ADMIN:DOCS

Linux Administration #4: Netzwerkkonfiguration und Verwaltung

Lerne die Grundlagen der Linux-Netzwerkkonfiguration: Von der physischen Netzwerkebene über DNS-Konfiguration bis hin zu Firewall-Einstellungen und Netzwerk-Monitoring.

Kategorie: Linux-Administration **Veröffentlicht:** 01.11.2024 **Zuletzt geprüft:** 02.11.2026

Willkommen zum vierten Teil unserer technischen Wiki-Serie über Linux-Administration!

Nachdem wir in den vorherigen Artikeln die Grundlagen der Systemadministration, Benutzerverwaltung und Prozessverwaltung behandelt haben, widmen wir uns heute der Netzwerkkonfiguration und -verwaltung.

Als Linux-Administrator ist es wichtig zu verstehen, wie Netzwerke funktionieren und wie du sie konfigurierst. Stell dir ein Netzwerk wie ein Postsystem vor: Jeder Computer hat eine eigene IP-Adresse, und die Netzwerkkonfiguration bestimmt, wie Datenpakete zwischen den Systemen vermittelt und übertragen werden.

> ⚠ **Hinweis:** In diesem Artikel verwenden wir Ubuntu/Debian als primäre Referenz-Distributionen. Die grundlegenden Netzwerk- und Kernelkonzepte (`iproute2` , `nftables` , `systemd-resolved`) sind auf allen modernen Linux-Systemen identisch; Paketmanagement und deklarative Konfigurationstools können sich jedoch unterscheiden.

Quelle: https://admindocs.de/de/linux-administration/linux-administration-netzwerkkonfiguration-und-verwaltung

Was ist ein Netzwerk?

Ein Netzwerk verbindet Computer und ermöglicht den zuverlässigen Datenaustausch. In Linux wird die Netzwerkfunktionalität direkt im Kernel über den Netzwerk-Stack abgewickelt und über moderne Userspace-Werkzeuge gesteuert:

- **Physische & Verbindungsebene:** Netzwerkschnittstellen (NICs), MAC-Hardwareadressen, Link-Status und Duplex-Modi.
- **Logische Netzwerkebene:** IP-Adressierung (IPv4/IPv6), Subnetze, Routing-Tabellen und Standard-Gateways.
- **Namensauflösung & Transport:** DNS-Resolution, Portverwaltung (TCP/UDP) und Paketfilterung.

Netzwerk-Grundlagen verstehen

In Linux-Systemen ist das Netzwerk in klar voneinander getrennte Schichten unterteilt:

Physische Ebene & Schnittstellen

Netzwerkschnittstellen repräsentieren reale oder virtuelle Netzwerkadapter:

- **Gerätenamen:** Moderne Linux-Systeme nutzen *Predictable Network Interface Names* (z. B. `enp3s0` für PCI-Ethernet, `wlp2s0` für WLAN oder `eth0` in virtualisierten Containern).
- **MAC-Adresse:** Weltweit eindeutige Hardware-Adresse (`link/ether`).
- **Verbindungsstatus:** Prüft, ob ein physikalischer Link anliegt (`LOWER_UP`) und das Interface administrativ aktiviert ist (`UP`).

```
# Alle Netzwerkschnittstellen und deren Link-Status anzeigen
ip link show

# Details einer bestimmten Schnittstelle (z. B. eth0 oder enp3s0)
ip link show eth0
```

Beispielausgabe:

