9. Logs, Monitoring und Fehlersuche

9.1 Logs, Monitoring und Fehlersuche

Logs und Monitoring sind zentrale Werkzeuge, um Linux-Systeme zu überwachen, Fehler zu finden und Sicherheitsereignisse nachzuvollziehen.

Unter Linux werden viele Ereignisse protokolliert:

```
- Systemstarts
- Dienststarts
- Fehler
- Warnungen
- Anmeldungen
- SSH-Zugriffe
- Kernelmeldungen
- Paketinstallationen
- Netzwerkprobleme
- Sicherheitsereignisse
```

Für Fachinformatiker Systemintegration ist dieses Thema besonders wichtig, weil man im Serverbetrieb nicht nur Befehle ausführt, sondern Probleme gezielt anhand von Logs, Messwerten und Statusinformationen eingrenzen muss.

Merksatz:

```
Gute Fehlersuche basiert auf Logs,
Messwerten
und systematischer Eingrenzung.
```

Lernziele

Nach dieser Seite solltest du erklären können:

```
- was Logs sind
- wo Linux-Logs häufig liegen
- was journalctl macht
- was /var/log enthält
- wie man Dienstlogs prüft
- wie man aktuelle Logeinträge live mitliest
```

- wie man CPU,
  RAM,
  Speicher
  und Prozesse prüft
- wie man typische Fehlerbilder systematisch eingrenzt
- warum Monitoring wichtig ist
- was Alerting bedeutet

Grundidee

Logs sind Protokolle von Ereignissen.

Sie helfen bei Fragen wie:

- Warum startet ein Dienst nicht?
- Warum schlägt eine Anmeldung fehl?
- Warum ist ein Server langsam?
- Warum ist Speicher voll?
- Warum wird eine Verbindung blockiert?
- Wann trat der Fehler erstmals auf?
- Welcher Benutzer war beteiligt?
- Welche Meldung schreibt der Dienst?

Merksatz:

Logs beantworten oft:
was,
wann,
wo
und warum etwas passiert ist.

Typische Logquellen

Quelle	Bedeutung
systemd-Journal	zentrale systemd-Logs
/var/log	klassische Logdateien
Dienstlogs	Logs einzelner Anwendungen
Kernelmeldungen	Hardware, Treiber, Kernel

Quelle	Bedeutung
Authentifizierungslogs	Login, sudo, SSH
Paketmanager-Logs	Installationen und Updates
Webserver-Logs	HTTP-Zugriffe und Fehler
Firewall-Logs	erlaubter oder blockierter Verkehr
Anwendungslogs	Fehler und Ereignisse von Anwendungen

Merksatz:

```
Nicht alle Logs liegen an derselben Stelle.
```

/var/log

Viele klassische Logdateien liegen unter:

```
/var/log
```

Beispiele:

Pfad	Bedeutung
/var/log/syslog	allgemeine Systemmeldungen, häufig Debian/Ubuntu
/var/log/messages	allgemeine Systemmeldungen, häufig RHEL/Fedora-Umfeld
/var/log/auth.log	Authentifizierung, häufig Debian/Ubuntu
/var/log/secure	Authentifizierung, häufig RHEL/Fedora-Umfeld
/var/log/kern.log	Kernelmeldungen, je nach Distribution
/var/log/dmesg	Boot- und Kernelmeldungen, je nach Distribution
/var/log/nginx	Nginx-Logs
/var/log/apache2	Apache-Logs
/var/log/apt	apt-Logs
/var/log/journal	dauerhaftes systemd-Journal, falls aktiviert

Wichtig:

```
Welche Dateien vorhanden sind,
hängt von Distribution,
```

```
Diensten
und Logging-Konfiguration ab.
```

Merksatz:

```
/var/log ist der klassische Ort für Logdateien.
```

---

systemd-Journal

Auf vielen modernen Linux-Systemen sammelt systemd Logs im Journal.

