Linux Networking Quick Reference

IP, NetworkManager, Wi-Fi, Bridging, SSH and Network Tests

Examples use `eth0`, `wlan0`, `192.168.10.20` and `pi@192.168.1.50`. Changes usually require `sudo`.

1. Status, IP and Reachability

Interfaces, Addresses and Routes

`ip -br link`	Interfaces and link state
`ip -br addr`	Compact IP address view
`ip route`	Routing table
`ip route get 1.1.1.1`	Show the selected route
`ip neigh`	ARP and neighbor table
`ss -lntup`	Listening TCP/UDP ports

NetworkManager

`nmcli general status`	Overall state
`nmcli device status`	Devices and connections
`nmcli con show -active`	Active profiles
`nmcli device show eth0`	Details for eth0
`nmcli networking off / on`	Networking off / on

Ping and Name Resolution

`ping -c 4 192.168.1.1`	Test the gateway
`ping -c 4 1.1.1.1`	Test Internet without DNS
`ping -c 4 example.com`	Test Internet and DNS
`getent hosts example.com`	Check name resolution
`ping -I eth0 -c 4 1.1.1.1`	Use a specific interface

Link Details and Packet Capture

`ifconfig eth0`	Legacy view; package `net-tools`
`ethtool eth0`	Link, speed and duplex
`ethtool -S eth0`	Driver and error counters
`sudo tcpdump -ni eth0`	Show all packets live
`sudo tcpdump -ni eth0 port 22`	Show SSH traffic only
`sudo tcpdump -ni eth0 -w capture.pcap`	Save capture as PCAP

2. Configure Wi-Fi, IP and Bridges

Connect Wi-Fi

`nmcli dev wifi list`	List networks
`nmcli dev wifi connect Lab-WLAN password geheim ifname wlan0`	Connect and save a profile
`nmcli con down Lab-WLAN`	Disconnect profile
`nmcli con up Lab-WLAN`	Activate saved profile
`nmcli dev disconnect wlan0`	Disconnect device now

Static IPv4 Address

`nmcli con add type ethernet ifname eth0 con-name lab-static ip4 192.168.10.20/24 gw4 192.168.10.1`	Create profile with address and gateway
`nmcli con mod lab-static ipv4.dns "192.168.10.1 1.1.1.1"`	Set DNS servers
`nmcli con up lab-static`	Activate profile
`nmcli con mod lab-static ipv4.method auto`	Return to DHCP

Ethernet Bridge

`nmcli con add type bridge con-name br0 ifname br0`	Create bridge profile
`nmcli con add type ethernet con-name br0-eth0 ifname eth0 master br0`	Add eth0 as a port
`nmcli con up br0`	Activate the bridge
`bridge link`	Inspect bridge ports
`ip -br addr show br0`	Address belongs on br0

Wi-Fi Hotspot and Bridge Note

`nmcli dev wifi hotspot ifname wlan0 con-name lab-hotspot ssid Lab-Hotspot password test12345`	Create a routed hotspot
`nmcli con down lab-hotspot`	Stop the hotspot

A Wi-Fi client normally cannot act as a transparent Layer-2 bridge port in standard 802.11 client mode. Use AP mode, routing/NAT, or hardware with 4-address/WDS support.

3. Remote Access and Network Tests

SSH

`ssh pi@192.168.1.50`	Log in
`ssh -p 2222 pi@192.168.1.50`	Use another port
`ssh-copy-id pi@192.168.1.50`	Install an SSH key
`ssh pi@192.168.1.50 'ip -br addr'`	Run a remote command
`ssh -v pi@192.168.1.50`	Debug the connection

SCP

`scp test.txt pi@192.168.1.50:/tmp/`	Upload a file
`scp pi@192.168.1.50:/var/log/app.log .`	Download a file
`scp -r results/ pi@192.168.1.50:/tmp/`	Copy a directory
`scp -P 2222 test.txt pi@192.168.1.50:/tmp/`	Use another SSH port

iperf3

`iperf3 -s`	Start server
`iperf3 -c 192.168.1.50`	TCP test to server
`iperf3 -c 192.168.1.50 -R`	Test reverse direction
`iperf3 -c 192.168.1.50 -t 30`	Measure for 30 seconds
`iperf3 -c 192.168.1.50 -t 0`	Send indefinitely; stop with Ctrl+C
`iperf3 -c 192.168.1.50 -u -b 8M`	UDP at 8 Mbit/s

Services and Logs

`systemctl status NetworkManager.service ssh.service`	Show service status
`sudo systemctl start ssh.service`	Start / stop now
`sudo systemctl stop ssh.service`	
`sudo systemctl restart NetworkManager.service`	Restart a service
`sudo systemctl enable -now ssh.service`	Change boot and runtime state together
`sudo systemctl disable -now ssh.service`	
`systemctl -failed`	Failed units
`sudo journalctl -b`	All or NetworkManager logs since boot
`sudo journalctl -u NetworkManager.service -b`	
`sudo journalctl -u ssh.service -f`	Follow SSH logs live

`start` changes only runtime state; `enable` only automatic startup at boot.

Units, Custom Services and Boot

`systemctl list-units -type=service`	Loaded service units
`systemctl list-unit-files -type=service`	List service unit files
`sudoedit /etc/systemd/system/my-app.service`	Edit a custom service
`sudo systemctl daemon-reload && sudo systemctl restart my-app.service`	Apply changes
`systemctl get-default`	Show boot target
`systemctl list-units -type=target`	List active targets
`systemctl list-timers -all`	List timers

Troubleshooting in Four Steps

1. Link and Device

`ip -br link` `nmcli device status`

Is the interface present and connected?

2. Address and Route

`ip -br addr` `ip route`

Is there an address, prefix and default route?

3. Reachability and DNS

`ping gateway` `ping 1.1.1.1` `getent hosts example.com`

This separates link, routing and DNS failures.

4. Service and Throughput

`ss -lntup` `systemctl status ssh.service` `iperf3`

Check port, service, firewall and performance.

BE-IIS
BRECHEL ELECTRONIC
Industrial Interface Solutions

Official NetworkManager reference:

networkmanager.dev/docs/api/latest/