Network Tools Cheatsheet

Network Tools Cheatsheet

A practical quick-reference guide for network diagnostics and troubleshooting on Linux. Commands are organized by tool with common flags and realistic examples.

ip

The modern replacement for `ifconfig`, `route`, and `arp`. Part of the `iproute2` suite.

Show Network Interfaces

```
# List all interfaces with addresses
ip addr show

# Show a specific interface
ip addr show eth0

# Brief listing (no extra details)
ip -brief addr show

# Show only assigned addresses
ip -4 addr show
```

Add and Remove Addresses

```
# Add an IP address to an interface
ip addr add 192.168.1.100/24 dev eth0

# Add a secondary IP
ip addr add 10.0.0.50/24 dev eth0 label eth0:1

# Remove an IP address
ip addr del 192.168.1.100/24 dev eth0

# Flush all addresses on an interface
ip addr flush dev eth0
```

Interface Link Management

```
# Bring an interface up
ip link set eth0 up

# Bring an interface down
ip link set eth0 down

# Set the MTU
ip link set eth0 mtu 9000

# Change the MAC address (requires link down first)
ip link set eth0 address 00:11:22:33:44:55

# Enable promiscuous mode
ip link set eth0 promisc on

# Disable promiscuous mode
ip link set eth0 promisc off
```

Routing

```
# Show the routing table
ip route show

# Show route to a specific destination
ip route get 8.8.8.8

# Add a default gateway
ip route add default via 192.168.1.1

# Add a static route
ip route add 10.0.0.0/24 via 192.168.1.254

# Add a route through a specific interface
ip route add 172.16.0.0/16 dev eth1

# Delete a route
ip route del 10.0.0.0/24 via 192.168.1.254

# Show routing cache
ip route show cache
```

Neighbor (ARP) Table

```
# Show ARP table
ip neigh show

# Show ARP for a specific device
ip neigh show dev eth0

# Add a static ARP entry
ip neigh add 192.168.1.10 lladdr aa:bb:cc:dd:ee:ff dev eth0

# Delete an ARP entry
ip neigh del 192.168.1.10 dev eth0
```

```
# Flush the ARP table
ip neigh flush all
```

tcpdump

Packet analyzer for capturing and inspecting network traffic. Requires root or `CAP_NET_RAW`.

Basic Capture

```
# Capture on a specific interface
tcpdump -i eth0

# Capture on any interface
tcpdump -i any

# Capture with verbose output
tcpdump -i eth0 -v

# Capture N packets then stop
tcpdump -i eth0 -c 100

# Capture with absolute sequence numbers
tcpdump -i eth0 -S
```

Filter by Host, Port, Protocol

```
# Capture traffic to/from a host
tcpdump -i eth0 host 192.168.1.10

# Capture traffic from a source only
tcpdump -i eth0 src 192.168.1.10

# Capture traffic to a destination only
tcpdump -i eth0 dst 192.168.1.10

# Capture traffic on a specific port
tcpdump -i eth0 port 80

# Capture on a range of ports
tcpdump -i eth0 portrange 80-90

# Capture specific protocol (tcp, udp, icmp, arp)
tcpdump -i eth0 icmp

# Combine filters with and/or/not
tcpdump -i eth0 'tcp port 80 and host 192.168.1.10'
tcpdump -i eth0 'tcp port 443 or tcp port 80'
tcpdump -i eth0 'not port 22'
```