Das Journal enthält unter anderem:

```
- Systemmeldungen
- Dienstmeldungen
- Bootmeldungen
- Fehler
- Warnungen
- Logs einzelner Units
```

Auswertung mit:

```
journalctl
```

Merksatz:

```
journalctl liest das systemd-Journal.
```

---

journalctl Grundlagen

Alle Journal-Einträge anzeigen:

```
journalctl
```

Letzte Einträge anzeigen:

```
journalctl -n 50
```

Live mitlesen:

```
journalctl -f
```

Logs seit letztem Boot:

```
journalctl -b
```

Logs eines Dienstes:

```
journalctl -u ssh
```

Live-Logs eines Dienstes:

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

Merksatz:

```
journalctl ist eines der wichtigsten Werkzeuge für Linux-Fehlersuche.
```

Dienstlogs mit journalctl

Wenn ein Dienst Probleme macht, prüft man zuerst:

```
systemctl status dienstname
```

Dann:

```
journalctl -u dienstname
```

Beispiel:

```
systemctl status nginx

journalctl -u nginx
```

Nur letzte Einträge:

```
journalctl -u nginx -n 50
```

Live mitlesen:

```
journalctl -u nginx -f
```

Seit letztem Boot:

```
journalctl -u nginx -b
```

Merksatz:

```
Dienstproblem:
status prüfen,
dann journalctl -u.
```

Zeitfilter bei journalctl

journalctl kann nach Zeit filtern.

Beispiele:

```
journalctl --since "1 hour ago"

journalctl --since "today"

journalctl --since "2026-07-08 10:00"

journalctl --until "2026-07-08 12:00"
```

Kombiniert mit Dienst:

```
journalctl -u ssh --since "today"
```

Merksatz:

Zeitfilter helfen,
Fehler auf einen Zeitraum einzugrenzen.

Prioritäten bei journalctl

Logs haben Prioritäten.

Nur Fehler anzeigen:

```
journalctl -p err
```

Warnungen und Schlimmeres anzeigen:

```
journalctl -p warning
```

Häufige Prioritäten:

Priorität	Bedeutung
emerg	System unbenutzbar
alert	sofortiges Eingreifen nötig
crit	kritischer Fehler
err	Fehler
warning	Warnung
notice	Hinweis
info	Information
debug	Debug-Ausgabe

Merksatz:

```
journalctl -p err zeigt Fehler.
```

---

dmesg

dmesg zeigt Kernelmeldungen.

Befehl:

```
dmesg
```

Menschenlesbare Zeitstempel:

```
dmesg -T
```

Typische Inhalte:

```
- Hardwareerkennung
- Treiberprobleme
- USB-Geräte
- Festplattenfehler
- Kernelwarnungen
- Netzwerkadapter
- Dateisystemfehler
```

Merksatz:

```
dmesg hilft bei Kernel-,
Hardware-
und Treiberproblemen.
```

Logdateien mit less lesen

Lange Logdateien liest man besser mit less.

Beispiel:

```
less /var/log/syslog
```

In less suchen:

```
/error
```

Nächster Treffer:

```
n
```

Verlassen:

```
q
```

Merksatz:

```
less ist besser als cat für lange Logs.
```

Logdateien live mitlesen

Mit tail kann man neue Logeinträge live verfolgen.

Beispiel:

```
tail -f /var/log/syslog
```

Letzte 100 Zeilen und live weiter:

```
tail -n 100 -f /var/log/syslog
```

Typische Nutzung:

```
Dienst neu starten

gleichzeitig Logs beobachten

Fehler direkt sehen
```

Merksatz:

```
tail -f liest neue Logeinträge live mit.
```

---

grep in Logs

Mit grep sucht man gezielt nach Begriffen.

Beispiele:

```
grep -i "error" /var/log/syslog

grep -i "failed" /var/log/auth.log

grep -i "denied" /var/log/auth.log
```

Mit Zeilennummer:

```
grep -n "error" datei.log
```

Groß-/Kleinschreibung ignorieren:

```
grep -i "error" datei.log
```

Merksatz:

```
grep filtert relevante Logzeilen.
```

Typische Suchbegriffe in Logs

Suchbegriff	Mögliche Bedeutung
error	Fehler
failed	fehlgeschlagen
denied	verweigert
refused	abgelehnt
timeout	Zeitüberschreitung
not found	nicht gefunden
permission	Rechteproblem
authentication	Anmeldung
invalid	ungültig
disconnect	Verbindung getrennt
no space	Speicherproblem
read-only	nur lesbares Dateisystem

Merksatz:

```
Gute Suchbegriffe beschleunigen Logauswertung.
```

Logrotation

Logs wachsen mit der Zeit.

Damit sie nicht unbegrenzt Speicher verbrauchen, werden sie rotiert.

