open http nginx 1.24.0\n443/tcp open ssl/http nginx 1.24.0
```

Detects service versions running on open ports.

Example: Network discovery

```
$ nmap -sn 192.168.1.0/24
```

Finds all active hosts on the local network.

Example: Full scan

```
$ sudo nmap -A -T4 target.com
```

Aggressive scan: OS, versions, scripts, and traceroute.

Tips & Best Practices

Warning: Get permission first: Scanning networks without permission is illegal in many jurisdictions. Only scan systems you own or have written authorization to test.

Pro Tip: -T4 for speed: Add -T4 for faster scanning. Timing levels: T0 (paranoid) to T5 (insane). T4 is good for most uses.

Note: NSE scripts: nmap includes scripting engine (NSE) with hundreds of scripts: --script vuln for vulnerability detection, --script http-enum for web enumeration.

$ nmcli

Intermediate

NetworkManager command-line client

The nmcli command is the primary command-line tool for controlling NetworkManager - the default network management daemon on most modern Linux distributions including Ubuntu, Fedora, RHEL, AlmaLinux, and CentOS Stream.

nmcli allows you to create, display, edit, delete, activate, and deactivate n...

Options & Flags

`general status`	Show overall NetworkManager status
`device status`	List all network devices and their state
`connection show`	List all connections (active and inactive)
`connection show --active`	List only active connections
`connection up NAME`	Activate a connection by name
`connection down NAME`	Deactivate a connection
`connection add`	Create a new connection profile
`connection modify`	Modify an existing connection
`connection delete`	Delete a connection profile
`-t`	Terse output for scripting (colon-separated)

Practical Examples

Example: Show all connections

```
$ nmcli connection show
NAME       UUID                                  TYPE      DEVICE\nWired 1    a1b2c3d4-e5f6-7890-abcd-ef1234567890  ethernet  eth0
```

Lists all configured connections with name, UUID, type, and device.

Example: Set static IP address

```
$ nmcli connection modify "Wired 1" ipv4.addresses 192.168.1.100/24 ipv4.gateway 192.168.1.1 ipv4.dns "1.1.1.1,8.8.8.8" ipv4.metho
```

Configure static IP, gateway, and DNS on an existing connection. Apply with: nmcli connection up "Wired 1"

Example: Switch to DHCP

```
$ nmcli connection modify "Wired 1" ipv4.method auto ipv4.addresses "" ipv4.gateway "" ipv4.dns ""
```

Reset connection to automatic (DHCP) configuration.

Example: Connect to WiFi

```
$ nmcli device wifi connect "MyNetwork" password "MyPassword"
```

Connect to a WiFi network. The connection profile is created and activated automatically.

Example: List available WiFi networks

```
$ nmcli device wifi list
```

Scan and display nearby WiFi access points with signal strength, security, and channel.

Tips & Best Practices

Pro Tip: Apply changes after modify: nmcli connection modify only saves the profile. You must run nmcli connection up "name" to apply the changes to the active connection.

Note: Tab completion: nmcli supports tab completion in bash/zsh. Type nmcli con<TAB> to auto-complete. Install bash-completion package if not working.

Warning: Connection name vs device name: Connection names and device names are different. A device (eth0) can have multiple connection profiles. Use nmcli connection show to see which profile is active on which device.

Pro Tip: Use -f for specific fields: Filter output fields: nmcli -f NAME,TYPE,DEVICE connection show - cleaner output for scripts and monitoring.

$ nslookup

Beginner

Query DNS name servers interactively

nslookup queries DNS name servers to find the IP address of a domain or the domain name for an IP address. It is one of the most commonly used DNS troubleshooting tools.

nslookup can query specific record types (A, AAAA, MX, TXT, CNAME, NS, SOA) and use specific DNS servers. It operates in inter...

Options & Flags

`domain`	Look up domain A record
`-type=`	Specify record type
`domain server`	Use specific DNS server
`-debug`	Show detailed query info

Practical Examples

Example: Look up domain

```
$ nslookup example.com
Server:  8.8.8.8
Name:    example.com
Address: 93.184.216.34
```

Finds the IP address for a domain.

Example: MX records

```
$ nslookup -type=MX example.com
example.com  mail exchanger = 10 mail.example.com
```

Finds mail server records for the domain.

Example: TXT records

```
$ nslookup -type=TXT example.com
```

Shows TXT records (SPF, DKIM, verification).

Example: Use specific DNS server

```
$ nslookup example.com 8.8.8.8
```

Queries Google DNS instead of your default server.

Example: Reverse lookup

