5.1 Prozesse, Dienste und systemd

Linux führt Programme als Prozesse aus.

Ein Prozess ist ein laufendes Programm. Ein Dienst ist ein Programm, das meist im Hintergrund läuft und eine bestimmte Aufgabe für das System oder Netzwerk bereitstellt.

Für Fachinformatiker Systemintegration ist dieses Thema wichtig, weil Serverdienste wie SSH, Webserver, Datenbanken, DNS, DHCP, Backupdienste oder Monitoring-Agenten auf Linux-Systemen als Prozesse und Dienste laufen.

Merksatz:

```
Prozess = laufendes Programm.
Dienst = Hintergrundprogramm mit Aufgabe.
```

Lernziele

Nach dieser Seite solltest du erklären können:

```
- was ein Prozess ist
- was eine PID ist
- was ein Dienst ist
- was ein Daemon ist
- wie man Prozesse anzeigt
- wie man Prozesse beendet
- wie man Systemressourcen prüft
- wie Dienste mit systemd verwaltet werden
- was systemctl macht
- was enable und disable bedeuten
- wie man typische Dienstfehler eingrenzt
```

Grundidee

Wenn unter Linux ein Programm gestartet wird, entsteht daraus ein Prozess.

Beispiele:

- Shell
- Texteditor
- Webserver
- Datenbankserver
- Backupskript
- SSH-Server
- Cronjob
- Monitoring-Agent

Jeder Prozess verbraucht Ressourcen, zum Beispiel:

- CPU
- Arbeitsspeicher
- Dateizugriffe
- Netzwerkverbindungen
- Prozess-IDs
- offene Dateien

Merksatz:

Alles,
was läuft,
ist als Prozess im System sichtbar.

Prozess

Ein Prozess ist eine laufende Instanz eines Programms.

Beispiel:

Das Programm nginx ist installiert.

Wenn nginx gestartet wird,
laufen ein oder mehrere nginx-Prozesse.

Ein Programm kann mehrfach als Prozess laufen.

Beispiele:

```
mehrere Shells

mehrere Webserver-Worker

mehrere Datenbankprozesse

mehrere SSH-Sitzungen
```

Merksatz:

```
Programm = Datei auf dem System.
Prozess = laufendes Programm.
```

PID

PID steht für:

```
Process ID
```

Die PID ist eine eindeutige Nummer eines laufenden Prozesses.

Beispiel:

```
PID 1234
```

Mit der PID kann ein Prozess gezielt angesprochen werden, zum Beispiel zum Beenden oder Prüfen.

Merksatz:

```
PID = eindeutige Nummer eines Prozesses.
```

PPID

PPID steht für:

```
Parent Process ID
```

Das ist die Prozess-ID des Elternprozesses.

Viele Prozesse werden von anderen Prozessen gestartet.

Beispiel:

```
Eine Shell startet ein Skript.

Die Shell ist der Elternprozess.

Das Skript ist der Kindprozess.
```

Merksatz:

```
Prozesse können Eltern-
und Kindprozesse haben.
```

Prozesse anzeigen mit ps

Der Befehl ps zeigt Prozesse an.

Beispiele:

```
ps

ps aux

ps -ef
```

Bedeutung:

Befehl	Bedeutung
ps	Prozesse der aktuellen Shell anzeigen
ps aux	viele Prozesse systemweit anzeigen
ps -ef	Prozesse in anderem verbreiteten Format anzeigen

Typische Spalten:

Spalte	Bedeutung

USER	Benutzer, unter dem der Prozess läuft
PID	Prozess-ID
%CPU	CPU-Nutzung
%MEM	Speicheranteil
COMMAND	gestarteter Befehl
PPID	Elternprozess-ID

Merksatz:

```
ps zeigt laufende Prozesse.
```

Prozesse suchen

Prozesse kann man mit ps und grep suchen.

