Linux Networking Quick Reference

IP, NetworkManager, WLAN, Bridge, SSH und Netzwerktests

Konkrete Beispiele verwenden `eth0`, `wlan0`, `192.168.10.20` und `pi@192.168.1.50`. Änderungen benötigen meist `sudo`.

1. Status, IP und Erreichbarkeit

Interfaces, Adressen und Routen

`ip -br link`	Interfaces und Link-Status
`ip -br addr`	IP-Adressen kompakt
`ip route`	Routing-Tabelle
`ip route get 1.1.1.1`	Gewählten Weg prüfen
`ip neigh`	ARP- und Nachbar-Tabelle
`ss -lntup`	Lauschende TCP/UDP-Ports

NetworkManager

`nmcli general status`	Gesamtzustand
`nmcli device status`	Geräte und Verbindung
`nmcli con show -active`	Aktive Profile
`nmcli device show eth0`	Details zu eth0
`nmcli networking off / on`	Netzwerk aus / ein

Ping und Namensauflösung

`ping -c 4 192.168.1.1`	Gateway testen
`ping -c 4 1.1.1.1`	Internet ohne DNS testen
`ping -c 4 example.com`	Internet und DNS testen
`getent hosts example.com`	DNS-Auflösung prüfen
`ping -I eth0 -c 4 1.1.1.1`	Bestimmtes Interface nutzen

Link-Details und Paketmitschnitt

`ifconfig eth0`	Klassische Ansicht; Paket `net-tools`
`ethtool eth0`	Link, Geschwindigkeit und Duplex
`ethtool -S eth0`	Treiber- und Fehlerzähler
`sudo tcpdump -ni eth0`	Alle Pakete live anzeigen
`sudo tcpdump -ni eth0 port 22`	Nur SSH-Verkehr anzeigen
`sudo tcpdump -ni eth0 -w capture.pcap`	Mitschnitt als PCAP speichern

2. WLAN, IP und Bridge konfigurieren

WLAN verbinden

`nmcli dev wifi list`	Netze anzeigen
`nmcli dev wifi connect Lab-WLAN password geheim ifname wlan0`	Verbinden und Profil speichern
`nmcli con down Lab-WLAN`	Verbindung trennen
`nmcli con up Lab-WLAN`	Gespeichertes Profil aktivieren
`nmcli dev disconnect wlan0`	Gerät sofort trennen

Statische IPv4-Adresse

`nmcli con add type ethernet ifname eth0 con-name lab-static ip4 192.168.10.20/24 gw4 192.168.10.1`	Profil mit Adresse und Gateway anlegen
`nmcli con mod lab-static ipv4.dns "192.168.10.1 1.1.1.1"`	DNS-Server setzen
`nmcli con up lab-static`	Profil aktivieren
`nmcli con mod lab-static ipv4.method auto`	Zurück auf DHCP

Ethernet-Bridge

`nmcli con add type bridge con-name br0 ifname br0`	Bridge-Profil anlegen
`nmcli con add type ethernet con-name br0-eth0 ifname eth0 master br0`	eth0 als Port hinzufügen
`nmcli con up br0`	Bridge aktivieren
`bridge link`	Bridge-Ports prüfen
`ip -br addr show br0`	Adresse liegt auf br0

WLAN-Hotspot und Bridge-Hinweis

`nmcli dev wifi hotspot ifname wlan0 con-name lab-hotspot ssid Lab-Hotspot password test12345`	Gerouteten Hotspot erstellen
`nmcli con down lab-hotspot`	Hotspot stoppen

Ein WLAN-Client lässt sich im normalen 802.11-Client-Modus meist nicht transparent als Layer-2-Bridge-Port verwenden. Nutze AP-Modus, Routing/NAT oder Hardware mit 4-Address/WDS-Unterstützung.

3. Remote-Zugriff und Netzwerktests

SSH

`ssh pi@192.168.1.50`	Anmelden
`ssh -p 2222 pi@192.168.1.50`	Anderen Port verwenden
`ssh-copy-id pi@192.168.1.50`	SSH-Schlüssel installieren
`ssh pi@192.168.1.50 'ip -br addr'`	Remote-Befehl ausführen
`ssh -v pi@192.168.1.50`	Verbindung debuggen

SCP

`scp test.txt pi@192.168.1.50:/tmp/`	Datei hochladen
`scp pi@192.168.1.50:/var/log/app.log .`	Datei herunterladen
`scp -r results/ pi@192.168.1.50:/tmp/`	Ordner kopieren
`scp -P 2222 test.txt pi@192.168.1.50:/tmp/`	Anderen SSH-Port verwenden

iperf3

`iperf3 -s`	Server starten
`iperf3 -c 192.168.1.50`	TCP zum Server testen
`iperf3 -c 192.168.1.50 -R`	Rückrichtung testen
`iperf3 -c 192.168.1.50 -t 30`	30 Sekunden messen
`iperf3 -c 192.168.1.50 -t 0`	Unbegrenzt senden; Abbruch mit Ctrl+C
`iperf3 -c 192.168.1.50 -u -b 8M`	UDP mit 8 Mbit/s

Dienste und Logs

`systemctl status NetworkManager.service ssh.service`	Dienststatus anzeigen
`sudo systemctl start ssh.service`	Jetzt starten / stoppen
`sudo systemctl stop ssh.service`	
`sudo systemctl restart NetworkManager.service`	Dienst neu starten
`sudo systemctl enable -now ssh.service`	Boot-Start und Laufzustand gemeinsam ändern
`sudo systemctl disable -now ssh.service`	
`systemctl -failed`	Fehlgeschlagene Units
`sudo journalctl -b`	Alle bzw. NM-Logs seit Boot
`sudo journalctl -u NetworkManager.service -b`	
`sudo journalctl -u ssh.service -f`	SSH-Logs live verfolgen

`start` ändert nur den aktuellen Zustand; `enable` nur den automatischen Boot-Start.

Units, eigene Dienste und Boot

`systemctl list-units -type=service`	Geladene Service-Units
`systemctl list-unit-files -type=service`	Service-Dateien auflisten
`sudoedit /etc/systemd/system/my-app.service`	Eigenen Dienst bearbeiten
`sudo systemctl daemon-reload && sudo systemctl restart my-app.service`	Änderungen übernehmen
`systemctl get-default`	Boot-Target anzeigen
`systemctl list-units -type=target`	Aktive Targets anzeigen
`systemctl list-timers -all`	Timer anzeigen

Fehleranalyse in vier Schritten

1. Link und Gerät

`ip -br link` `nmcli device status`

Ist das Interface vorhanden und verbunden?

2. Adresse und Route

`ip -br addr` `ip route`

Gibt es Adresse, Netzmaske und Default-Route?

3. Erreichbarkeit und DNS

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

So trennst du Link-, Routing- und DNS-Fehler.

4. Dienst und Datenrate

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

Port, Dienst, Firewall und Leistung prüfen.

BE-IIS
BRECHEL ELECTRONIC
Industrial Interface Solutions

Offizielle NetworkManager-Referenz:

networkmanager.dev/docs/api/latest/