Advanced Filters with BPF

```
# Capture TCP SYN packets (new connections)
tcpdump -i eth0 'tcp[tcpflags] & (tcp-syn) != 0'
```

```
# Capture only SYN-ACK packets
tcpdump -i eth0 'tcp[tcpflags] & (tcp-syn|tcp-ack) = (tcp-syn|tcp-ack)'

# Capture TCP RST packets (connection resets)
tcpdump -i eth0 'tcp[tcpflags] & (tcp-rst) != 0'

# Capture packets larger than 1000 bytes
tcpdump -i eth0 'greater 1000'

# Capture packets smaller than 100 bytes
tcpdump -i eth0 'less 100'

# Capture traffic on a specific subnet
tcpdump -i eth0 'net 10.0.0.0/24'

# Capture HTTP GET requests
tcpdump -i eth0 -A -s 0 'tcp port 80 and ((tcp[((tcp[12:1] & 0xf0) >> 2):4] = 0x47455420))'

# Capture DNS queries
tcpdump -i eth0 -s 0 'port 53'
```

Save and Read Capture Files

```
# Save capture to pcap file
tcpdump -i eth0 -w capture.pcap

# Save with ring buffer (rotate files at 100MB, keep 5)
tcpdump -i eth0 -w capture.pcap -C 100 -W 5

# Read a pcap file
tcpdump -r capture.pcap

# Read with verbose output
tcpdump -r capture.pcap -v

# Read with hex and ASCII content
tcpdump -r capture.pcap -XX
```

Display Options

```
# Show ASCII content alongside hex
tcpdump -i eth0 -A

# Show hex and ASCII side by side
tcpdump -i eth0 -XX

# Print each packet in detail
tcpdump -i eth0 -vv

# Don't resolve hostnames (faster)
tcpdump -i eth0 -n

# Don't resolve port names
tcpdump -i eth0 -nn
```

```
# Print absolute timestamps
tcpdump -i eth0 -tttt
```

dig

DNS lookup utility from the BIND suite. Preferred over `nslookup` for scripting.

Basic Lookups

```
# Simple A record lookup
dig example.com

# Lookup with short output
dig +short example.com

# Lookup a specific record type
dig example.com A
dig example.com AAAA
dig example.com MX
dig example.com NS
dig example.com TXT
dig example.com CNAME
dig example.com SOA

# Lookup against a specific DNS server
dig @8.8.8.8 example.com

# Lookup from a specific DNS server with short output
dig @1.1.1.1 +short example.com
```

Reverse DNS

```
# Reverse lookup (PTR record)
dig -x 8.8.8.8

# Reverse lookup with short output
dig -x 8.8.8.8 +short
```

Trace DNS Resolution

```
# Trace the full resolution path
dig +trace example.com

# Trace from a specific root server
dig +trace @a.root-servers.net example.com
```

Query Details and Debugging

```
# Show the full DNS response (header, question, answer, authority, additional)
dig example.com +noall +answer
```

```
# Show only the answer section
dig example.com +answer

# Show DNSSEC chain of trust
dig example.com DNSKEY +short
dig example.com DS +short

# Check DNSSEC validation
dig example.com +dnssec

# Query with TCP instead of UDP
dig +tcp example.com

# Show timing information
dig example.com +stats

# Query over a specific port
dig @localhost -p 5353 example.com
```

Bulk and Batch Lookups

```
# Query multiple record types at once
dig example.com ANY +noall +answer

# Query from a file (one domain per line)
dig -f domains.txt +short

# Query with specific EDNS options
dig +edns=0 example.com
```

nslookup

Interactive DNS lookup tool. Simpler than `dig` but less scriptable.

Basic Usage

```
# Lookup a domain
nslookup example.com

# Lookup against a specific server
nslookup example.com 8.8.8.8

# Lookup a specific record type
nslookup -type=MX example.com
nslookup -type=NS example.com
nslookup -type=TXT example.com

# Reverse lookup
nslookup 8.8.8.8

# Interactive mode
nslookup
> server 8.8.8.8
> set type=MX
```

```
> example.com
> exit
```

curl

Versatile tool for transferring data with URL syntax. Supports HTTP, HTTPS, FTP, and more.