Logrotation bedeutet:

```
alte Logs werden umbenannt,
komprimiert,
gelöscht
oder archiviert.
```

Typisches Werkzeug:

```
logrotate
```

Beispielhafte rotierte Dateien:

```
syslog

syslog.1

syslog.2.gz
```

Merksatz:

```
Logrotation verhindert,
dass Logs den Speicher füllen.
```

Warum Logs Speicher füllen können

Logs können sehr groß werden, wenn:

```
ein Dienst ständig Fehler schreibt

Debug-Logging aktiviert ist

Angriffe viele Einträge erzeugen

Logrotation nicht funktioniert

ein Dienst in einer Fehlerschleife hängt
```

Folgen:

```
Dateisystem voll

Dienste können nicht schreiben

Updates schlagen fehl

System wird instabil
```

Prüfen:

```
df -h

du -sh /var/log
```

Merksatz:

```
Viele Logs können Ursache und Folge eines Problems sein.
```

Monitoring

Monitoring bedeutet:

```
Systeme,
Dienste
Ressourcen
```

und Zustände regelmäßig überwachen.

Überwacht werden zum Beispiel:

- CPU
- RAM
- Speicherplatz
- Inodes
- Netzwerk
- Dienste
- Ports
- Antwortzeiten
- Backups
- Zertifikate
- Logs
- Sicherheitsereignisse

Merksatz:

Monitoring soll Probleme erkennen,
bevor Benutzer sie melden.

Alerting

Alerting bedeutet:

Bei bestimmten Ereignissen oder Grenzwerten wird eine Benachrichtigung ausgelöst.

Beispiele:

Speicher über 90 Prozent

Backup fehlgeschlagen

Dienst nicht erreichbar

Zertifikat läuft bald ab

CPU dauerhaft sehr hoch

viele fehlgeschlagene Logins

Wichtig:

Alarme müssen sinnvoll sein.
Zu viele unwichtige Alarme führen zu Alarmmüdigkeit.

Merksatz:

Alerting macht aus Monitoring eine aktive Warnung.

Wichtige Systemwerte

Wert	Warum wichtig?
CPU	hohe Last kann Dienste verlangsamen
RAM	Speichermangel führt zu Swap oder Fehlern
Swap	starke Nutzung kann System verlangsamen
Speicherplatz	volle Dateisysteme verhindern Schreibzugriffe
Inodes	zu viele Dateien verhindern neue Dateien
Load Average	zeigt Systemlast
Netzwerk	Verbindungsprobleme oder Paketverlust
Dienststatus	zeigt, ob Dienste laufen
Logs	zeigen Ursachen und Fehler

Merksatz:

Langsame Systeme brauchen Messwerte,
keine Vermutungen.

CPU und Prozesse prüfen

Prozesse und CPU prüfen mit:

```
top
```

oder:

```
htop
```

Prozesse anzeigen:

```
ps aux
```

Nach Prozess suchen:

```
ps aux | grep nginx
```

Mögliche Ursachen hoher CPU:

```
hohe Nutzerlast

Endlosschleife

fehlerhafter Prozess

Angriff

falsche Konfiguration
```

Merksatz:

```
Hohe CPU immer pro Prozess einordnen.
```

RAM und Swap prüfen

RAM prüfen:

```
free -h
```

Live prüfen:

```
top
```

Swap prüfen:

```
swapon --show
```

Mögliche Fehlerbilder:

```
System langsam

Prozesse werden beendet

starke Swap-Nutzung

Dienst startet nicht

Out-of-Memory-Ereignis
```

Merksatz:

```
Swap ist langsamer als RAM.
Viel Swap kann System stark verlangsamen.
```

Out of Memory

Wenn der Arbeitsspeicher knapp wird, kann Linux Prozesse beenden, um das System zu stabilisieren.

Das nennt man häufig:

```
OOM
```

oder:

```
Out of Memory
```

Suchen in Logs:

```
journalctl -k | grep -i "out of memory"

dmesg -T | grep -i "killed process"
```

Merksatz:

```
Wenn Prozesse plötzlich verschwinden,
OOM in Kernelmeldungen prüfen.
```

Speicherplatz prüfen

Dateisysteme prüfen:

```
df -h
```

Inodes prüfen:

```
df -i
```

Große Ordner prüfen:

```
du -sh /var/log

du -sh /var/lib

du -sh /home/*
```

Typische Ursachen:

```
Logs

Backups

Datenbanken

Container-Images
```

```
Cache

temporäre Dateien
```

Merksatz:

```
No space left on device:
df -h und df -i prüfen.
```

Netzwerk prüfen

IP-Adressen:

```
ip addr
```

Routen:

```
ip route
```

Gateway testen:

```
ping -c 4 gateway-ip
```

DNS prüfen:

```
dig example.com
```

Ports prüfen:

```
ss -tulpen
```

Webdienst prüfen:

```
curl -I http://localhost
```

Merksatz:

Netzwerkfehler Schritt für Schritt prüfen:
IP,
Route,
DNS,
Port,
Dienst.

