**You said:**

linux networking cmds with all the flags and example - how to use them in real world please - please include all from basic to advance cmds

**ChatGPT said:**

Here is a comprehensive list of Linux networking commands, starting from basic to advanced, along with examples and real-world use cases:

**1. ifconfig (Interface Configuration)**

Used to configure and display network interfaces.

* **Basic usage**:

bash

CopyEdit

ifconfig

Displays all active network interfaces.

* **Real-world example**: To check the IP address of a specific network interface (e.g., eth0):

bash

CopyEdit

ifconfig eth0

* **Flags**:
  + up: Activate the interface.
  + down: Deactivate the interface.
  + inet: Show the IP address associated with the interface.
  + netmask: Show or set the network mask.
  + broadcast: Show or set the broadcast address.

**2. ip (Network Configuration)**

More powerful and flexible replacement for ifconfig.

* **Basic usage**:

bash

CopyEdit

ip addr show

Displays all network interfaces and their addresses.

* **Real-world example**: Display IP address for interface eth0:

bash

CopyEdit

ip addr show dev eth0

* **Flags**:
  + link: Show link-layer information.
  + addr: Show IP addresses.
  + route: Show routing table.
  + link set: Modify interface status (up/down).
  + addr add/del: Add or remove IP addresses.

**3. ping**

Used to check the network connectivity to a host.

* **Basic usage**:

bash

CopyEdit

ping google.com

Sends ICMP echo requests to the specified host.

* **Real-world example**: Check network latency to a remote server:

bash

CopyEdit

ping -c 4 google.com

* **Flags**:
  + -c <count>: Send a specific number of pings.
  + -i <interval>: Set interval between pings.
  + -t <TTL>: Set Time-to-Live (TTL) for the packets.

**4. netstat (Network Statistics)**

Displays various network statistics and information about network connections.

* **Basic usage**:

bash

CopyEdit

netstat

Shows active connections and listening ports.

* **Real-world example**: Display network connections along with the process name:

bash

CopyEdit

netstat -tuln

* **Flags**:
  + -t: Show TCP connections.
  + -u: Show UDP connections.
  + -l: Show listening ports.
  + -n: Show numerical addresses instead of resolving hostnames.

**5. ss (Socket Statistics)**

A utility to investigate sockets, often considered a faster alternative to netstat.

* **Basic usage**:

bash

CopyEdit

ss

Displays a list of open sockets.

* **Real-world example**: Display TCP sockets in use:

bash

CopyEdit

ss -t

* **Flags**:
  + -t: Show TCP sockets.
  + -u: Show UDP sockets.
  + -l: Show listening sockets.
  + -p: Show the process associated with each socket.

**6. traceroute**

Shows the route packets take to a destination.

* **Basic usage**:

bash

CopyEdit

traceroute google.com

* **Real-world example**: Trace the route to a specific website:

bash

CopyEdit

traceroute -m 30 google.com

* **Flags**:
  + -m <max hops>: Set the maximum number of hops.
  + -w <timeout>: Set the timeout for each probe.
  + -I: Use ICMP ECHO instead of UDP.

**7. route**

Displays and modifies the IP routing table.

* **Basic usage**:

bash

CopyEdit

route

Displays the current routing table.

* **Real-world example**: Add a new route to a specific network:

bash

CopyEdit

sudo route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.0.1

* **Flags**:
  + -n: Show numerical addresses instead of hostnames.
  + add: Add a route to the routing table.
  + del: Delete a route from the routing table.

**8. nslookup (Name Server Lookup)**

Queries DNS to obtain domain name or IP address mapping.

* **Basic usage**:

bash

CopyEdit

nslookup google.com

* **Real-world example**: Find the IP address of a domain:

bash

CopyEdit

nslookup -type=A google.com

* **Flags**:
  + -type=<type>: Specify the query type (A, MX, etc.).
  + -debug: Display detailed debugging information.

**9. dig (Domain Information Groper)**

A flexible tool for querying DNS servers.

* **Basic usage**:

bash

CopyEdit

dig google.com

* **Real-world example**: Query for MX records (Mail Exchange) for a domain:

bash

CopyEdit

dig google.com MX

* **Flags**:
  + +short: Return a short answer without extra details.
  + +trace: Trace the DNS resolution path.
  + -x: Reverse lookup (IP to hostname).

**10. iptables**

Used for setting up, maintaining, and inspecting the IP packet filter rules.

* **Basic usage**:

bash

CopyEdit

sudo iptables -L

* **Real-world example**: Block incoming connections on port 80 (HTTP):

bash

CopyEdit

sudo iptables -A INPUT -p tcp --dport 80 -j DROP

* **Flags**:
  + -L: List all rules.
  + -A: Append a rule to the chain.
  + -D: Delete a rule from the chain.
  + -P: Set default policies (e.g., ACCEPT, DROP).

**11. tcpdump**

Network traffic analyzer.

* **Basic usage**:

bash

CopyEdit

sudo tcpdump

* **Real-world example**: Capture traffic on interface eth0:

bash

CopyEdit

sudo tcpdump -i eth0

* **Flags**:
  + -i <interface>: Specify the network interface.
  + -w <file>: Write the output to a file.
  + -n: Show numerical addresses instead of resolving hostnames.
  + -v: Show detailed packet information.

**12. mtr (My Traceroute)**

Combines traceroute and ping to show network performance over time.

* **Basic usage**:

bash

CopyEdit

mtr google.com

* **Real-world example**: Continuous network diagnostics with mtr:

bash

CopyEdit

mtr -rw google.com

* **Flags**:
  + -r: Generate a report after the test.
  + -w: Show the report with wide output.

**13. ethtool**

Used to query and control network driver and hardware settings.