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT
group default qlen 1000
    link/ether 00:11:22:33:44:55 brd ff:ff:ff:ff:ff:ff
```

> **Hinweis: `net-tools` (`ifconfig`) ist Legacy** Befehle wie `ifconfig` , `route` und `netstat` stammen aus dem Paket `net-tools` und sind seit über 15 Jahren veraltet. Auf modernen Linux-Servern ist `net-tools` standardmäßig nicht mehr installiert. Verwende stattdessen ausschließlich die Befehle der modernen **`iproute2`** -Suite (`ip link` , `ip addr` , `ip route`) sowie **`ss`** .

Logische Ebene

Die logische Ebene regelt Adressierung, Routing und Namensauflösung:

- **IP-Adressen:** Eindeutige logische Identifikatoren im Netzwerk (IPv4 mit 32 Bit, IPv6 mit 128 Bit).
- **Subnetzmaske / CIDR:** Definiert, welcher Teil der IP das Netzwerk und welcher den einzelnen Host adressiert (z. B. `/24` = `255.255.255.0`).
- **Default Gateway:** Der Router, an den alle Pakete weitergeleitet werden, deren Zieladresse nicht im lokalen Subnetz liegt.

Die logische Netzwerkebene verstehen

IP-Adressierung (IPv4 & Private Netze)

IPv4-Adressen bestehen aus vier Dezimalblöcken (Oktetten) von 0 bis 255. Nach RFC 1918 sind bestimmte Adressbereiche für private lokale Netzwerke reserviert und werden im Internet nicht geroutet:

- **Klasse A (Große Netze):** `10.0.0.0/8` (`10.0.0.0` bis `10.255.255.255`)
- **Klasse B (Mittlere Netze):** `172.16.0.0/12` (`172.16.0.0` bis `172.31.255.255`)
- **Klasse C (Heim- & Büronetze):** `192.168.0.0/16` (`192.168.0.0` bis `192.168.255.255`)

Beispiel für Subnetz-Aufteilung (`192.168.1.100/24`):

- **Netzwerk-Teil:** `192.168.1.0` (definiert durch Subnetzmaske `255.255.255.0` bzw. `/24`)
- **Host-Teil:** `.100` (die individuelle Nummer dieses Computers im Netz)
- **Broadcast-Adresse:** `192.168.1.255` (spricht alle Hosts im Subnetz an)

DNS (Domain Name System)

DNS übersetzt menschenlesbare Hostnamen (wie `admindocs.de`) in maschinenlesbare IP-Adressen (`93.184.216.34`).

DNS-Auflösung im Detail

Die DNS-Auflösung unter Linux erfolgt in einer fest definierten Kette:

1. **Lokaler Client & Cache** Anwendung prüft internen Cache und sendet Anfrage an den Resolver
2. **Lokale Namensauflösung** Abgleich mit statischen Host-Einträgen in `/etc/hosts`
3. **Lokaler Stub-Resolver** Anfrage an `systemd-resolved` (127.0.0.53:53) oder Upstream-Nameserver
4. **Rekursiver DNS-Server** Resolver des Providers oder öffentlicher DNS (z. B. 1.1.1.1, 8.8.8.8)
5. **Root- & TLD-Nameserver** Hierarchische Ermittlung des autoritativen Nameservers der Domain
6. **Antwort & Caching** IP-Adresse wird geliefert und für TTL Sekunden lokal gecacht

> ⚠ **Wichtig in modernen Distributionen:** `/etc/resolv.conf` **ist dynamisch!** Auf aktuellen Versionen von Ubuntu (ab 22.04 / 24.04 LTS), Debian und systemd-basierten Systemen darf `/etc/resolv.conf` nicht mehr manuell überschrieben werden. Sie ist standardmäßig ein Symlink auf `/run/systemd/resolve/stub-resolv.conf` und bindet den lokalen Stub-Resolver `127.0.0.53` ein. Manuelle Änderungen gehen bei jedem Netzwerk-Neustart verloren!
>
> - **DNS-Status prüfen:** `resolvectl status`
> - **DNS zur Laufzeit setzen:** `resolvectl dns eth0 1.1.1.1 8.8.8.8`
> - **Persistente Konfiguration:** Über **Netplan** (`/etc/netplan/*.yaml`)

Lokale Hosts in `/etc/hosts` definieren

Für lokale Testserver oder Entwicklungsinstanzen können Hostnamen ohne DNS-Server direkt in `/etc/hosts` abgebildet werden:

```
127.0.0.1       localhost
127.0.1.1       mein-server
192.168.1.10    database.local db1
192.168.1.20    webserver.local web1
```

DNS-Abfragewerkzeuge im Vergleich

- **`dig` (Domain Information Groper):** Das Standard-Werkzeug für professionelle DNS-Analysen. Liefert detaillierte Record-Informationen, TTL-Werte und autoritative Header.
- **`resolvectl query`:** Fragt direkt den systemd-Resolver inklusive DNSSEC-Validierung ab.
- **`nslookup` & `host`:** Schnelle Abfragen für einfache Adress- und Reverse-Lookups.

Quelle: https://admindocs.de/de/linux-administration/linux-administration-netzwerkkonfiguration-und-verwaltung