Dienststatus prüfen

Dienststatus:

```
systemctl status dienstname
```

Autostart prüfen:

```
systemctl is-enabled dienstname
```

Aktiv prüfen:

```
systemctl is-active dienstname
```

Logs prüfen:

```
journalctl -u dienstname
```

Merksatz:

```
Dienstfehler:
systemctl status
und journalctl -u.
```

HTTP-Fehler einordnen

Code	Bedeutung	Wahrscheinliche Richtung
200	OK	Dienst antwortet
301 / 302	Weiterleitung	Webserver oder Anwendung

Code	Bedeutung	Wahrscheinliche Richtung
401	nicht authentifiziert	Login fehlt oder falsch
403	verboten	Rechteproblem
404	nicht gefunden	Pfad oder Ressource falsch
500	interner Serverfehler	Anwendung oder Backend
502	Bad Gateway	Proxy erreicht Backend nicht
503	Dienst nicht verfügbar	Backend oder Dienst nicht bereit

Merksatz:

```
HTTP-Code zeigt,
auf welcher Ebene man weiter suchen sollte.
```

Fehlersuche nach Änderung

Wenn ein Fehler direkt nach einer Änderung auftritt, ist diese Änderung besonders relevant.

Mögliche Änderungen:

```
Update

Konfigurationsänderung

Firewall-Regel

Rechteänderung

Zertifikatswechsel

Mountänderung

DNS-Änderung

Neustart
```

Sinnvolle Fragen:

```
Was wurde geändert?

Wann wurde geändert?

Wer hat geändert?

Gibt es ein Backup?

Gibt es einen Rollback-Plan?
```

Merksatz:

```
Fehler nach Änderung:
Änderung zuerst prüfen.
```

Systematische Fehlersuche

Eine gute Reihenfolge:

```
1. Fehler genau beschreiben

2. Zeitpunkt feststellen

3. betroffene Systeme eingrenzen

4. letzte Änderungen prüfen

5. Status prüfen

6. Logs prüfen

7. Ressourcen prüfen

8. Netzwerk prüfen

9. Berechtigungen prüfen

10. Maßnahme testen und dokumentieren
```

Merksatz:

> Nicht raten,
> sondern eingrenzen.

Einzelnes System oder viele Systeme?

Wichtige Eingrenzung:

Betroffen	Wahrscheinliche Richtung
ein Benutzer	Benutzerrechte, Client, Konto
alle Benutzer	Dienst, Server, Netzwerk, Authentifizierung
ein Server	Dienst, Ressourcen, lokale Konfiguration
alle Server	Netzwerk, DNS, zentrale Dienste
eine Anwendung	Anwendung, Datenbank, Backend
mehrere Anwendungen	Infrastruktur, DNS, Storage, Netzwerk

Merksatz:

> Anzahl der Betroffenen hilft bei der Eingrenzung.

Typische Linux-Fehlerbilder

Fehlerbild	Wahrscheinliche Ursache
Dienst startet nicht	Konfiguration, Rechte, Port, Abhängigkeit
Permission denied	Rechteproblem
No such file or directory	falscher Pfad oder Datei fehlt
No space left on device	Speicherplatz oder Inodes voll
Read-only file system	Dateisystem nur lesbar oder Fehler
Connection refused	Dienst läuft nicht oder Port geschlossen
Connection timed out	Firewall, Routing oder Ziel nicht erreichbar
Name or service not known	DNS oder Name falsch
Too many open files	Limit für offene Dateien erreicht
Out of memory	RAM knapp, OOM-Killer aktiv
Target is busy	Mount oder Datei wird noch verwendet

Connection refused

Bedeutung:

```
Das Zielsystem ist erreichbar,
aber der Dienst nimmt auf diesem Port keine Verbindung an.
```

Mögliche Ursachen:

```
Dienst läuft nicht

falscher Port

Dienst lauscht nur auf anderer Adresse

lokale Firewall lehnt aktiv ab
```

Prüfen:

```
systemctl status dienst

ss -tulpen

journalctl -u dienst
```

Merksatz:

```
Connection refused:
Ziel erreichbar,
Dienst oder Port problematisch.
```