* **Basic usage**:

bash

CopyEdit

sudo ethtool eth0

* **Real-world example**: Display network interface statistics:

bash

CopyEdit

sudo ethtool -S eth0

* **Flags**:
  + -s <interface>: Set interface speed, duplex, etc.
  + -p: Blink the LEDs on the interface.
  + -e: Show EEPROM contents.

**14. ip link**

Shows and manipulates network interfaces.

* **Basic usage**:

bash

CopyEdit

ip link

* **Real-world example**: Bring down an interface:

bash

CopyEdit

sudo ip link set eth0 down

* **Flags**:
  + set <interface> up/down: Activate or deactivate the interface.

**15. nmap (Network Mapper)**

Used for network discovery and security auditing.

* **Basic usage**:

bash

CopyEdit

nmap 192.168.1.1

* **Real-world example**: Scan for open ports on a remote server:

bash

CopyEdit

nmap -p 80,443 192.168.1.1

* **Flags**:
  + -p <ports>: Scan specific ports.
  + -sS: TCP SYN scan (stealth).
  + -A: Enable OS detection, version detection, script scanning, and traceroute.

These commands and their flags will cover most of your networking needs, from basic interface management to advanced troubleshooting and security audits.

4o

**You said:**

continue with any command if left - please cover all related to networking linux cmds

**ChatGPT said:**

Sure! Here are some additional Linux networking commands that are useful for networking management and troubleshooting. These cover a variety of scenarios, from DNS management to advanced network diagnostics.

**16. hostname**

This command displays or sets the system's hostname.

* **Basic usage**:

bash

CopyEdit

hostname

Displays the current system hostname.

* **Real-world example**: Change the system hostname to "my-server":

bash

CopyEdit

sudo hostname my-server

* **Flags**:
  + -i: Show the IP address of the system.
  + -f: Show the fully qualified domain name (FQDN).
  + -d: Show the DNS domain name.

**17. curl (Client URL)**

Used for transferring data from or to a server, commonly used for testing APIs, web pages, or fetching files.

* **Basic usage**:

bash

CopyEdit

curl google.com

Fetch the HTML content of the webpage.

* **Real-world example**: Check a server’s status with a GET request:

bash

CopyEdit

curl -I http://example.com

* **Flags**:
  + -I: Fetch the HTTP header only.
  + -o <filename>: Save output to a file.
  + -L: Follow redirects.
  + -u <user:password>: Send basic authentication credentials.
  + -X <request>: Specify the request method (GET, POST, etc.).

**18. wget**

A non-interactive command-line tool used to download files from the web.

* **Basic usage**:

bash

CopyEdit

wget http://example.com/file.zip

* **Real-world example**: Download a file and save it under a specific filename:

bash

CopyEdit

wget -O myfile.zip http://example.com/file.zip

* **Flags**:
  + -O <filename>: Save the file as a specific name.
  + -c: Continue an interrupted download.
  + -r: Download files recursively.
  + -q: Quiet mode (do not show output).

**19. ip route**

Used to display or manipulate the system’s routing table.

* **Basic usage**:

bash

CopyEdit

ip route

Shows the routing table for the system.

* **Real-world example**: Add a new route:

bash

CopyEdit

sudo ip route add 192.168.1.0/24 via 192.168.0.1

* **Flags**:
  + add: Add a new route.
  + del: Delete a route.
  + show: Display routing table.

**20. arp (Address Resolution Protocol)**

Displays or modifies the system’s ARP cache, which maps IP addresses to MAC addresses.

* **Basic usage**:

bash

CopyEdit

arp

Displays the ARP cache.

* **Real-world example**: Show the ARP entry for a specific IP:

bash

CopyEdit

arp -a 192.168.1.1

* **Flags**:
  + -a: Display all ARP entries.
  + -d: Delete an entry from the ARP table.
  + -s: Manually add an entry to the ARP table.

**21. mii-tool**

Used for querying and controlling the media-independent interface (MII) status.

* **Basic usage**:

bash

CopyEdit

mii-tool

Displays the status of the network interfaces.

* **Real-world example**: Check the link status of a specific interface:

bash

CopyEdit

mii-tool eth0

* **Flags**:
  + -v: Verbose mode.
  + -r: Reset interface.

**22. tcpdump (Advanced Usage)**

Used to capture and analyze packets on a network interface.

* **Basic usage**:

bash

CopyEdit

sudo tcpdump

* **Real-world example**: Capture traffic on a specific port (e.g., port 80):

bash

CopyEdit

sudo tcpdump port 80

* **Flags**:
  + -i <interface>: Specify the network interface to capture from.
  + -v: Verbose output.
  + -w <file>: Write the captured data to a file.
  + -nn: Display IP addresses and port numbers instead of resolving them to hostnames.

**23. iwconfig**

Used for configuring wireless network interfaces.

* **Basic usage**:

bash

CopyEdit

iwconfig

Display wireless network interface settings.

* **Real-world example**: Set the SSID for a wireless interface:

bash

CopyEdit

sudo iwconfig wlan0 essid "MyWiFi"

* **Flags**:
  + essid: Set or display the ESSID (network name).
  + mode: Set or display the mode (e.g., Managed, Ad-Hoc).
  + key: Set or display the encryption key.

**24. nmcli (NetworkManager CLI)**

A command-line interface used to manage network connections via NetworkManager.

* **Basic usage**:

bash

CopyEdit

nmcli

* **Real-world example**: Connect to a Wi-Fi network:

bash

CopyEdit

nmcli dev wifi connect "MyWiFi" password "mypassword"

* **Flags**:
  + dev status: Show device statuses.
  + connection up/down: Bring a network connection up or down.
  + device wifi: Show available wireless networks.

**25. nft (Netfilter)**

A more modern and flexible tool for packet filtering, replacing iptables in many cases.

* **Basic usage**:

bash

CopyEdit

sudo nft list ruleset

* **Real-world example**: Create a simple NAT rule to forward packets:

bash

CopyEdit

sudo nft add rule ip nat prerouting ip daddr 192.168.1.1 tcp dport 80 accept

* **Flags**:
  + add: Add a rule to the set.
  + delete: Remove a rule.
  + list: Show existing rules.