HTTP Methods

```
# GET request (default)
curl https://api.example.com/users

# POST request with data
curl -X POST https://api.example.com/users -d "name=alice"

# POST with JSON body
curl -X POST https://api.example.com/users \
  -H "Content-Type: application/json" \
  -d '{"name": "alice", "role": "admin"}'

# PUT request
curl -X PUT https://api.example.com/users/1 -d "name=bob"

# PATCH request
curl -X PATCH https://api.example.com/users/1 -d "role=editor"

# DELETE request
curl -X DELETE https://api.example.com/users/1

# HEAD request (headers only)
curl -I https://api.example.com/users

# OPTIONS request
curl -X OPTIONS https://api.example.com/users -i
```

Headers

```
# Set a custom header
curl -H "Authorization: Bearer TOKEN" https://api.example.com/me

# Set multiple headers
curl -H "Accept: application/json" -H "X-Request-ID: 123" https://api.example.com

# Show response headers
curl -i https://api.example.com

# Show only response headers
curl -s -D - https://api.example.com -o /dev/null

# Send a cookie
curl -b "session=abc123" https://api.example.com/me

# Send cookies from a file
curl -b cookies.txt https://api.example.com
```

```
# Save response cookies to a file
curl -c cookies.txt https://api.example.com/login
```

Authentication

```
# Basic auth
curl -u user:password https://api.example.com

# Bearer token
curl -H "Authorization: Bearer TOKEN" https://api.example.com

# OAuth2 client credentials (fetch token first)
curl -X POST https://auth.example.com/token \
  -d "grant_type=client_credentials" \
  -u "CLIENT_ID:CLIENT_SECRET"

# Use the token
TOKEN="eyJhbGci..."
curl -H "Authorization: Bearer $TOKEN" https://api.example.com/me

# Client certificate authentication
curl --cert client.pem --key key.pem https://secure.example.com

# Skip TLS verification (insecure, for testing only)
curl -k https://self-signed.example.com
```

Output Control

```
# Save response to file
curl -o output.html https://example.com

# Follow the original remote filename
curl -O https://example.com/file.zip

# Silent mode (no progress or errors)
curl -s https://example.com

# Show verbose output (headers, connection info)
curl -v https://example.com

# Show only HTTP status code
curl -s -o /dev/null -w "%{http_code}" https://example.com

# Format response details
curl -s -o /dev/null -w "Status: %{http_code}\nTime: %{time_total}s\nSize: %{size_download}\n

# Fail silently on server errors
curl -f https://example.com

# Write a debug log
curl --trace-ascii debug.log https://example.com
```

Advanced Requests

```
# Follow redirects
curl -L https://example.com

# Follow redirects with a max limit
curl -L --max-redirs 5 https://example.com

# Set a timeout (seconds)
curl --max-time 10 https://example.com

# Set connect timeout only
curl --connect-timeout 5 https://example.com

# Retry on failure (up to 3 times)
curl --retry 3 https://example.com

# Upload a file with PUT
curl -T upload.tar.gz https://example.com/upload

# Send multipart form data
curl -F "file=@photo.jpg" https://example.com/upload

# Send form data (application/x-www-form-urlencoded)
curl -d "user=alice&action=submit" https://example.com/form

# Use a proxy
curl --proxy http://proxy.example.com:8080 https://example.com

# Set the Referer header
curl -e "https://example.com" https://other.example.com

# Set User-Agent
curl -A "MyBot/1.0" https://example.com

# Compress request (send Accept-Encoding)
curl --compressed https://example.com
```

ss

Modern replacement for `netstat`. Inspect sockets, connections, and network statistics.