Connection timed out

Bedeutung:

```
Die Verbindung erhält keine Antwort innerhalb der Zeit.
```

Mögliche Ursachen:

```
Firewall verwirft Pakete

Routingproblem

Ziel nicht erreichbar

falsche IP

Netzwerkproblem

Cloud Security Group blockiert
```

Prüfen:

```
ping

ip route

Firewall-Regeln

Netzwerkpfad
```

Merksatz:

```
Timeout deutet oft auf Firewall,
Routing
oder Nichterreichbarkeit hin.
```

Permission denied

Bedeutung:

```
Zugriff verweigert.
```

Mögliche Ursachen:

```
fehlende Dateirechte

falscher Besitzer

falsche Gruppe

fehlendes x-Recht auf Verzeichnis

sudo nötig

Dienstbenutzer hat keine Rechte
```

Prüfen:

```
whoami

id

ls -l datei

ls -ld verzeichnis
```

Merksatz:

```
Permission denied ist meist ein Rechteproblem.
```

No such file or directory

Bedeutung:

```
Datei oder Verzeichnis existiert nicht.
```

Mögliche Ursachen:

```
falscher Pfad

Tippfehler
```

falsche Groß-/Kleinschreibung

Datei wurde gelöscht

Mount fehlt

relativer Pfad falsch verstanden

Prüfen:

```
pwd

ls -la

find

mount
```

Merksatz:

```
Linux unterscheidet Groß-
und Kleinschreibung.
```

Too many open files

Bedeutung:

```
Ein Prozess oder Benutzer hat zu viele Dateien,
Sockets
oder Verbindungen geöffnet.
```

Mögliche Ursachen:

```
hohe Last

Fehler in Anwendung
```

```
zu niedriges Limit

Verbindungsleck
```

Prüfen:

```
Logs

Prozessstatus

Limits

offene Dateien
```

Merksatz:

```
Unter Linux zählen auch Netzwerkverbindungen als offene Dateien.
```

Logs und Sicherheit

Logs sind auch für Sicherheit wichtig.

Wichtige Ereignisse:

```
fehlgeschlagene Logins

sudo-Nutzung

SSH-Zugriffe

neue Benutzer

Rechteänderungen

Dienststarts

Firewall-Blockierungen
```

ungewöhnliche Prozesse

viele Anmeldeversuche

Beispiele:

```
journalctl -u ssh

grep -i "failed" /var/log/auth.log

grep -i "sudo" /var/log/auth.log
```

Merksatz:

```
Sicherheitsanalyse beginnt oft in Auth-
und Dienstlogs.
```

Backup und Logs

Logs können nach einem Vorfall wichtig sein.

Aber:

```
Logs auf demselben kompromittierten System können manipuliert werden.
```

Besser:

```
zentrale Logsammlung

SIEM

manipulationsgeschützte Logs

Zeitserver verwenden

ausreichende Aufbewahrung
```

Merksatz:

```
Für Sicherheitsvorfälle sind zentrale und geschützte Logs wertvoll.
```

Zeit und Logs

Korrekte Uhrzeit ist für Logs wichtig.

Wenn die Systemzeit falsch ist, werden Ereignisse schwer nachvollziehbar.

Prüfen:

```
timedatectl
```

Zeitdienst prüfen:

```
systemctl status systemd-timesyncd
```

oder je nach System:

```
chrony


ntpd
```

Merksatz:

```
Ohne korrekte Zeit sind Logs schwer auswertbar.
```

Dokumentation bei Fehlersuche

Bei Fehlersuche dokumentieren:

```
- Fehlerbild
- Zeitpunkt
- betroffene Systeme
- ausgeführte Prüfungen
- relevante Logmeldungen
- Ursache
- Maßnahme
```

- Ergebnis
- offene Punkte

Warum?

Nachvollziehbarkeit

Wiederverwendbarkeit

Übergabe an Kollegen

spätere Analyse

Prüfungsrelevanz

Merksatz:

Gute Administration dokumentiert Fehler und Lösung.