**26. systemctl (Systemd Control)**

Although primarily used for managing services, systemctl is often used to restart or control network-related services.

* **Basic usage**:

bash

CopyEdit

systemctl restart network.service

* **Real-world example**: Restart the network service after changing network configurations:

bash

CopyEdit

sudo systemctl restart networking

* **Flags**:
  + start: Start a service.
  + stop: Stop a service.
  + status: Display the current status of a service.
  + restart: Restart a service.

**27. fping**

Used to send ICMP echo requests to multiple hosts and display the results.

* **Basic usage**:

bash

CopyEdit

fping google.com

* **Real-world example**: Ping a range of IPs in a network:

bash

CopyEdit

fping -g 192.168.1.1 192.168.1.255

* **Flags**:
  + -g: Specify a range of IPs to ping.
  + -a: Show only reachable hosts.

**28. ethtool (Advanced Usage)**

Used to query and modify network interface parameters, especially for Ethernet devices.

* **Basic usage**:

bash

CopyEdit

ethtool eth0

* **Real-world example**: Change the speed of the interface eth0:

bash

CopyEdit

sudo ethtool -s eth0 speed 1000 duplex full autoneg off

* **Flags**:
  + -s <interface>: Set interface parameters.
  + -p: Blink the interface's LEDs.
  + -S: Display extended statistics.

**29. iftop**

A real-time console-based network bandwidth monitoring tool.

* **Basic usage**:

bash

CopyEdit

sudo iftop

* **Real-world example**: Display network traffic on a specific interface:

bash

CopyEdit

sudo iftop -i eth0

* **Flags**:
  + -i <interface>: Specify the network interface to monitor.
  + -B: Display bandwidth in bytes.
  + -t: Run in text-only mode.

**30. bmon (Bandwidth Monitor)**

Used to monitor bandwidth in real time.

* **Basic usage**:

bash

CopyEdit

bmon

* **Real-world example**: Display bandwidth usage for all interfaces:

bash

CopyEdit

bmon -p

* **Flags**:
  + -p: Show all available interfaces.
  + -c: Set the color scheme for output.

These additional commands round out the most commonly used networking commands in Linux. They cover a wide range of scenarios, from basic network configuration to advanced monitoring, troubleshooting, and security tasks.

what is interface in simple words tell me now

###### ChatGPT said:

An **interface** in simple terms is like a "connection point" that allows your computer or device to communicate with other devices or networks.

For example, if you're using a computer, the network interface could be the Wi-Fi or Ethernet connection that lets your computer connect to the internet or a local network. Just like a plug that connects your device to power or data, a network interface connects your computer to a network.

In short, it's the "bridge" between your device and the network, whether it's wireless (Wi-Fi) or wired (Ethernet).