```
# Detaillierten A-Record einer Domain abfragen
dig admindocs.de

# Kurze IP-Rückgabe ohne Header
dig +short admindocs.de

# Namensauflösung über systemd-resolved testen
resolvectl query admindocs.de
```

DNS-Troubleshooting in der Praxis

```
# 1. Basis-IP-Konnektivität ohne DNS testen
ping -c 3 1.1.1.1

# 2. Namensauflösung testen
ping -c 3 admindocs.de

# 3. DNS-Auflösung mit dig im Detail analysieren
dig admindocs.de +trace
```

Routing verstehen

Das Routing in Linux bestimmt, wie Netzwerkpakete ihren Weg durch das Netzwerk finden:

- **Default Gateway (Standard-Route):** Wenn ein Ziel nicht im lokalen Subnetz liegt, wird das Paket an das Standard-Gateway (den Router) geschickt.
- **Statische Routen:** Manuell konfigurierte Pfade zu spezifischen Netzwerken (z. B. zu VPNs oder internen VLAN-Segmenten).
- **Kernel-Routing-Tabelle:** Der Kernel entscheidet anhand der Ziel-IP und der Subnetzmaske (Prefix-Length), über welches Interface ein Paket den Host verlässt (Longest Prefix Match).

Routing-Tabelle anzeigen und analysieren

```
# Aktive Kernel-Routingtabelle anzeigen (iproute2)
ip route show

# Zielgerichteten Next-Hop für eine bestimmte IP-Adresse ermitteln
ip route get 1.1.1.1
```

Beispielausgabe:

```
default via 192.168.1.1 dev eth0 proto dhcp src 192.168.1.100 metric 100
192.168.1.0/24 dev eth0 proto kernel scope link src 192.168.1.100 metric 100
```

Flüchtige Routen zur Laufzeit manipulieren

```
# Standard-Gateway manuell setzen
sudo ip route add default via 192.168.1.1

# Statische Route für ein internes Subnetz über ein Gateway hinzufügen
sudo ip route add 10.10.0.0/16 via 192.168.1.254 dev eth0

# Statische Route wieder entfernen
sudo ip route del 10.10.0.0/16
```

Netzwerk-Konfiguration

In der Linux-Administration muss klar zwischen **flüchtiger Ad-hoc-Konfiguration** (sofort aktiv, aber nach Reboot verloren) und **persistenter Konfiguration** über Konfigurationsdienste unterschieden werden:

1. Ad-hoc-Konfiguration via `iproute2` (Laufzeit)

Für schnelle Fehlerdiagnosen und temporäre Tests können Adressen direkt über den Kernel zugewiesen werden:

```
# Temporäre IP-Adresse und Subnetzmaske zuweisen
sudo ip addr add 192.168.1.100/24 dev eth0

# Default-Gateway hinzufügen
sudo ip route add default via 192.168.1.1

# Konfiguration überprüfen
ip addr show eth0
ip route show
```

2. Persistente Konfiguration mit Netplan 1.0 (Standard)

Auf modernen Debian- und Ubuntu-Systemen (insbesondere Ubuntu Server ab 22.04 und 24.04/26.04 LTS mit Netplan 1.0) erfolgt die Netzwerkkonfiguration deklarativ in YAML-Dateien unter `/etc/netplan/*.yaml`. Netplan übersetzt diese in Low-Level-Konfigurationen für **`systemd-networkd`** (Server) oder **`NetworkManager`** (Desktop).

Beispiel: Statische IP-Konfiguration (`/etc/netplan/01-netcfg.yaml`):

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: false
      addresses:
        - 192.168.1.100/24
      routes:
        - to: default
          via: 192.168.1.1
      nameservers:
        addresses:
          - 1.1.1.1
          - 8.8.8.8
        search:
          - local.domain
```