Beispiel:

```
ps aux | grep nginx
```

Bedeutung:

```
ps aux zeigt Prozesse.

grep nginx filtert nach nginx.
```

Alternative Befehle:

```
pgrep nginx

pidof nginx
```

Bedeutung:

Befehl	Bedeutung
pgrep nginx	sucht Prozess-IDs nach Namen
pidof nginx	zeigt PID eines laufenden Programms

Merksatz:

```
ps zeigt,
grep filtert.
```

Prozesse live überwachen mit top

top zeigt laufende Prozesse und Systemauslastung live an.

Starten:

```
top
```

Typische Informationen:

```
CPU-Auslastung

RAM-Nutzung

laufende Prozesse

Load Average

Prozessliste

Benutzer

Prozess-IDs
```

Beenden:

```
q
```

Merksatz:

```
top zeigt Systemlast live.
```

---

htop

htop ist eine komfortablere Alternative zu top.

Starten:

```
htop
```

Vorteile:

```
übersichtlicher

farbliche Anzeige

einfachere Prozessauswahl

leichteres Beenden von Prozessen
```

Wichtig:

```
htop ist nicht immer standardmäßig installiert.
```

Merksatz:

```
htop ist ein komfortabler Prozessmonitor,
falls installiert.
```

CPU, RAM und Load Average

Bei Prozessproblemen sind drei Werte besonders wichtig:

Wert	Bedeutung
CPU	Rechenleistung
RAM	Arbeitsspeicher
Load Average	durchschnittliche Systemlast

Hohe CPU-Auslastung kann bedeuten:

```
Prozess rechnet stark

Endlosschleife
```

hohe Nutzerlast

Angriff oder Fehlkonfiguration

Hohe RAM-Nutzung kann bedeuten:

Dienst benötigt viel Speicher

Speicherleck

zu viele Prozesse

falsche Konfiguration

Merksatz:

Langsame Systeme immer mit Messwerten prüfen.

Load Average einfach erklärt

Load Average zeigt, wie viele Prozesse im Durchschnitt auf CPU oder I/O warten.

Typische Anzeige:

load average: 0.50, 1.20, 2.10

Die drei Werte stehen für:

letzte 1 Minute

letzte 5 Minuten

letzte 15 Minuten

Ein hoher Wert kann auf Überlastung hindeuten.

Wichtig:

```
Bewertung hängt von der Anzahl der CPU-Kerne ab.
```

Merksatz:

```
Load Average zeigt durchschnittliche Systemlast.
```

Prozesse beenden mit kill

Mit kill sendet man ein Signal an einen Prozess.

Beispiel:

```
kill 1234
```

Das sendet standardmäßig ein TERM-Signal.

TERM bedeutet:

```
Prozess soll sauber beendet werden.
```

Wenn das nicht funktioniert:

```
kill -9 1234
```

Das sendet ein KILL-Signal.

Wichtig:

```
kill -9 beendet hart
und sollte nur verwendet werden,
wenn normales Beenden nicht funktioniert.
```

Merksatz:

```
Erst normal beenden,
dann nur notfalls kill -9.
```

Wichtige Signale

Signal	Nummer	Bedeutung
TERM	15	sauber beenden
KILL	9	sofort hart beenden
HUP	1	häufig Konfiguration neu laden
INT	2	Unterbrechung, ähnlich Strg+C

Beispiele:

```
kill -15 1234

kill -9 1234

kill -HUP 1234
```

Merksatz:

```
TERM bittet um Beenden.
KILL erzwingt Beenden.
```

Prozess mit Namen beenden

Mit pkill kann man Prozesse anhand des Namens beenden.

Beispiel:

```
pkill nginx
```

Wichtig:

```
Vorsichtig verwenden,
weil mehrere Prozesse betroffen sein können.
```

Besser vorher prüfen:

```
pgrep -a nginx
```

Merksatz:

```
pkill beendet nach Namen,
deshalb vorher genau prüfen.
```

Prozesse im Vordergrund und Hintergrund

Ein Prozess kann im Vordergrund oder Hintergrund laufen.