1. ifconfig Purpose: Display and configure network interfaces. • ifconfig – Show all network interfaces and their IP addresses. • ifconfig <interface> – Show specific interface (e.g., eth0, wlan0). • ifconfig <interface> up – Bring up a network interface. • ifconfig <interface> down – Bring down a network interface. • ifconfig <interface> <IP> – Assign an IP address to an interface. • ifconfig <interface> netmask <netmask> – Set the netmask for the interface. 2. ip Purpose: A more modern and powerful alternative to ifconfig for network management. • ip a – Show all network interfaces and their IP addresses. • ip link show – Display link information (e.g., interface status). • ip addr show – Show IP addresses assigned to all interfaces. • ip link set <interface> up – Bring the interface up. • ip link set <interface> down – Bring the interface down. • ip addr add <IP>/24 dev <interface> – Assign a new IP to an interface. • ip route show – Display the routing table. • ip route add <destination> via <gateway> – Add a new route. 3. ping Purpose: Send ICMP echo requests to check network connectivity. • ping <hostname or IP> – Ping a host by its IP address or hostname. • ping -c <count> <hostname> – Send a specific number of packets. • ping -i <interval> <hostname> – Set the interval between packets. • ping -t <ttl> <hostname> – Set the TTL (Time To Live) for packets. • ping -s <size> <hostname> – Send packets of a specific size. • ping -4 <hostname> – Force the use of IPv4. • ping -6 <hostname> – Force the use of IPv6. 4. traceroute Purpose: Trace the route packets take to a destination. • traceroute <hostname> – Trace the route to a remote host. • traceroute -m <max\_hops> <hostname> – Set the maximum number of hops. • traceroute -T – Use TCP packets for tracing. • traceroute -I – Use ICMP packets for tracing. 5. nslookup Purpose: Query DNS to obtain domain name or IP address information. • nslookup <hostname> – Query the IP address of a given hostname. • nslookup <IP> – Query the domain name for a given IP. • nslookup -type=MX <domain> – Query MX (mail) records for a domain. • nslookup -type=NS <domain> – Query the name servers for a domain. 6. dig Purpose: Perform DNS lookups and query DNS servers. • dig <hostname> – Perform a basic DNS lookup for a domain. • dig @<DNS server> <hostname> – Query a specific DNS server. • dig <hostname> A – Query A (address) record for a domain. • dig <hostname> MX – Query MX (mail) record for a domain. • dig +short <hostname> – Shorten output to show only the result. 7. route Purpose: View and configure network routing tables. • route – Show the current routing table. • route -n – Show the routing table with numeric addresses (no DNS resolution). • route add <destination> gw <gateway> – Add a new route to a destination. • route del <destination> – Delete a specific route. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Advanced Networking Commands \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 8. netstat Purpose: Display network connections, routing tables, interface statistics, etc. • netstat – Show active connections and listening ports. • netstat -tuln – Display TCP and UDP ports that are open and listening. • netstat -an – Display all network connections (IP and port). • netstat -s – Show network statistics. • netstat -i – Show network interfaces and packet statistics. • netstat -r – Show the routing table. 9. ss Purpose: Utility to investigate sockets. • ss – Display all sockets. • ss -tuln – Display listening TCP and UDP sockets. • ss -a – Show all sockets (listening and non-listening). • ss -p – Display processes using the sockets. • ss -r – Display routing information. 10. netcat (nc) Purpose: A versatile tool for network debugging and banner grabbing. • nc -zv <hostname> <port> – Scan a specific port or range of ports. • nc -l <port> – Listen on a specific port for incoming connections. • nc <hostname> <port> – Connect to a host on a specific port. • nc -u <hostname> <port> – Use UDP instead of TCP. • nc -v <hostname> <port> – Verbose output for connection details. 11. tcpdump Purpose: Capture network traffic and display the packets. • tcpdump – Capture packets on the default interface. • tcpdump -i <interface> – Capture on a specific network interface. • tcpdump -n – Do not resolve hostnames or port names. • tcpdump -v – Show more verbose packet information. • tcpdump -c <count> – Capture a specific number of packets. • tcpdump -w <file> – Save the captured packets to a file. • tcpdump 'port 80' – Capture HTTP traffic. • tcpdump -A – Display packet contents in ASCII format. • tcpdump -X – Display packet contents in both hex and ASCII. 12. iftop Purpose: Display bandwidth usage on a network interface in real-time. • iftop – Display real-time bandwidth usage. • iftop -i <interface> – Specify the network interface to monitor. • iftop -n – Show numerical addresses (do not resolve hostnames). • iftop -P – Show port numbers along with IP addresses. 13. nmap Purpose: Network exploration and security auditing tool. • nmap <hostname> – Perform a simple scan of a host. • nmap -sP <IP-range> – Perform a ping scan to identify live hosts. • nmap -p <ports> <hostname> – Scan specific ports (e.g., -p 80,443). • nmap -sS <hostname> – Perform a SYN scan (stealth scan). • nmap -O <hostname> – OS detection. • nmap -A <hostname> – Enable OS detection, version detection, script scanning, and traceroute. 14. mtr Purpose: A network diagnostic tool combining traceroute and ping. • mtr <hostname> – Continuous traceroute and ping to a destination. • mtr -r <hostname> – Generate a report instead of continuous output. • mtr -c <count> <hostname> – Set the number of pings to send. 15. arp Purpose: View and manipulate the system's ARP (Address Resolution Protocol) cache. • arp -a – Display the current ARP table. • arp -d <IP> – Delete an entry from the ARP table. • arp -s <IP> <MAC> – Manually add a static entry to the ARP table. 16. ethtool Purpose: Query and control network device settings. • ethtool <interface> – Display information about a network interface. • ethtool -s <interface> speed <speed> duplex <duplex> – Change the speed and duplex settings of a network interface. • ethtool -p <interface> – Blink the LED on a network interface to help locate it. 17. iwconfig Purpose: Configure wireless network interfaces. • iwconfig – Display wireless network information. • iwconfig <interface> essid <SSID> – Connect to a specific Wi-Fi network. • iwconfig <interface> mode <mode> – Set the mode of the wireless interface (e.g., managed, ad-hoc). 18. ip link Purpose: Display or modify network device attributes. • ip link show – Display detailed information about network interfaces. • ip link set <interface> up/down – Bring the interface up or down. • ip link set <interface> mtu <mtu\_size> – Change the MTU size for an interface. 19. sshd (SSH Daemon) Purpose: Manage SSH access and configurations. • sshd -T – Show current configuration options. • sshd -f <config\_file> – Use a specific SSH configuration file. • sshd -D – Run in the foreground (useful for debugging). \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Cloud/Container Specific Networking Commands \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20. docker network Purpose: Manage Docker network settings. • docker network ls – List all Docker networks. • docker network inspect <network\_name> – Inspect a specific Docker network. • docker network create <network\_name> – Create a new Docker network. • docker network connect <network\_name> <container\_name> – Connect a container to a network. 21. kubectl get svc Purpose: Retrieve the details of Kubernetes services. • kubectl get svc – List all services in the current namespace. • kubectl get svc <service\_name> – Get details of a specific service. • kubectl describe svc <service\_name> – Detailed information on the service, including networking details. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ These commands, along with their options and flags, are crucial for diagnosing, configuring, and optimizing networking in any system or infrastructure. DevOps Engineers often use them to ensure reliable network performance, security, and troubleshooting in both on-premise and cloud-based environments. You said: please continue with more advance linux networking cmds used by devops engineer ChatGPT said: Here’s a continuation of advanced Linux networking commands that are typically used by DevOps Engineers for deep network diagnostics, configuration, performance monitoring, and troubleshooting. These commands are important when working with complex network setups, performance tuning, and cloud or containerized environments. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 22. ss (Socket Stat) Purpose: A more modern and faster alternative to netstat for analyzing socket connections. • ss -tuln – Display all listening TCP and UDP sockets. • ss -an – Show all network connections in numeric form (no DNS resolution). • ss -p – Show the processes associated with each socket. • ss -s – Display socket statistics (summary). • ss -o state established – Show only established connections. • ss -r – Display routing information along with socket statistics. • ss -A – Show application layer statistics. 23. nft (Netfilter) Purpose: Interface for configuring packet filtering and network address translation (NAT). • nft list ruleset – Display all the rules in the firewall. • nft add rule inet filter input tcp dport 22 accept – Add a rule to allow incoming SSH connections. • nft flush ruleset – Flush all rules from the ruleset. • nft delete rule inet filter input handle 4 – Delete a specific rule by its handle number. • nft list table inet filter – List the rules in the filter table. • nft add chain inet filter input { type filter hook input priority 0 \; } – Add a chain to filter incoming packets. 24. iptables Purpose: Configure packet filtering rules in the Linux kernel firewall. • iptables -L – List all active firewall rules. • iptables -A INPUT -p tcp --dport 80 -j ACCEPT – Allow incoming HTTP traffic. • iptables -A INPUT -p tcp --dport 443 -j ACCEPT – Allow incoming HTTPS traffic. • iptables -A INPUT -p tcp --dport 22 -s <IP> -j ACCEPT – Allow SSH traffic from a specific IP. • iptables -D INPUT -p tcp --dport 80 -j ACCEPT – Remove the rule allowing HTTP traffic. • iptables -F – Flush all rules in the filter table. • iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 192.168.1.2:8080 – Redirect HTTP traffic to a different destination. 25. firewalld Purpose: Manage firewall rules dynamically with firewalld. • firewall-cmd --list-all – Show the current firewall settings and rules. • firewall-cmd --zone=public --add-port=8080/tcp --permanent – Open port 8080 permanently. • firewall-cmd --reload – Reload the firewall to apply new rules. • firewall-cmd --zone=public --remove-port=8080/tcp --permanent – Close port 8080. • firewall-cmd --list-ports – Show open ports. 26. bpftrace Purpose: Dynamic tracing tool for observing and troubleshooting kernel-level events using BPF (Berkeley Packet Filter). • bpftrace -e 'tracepoint:syscalls:sys\_enter\_open { printf("%s opened\n", str(args->filename)); }' – Trace all open system calls. • bpftrace -e 'kprobe:vfs\_read { printf("PID %d read %d bytes\n", pid, args->count); }' – Trace file reads. • bpftrace -e 'tracepoint:net:net\_dev\_xmit { @[args->dev\_name] = count(); }' – Count network transmissions for each device. 27. tcpdump with Advanced Filtering Purpose: Advanced packet capture with specific filters. • tcpdump -i eth0 port 443 – Capture packets on port 443 (HTTPS). • tcpdump -i eth0 'tcp[tcpflags] & tcp-push != 0' – Capture TCP packets with the PUSH flag set. • tcpdump -i eth0 'src host <IP>' – Capture packets from a specific source IP. • tcpdump -i eth0 'dst host <IP>' – Capture packets destined to a specific IP. • tcpdump -i eth0 -w capture.pcap – Write captured packets to a file. • tcpdump -r capture.pcap – Read packets from a capture file. • tcpdump -i eth0 -v – Increase verbosity of the output. 28. ip rule Purpose: Used to configure policy-based routing. • ip rule show – Display the current routing rules. • ip rule add from <IP> table <table\_id> – Add a rule to route traffic from a specific IP through a custom routing table. • ip rule delete from <IP> – Delete the rule for traffic from a specific IP. • ip rule add iif <interface> table <table\_id> – Route traffic coming from a specific interface through a particular table. 29. tc (Traffic Control) Purpose: Control network traffic on interfaces. • tc qdisc show – Show the current queuing disciplines (QDisc) for interfaces. • tc qdisc add dev eth0 root netem delay 100ms – Add network delay to interface eth0. • tc qdisc change dev eth0 root netem loss 5% – Simulate 5% packet loss on eth0. • tc class add dev eth0 parent 1:1 classid 1:10 htb rate 1mbit – Add a traffic class with a rate limit. • tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 match ip dport 80 0xffff flowid 1:10 – Filter HTTP traffic and assign it to a specific traffic class. 30. ethtool (Advanced) Purpose: Query and modify network interface driver and hardware settings. • ethtool eth0 – Display settings for the eth0 network interface. • ethtool -s eth0 speed 1000 duplex full – Set the speed and duplex mode for the interface. • ethtool -p eth0 – Blink the LED to locate the physical interface. • ethtool -i eth0 – Display driver information for the network interface. • ethtool -S eth0 – Show statistics about the network interface. 31. nmcli Purpose: Command-line interface for NetworkManager (useful for managing network connections). • nmcli connection show – List all active network connections. • nmcli device status – Show the status of all network devices. • nmcli connection add type ethernet con-name eth0 ifname eth0 – Add a new Ethernet connection. • nmcli device disconnect eth0 – Disconnect a specific network interface. • nmcli device wifi list – List available Wi-Fi networks. 32. conntrack Purpose: Used to view and manage the connection tracking subsystem in the Linux kernel. • conntrack -L – List all tracked connections. • conntrack -F – Flush all tracked connections. • conntrack -D – Delete a specific connection from the connection table. • conntrack -L -p tcp – List all tracked TCP connections. • conntrack -E – Display connection events in real-time. 33. wireshark Purpose: A GUI-based network protocol analyzer (often used in tandem with tcpdump for deep packet inspection). • Launch Wireshark and capture packets from a specific interface. • wireshark – Start Wireshark GUI. • wireshark -i <interface> – Start Wireshark and capture packets from a specific interface. • Wireshark supports advanced filtering using display filters (e.g., http, ip.addr==192.168.1.1). 34. bridge Purpose: Configure and manage network bridges. • bridge link – Show the status of all network bridges. • bridge fdb show – Show the Forwarding Database (FDB) for a bridge. • bridge vlan – Show or configure VLANs on a bridge. • bridge addbr br0 – Create a new network bridge. • bridge addif br0 eth0 – Add an interface to the bridge. 35. bmon Purpose: Bandwidth monitor and rate estimator for Linux. • bmon – Display bandwidth usage in real-time. • bmon -p – Display statistics for all interfaces. • bmon -c – Show detailed statistics in a continuous mode. - create examples from scratch for all these commands and apply all these commands - with all the flags show examplse - first create and then apply all thse commands please -so that i can clearly understand