```
$ nslookup 8.8.8.8
8.8.8.8.in-addr.arpa  name = dns.google
```

Finds the domain name for an IP address.

Tips & Best Practices

Pro Tip: Test different DNS servers: Compare: nslookup domain 8.8.8.8 vs nslookup domain 1.1.1.1. Different results may indicate DNS propagation delays.

Note: dig is more detailed: dig provides more detailed output with TTLs and authority info. nslookup is simpler for quick lookups.

Warning: Deprecated notice: nslookup shows a deprecation warning on some systems. It still works fine, but dig and host are the recommended alternatives.

$ ping

Beginner

Send ICMP echo requests to test network connectivity

The ping command tests network connectivity by sending ICMP echo requests to a host and measuring response times. It is the first tool used when diagnosing network problems - if ping works, basic network connectivity is established.

ping measures round-trip time (RTT), packet loss, and jitter. I...

Options & Flags

`-c`	Send N packets then stop
`-i`	Set interval between packets (seconds)
`-W`	Set timeout for each reply (seconds)
`-s`	Set packet size in bytes
`-f`	Flood ping (requires root)
`-q`	Quiet - show only summary statistics
`-4/-6`	Force IPv4 or IPv6
`-t`	Set TTL (Time To Live)

Practical Examples

Example: Basic connectivity test

```
$ ping -c 4 google.com
PING google.com (142.250.74.14): 64 bytes, icmp_seq=0 ttl=117 time=12.3 ms
--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss
rtt min/avg/max = 11.2/12.5/14.1 ms
```

Sends 4 packets to Google and shows round-trip times.

Example: Test local network

```
$ ping -c 3 192.168.1.1
```

Pings the default gateway to verify local network works.

Example: Quick host check

```
$ ping -c 1 -W 2 server.example.com
```

Single packet with 2-second timeout - fast alive/dead check for scripts.

Example: Continuous monitoring

```
$ ping google.com
```

Runs continuously until Ctrl+C. Shows real-time latency monitoring.

Example: Test with specific packet size

```
$ ping -c 3 -s 1472 192.168.1.1
```

Tests with large packets to verify MTU and detect fragmentation issues.

Tips & Best Practices

Pro Tip: Script-friendly check: Use ping -c 1 -W 1 host > /dev/null 2>&1 && echo "UP" || echo "DOWN" for quick host availability checking in scripts.

Note: Ping blocked by firewalls: Many servers block ICMP ping. No response does not always mean the host is down - the firewall may be blocking ICMP.

Warning: Flood ping caution: ping -f sends packets as fast as possible. Only use on networks you own. It can saturate network links and trigger security alerts.

$ resolvectl

Intermediate

DNS resolution and systemd-resolved management

The resolvectl command is the management tool for systemd-resolved, the DNS resolver service built into modern systemd-based Linux distributions. It provides DNS lookups, cache management, DNSSEC validation status, and per-interface DNS configuration.

resolvectl replaces the older systemd-resolv...

Options & Flags

`query HOSTNAME`	Resolve a hostname to IP address(es)
`status`	Show current DNS configuration per interface
`statistics`	Show DNS cache and query statistics
`flush-caches`	Flush all DNS caches
`dns INTERFACE SERVER`	Set DNS server for an interface
`domain INTERFACE DOMAIN`	Set search domain for an interface
`dnssec INTERFACE MODE`	Set DNSSEC mode (yes/no/allow-downgrade)
`monitor`	Monitor DNS queries in real-time

Practical Examples

Example: Resolve a hostname

```
$ resolvectl query example.com
```

Resolve hostname showing IP addresses, DNSSEC status, and which interface/server handled the query.

Example: View DNS configuration

```
$ resolvectl status
```

Show DNS servers, search domains, and DNSSEC/DoT settings for each network interface.

Example: Flush DNS cache

```
$ sudo resolvectl flush-caches && resolvectl statistics
```

Clear all cached DNS entries. Verify with statistics showing zero cache entries.

Example: Set DNS servers

```
$ sudo resolvectl dns eth0 1.1.1.1 1.0.0.1 && resolvectl status eth0
```

Set Cloudflare DNS servers for eth0. Takes effect immediately.

Example: Monitor DNS queries