List Connections

```
# List all TCP connections
ss -t

# List all TCP connections with process info
ss -tp

# List all listening TCP sockets
ss -tln

# List all UDP sockets
ss -u

# List all listening UDP sockets
```

```
ss -uln

# List all sockets (TCP, UDP, UNIX)
ss -a

# List all sockets with process info
ss -tap
```

Filter by State

```
# Show established connections
ss -t state established

# Show TIME-WAIT connections
ss -t state time-wait

# Show closed connections
ss -t state closed

# Show all states
ss -t state all

# Show connections in a specific state with process info
ss -tp state established
```

Filter by Address and Port

```
# Show connections to a specific port
ss -t 'sport = :443'

# Show connections from a specific source
ss -t 'src = 192.168.1.0/24'

# Show connections to a specific destination
ss -t 'dst = 10.0.0.1'

# Combine filters
ss -t 'dst = 10.0.0.1 and dport = :443'

# Show connections on a specific interface
ss -t 'dev = eth0'
```

Statistics

```
# Show socket statistics summary
ss -s

# Show TCP statistics
ss -st

# Show UDP statistics
ss -su
```

```
# Show memory usage per socket
ss -tm
```

Advanced Options

```
# Show numeric addresses and ports (no DNS resolution)
ss -tn

# Show extended info (uid, ino, sk, etc.)
ss -te

# Show socket memory info
ss -tm

# Show timer info (keepalive, retransmission)
ss -to

# Show only IPv4
ss -4 -tln

# Show only IPv6
ss -6 -tln

# Continuous monitoring (every 1 second)
ss -tlnp -i 1
```

netstat

Legacy network statistics tool. Use `ss` on modern systems, but `netstat` is still found on many servers.

List Connections

```
# List all TCP connections
netstat -t

# List all listening TCP sockets
netstat -tln

# List all UDP sockets
netstat -uln

# List all sockets
netstat -a

# Show process info (PID/program name)
netstat -tlnp

# Show numeric addresses only (skip DNS lookup)
netstat -tn
```

Network Interfaces

```
# Show interface statistics
netstat -i

# Show kernel routing table
netstat -rn

# Show kernel interface table
netstat -ie
```

Statistics

```
# Show TCP statistics
netstat -st

# Show UDP statistics
netstat -su

# Show all statistics
netstat -s
```

Common Patterns

```
# Find what's listening on a port
netstat -tlnp | grep :8080

# Count established connections per IP
netstat -tn | grep ESTABLISHED | awk '{print $5}' | cut -d: -f1 | sort | uniq -c | sort -rn
```

traceroute

Trace the route packets take to a network host, showing each hop along the path.

Basic Usage

```
# Trace route to a host
traceroute example.com

# Trace with a maximum of 20 hops
traceroute -m 20 example.com

# Trace with ICMP probes instead of UDP (more reliable through firewalls)
traceroute -I example.com

# Trace using TCP SYN on a specific port
traceroute -T -p 443 example.com

# Trace with a specific number of queries per hop
traceroute -q 3 example.com

# Trace and resolve hostnames
traceroute example.com
```

```
# Trace without resolving hostnames (faster)
traceroute -n example.com
```

mtr

Combines `traceroute` and `ping` into a single tool with real-time updating output.

Basic Usage

```
# Run mtr (real-time updating display)
mtr example.com

# Run in report mode (no interactive display)
mtr --report example.com

# Run report with a specific number of pings
mtr --report --report-cycles 50 example.com

# Use ICMP instead of UDP
mtr --icmp example.com

# Use TCP on a specific port
mtr --tcp --port 443 example.com

# Resolve hostnames (or skip with -n)
mtr --report --no-dns example.com

# Show both ASN and IP info
mtr --report --aslookup example.com
```

nmap

Network exploration and security auditing tool. Used for port scanning, service detection, and OS fingerprinting.