**ChatGPT said:**

Let's break down the advanced Linux networking commands, step-by-step, to understand how to use them effectively in a real-world scenario. We'll create some basic setup for network interfaces, traffic, and connections, and then apply each of the commands and their flags to monitor, configure, and troubleshoot network operations.

**Step 1: Network Setup and Configuration for Testing**

We'll begin by creating some basic configurations for network interfaces and simulate traffic.

bash

CopyEdit

# Create network interfaces (use actual interface names in your environment, e.g., eth0 or wlan0)

# Let's assume the interfaces are already up

# (For example, using eth0 and wlan0 in the tests)

# Set static IPs for testing

sudo ifconfig eth0 192.168.1.100 netmask 255.255.255.0 up

sudo ifconfig wlan0 192.168.1.101 netmask 255.255.255.0 up

# Bring down the interfaces to simulate changes

sudo ifconfig eth0 down

sudo ifconfig wlan0 down

# Bring them back up

sudo ifconfig eth0 up

sudo ifconfig wlan0 up

# Test with a dummy interface to simulate IP assignment

sudo ifconfig dummy0 192.168.1.102 netmask 255.255.255.0 up

# Create some traffic (ping the gateway or another device in your network)

ping 192.168.1.1 -c 4

Now that we have the setup, let's move to each command and apply them one by one.

**1. ifconfig - Display and Configure Network Interfaces**

bash

CopyEdit

# Show all network interfaces and their IP addresses

ifconfig

# Show specific interface information (e.g., eth0)

ifconfig eth0

# Bring up a network interface

ifconfig eth0 up

# Bring down a network interface

ifconfig eth0 down

# Assign an IP address to an interface

ifconfig eth0 192.168.1.200

# Set the netmask for the interface

ifconfig eth0 netmask 255.255.255.0

**2. ip - A Modern Alternative for Network Management**

bash

CopyEdit

# Show all network interfaces and their IP addresses

ip a

# Display link information (status of the interfaces)

ip link show

# Show IP addresses assigned to all interfaces

ip addr show

# Bring an interface up

ip link set eth0 up

# Bring an interface down

ip link set eth0 down

# Assign a new IP to an interface

ip addr add 192.168.1.201/24 dev eth0

# Display the routing table

ip route show

# Add a new route

ip route add 10.0.0.0/24 via 192.168.1.1

**3. ping - Send ICMP Echo Requests**

bash

CopyEdit

# Ping a host by IP address

ping 192.168.1.1

# Ping a host by hostname

ping google.com

# Send a specific number of packets (e.g., 4 packets)

ping -c 4 google.com

# Set the interval between packets (e.g., 1 second)

ping -i 1 google.com

# Set the TTL (Time To Live) for packets

ping -t 64 google.com

# Send packets of a specific size (e.g., 128 bytes)

ping -s 128 google.com

# Force the use of IPv4

ping -4 google.com

# Force the use of IPv6

ping -6 google.com

**4. traceroute - Trace the Route of Packets**

bash

CopyEdit

# Trace the route to a remote host

traceroute google.com

# Set the maximum number of hops

traceroute -m 30 google.com

# Use TCP packets for tracing

traceroute -T google.com

# Use ICMP packets for tracing

traceroute -I google.com

**5. nslookup - Query DNS for Domain or IP Information**

bash

CopyEdit

# Query the IP address of a given hostname

nslookup google.com

# Query the domain name for a given IP address

nslookup 8.8.8.8

# Query MX records for a domain

nslookup -type=MX example.com

# Query the name servers for a domain

nslookup -type=NS example.com

**6. dig - Perform DNS Lookups**

bash

CopyEdit

# Perform a basic DNS lookup for a domain

dig google.com

# Query a specific DNS server

dig @8.8.8.8 google.com

# Query A (address) record for a domain

dig google.com A

# Query MX (mail) record for a domain

dig google.com MX

# Shorten output to show only the result

dig +short google.com

**7. route - View and Configure Routing Tables**

bash

CopyEdit

# Show the current routing table

route

# Show the routing table with numeric addresses (no DNS resolution)

route -n

# Add a new route to a destination

route add -net 10.0.0.0 netmask 255.255.255.0 gw 192.168.1.1

# Delete a specific route

route del -net 10.0.0.0 netmask 255.255.255.0

**8. netstat - Display Network Connections, Routing Tables, etc.**

bash

CopyEdit

# Show active connections and listening ports

netstat

# Display TCP and UDP ports that are open and listening

netstat -tuln

# Display all network connections with IP and port

netstat -an

# Show network statistics

netstat -s

# Show network interfaces and packet statistics

netstat -i

# Show the routing table

netstat -r

**9. ss - Analyze Socket Connections (Faster than netstat)**

bash

CopyEdit

# Display all sockets

ss

# Display listening TCP and UDP sockets

ss -tuln

# Show all sockets (listening and non-listening)

ss -a

# Display processes using the sockets

ss -p

# Display routing information along with socket statistics

ss -r

# Show application layer statistics

ss -A

**10. netcat (nc) - Network Debugging and Banner Grabbing**

bash

CopyEdit

# Scan a specific port or range of ports

nc -zv 192.168.1.1 80-443

# Listen on a specific port for incoming connections

nc -l 8080

# Connect to a host on a specific port

nc 192.168.1.1 80

# Use UDP instead of TCP

nc -u 192.168.1.1 1234

# Verbose output for connection details

nc -v 192.168.1.1 80

**11. tcpdump - Capture Network Traffic**

bash

CopyEdit

# Capture packets on the default interface

tcpdump

# Capture on a specific network interface

tcpdump -i eth0

# Do not resolve hostnames or port names

tcpdump -n

# Show more verbose packet information

tcpdump -v

# Capture a specific number of packets

tcpdump -c 10

# Save captured packets to a file

tcpdump -w capture.pcap

# Capture HTTP traffic

tcpdump 'port 80'