 Netplan sicher anwenden: Niemals ohne Rollback! Auf entfernten Servern kann ein YAML-Einrückungsfehler die Netzwerkverbindung kappen. Verwende daher immer:

```
sudo netplan try
```

Netplan wendet die Einstellungen temporär an und erwartet innerhalb von 120 Sekunden eine Bestätigung durch Tastendruck. Bricht die SSH-Verbindung ab, setzt Netplan den vorherigen Zustand automatisch zurück.

```
# Aktuellen Netplan-Status und Schnittstellen prüfen
netplan status

# Konfiguration dauerhaft anwenden
sudo netplan apply
```

Namensauflösung im System (`/etc/nsswitch.conf`)

Die Datei `/etc/nsswitch.conf` steuert die Priorität der Namensauflösungs-Quellen:

```
hosts:          files mdns4_minimal [NOTFOUND=return] resolve [!UNAVAIL=return] dns
```

- `files` : Zuerst werden statische Einträge aus `/etc/hosts` abgefragt
- `resolve` : Nutzt den Cache und DNS-Stack von `systemd-resolved`
- `dns` : Klassischer Fallback auf konfigurierte Nameserver

Netzwerk-Sicherheit

Die Absicherung offener Ports und Verbindungen ist eine fundamentale Pflichtaufgabe. Linux bietet hierfür hochperformante Paketfilter direkt im Kernel.

Firewall-Architektur: nftables vs. iptables

- **`nftables`:** Das moderne, offizielle Paketfilter-Framework des Linux-Kernels (seit Debian 10/11/12 und aktuellem Ubuntu Standard). Es vereint IPv4, IPv6, ARP und Bridge-Filterung in einer einzigen, performanten Syntax.

- **`iptables`:** Das historische Paketfilter-Werkzeug. Auf modernen Systemen ist `iptables` meist nur noch ein Wrapper (`iptables-nft`), der Befehle intern in nftables-Regeln übersetzt.
- **`UFW` (Uncomplicated Firewall):** Eine einsteigerfreundliche CLI-Oberfläche für Debian/Ubuntu, die Regeln im Hintergrund für nftables bzw. iptables generiert.

UFW (Uncomplicated Firewall) in der Praxis

UFW eignet sich ideal für Standard-Server:

```
# 1. Standard-Policies setzen (eingehend alles blockieren, ausgehend erlauben)
sudo ufw default deny incoming
sudo ufw default allow outgoing

# 2. Wichtige Dienste freigeben VOR dem Aktivieren!
sudo ufw allow 22/tcp comment "SSH Zugang"
sudo ufw allow 80/tcp comment "HTTP Webserver"
sudo ufw allow 443/tcp comment "HTTPS Webserver"

# 3. UFW aktivieren
sudo ufw enable

# 4. Status und aktive Regeln prüfen
sudo ufw status verbose
```

Erweiterte UFW-Regeln

```
# Port-Bereiche für Anwendungen freigeben
sudo ufw allow 3000:3050/tcp

# Zugriff auf SSH nur aus dem internen Admin-Subnetz erlauben
```

```
sudo ufw allow from 192.168.1.0/24 to any port 22 proto tcp

# Rate-Limiting gegen SSH-Brute-Force (blockt IPs nach 6 Fehlversuchen in 30s)
sudo ufw limit 22/tcp
```

nftables (Nativer Kernel-Paketfilter)

`nftables` arbeitet mit hierarchischen Tabellen, Ketten (*Chains*) und Regeln:

```
# Neue Filter-Tabelle für IPv4 und IPv6 (inet) erstellen
sudo nft add table inet firewall

# Input- und Output-Chains mit Standard-Hooks anlegen
sudo nft add chain inet firewall input { type filter hook input priority 0 \; policy drop
\; }
sudo nft add chain inet firewall output { type filter hook output priority 0 \; policy
accept \; }

# Loopback-Traffic und bestehende Verbindungen erlauben
sudo nft add rule inet firewall input iif "lo" accept
sudo nft add rule inet firewall input ct state established,related accept
sudo nft add rule inet firewall input ct state invalid drop

# SSH und Webserver für neue Verbindungen freigeben
sudo nft add rule inet firewall input tcp dport 22 accept
sudo nft add rule inet firewall input tcp dport { 80, 443 } accept

# Aktives nftables-Regelwerk ausgeben
sudo nft list ruleset
```