Vordergrund:

```
blockiert die aktuelle Shell,
bis er beendet ist.
```

Hintergrund:

```
läuft weiter,
während man die Shell weiter nutzen kann.
```

Befehl im Hintergrund starten:

```
befehl &
```

Beispiel:

```
sleep 60 &
```

Jobs anzeigen:

```
jobs
```

Merksatz:

```
& startet einen Befehl im Hintergrund.
```

Strg+C und Strg+Z

Wichtige Tastenkombinationen:

Tastenkombination	Bedeutung
Strg+C	laufenden Vordergrundprozess abbrechen
Strg+Z	Prozess anhalten
fg	angehaltenen Job in Vordergrund holen
bg	angehaltenen Job im Hintergrund fortsetzen

Merksatz:

```
Strg+C beendet meist.
Strg+Z hält an.
```

Daemon

Ein Daemon ist ein Hintergrundprozess, der dauerhaft oder bei Bedarf eine Systemaufgabe erfüllt.

Typische Daemons:

```
sshd

cron

nginx

mysqld

systemd

rsyslogd
```

Viele Daemon-Namen enden auf:

```
d
```

Beispiele:

```
sshd = SSH-Daemon

cron oder crond = Cron-Daemon
```

Merksatz:

```
Daemon = Hintergrunddienst.
```

Dienst

Ein Dienst ist eine Systemfunktion, die meist im Hintergrund bereitgestellt wird.

Beispiele:

Dienst	Zweck
ssh	Fernadministration
nginx	Webserver
apache2	Webserver
mariadb	Datenbank
postgresql	Datenbank
cron	zeitgesteuerte Aufgaben
systemd-journald	Logging
docker	Containerdienst

Merksatz:

```
Dienste stellen Funktionen bereit.
```

---

systemd

systemd ist auf vielen modernen Linux-Distributionen das zentrale Init- und Dienstverwaltungssystem.

systemd ist zuständig für:

```
- Systemstart
- Dienste
```

- Abhängigkeiten
- Units
- Logging-Anbindung
- Timer
- Mounts
- Targets
- Systemzustände

Wichtig:

Nicht jede Linux-Distribution nutzt systemd.
Viele Serverdistributionen tun es aber.

Merksatz:

systemd verwaltet auf vielen Linux-Systemen Dienste und Systemstart.

Unit

Eine Unit ist eine Verwaltungseinheit von systemd.

Typische Unit-Arten:

Unit-Art	Bedeutung
.service	Dienst
.socket	Socket-Aktivierung
.timer	zeitgesteuerte Aufgabe
.mount	Mountpoint
.target	Zielzustand des Systems

Beispiel:

```
ssh.service

nginx.service

docker.service
```

Merksatz:

```
.service steht für einen Dienst.
```

---

systemctl

systemctl ist das wichtigste Werkzeug zur Verwaltung von systemd.

Typische Befehle:

```
systemctl status ssh

systemctl start ssh

systemctl stop ssh

systemctl restart ssh

systemctl reload ssh

systemctl enable ssh

systemctl disable ssh
```

Merksatz:

```
systemctl verwaltet systemd-Units.
```

Dienststatus prüfen

Status eines Dienstes prüfen:

```
systemctl status ssh
```

Typische Informationen:

```
Dienstname
```

aktiv oder inaktiv

Fehlerstatus

PID

Startzeit

letzte Logzeilen

Pfad zur Unit-Datei

Wichtige Zustände:

Zustand	Bedeutung
active	Dienst läuft
inactive	Dienst läuft nicht
failed	Dienst ist fehlgeschlagen
enabled	startet automatisch beim Boot
disabled	startet nicht automatisch beim Boot

Merksatz:

systemctl status ist die erste Prüfung bei Dienstproblemen.