# Display packet contents in ASCII format

tcpdump -A

# Display packet contents in both hex and ASCII

tcpdump -X

**12. iftop - Display Real-Time Bandwidth Usage**

bash

CopyEdit

# Display real-time bandwidth usage

iftop

# Specify the network interface to monitor

iftop -i eth0

# Show numerical addresses (do not resolve hostnames)

iftop -n

# Show port numbers along with IP addresses

iftop -P

**13. nmap - Network Exploration and Security Auditing**

bash

CopyEdit

# Perform a simple scan of a host

nmap 192.168.1.1

# Perform a ping scan to identify live hosts

nmap -sP 192.168.1.0/24

# Scan specific ports (e.g., HTTP and HTTPS ports)

nmap -p 80,443 192.168.1.1

# Perform a SYN scan (stealth scan)

nmap -sS 192.168.1.1

# OS detection

nmap -O 192.168.1.1

# Enable OS detection, version detection, script scanning, and traceroute

nmap -A 192.168.1.1

**14. mtr - Network Diagnostic Tool Combining Traceroute and Ping**

bash

CopyEdit

# Continuous traceroute and ping to a destination

mtr google.com

# Generate a report instead of continuous output

mtr -r google.com

# Set the number of pings to send

mtr -c 10 google.com

**15. arp - View and Manipulate ARP Cache**

bash

CopyEdit

# Display the current ARP table

arp -a

# Delete an entry from the ARP table

arp -d 192.168.1.1

# Manually add a static entry to the ARP table

arp -s 192.168.1.50 00:14:22:01:23:45

**16. ethtool - Query and Control Network Device Settings**

bash

CopyEdit

# Display settings for the eth0 network interface

ethtool eth0

# Set the speed and duplex mode for the interface

ethtool -s eth0 speed 1000 duplex full

# Blink the LED to locate the physical interface

ethtool -p eth0

# Display driver information for the network interface

ethtool -i eth0

# Show statistics about the network interface

ethtool -S eth0

**17. iwconfig - Configure Wireless Network Interfaces**

bash

CopyEdit

# Display wireless network information

iwconfig

# Connect to a specific Wi-Fi network

iwconfig wlan0 essid "MyWiFiNetwork"

# Set the mode of the wireless interface

iwconfig wlan0 mode managed

**18. ip link - Display/Modify Network Device Attributes**

bash

CopyEdit

# Display detailed information about network interfaces

ip link show

# Bring an interface up or down

ip link set eth0 up

ip link set eth0 down

# Change the MTU size for an interface

ip link set eth0 mtu 1500

**19. sshd - Manage SSH Daemon Configurations**

bash

CopyEdit

# Show current configuration options

sshd -T

# Use a specific SSH configuration file

sshd -f /etc/ssh/sshd\_config

# Run SSH daemon in the foreground (useful for debugging)

sshd -D

**Cloud/Container-Specific Networking Commands**

**20. docker network - Manage Docker Network Settings**

bash

CopyEdit

# List all Docker networks

docker network ls

# Inspect a specific Docker network

docker network inspect bridge

# Create a new Docker network

docker network create my\_network

# Connect a container to a network

docker network connect my\_network my\_container

**21. kubectl get svc - Retrieve Kubernetes Service Details**

bash

CopyEdit

# List all services in the current namespace

kubectl get svc

# Get details of a specific service

kubectl get svc my-service

# Describe a specific service

kubectl describe svc my-service

These commands, when applied in a real environment, help a DevOps engineer manage and monitor network connectivity, performance, and troubleshooting in Linux-based systems, both on-premise and in cloud/container environments. By practicing these commands in real-time, you can effectively troubleshoot and optimize network settings across systems.

4o

**You said:**

please compleete all these cmds of networking

**ChatGPT said:**

Sure! Below is a detailed continuation of the advanced Linux networking commands for DevOps engineers. I'll create examples for each of the commands, and provide explanations for how they work and their use cases.

**22. ss - Socket Stat**

The ss command is used to analyze socket connections and is a faster and more modern alternative to netstat.

bash

CopyEdit

# Display all sockets

ss

# Display listening TCP and UDP sockets

ss -tuln

# Show all sockets (listening and non-listening)

ss -a

# Display processes using the sockets

ss -p

# Display routing information along with socket statistics

ss -r

# Show application layer statistics

ss -A

# Show only established connections

ss -o state established

**23. nft - Netfilter (Firewall)**

nft is the tool to configure packet filtering and NAT (Network Address Translation) in the Linux kernel. It is used for advanced firewall configurations.

bash

CopyEdit

# Display all the rules in the firewall

nft list ruleset

# Add a rule to allow incoming SSH connections on port 22

nft add rule inet filter input tcp dport 22 accept

# Flush all rules from the ruleset

nft flush ruleset

# Delete a specific rule by its handle number (replace 4 with actual handle)

nft delete rule inet filter input handle 4

# List the rules in the filter table

nft list table inet filter

# Add a chain to filter incoming packets

nft add chain inet filter input { type filter hook input priority 0 ; }

**24. iptables - Configure Packet Filtering**

iptables is a classic utility to set up, maintain, and inspect the IP packet filter rules in the Linux kernel.

bash

CopyEdit

# List all active firewall rules

iptables -L

# Allow incoming HTTP traffic on port 80

iptables -A INPUT -p tcp --dport 80 -j ACCEPT

# Allow incoming HTTPS traffic on port 443

iptables -A INPUT -p tcp --dport 443 -j ACCEPT

# Allow SSH traffic from a specific IP address

iptables -A INPUT -p tcp --dport 22 -s 192.168.1.100 -j ACCEPT

# Remove the rule allowing HTTP traffic

iptables -D INPUT -p tcp --dport 80 -j ACCEPT

# Flush all rules in the filter table

iptables -F

# Redirect HTTP traffic to a different destination IP and port

iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 192.168.1.2:8080

**25. firewalld - Manage Firewall Rules Dynamically**

firewalld is a firewall management tool available on many Linux distributions, providing a dynamic way to manage firewall rules.