IPtables (Historischer Kompatibilitätsmodus)

Auf älteren Systemen oder wenn Legacy-Skripte `iptables` erzwingen:

```
# Basis-Filterung: Eingehend verwerfen, ausgehend erlauben
sudo iptables -P INPUT DROP
sudo iptables -P FORWARD DROP
sudo iptables -P OUTPUT ACCEPT

# Loopback und bestehende Verbindungen erlauben
sudo iptables -A INPUT -i lo -j ACCEPT
sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

# Ports freigeben
sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
sudo iptables -A INPUT -p tcp -m multiport --dports 80,443 -j ACCEPT
```

SSH-Sicherheit und Härtung

SSH (Secure Shell) ist der zentrale Administrationszugang zu Linux-Servern und muss nach aktuellen Standards abgesichert werden.

> □□ **Ubuntu 24.04 / 26.04 LTS: SSH Socket Activation (`ssh.socket`)** In aktuellen Ubuntu-Releases läuft OpenSSH standardmäßig nicht mehr als permanent aktiver Daemon (`ssh.service`), sondern wird über **`ssh.socket`** bei eingehenden Verbindungen bedarfsgerecht gestartet (*Socket-based activation*).
>
> - **Socket-Status prüfen:** `sudo systemctl status ssh.socket`
> - **Neustart nach Portänderungen:** `sudo systemctl daemon-reload && sudo systemctl restart ssh.socket`
> - **Konfigurationssyntax testen:** `sudo sshd -t`

Schlüssel-basierte Authentifizierung (Ed25519)

Klassische Passwörter sind anfällig für Brute-Force-Angriffe. Verwende moderne Ed25519-Schlüsselpaare:

```
# Modernes Ed25519-Schlüsselpaar auf dem Client generieren
ssh-keygen -t ed25519 -C "admin@workstation"

# Öffentlichen Schlüssel auf den Zielserver übertragen
ssh-copy-id -i ~/.ssh/id_ed25519.pub benutzer@server-ip

# Berechtigungen auf dem Server verifizieren
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys
```

SSH-Server härten (`/etc/ssh/sshd_config.d/security.conf`)

Anstatt die Hauptkonfiguration `/etc/ssh/sshd_config` direkt zu bearbeiten, legst du eine modulare Konfigurationsdatei an:

```
# Root-Login über Passwort verbieten
PermitRootLogin prohibit-password

# Passwort-Authentifizierung komplett abschalten (nur SSH-Keys erlauben)
PasswordAuthentication no

# Leere Passwörter verbieten
PermitEmptyPasswords no

# Maximale Authentifizierungsversuche pro Verbindung
MaxAuthTries 3

# Timeout für den Anmeldevorgang in Sekunden
LoginGraceTime 30
```

Netzwerk-Monitoring

Das Monitoring des Netzwerkverkehrs und der aktiven Sockets ist essenziell für die Erkennung von Lastspitzen, Fehlkonfigurationen und unerwünschten Verbindungen.

Wichtige Monitoring-Werkzeuge

Moderne Linux-Systeme nutzen den Kernel-Netlink-Mechanismus anstelle veralteter Textdateien:

- **`ss` (Socket Statistics):** Der moderne Ersatz für `netstat`. Fragt Sockets direkt aus dem Kernel ab – extrem schnell auch bei tausenden parallelen Verbindungen.
- **`iftop`:** Bandbreitennutzung in Echtzeit nach Quell- und Ziel-IP aufgeschlüsselt.
- **`nethogs`:** Ordnet Bandbreite direkt den verursachenden Prozessen und PIDs zu.
- **`vnstat`:** Leichtgewichtiger Hintergrund-Daemon zur kontinuierlichen Erfassung des monatlichen und täglichen Datenvolumens.

```
# Nützliche Diagnose-Tools installieren
sudo apt update && sudo apt install -y iftop nethogs vnstat
```

Verbindungs- und Port-Monitoring mit `ss`

```
# Alle lauschenden TCP- und UDP-Sockets mit Prozess-Informationen auflisten
sudo ss -tulpn

# Nur aktive (hergestellte) TCP-Verbindungen anzeigen
ss -t state established
```

```
# Verbindung zu einem bestimmten Zielport (z. B. 443) filtern
ss -tan '( dport = :443 )'
```