Dienst starten

Dienst starten:

```
systemctl start ssh
```

Bedeutung:

Der Dienst wird jetzt gestartet.

Wichtig:

```
start bedeutet nicht automatisch,
dass der Dienst beim nächsten Neustart wieder startet.
```

Merksatz:

```
start startet jetzt.
```

Dienst stoppen

Dienst stoppen:

```
systemctl stop ssh
```

Bedeutung:

```
Der Dienst wird jetzt gestoppt.
```

Wichtig:

```
stop deaktiviert nicht automatisch den Autostart.
```

Merksatz:

```
stop stoppt jetzt.
```

Dienst neu starten

Dienst neu starten:

```
systemctl restart ssh
```

Bedeutung:

```
Der Dienst wird gestoppt und neu gestartet.
```

Typische Nutzung:

nach Konfigurationsänderungen

nach Fehlern

nach Updates

Wichtig:

Bei SSH vorsichtig sein,
besonders wenn man gerade per SSH verbunden ist.

Merksatz:

restart startet Dienst komplett neu.

Dienst neu laden

Dienst neu laden:

```
systemctl reload nginx
```

Bedeutung:

Der Dienst lädt seine Konfiguration neu,
ohne vollständig beendet zu werden.

Nicht jeder Dienst unterstützt reload.

Unterschied:

Befehl	Bedeutung
restart	Dienst komplett neu starten
reload	Konfiguration neu laden, falls unterstützt

Merksatz:

```
reload ist oft schonender als restart,
wenn der Dienst es unterstützt.
```

Autostart aktivieren

Dienst beim Systemstart automatisch starten:

```
systemctl enable ssh
```

Wichtig:

```
enable startet den Dienst nicht unbedingt sofort.
```

Dafür braucht man:

```
systemctl start ssh
```

Oder kombiniert:

```
systemctl enable --now ssh
```

Merksatz:

```
enable aktiviert Autostart.
```

Autostart deaktivieren

Autostart deaktivieren:

```
systemctl disable ssh
```

Wichtig:

```
disable stoppt den aktuell laufenden Dienst nicht unbedingt sofort.
```

Dafür braucht man:

```
systemctl stop ssh
```

Merksatz:

```
disable entfernt Autostart.
```

---

enable, disable, start und stop unterscheiden

Befehl	Wirkung
start	startet Dienst jetzt
stop	stoppt Dienst jetzt
enable	startet Dienst künftig automatisch beim Boot
disable	startet Dienst künftig nicht automatisch beim Boot
restart	stoppt und startet Dienst jetzt neu
reload	lädt Konfiguration neu, falls unterstützt

Merksatz:

```
start und stop gelten jetzt.
enable und disable gelten für den Systemstart.
```

Dienst prüfen nach Änderung

Nach Änderung an einem Dienst:

```
systemctl restart dienstname

systemctl status dienstname

journalctl -u dienstname
```

Typisches Vorgehen:

```
1. Konfiguration ändern

2. Syntax prüfen,
```

```
   falls möglich

3. Dienst neu laden oder neu starten

4. Status prüfen

5. Logs prüfen
```

Merksatz:

```
Nach Dienständerung immer Status und Logs prüfen.
```

---

journalctl für Dienste

Logs eines Dienstes anzeigen:

```
journalctl -u ssh
```

Live mitlesen:

```
journalctl -u ssh -f
```

Seit letztem Boot:

```
journalctl -u ssh -b
```

Nur letzte Einträge:

```
journalctl -u ssh -n 50
```

Merksatz:

```
journalctl -u zeigt Logs einer Unit.
```

Dienst läuft, aber Port ist nicht erreichbar

Mögliche Ursachen:

```
Dienst lauscht nicht auf erwarteter Adresse

Dienst lauscht auf anderem Port

lokale Firewall blockiert

Netzwerkfirewall blockiert

Dienst ist nur lokal gebunden

falsche Konfiguration

IPv4/IPv6-Verwechslung
```

Prüfen:

```
systemctl status dienst

ss -tulpen

journalctl -u dienst

Firewall-Regeln
```

Merksatz:

```
Dienst läuft heißt nicht automatisch:
Port ist erreichbar.
```

Offene Ports anzeigen

Mit ss kann man offene Ports und Verbindungen anzeigen.