Port Scanning Basics

```
# Scan a single host (common ports)
nmap example.com

# Scan a specific port
nmap -p 80 example.com

# Scan a range of ports
nmap -p 1-1024 example.com

# Scan specific ports
nmap -p 22,80,443,3306 example.com

# Scan all 65535 ports
nmap -p- example.com
```

```
# Fast scan (top 100 ports)
nmap -F example.com
```

Scan Types

```
# TCP SYN scan (stealth, default if root)
nmap -sS example.com

# TCP connect scan (full handshake, works without root)
nmap -sT example.com

# UDP scan (slow, requires root)
nmap -sU example.com

# Ping scan (host discovery only)
nmap -sn 192.168.1.0/24

# Skip host discovery (scan directly)
nmap -Pn example.com

# ARP scan on local network (fast)
nmap -PR 192.168.1.0/24
```

Service and OS Detection

```
# Detect service versions
nmap -sV example.com

# Detect operating system
nmap -O example.com

# Aggressive scan (OS, version, scripts, traceroute)
nmap -A example.com

# Service detection with intensity level (0-9)
nmap -sV --version-intensity 7 example.com
```

Network and Subnet Scanning

```
# Scan a full subnet
nmap 192.168.1.0/24

# Scan a range of IPs
nmap 192.168.1.1-50

# Scan from a file
nmap -iL hosts.txt

# Exclude a host from scan
nmap 192.168.1.0/24 --exclude 192.168.1.100

# Scan multiple targets
nmap 192.168.1.1 192.168.1.10 10.0.0.1
```

NSE Scripts

```
# Run default scripts
nmap -sC example.com

# Run a specific script
nmap --script http-headers example.com

# Run multiple scripts
nmap --script=http-headers,ssl-heartbleed example.com

# Run scripts with arguments
nmap --script http-title --script-args http.useragent="MyBot" example.com

# List all available scripts
nmap --script-help="http-*"

# Vuln scan (all vulnerability scripts)
nmap --script vuln example.com
```

Output Formats

```
# Save output to normal text file
nmap -oN scan.txt example.com

# Save output to XML
nmap -oX scan.xml example.com

# Save output in grepable format
nmap -oG scan.gnmap example.com

# Save to all formats at once
nmap -oA scan_results example.com
```

nc (netcat)

Swiss-army knife for TCP/UDP connections. Useful for testing ports, transferring files, and quick ad-hoc services.

Port Scanning and Testing

```
# Scan a single port
nc -zv example.com 80

# Scan a range of ports
nc -zv example.com 20-100

# Scan specific ports
nc -zv example.com 22 80 443

# Test a UDP port
nc -zuv example.com 53
```

```
# Connect to a port (interactive)
nc example.com 80
```

Listen and Serve

```
# Listen on a port
nc -l 8080

# Listen and keep running after client disconnects
nc -lk 8080

# Listen on a specific address
nc -l 192.168.1.10 8080

# Serve a file over a port
nc -l 8080 < myfile.txt

# Receive a file
nc example.com 8080 > received.txt
```

Chat and Debugging

```
# Simple chat between two machines
# Machine A:
nc -l 4444
# Machine B:
nc machine-a-ip 4444

# Send an HTTP request manually
nc example.com 80
GET / HTTP/1.1
Host: example.com
# Press Enter twice

# Test a TLS connection (using -l with --ssl)
nc -l --ssl 8443
nc --ssl example.com 8443
```

Proxy and Relay

```
# Simple port forwarding (relay)
nc -l 8080 | nc example.com 80

# Port scan with timeout
nc -zv -w 3 example.com 80
```

iptables

Linux kernel firewall for filtering and NAT. Requires root.

Basic Concepts

```
# List all rules in the filter table
iptables -L

# List rules with line numbers and packet counts
iptables -L -v -n --line-numbers

# List rules in a specific chain
iptables -L INPUT -v -n

# List NAT table rules
iptables -t nat -L -v -n

# Flush all rules in a chain
iptables -F

# Flush a specific chain
iptables -F INPUT
```

Filter Rules

```
# Allow established and related traffic
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

# Allow incoming SSH
iptables -A INPUT -p tcp --dport 22 -j ACCEPT

# Allow incoming HTTP and HTTPS
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -j ACCEPT

# Allow ping (ICMP)
iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT

# Allow traffic from a specific IP
iptables -A INPUT -s 192.168.1.0/24 -j ACCEPT

# Drop all other incoming traffic
iptables -A INPUT -j DROP

# Allow all outgoing traffic
iptables -A OUTPUT -j ACCEPT
```