Bandbreite in Echtzeit beobachten

Prozess-basiertes Bandbreiten-Monitoring (`nethogs`):

```
# Zeigt Echtzeit-Übertragungsraten pro Prozess-ID (PID)
sudo nethogs eth0
```

Schnittstellen-Traffic nach Gegenstellen (`iftop`):

```
# Zeigt Host-zu-Host Übertragungsraten auf dem Interface
sudo iftop -i eth0
```

Langzeit-Volumenerfassung mit `vnstat` :

```
# Tägliche und monatliche Traffic-Statistiken abrufen
vnstat -i eth0
```

Troubleshooting

Wenn Netzwerkprobleme auftreten, empfiehlt sich eine strukturierte Fehlersuche von der physischen Schicht bis zur Anwendungsebene:

1. **Link- & Adress-Status prüfen:** Ist das Interface `UP` und hat es eine gültige IP? (`ip addr show`)
2. **Gateway-Erreichbarkeit:** Antwortet der lokale Router? (`ping -c 3 <gateway-ip>`)
3. **Internet-Konnektivität (ohne DNS):** Ist eine öffentliche IP erreichbar? (`ping -c 3 1.1.1.1`)
4. **DNS-Auflösung:** Funktioniert die Namensauflösung? (`resolvectl query admindocs.de` oder `dig admindocs.de`)
5. **Dienst- & Firewall-Status:** Lauscht der Ziel-Port und lässt die Firewall Pakete durch? (`sudo ss -tulpn` , `sudo ufw status`)

Praktische Diagnose-Befehle

1. Link-Status und MTU analysieren:

```
# Interface-Status und Fehlerzähler (Drop/Errors) prüfen
ip -s link show eth0
```

2. Routing-Weg mit MTR oder Traceroute verfolgen:

```
# Routenverlauf und Paketverlust an jedem Hop testen
traceroute -n 1.1.1.1
```

3. Port-Erreichbarkeit mit Netcat prüfen (ohne Portscan):

```
# Prüft in 2 Sekunden, ob Port 443 erreichbar ist
nc -zv -w 2 admindocs.de 443
```

Übung

Wende die gelernten Konzepte in einer praxisnahen Konfiguration an.

Aufgabe 1: Netzwerk-Status und Routing analysieren

```
# 1. Alle Interfaces auflisten
ip link show

# 2. Aktive IP-Adressen anzeigen
ip addr show

# 3. Default-Gateway und Routing-Tabelle prüfen
ip route show
```

Aufgabe 2: DNS-Status prüfen und testen

```
# 1. Aktiven systemd-resolved DNS-Status abfragen
resolvectl status

# 2. Namensauflösung mit dig im Kurzformat testen
dig +short admindocs.de
```

Aufgabe 3: Firewall mit UFW absichern

```
# 1. Standard-Policy setzen (Alles Eingehende blockieren)
sudo ufw default deny incoming
sudo ufw default allow outgoing

# 2. SSH und Webserver erlauben
sudo ufw allow 22/tcp
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp

# 3. UFW aktivieren und Status verifizieren
sudo ufw enable
sudo ufw status verbose
```

Befehlsreferenz (Cheatsheet)

Für den schnellen Zugriff bei der Netzwerkadministration, Fehlerdiagnose und Firewall-Konfiguration fasst die folgende Referenz die wichtigsten Linux-Kommandos zusammen:

Netzwerkadapter und Link-Status anzeigen Zeigt alle Interfaces und physischen Verbindungsstatus (UP/DOWN)

$ `ip link show`

IP-Adressen aller Schnittstellen auflisten Listet alle konfigurierten IPv4- und IPv6-Adressen auf

$ `ip addr show`

Kernel-Routingtabelle anzeigen Zeigt Routen, Subnetze und das aktive Standard-Gateway

$ `ip route show`

Next-Hop für Ziel-IP ermitteln Ermittelt ausgehendes Interface und Gateway für eine Zieladresse