Beispiele:

```
ss -tulpen

ss -tulpn
```

Bedeutung der Optionen:

Option	Bedeutung
-t	TCP
-u	UDP
-l	listening
-p	Prozess anzeigen
-e	erweiterte Informationen
-n	numerische Anzeige

Merksatz:

```
ss zeigt,
welche Dienste auf welchen Ports lauschen.
```

Dienst lauscht nur auf localhost

Wenn ein Dienst nur auf 127.0.0.1 lauscht, ist er nur lokal erreichbar.

Beispiel:

```
127.0.0.1:8080
```

Bedeutung:

```
Nur Programme auf demselben System können direkt darauf zugreifen.
```

Wenn ein Dienst auf allen IPv4-Schnittstellen lauscht:

```
0.0.0.0:8080
```

Bedeutung:

```
Dienst lauscht auf allen IPv4-Interfaces.
```

Merksatz:

127.0.0.1 = nur lokal.

0.0.0.0 = alle IPv4-Interfaces.

Dienst startet nicht

Wenn ein Dienst nicht startet, sollte man prüfen:

systemctl status dienst

journalctl -u dienst

Konfigurationsdatei

Syntaxprüfung

Ports bereits belegt?

Rechte auf Dateien

Abhängigkeiten

Speicherplatz

Benutzer des Dienstes

Pfade

Zertifikate

Merksatz:

Dienstfehler immer mit Status und Logs eingrenzen.

Port bereits belegt

Ein Dienst kann nicht starten, wenn ein anderer Prozess denselben Port bereits nutzt.

Beispiel:

```
nginx soll TCP 80 nutzen,
aber Apache läuft bereits auf TCP 80.
```

Prüfen:

```
ss -tulpen
```

Lösung:

```
einen Dienst stoppen

Port ändern

Konfiguration anpassen
```

Merksatz:

```
Ein Port kann pro IP und Protokoll nur passend gebunden werden.
```

Dienst hat Rechteproblem

Ein Dienst läuft oft unter einem eigenen Benutzer.

Beispiel:

```
www-data

nginx

postgres
```

Wenn dieser Benutzer keine Rechte auf Dateien oder Verzeichnisse hat, kann der Dienst fehlschlagen.

Prüfen:

```
systemctl status dienst
```

```
journalctl -u dienst

ls -l datei

ls -ld verzeichnis
```

Merksatz:

```
Dienstprobleme können Rechteprobleme sein.
```

Dienst hat Konfigurationsfehler

Viele Dienste starten nicht, wenn die Konfiguration fehlerhaft ist.

Beispiele:

```
falsche Syntax

falscher Pfad

falscher Port

Zertifikat fehlt

Benutzer existiert nicht

ungültige Option
```

Viele Dienste bieten Syntaxprüfungen.

Beispiele:

```
nginx -t

apachectl configtest
```

Wichtig:

```
Vor restart möglichst Syntax prüfen,
wenn der Dienst das unterstützt.
```

Merksatz:

```
Konfiguration prüfen,
bevor man produktive Dienste neu startet.
```

Abhängigkeiten

Dienste können voneinander abhängig sein.

Beispiele:

```
Webanwendung braucht Datenbank.

Backupdienst braucht Netzwerkshare.

Monitoring braucht Netzwerk.

Containerdienst braucht Storage.
```

Wenn eine Abhängigkeit fehlt, startet der Dienst eventuell nicht korrekt oder funktioniert nur teilweise.

Merksatz:

```
Dienste funktionieren oft nur mit ihren Abhängigkeiten.
```

Dienste beim Boot

Beim Systemstart werden Dienste automatisch gestartet, wenn sie aktiviert sind.