bash

CopyEdit

# Show the current firewall settings and rules

firewall-cmd --list-all

# Open port 8080 permanently in the public zone

firewall-cmd --zone=public --add-port=8080/tcp --permanent

# Reload the firewall to apply new rules

firewall-cmd --reload

# Close port 8080 permanently

firewall-cmd --zone=public --remove-port=8080/tcp --permanent

# Show open ports in the firewall

firewall-cmd --list-ports

**26. bpftrace - Dynamic Tracing Tool**

bpftrace allows users to trace kernel-level events using BPF (Berkeley Packet Filter). It is used for real-time debugging and performance analysis.

bash

CopyEdit

# Trace all open system calls

bpftrace -e 'tracepoint:syscalls:sys\_enter\_open { printf("%s opened\n", str(args->filename)); }'

# Trace file reads

bpftrace -e 'kprobe:vfs\_read { printf("PID %d read %d bytes\n", pid, args->count); }'

# Count network transmissions for each device

bpftrace -e 'tracepoint:net:net\_dev\_xmit { @[args->dev\_name] = count(); }'

**27. tcpdump with Advanced Filtering**

tcpdump is used to capture network traffic and display the captured packets. It can apply advanced filtering for deeper analysis.

bash

CopyEdit

# Capture packets on port 443 (HTTPS)

tcpdump -i eth0 port 443

# Capture TCP packets with the PUSH flag set

tcpdump -i eth0 'tcp[tcpflags] & tcp-push != 0'

# Capture packets from a specific source IP

tcpdump -i eth0 'src host 192.168.1.100'

# Capture packets destined to a specific IP

tcpdump -i eth0 'dst host 192.168.1.100'

# Write captured packets to a file

tcpdump -i eth0 -w capture.pcap

# Read packets from a capture file

tcpdump -r capture.pcap

# Increase verbosity of the output

tcpdump -i eth0 -v

# Display packet contents in ASCII format

tcpdump -A

# Display packet contents in both hex and ASCII

tcpdump -X

**28. ip rule - Policy-Based Routing**

ip rule is used to configure and manage policy-based routing, which allows routing decisions to be based on various criteria like source IP, interface, or destination.

bash

CopyEdit

# Display the current routing rules

ip rule show

# Add a rule to route traffic from a specific IP through a custom routing table

ip rule add from 192.168.1.100 table 100

# Delete the rule for traffic from a specific IP

ip rule delete from 192.168.1.100

# Route traffic coming from a specific interface through a particular table

ip rule add iif eth0 table 100

**29. tc - Traffic Control**

tc is used to control the traffic on network interfaces, such as adding network delay or limiting bandwidth.

bash

CopyEdit

# Show the current queuing disciplines (QDisc) for interfaces

tc qdisc show

# Add network delay to interface eth0

tc qdisc add dev eth0 root netem delay 100ms

# Simulate 5% packet loss on eth0

tc qdisc change dev eth0 root netem loss 5%

# Add a traffic class with a rate limit

tc class add dev eth0 parent 1:1 classid 1:10 htb rate 1mbit

# Filter HTTP traffic and assign it to a specific traffic class

tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 match ip dport 80 0xffff flowid 1:10

**30. ethtool (Advanced)**

ethtool is used to query and modify network interface settings, such as changing the interface speed and duplex settings.

bash

CopyEdit

# Display settings for the eth0 network interface

ethtool eth0

# Set the speed and duplex mode for the interface

ethtool -s eth0 speed 1000 duplex full

# Blink the LED to locate the physical interface

ethtool -p eth0

# Display driver information for the network interface

ethtool -i eth0

# Show statistics about the network interface

ethtool -S eth0

**31. nmcli - NetworkManager Command-Line Interface**

nmcli is used to manage network connections on systems with NetworkManager.

bash

CopyEdit

# List all active network connections

nmcli connection show

# Show the status of all network devices

nmcli device status

# Add a new Ethernet connection

nmcli connection add type ethernet con-name eth0 ifname eth0

# Disconnect a specific network interface

nmcli device disconnect eth0

# List available Wi-Fi networks

nmcli device wifi list

**32. conntrack - Connection Tracking Subsystem**

conntrack is used to manage the connection tracking table in the Linux kernel, useful for firewall and NAT configurations.

bash

CopyEdit

# List all tracked connections

conntrack -L

# Flush all tracked connections

conntrack -F

# Delete a specific connection from the connection table

conntrack -D

# List all tracked TCP connections

conntrack -L -p tcp

# Display connection events in real-time

conntrack -E

**33. wireshark - GUI-Based Network Protocol Analyzer**

Wireshark is a GUI-based tool for analyzing network traffic, often used in conjunction with tcpdump for deep packet inspection.

bash

CopyEdit

# Launch Wireshark and capture packets from a specific interface

wireshark -i eth0

# Wireshark also supports advanced filtering using display filters

# For example, filter HTTP traffic

wireshark -i eth0 "http"

# Filter packets by source IP

wireshark -i eth0 "ip.addr==192.168.1.1"

**34. bridge - Manage Network Bridges**

bridge is used to configure network bridges, such as adding interfaces to a bridge or viewing the Forwarding Database (FDB).

bash

CopyEdit

# Show the status of all network bridges

bridge link

# Show the Forwarding Database (FDB) for a bridge

bridge fdb show

# Show or configure VLANs on a bridge

bridge vlan

# Create a new network bridge

bridge addbr br0

# Add an interface to the bridge

bridge addif br0 eth0

**35. bmon - Bandwidth Monitor and Rate Estimator**

bmon is a tool used for monitoring bandwidth usage and estimating network traffic.

bash

CopyEdit

# Display real-time bandwidth usage

bmon

# Display statistics for all interfaces

bmon -p

# Show detailed statistics in continuous mode

bmon -c

These are the advanced Linux networking commands used by De