$ `ip route get <IP>`

Netplan-Konfiguration mit Auto-Rollback testen Testet deklarative Netzwerkkonfiguration mit 120s Rollback-Schutz

$ `sudo netplan try`

Netplan-Konfiguration dauerhaft anwenden Aktiviert die YAML-Netzwerkkonfiguration für den System-Renderer

$ `sudo netplan apply`

Netplan-Status und Interfaces prüfen Zeigt den aktuellen Laufzeitstatus aller Netplan-Schnittstellen

$ `netplan status`

NetworkManager-Gerätestatus anzeigen Listet Status und Verbindungsprofile aller NetworkManager-Geräte

$ `nmcli device status`

DNS-Resolver-Status abfragen Zeigt aktive Upstream-DNS-Server und Domains von systemd-resolved

$ `resolvectl status`

DNS-Eintrag einer Domain abfragen Liefert A-Record und IP-Adressen via DNS

$ `dig +short <domain>`

Lauschende Ports und Sockets anzeigen Listet alle lauschenden TCP/UDP-Sockets mit zugehöriger PID via Netlink

$ `sudo ss -tulpn`

Eingehenden Datenverkehr standardmäßig blockieren Setzt restriktive Standard-Policy für alle eingehenden Pakete

```
$ sudo ufw default deny incoming
```

SSH-Port in Firewall freigeben Öffnet TCP-Port 22 in der Uncomplicated Firewall

```
$ sudo ufw allow 22/tcp
```

Rate-Limiting gegen Brute-Force aktivieren Blockt IPs bei mehr als 6 Fehlversuchen innerhalb von 30 Sekunden

```
$ sudo ufw limit 22/tcp
```

Detaillierten Firewall-Status anzeigen Listet alle aktiven UFW-Regeln, Profile und Policies auf

```
$ sudo ufw status verbose
```

TCP-Port-Erreichbarkeit prüfen Testet schnelle Socket-Verbindung zu einem entfernten Port (Netcat)

```
$ nc -zv <host> <port>
```

Weiterführende Ressourcen

Die folgenden Leitfäden, Spezifikationen und internen Kursmodule vertiefen die Linux-Netzwerkkonfiguration und Fehlersuche:

Netplan Documentation Offizielle Dokumentation zur deklarativen YAML-Netzwerkkonfiguration

iproute2 Documentation Offizieller Leitfaden zu ip, ss, tc und Netzwerk-Namespaces

systemd-resolved Manual Offizielle Dokumentation zum systemd Name Resolution Service

UFW Community Documentation Ausführlicher Leitfaden zur Uncomplicated Firewall auf Ubuntu/Debian

Linux Administration #3: Prozesse Das vorherige Modul: Prozesse, Signale, Nice & cgroups

Linux Administration #5: Shell-Scripting Das nachfolgende Modul: Automatisierung, Cron, Traps & Bash

Befehlszeilenprozessor in Linux Fundamentales Wissen zu Shells, I/O-Streams und Pipes

Fazit

Die Netzwerkverwaltung ist das fundamentale Bindeglied, das eigenständige Linux-Server in ausfallsichere Cloud- und Rechenzentrumsinfrastrukturen integriert. Durch die Beherrschung moderner `iproute2`-Befehle, deklarativer Netzwerkkonfiguration über Netplan und NetworkManager, transparenter DNS-Auflösung via `systemd-resolved` und solider Firewall-Regeln mit UFW und nftables bist du in der Lage, Servernetzwerke hochverfügbar aufzusetzen und Störungen im Betrieb systematisch zu lokalisieren.

> **Praxis-Tipp:** Teste neue Netzwerkkonfigurationen über Netplan immer mit `sudo netplan try`. Dieser Befehl wendet die Konfiguration an und setzt sie nach 120 Sekunden automatisch zurück, falls du dich durch einen Tippfehler selbst vom SSH-Zugang aussperrst und den Timeout nicht rechtzeitig mit Enter bestätigen kannst.

Im nächsten Modul unseres Administrationskurses widmen wir uns der Automatisierung wiederkehrender Aufgaben: **Weiter geht es mit:** Linux Administration #5: Shell-Scripting und Automatisierung

Zur Kursübersicht: Alle Linux-Administrations-Artikel & Module