NAT Rules

```
# Masquerade (SNAT for dynamic IPs, common for routers)
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

# Destination NAT (port forwarding)
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 192.168.1.10:8080

# Source NAT (static)
iptables -t nat -A POSTROUTING -j SNAT --to-source 203.0.113.5
```

Rule Management

```
# Insert a rule at a specific position
iptables -I INPUT 3 -p tcp --dport 3306 -j ACCEPT

# Delete a rule by line number
iptables -D INPUT 3

# Delete a rule by specification
iptables -D INPUT -p tcp --dport 80 -j ACCEPT

# Set the default policy for a chain
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT
```

Save and Restore

```
# Save rules to a file
iptables-save > /etc/iptables/rules.v4

# Restore rules from a file
iptables-restore < /etc/iptables/rules.v4

# Make rules persistent (Debian/Ubuntu)
apt install iptables-persistent
netfilter-persistent save
netfilter-persistent reload
```

wget

Non-interactive network downloader. Ideal for scripts, recursive downloads, and mirroring.

Download Files

```
# Download a file
wget https://example.com/file.zip

# Download with a custom filename
wget -O myfile.zip https://example.com/file.zip

# Download to a specific directory
wget -P /tmp https://example.com/file.zip

# Download in the background
wget -b https://example.com/large-file.iso

# Resume an interrupted download
wget -c https://example.com/large-file.iso
```

Recursive Download

```
# Mirror a website (download entire site)
wget --mirror --convert-links --adjust-extension --page-requisites https://example.com
```

```
# Download all files in a directory (no parent)
wget -r -np -nH --cut-dirs=2 https://example.com/docs/files/

# Download specific file types recursively
wget -r -l1 -A pdf https://example.com/docs/

# Set download recursion depth
wget -r -l 3 https://example.com/
```

Authentication and Headers

```
# Download with basic auth
wget --user=admin --password=secret https://example.com/admin.zip

# Download with a custom header
wget --header="Authorization: Bearer TOKEN" https://example.com/api/data

# Download with a custom User-Agent
wget -U "MyBot/1.0" https://example.com
```

Rate Limiting and Timeout

```
# Limit download speed to 1 MB/s
wget --limit-rate=1M https://example.com/large-file.iso

# Set timeout (seconds)
wget --timeout=10 https://example.com/file.zip

# Set number of retries
wget --tries=3 https://example.com/file.zip

# Continue on error (skip broken links in recursive mode)
wget -r -nc https://example.com/
```

Debugging

```
# Verbose output
wget -v https://example.com/file.zip

# Debug output (even more verbose)
wget -d https://example.com/file.zip

# Quiet mode (no output)
wget -q https://example.com/file.zip

# Log output to a file
wget -o download.log https://example.com/file.zip
```

arp

View and manipulate the ARP cache.

View ARP Table

```
# Show the ARP table
arp -a

# Show ARP table in numeric mode (no DNS resolution)
arp -an

# Show ARP for a specific host
arp -a 192.168.1.10

# Show ARP for a specific interface
arp -i eth0
```

Modify ARP Table

```
# Add a static ARP entry
arp -s 192.168.1.10 aa:bb:cc:dd:ee:ff

# Delete an ARP entry
arp -d 192.168.1.10

# Delete all entries for an interface
arp -d -i eth0 -a
```

Hostname Resolution Debugging

Tools and techniques for diagnosing DNS and hostname resolution issues.

getent

```
# Resolve a hostname using system resolver (respects /etc/nsswitch.conf)
getent hosts example.com

# Resolve using only DNS
getent ahosts example.com

# Resolve with IPv4 only
getent ahostsv4 example.com

# Resolve with IPv6 only
getent ahostsv6 example.com
```