```
$ resolvectl monitor
```

Watch DNS queries in real-time - shows what hostnames the system is resolving and the results.

Tips & Best Practices

Note: systemd-resolve is deprecated: The older systemd-resolve command is deprecated. Use resolvectl instead. On older systems, systemd-resolve still works as a symlink.

Pro Tip: Check /etc/resolv.conf: On systemd-resolved systems, /etc/resolv.conf should be a symlink to /run/systemd/resolve/stub-resolv.conf. If not, DNS may not use systemd-resolved.

Warning: Per-interface settings are temporary: DNS servers set with resolvectl dns are lost on reboot. For persistent config, use NetworkManager (nmcli) or systemd-networkd .network files.

Pro Tip: Enable DNS-over-TLS: Set DNSOverTLS=yes in /etc/systemd/resolved.conf to encrypt all DNS queries. Use Cloudflare (1.1.1.1) or Google (8.8.8.8) DNS which support DoT.

$ ss

Intermediate

Show socket statistics (modern replacement for netstat)

ss (socket statistics) is the modern replacement for netstat, providing faster and more detailed network socket information. It is part of the iproute2 package and is the recommended tool for inspecting network connections.

ss can display TCP, UDP, Unix, and raw socket information. It supports p...

Options & Flags

`-t`	Show TCP sockets
`-u`	Show UDP sockets
`-l`	Show listening sockets only
`-n`	Show numerical addresses
`-p`	Show process using socket
`-a`	Show all sockets (listening and non-listening)
`-s`	Show socket statistics summary
`-i`	Show internal TCP information
`-o`	Show timer information

Practical Examples

Example: Show listening ports

```
$ sudo ss -tulnp
Netid State  Local Address:Port  Process
tcp   LISTEN 0.0.0.0:22          users:(("sshd",pid=1234))
tcp   LISTEN 0.0.0.0:443         users:(("nginx",pid=5678))
```

Shows all listening TCP/UDP ports with process names - the most common usage.

Example: Socket statistics summary

```
$ ss -s
Total: 234\nTCP:   156 (estab 89, closed 12, orphaned 0)
```

Quick overview of all socket types and their states.

Example: Filter by port

```
$ ss -tlnp sport = :443
```

Shows only connections on port 443.

Example: Show established connections

```
$ ss -tn state established
```

Shows only currently established TCP connections.

Example: Find process on specific port

```
$ sudo ss -tlnp | grep ':80'
```

Identifies what process is listening on port 80.

Tips & Best Practices

Pro Tip: ss filtering syntax: ss supports powerful filters: ss -tn 'sport = :80 or sport = :443' shows connections on web ports. Use state filters: ss -t state established.

Note: ss vs netstat: ss is faster (reads kernel directly), shows more details, and supports advanced filtering. netstat parses /proc. Always prefer ss on modern systems.

Warning: Root for process info: Like netstat, ss -p requires root to show processes owned by other users. Use sudo.

$ ss -tulnp

Intermediate

Show listening TCP/UDP ports with process info

ss -tulnp is the most commonly used network diagnostic command in Linux. It combines flags to show all listening TCP and UDP ports with their associated process names, providing a complete picture of network services running on a system.

This command is typically the first step in network troubl...

Options & Flags

`-t`	Show TCP sockets
`-u`	Show UDP sockets
`-l`	Listening sockets only
`-n`	Numerical addresses (no DNS resolution)
`-p`	Show process using each socket
`-4/-6`	Show only IPv4 or IPv6

Practical Examples

Example: Show all listening services

```
$ sudo ss -tulnp
Netid State  Local Address:Port   Process
tcp   LISTEN 0.0.0.0:22            sshd
tcp   LISTEN 0.0.0.0:80            nginx
tcp   LISTEN 0.0.0.0:443           nginx
```

The standard command for checking open ports and services.

Example: Check specific port

```
$ sudo ss -tulnp | grep ':3306'
tcp  LISTEN  0.0.0.0:3306  users:(("mysqld",pid=1234))
```

Checks if MySQL is listening on port 3306.

Example: IPv4 only

```
$ sudo ss -4tulnp
```

Shows only IPv4 listening ports.

Example: Count listening ports

```
$ ss -tln | wc -l
12
```

Counts the number of listening TCP ports.

Example: Check for port conflicts

```
$ sudo ss -tulnp | grep ':8080'
```

Verifies if port 8080 is already in use before starting a new service.

Tips & Best Practices

Pro Tip: Memorize the flags: tulnp = TCP, UDP, Listening, Numerical, Process. This is the single most useful network command to memorize.

Note: Alternative: netstat -tulnp: The equivalent netstat command is netstat -tulnp. ss is preferred on modern systems as it is faster.

Warning: Requires root: The -p flag needs root to show all process names. Without sudo, you only see your own processes.