Prüfen, ob ein Dienst aktiviert ist:

```
systemctl is-enabled ssh
```

Prüfen, ob ein Dienst läuft:

```
systemctl is-active ssh
```

Merksatz:

```
enabled heißt Autostart.
active heißt läuft gerade.
```

Targets

Targets sind systemd-Zielzustände.

Beispiele:

Target	Bedeutung
multi-user.target	Mehrbenutzerbetrieb ohne grafische Oberfläche
graphical.target	Mehrbenutzerbetrieb mit grafischer Oberfläche
rescue.target	Rettungsmodus
emergency.target	Minimaler Notfallmodus

Merksatz:

```
Targets beschreiben Systemzustände.
```

Cron und systemd Timer

Zeitgesteuerte Aufgaben können unter Linux über cron oder systemd Timer laufen.

cron:

```
klassischer Zeitplaner
```

systemd Timer:

```
systemd-eigene zeitgesteuerte Units
```

Beispiele für Aufgaben:

Backup starten

Logs aufräumen

Updates prüfen

Skript regelmäßig ausführen

Merksatz:

Wiederkehrende Aufgaben laufen oft über cron oder systemd Timer.

Typische Prozessbefehle

Befehl	Zweck
ps	Prozesse anzeigen
ps aux	viele Prozesse anzeigen
pgrep	Prozesse nach Namen suchen
pidof	PID eines Programms anzeigen
top	Prozesse live überwachen
htop	komfortabler Prozessmonitor
kill	Signal an Prozess senden
pkill	Prozess nach Namen beenden
jobs	Hintergrundjobs anzeigen
fg	Job in Vordergrund holen
bg	Job im Hintergrund fortsetzen

Typische Dienstbefehle

Befehl	Zweck
systemctl status dienst	Dienststatus prüfen
systemctl start dienst	Dienst jetzt starten
systemctl stop dienst	Dienst jetzt stoppen
systemctl restart dienst	Dienst neu starten

Befehl	Zweck
systemctl reload dienst	Konfiguration neu laden
systemctl enable dienst	Autostart aktivieren
systemctl disable dienst	Autostart deaktivieren
systemctl enable --now dienst	Autostart aktivieren und sofort starten
systemctl is-active dienst	prüfen, ob Dienst läuft
systemctl is-enabled dienst	prüfen, ob Autostart aktiv ist
journalctl -u dienst	Dienstlogs anzeigen

Typische Fehlerbilder

Fehlerbild	Wahrscheinliche Ursache
Dienst ist inactive	Dienst läuft nicht
Dienst ist failed	Startfehler oder Absturz
Dienst startet nicht	Konfiguration, Rechte, Port, Abhängigkeit
Dienst läuft, aber nicht erreichbar	Firewall, Port, Bind-Adresse, Netzwerk
Port bereits belegt	anderer Prozess nutzt denselben Port
Dienst nach Boot nicht gestartet	Autostart nicht enabled
Dienst kann Datei nicht lesen	Rechteproblem
Dienst findet Datei nicht	falscher Pfad
Dienst bricht nach Änderung ab	Konfigurationsfehler
System langsam	CPU, RAM, I/O oder Prozesslast prüfen

Sichere Arbeitsweise bei Dienständerungen

Vor Änderung:

```
aktuelle Konfiguration sichern

Dokumentation prüfen

Wartungsfenster beachten

aktive Verbindung berücksichtigen
```

Während Änderung:

Syntax prüfen,
falls möglich

Dienst reload statt restart nutzen,
wenn ausreichend

Nach Änderung:

systemctl status prüfen

journalctl -u prüfen

Funktion testen

Rollback bereithalten

Merksatz:

Dienständerungen immer prüfbar und rückgängig planbar machen.