Typische Befehle dieser Seite

Befehl	Zweck
journalctl	systemd-Journal anzeigen
journalctl -u dienst	Logs eines Dienstes anzeigen
journalctl -f	Logs live mitlesen
journalctl -b	Logs seit letztem Boot
dmesg	Kernelmeldungen anzeigen
dmesg -T	Kernelmeldungen mit lesbarer Zeit
less logfile	lange Logdatei lesen
tail -f logfile	Log live verfolgen
grep muster logfile	Log nach Muster filtern
systemctl status dienst	Dienststatus prüfen
top	Prozesse und Last live prüfen
free -h	RAM und Swap prüfen
df -h	Speicherplatz prüfen

Befehl	Zweck
df -i	Inodes prüfen
du -sh ordner	Ordnergröße prüfen
ip addr	IP-Adressen prüfen
ip route	Routing prüfen
ss -tulpen	Ports prüfen
curl -I URL	Webantwort prüfen
timedatectl	Uhrzeit und Zeitsynchronisation prüfen

Typische Prüfungsfragen

Frage	Kurzantwort
Was sind Logs?	Protokolle von Ereignissen
Wo liegen viele klassische Logs?	/var/log
Was macht journalctl?	systemd-Journal anzeigen
Wie zeigt man Dienstlogs?	journalctl -u dienst
Wie liest man Logs live mit?	tail -f oder journalctl -f
Was macht dmesg?	Kernelmeldungen anzeigen
Was ist Monitoring?	regelmäßige Überwachung
Was ist Alerting?	Benachrichtigung bei Ereignissen
Warum ist Logrotation wichtig?	verhindert zu große Logs
Was prüft df -h?	Speicherplatz
Was prüft free -h?	RAM und Swap
Was bedeutet Connection refused?	Ziel erreichbar, Dienst/Port lehnt ab
Was bedeutet Connection timed out?	keine Antwort, oft Firewall/Routing
Warum ist korrekte Zeit wichtig?	Logs müssen zeitlich stimmen

Typische Prüfungsfallen

Falle	Richtig denken
Logs ignorieren	Logs liefern Ursache und Zeitbezug
nur Dienst neu starten	erst Status und Logs prüfen
cat für riesige Logs nutzen	less, tail oder grep verwenden
Monitoring mit Logging verwechseln	Logs protokollieren, Monitoring überwacht

Falle	Richtig denken
Alerting vergessen	Monitoring ohne Alarm wird leicht übersehen
hohe CPU ohne Prozessbezug nennen	verursachenden Prozess ermitteln
Speicher frei, aber Fehler beim Schreiben	Inodes prüfen
Zeitfehler ignorieren	falsche Zeit erschwert Loganalyse
Connection refused und timed out gleichsetzen	unterschiedliche Ursachen
nach Änderung nicht an Rollback denken	Änderungen dokumentieren und absichern

---

IHK-sichere Kurzformulierung

Logs sind Protokolle von Ereignissen und dienen der Fehlersuche, Überwachung und Sicherheitsanalyse. Viele klassische Logdateien liegen unter /var/log, während auf systemd-Systemen journalctl das zentrale Journal auswertet. Dienstprobleme werden typischerweise mit systemctl status und journalctl -u geprüft. tail -f oder journalctl -f können neue Logeinträge live anzeigen. Monitoring überwacht Ressourcen und Dienste wie CPU, RAM, Speicherplatz, Inodes, Netzwerk, Ports, Backups und Zertifikate. Alerting benachrichtigt bei Grenzwerten oder Fehlern. Eine systematische Fehlersuche grenzt das Problem anhand von Fehlerbild, Zeitpunkt, betroffenen Systemen, Logs, Ressourcen, Netzwerk, Rechten und letzten Änderungen ein.

Merksätze

Logs zeigen Ereignisse.

Monitoring überwacht Zustände.

Alerting warnt aktiv.

/var/log enthält viele klassische Logs.

journalctl liest das systemd-Journal.

journalctl -u zeigt Dienstlogs.

journalctl -f liest live mit.

journalctl -b zeigt Logs seit letztem Boot.

dmesg zeigt Kernelmeldungen.

less ist gut für lange Logs.

tail -f liest Logdateien live mit.

grep filtert relevante Logzeilen.

Logrotation verhindert zu große Logs.

df -h prüft Speicherplatz.

df -i prüft Inodes.

free -h prüft RAM und Swap.

top zeigt Last und Prozesse.

ss zeigt Ports.

curl prüft HTTP oder HTTPS.

systemctl status zeigt Dienstzustand.

Connection refused:
Dienst oder Port problematisch.

Connection timed out:
Firewall,
Routing
oder Nichterreichbarkeit.

Permission denied:
Rechteproblem.

No space left on device:
Speicherplatz oder Inodes prüfen.

Nach Änderung zuerst Änderung prüfen.

Nicht raten,
sondern eingrenzen.

Gute Fehlersuche wird dokumentiert.

Korrekte Uhrzeit ist wichtig für Logs.