/etc/hosts and /etc/resolv.conf

```
# Check local hostname mappings
cat /etc/hosts

# Check DNS resolver configuration
cat /etc/resolv.conf
```

```
# Check the name service switch order
cat /etc/nsswitch.conf | grep hosts
```

host Command

```
# Simple lookup
host example.com

# Lookup a specific record type
host -t MX example.com
host -t NS example.com
host -t TXT example.com

# Lookup against a specific server
host example.com 8.8.8.8

# Verbose mode (show TTL)
host -v example.com
```

Debugging Steps

```
# 1. Check local hosts file first
getent hosts example.com

# 2. Query DNS directly (bypass local cache)
dig +short example.com

# 3. Query a specific DNS server
dig @8.8.8.8 example.com

# 4. Check if the domain exists at all
dig example.com NS +short

# 5. Trace the full resolution path
dig +trace example.com

# 6. Verify reverse DNS
dig -x 93.184.216.34

# 7. Check if your system's resolver is working
dig +short resolver1.opendns.com

# 8. Test connectivity to DNS server on port 53
nc -zuv 8.8.8.8 53
```

Quick Reference

Tool	Purpose	Key Flags
ip	Network interfaces & routing	`addr`, `link`, `route`, `neigh`
tcpdump	Packet capture & analysis	`-i`, `-w`, `-r`, `-n`, host/port filters
dig	DNS lookups	`+short`, `+trace`, `-x`, `-t`, `@server`

Tool	Purpose	Key Flags
curl	HTTP requests	`-X`, `-H`, `-d`, `-o`, `-I`, `-L`
ss	Socket statistics	`-t`, `-u`, `-l`, `-n`, `-p`, `state`
netstat	Legacy socket stats	`-tlnp`, `-rn`, `-s`, `-i`
traceroute	Trace packet path	`-I`, `-T`, `-n`, `-m`, `-q`
mtr	Combined traceroute + ping	`--report`, `--tcp`, `--icmp`
nmap	Port scanning & discovery	`-sS`, `-sV`, `-O`, `-A`, `-p`, `-sn`
nc	TCP/UDP Swiss-army knife	`-zv`, `-l`, `-lk`, `-u`
iptables	Firewall rules	`-A`, `-D`, `-I`, `-L`, `-F`, `-t nat`
wget	File downloads	`-O`, `-c`, `-b`, `-r`, `-q`
arp	ARP cache	`-a`, `-an`, `-s`, `-d`

Common Troubleshooting Workflows

Can't Reach a Remote Host

```
# 1. Can you resolve the hostname?
dig +short example.com

# 2. Can you reach the IP?
ping -c 4 93.184.216.34

# 3. Is the port open?
nc -zv example.com 443

# 4. Where does the route drop?
traceroute -n -m 20 example.com

# 5. What does the server return?
curl -vI https://example.com
```

High Connection Count

```
# 1. Count connections per IP
ss -tn state established | awk '{print $5}' | cut -d: -f1 | sort | uniq -c | sort -rn | head

# 2. Check for TIME-WAIT accumulation
ss -tn state time-wait | wc -l

# 3. Check for SYN flood (many SYN_RECV)
ss -tn state syn-recv | wc -l

# 4. Show sockets per process
ss -tpn | awk '{print $6}' | sort | uniq -c | sort -rn
```

DNS Resolution Problems

```
# 1. Check system resolver
cat /etc/resolv.conf

# 2. Test with dig against the configured server
dig @$(grep nameserver /etc/resolv.conf | awk '{print $2}') example.com

# 3. Test with a known-good server
dig @8.8.8.8 example.com

# 4. Check for stale local cache
dig +trace example.com

# 5. Verify /etc/hosts isn't interfering
grep example.com /etc/hosts
```

Next Steps

- Bash CLI Tools Cheat Sheet — Text processing and CLI utilities
- Docker CLI Cheat Sheet — Container networking commands
- Kubernetes kubectl Cheat Sheet — K8s network policies and services