Typische Prüfungsfragen

Frage	Kurzantwort
Was ist ein Prozess?	laufendes Programm
Was ist eine PID?	eindeutige Prozess-ID
Was ist ein Dienst?	Hintergrundprogramm mit Aufgabe
Was ist ein Daemon?	Hintergrundprozess
Was macht ps?	Prozesse anzeigen
Was macht top?	Prozesse live überwachen
Was macht kill?	Signal an Prozess senden
Was macht systemctl?	systemd-Units verwalten
Was bedeutet active?	Dienst läuft
Was bedeutet inactive?	Dienst läuft nicht
Was bedeutet failed?	Dienst ist fehlgeschlagen

Frage	Kurzantwort
Was bedeutet enabled?	Autostart aktiv
Was bedeutet disabled?	Autostart nicht aktiv
Was macht journalctl -u?	Logs einer Unit anzeigen

Typische Prüfungsfallen

Falle	Richtig denken
Programm und Prozess gleichsetzen	Programm ist Datei, Prozess läuft
kill immer als hartes Beenden verstehen	kill sendet Signale, Standard ist TERM
kill -9 sofort verwenden	erst sauber beenden
Dienst läuft = erreichbar	Firewall, Port und Bind-Adresse prüfen
enable mit start verwechseln	enable ist Autostart, start ist jetzt
disable mit stop verwechseln	disable entfernt Autostart, stop stoppt jetzt
Logs nicht prüfen	journalctl -u ist zentral
reload und restart gleichsetzen	reload lädt Konfiguration, restart startet neu
SSH unvorsichtig neu starten	Remote-Zugriff kann abbrechen
Portproblem nur am Dienst suchen	auch Firewall und Bind-Adresse prüfen

IHK-sichere Kurzformulierung

Ein Prozess ist eine laufende Instanz eines Programms und besitzt eine eindeutige Prozess-ID, die PID. Dienste sind Hintergrundprogramme, die System- oder Netzwerkfunktionen bereitstellen, zum Beispiel SSH, Webserver oder Datenbanken. Auf vielen Linux-Systemen werden Dienste mit systemd verwaltet. Das Werkzeug systemctl dient zum Starten, Stoppen, Neustarten, Aktivieren und Prüfen von Diensten. active bedeutet, dass ein Dienst läuft, enabled bedeutet, dass er beim Systemstart automatisch gestartet wird. Bei Dienstproblemen sollten systemctl status, journalctl -u, offene Ports, Rechte, Konfiguration und Abhängigkeiten geprüft werden. Ein laufender Dienst ist nicht automatisch von außen erreichbar, weil Firewall-Regeln, Bind-Adresse und Netzwerkpfade ebenfalls passen müssen.

Merksätze

```
Prozess = laufendes Programm.

Programm = Datei auf dem System.
```

PID = Prozess-ID.

PPID = Elternprozess-ID.

Dienst = Hintergrundprogramm mit Aufgabe.

Daemon = Hintergrunddienst.

ps zeigt Prozesse.

top zeigt Prozesse live.

htop ist komfortabler,
falls installiert.

kill sendet Signale.

TERM beendet sauber.

KILL beendet hart.

kill -9 nur im Notfall.

systemd verwaltet Dienste und Systemstart.

systemctl verwaltet systemd-Units.

.service steht für Dienst.

status prüft Dienstzustand.

start startet jetzt.

stop stoppt jetzt.

restart startet neu.

reload lädt Konfiguration neu.

enable aktiviert Autostart.

disable deaktiviert Autostart.

active heißt:
läuft gerade.

enabled heißt:
startet beim Boot.

Dienst läuft heißt nicht automatisch:
Port erreichbar.

127.0.0.1 heißt:
nur lokal.

0.0.0.0 heißt:
alle IPv4-Interfaces.

journalctl -u zeigt Dienstlogs.

Dienstfehler mit Status,
Logs,
Ports,
Rechten,
Konfiguration
und Abhängigkeiten prüfen.